ORACLE 12c
FORMS y REPORTS

Curso práctico de formación

Antolín Muñoz Chaparro

Libro disponible en: eybooks.com

Alfaomega

Oracle 12c
FORMS y REPORTS

Curso práctico de formación

Oracle 12c
FORMS y REPORTS

Curso práctico de formación

Antolín Muñoz Chaparro

Alfaomega

libros RC

Diseño de colección y pre-impresión: Grupo RC
Diseño de cubierta: Cuadratín

Datos catalográficos
Muñoz, Antolín
Oracle 12c FORMS y REPORTS. Curso práctico de formación
Primera Edición
Alfaomega Grupo Editor, S.A. de C.V., México

ISBN: 978-607-622-875-3

Formato: 17 x 23 cm Páginas: 580

Oracle 12c FORMS y REPORTS. Curso práctico de formación
Antolín Muñoz Chaparro
ISBN: 978-84-943055-7-3 edición original publicada por RC Libros, Madrid, España.

Primera edición: Alfaomega Grupo Editor, México, marzo 2017

Dr. Isidoro Olvera (Eje 2 sur) No. 74, Col. Doctores, 06720, Ciudad de México.

Miembro de la Cámara Nacional de la Industria Editorial Mexicana
Registro No. 2317

Pág. Web: **http://www.alfaomega.com.mx**
E-mail: **atencionalcliente@alfaomega.com.mx**

ISBN: 978-607-622-875-3

Nota importante:
La información contenida en esta obra tiene un fin exclusivamente didáctico y, por lo tanto, no está previsto su aprovechamiento a nivel profesional o industrial. Las indicaciones técnicas y programas incluidos, han sido elaborados con gran cuidado por el autor y reproducidos bajo estrictas normas de control. ALFAOMEGA GRUPO EDITOR, S.A. de C.V. no será jurídicamente responsable por: errores u omisiones; daños y perjuicios que se pudieran atribuir al uso de la información comprendida en este libro, ni por la utilización indebida que pudiera dársele. d e s c a r g a do en: e y b o oks. c o m

Edición autorizada para venta en México y todo el continente americano.

Impreso en México. Printed in Mexico.

Empresas del grupo:

México: Alfaomega Grupo Editor, S.A. de C.V. – Dr. Isidoro Olvera (Eje 2 sur) No. 74, Col. Doctores, C.P. 06720, Del. Cuauhtémoc, Ciudad de México. Tel.: (52-55) 5575-5022 – Fax: (52-55) 5575-2420 / 2490. Sin costo: 01-800-020-4396 E-mail: atencionalcliente@alfaomega.com.mx

Colombia: Alfaomega Colombiana S.A. – Calle 62 No. 20-46, Barrio San Luis, Bogotá, Colombia,
Tels.: (57-1) 746 0102 / 210 0415 – E-mail: cliente@alfaomega.com.co

Chile: Alfaomega Grupo Editor, S.A. – Av. Providencia 1443. Oficina 24, Santiago, Chile
Tel.: (56-2) 2235-4248 – Fax: (56-2) 2235-5786 – E-mail: agechile@alfaomega.cl

Argentina: Alfaomega Grupo Editor Argentino, S.A. – Av. Córdoba 1215 piso 10, C.P. 1055, Buenos Aires, Argentina. – Tel./Fax: (54-11) 4811-0887 y 4811 7183 – E-mail: ventas@alfaomegaeditor.com.ar

ÍNDICE

PRÓLOGO

Con la edición de este libro culmina el trabajo que comencé en el año 2011 con objeto de crear una trilogía de cursos basados en mi experiencia tanto didáctica como laboral con el software de Oracle.

Tras la publicación de los cursos de SQL y PL/SQL (en su versión 11g) ahora presento este último trabajo actualizado a la versión 12c. La versión 11g de los cursos SQL y PL/SQL que publiqué con anterioridad es plenamente compatible con la versión 12c de la base de datos Oracle, por lo que no es necesario ningún conocimiento nuevo para la realización de este curso de Forms y Reports en la versión 12c, si ya realizaron anteriormente los cursos de SQL y PL/SQL; no obstante, estoy trabajando en su actualización y en breve también saldrán publicados.

Este manual pretende dar una formación en el manejo de las herramientas Oracle Forms y Oracle Reports, que siguen después de varias décadas formando parte de los aplicativos de multitud de empresas relevantes en el mercado nacional e internacional.

Si no se tiene ningún conocimiento en el lenguaje de base de datos SQL de Oracle, ni en el lenguaje de Programación PL/SQL de Oracle, recomiendo primero la lectura de los cursos que he editado para estos productos, antes de abordar el estudio de este curso sobre Oracle Forms y Reports, de lo contrario le resultará

complicada la comprensión de algunos de los ejemplos prácticos incluidos en este manual.

Los archivos para la resolución de los supuestos se encuentran disponibles en la página de la editorial.

Plataforma de contenidos interactivos

Para tener acceso al material de la plataforma de contenidos interactivos del libro: *Oracle 12c, Forms y Reports*, 1a. edición, siga los siguientes pasos:

1. Ir a la página: http://libroweb.alfaomega.com.mx

2. Ir a la sección Catálogo y seleccionar la imagen de la portada del libro, al dar doble clic sobre ella, tendrá acceso al material descargable.

NOTA: Se recomienda respaldar los archivos descargados de las páginas web en un soporte físico.

CONCEPTOS SOBRE FORMS Y REPORTS 1

ORACLE FUSION MIDDLEWARE

Oracle Fusion Middleware 12c es la base de infraestructuras de aplicaciones de mayor aceptación hoy en día. Permite a las empresas crear y utilizar aplicaciones empresariales ágiles e inteligentes, y al mismo tiempo potenciar al máximo la eficacia informática aprovechando plenamente las arquitecturas modernas de hardware y software.

Componentes de Oracle Fusion Middleware

Las herramientas y software que pone a disposición Oracle dentro del conjunto Oracle Fusion Middleware se catalogan en los siguientes grupos:

- Cloud Application Foundation.
- Service-Oriented Architecture (SOA).
- Business Process Management.
- Data Integration.
- Development Tools.
- Enterprise Performance Management.
- Business Intelligence.
- System Management.
- Social Business & Collaboration.
- Identity Management.
- High Availability.
- Upgrade.

De este conjunto de aplicaciones y herramientas nos vamos a fijar en el apartado Development Tools, donde se encuentra englobado el software de Forms y Reports 12c que se trata en este libro.

ORACLE DEVELOPMENT TOOLS

Oracle ofrece la gama más completa e integrada de herramientas para el desarrollo de aplicaciones, desarrollo de base de datos e inteligencia de negocio, compatibles con cualquier enfoque de desarrollo, plataforma tecnológica o sistema operativo.

Oracle también ofrece diversas herramientas como Oracle JDeveloper, Oracle ADF y Oracle Developer Tools para Visual Studio que facilitan el desarrollo de aplicaciones web y bases de datos, en otros entornos de programación.

Dentro de este conjunto de herramientas, Oracle distingue las siguientes categorías:

- Application Development Framework.
- Developer Suite.
- Enterprise Pack for Eclipse.
- Forms Services.
- JDeveloper
- Mapviewer.
- User Productivity Kit.
- Virtual Assembly Builder.
- Workshop.

De este conjunto de herramientas nos vamos a centrar en Oracle Developer Suite porque es el grupo que contiene Reports y Forms, respectivamente.

ORACLE DEVELOPER SUITE

Oracle Developer Suite es el entorno de desarrollo más completo e integrado que combina la potencia de desarrollo de aplicaciones y las herramientas de inteligencia del negocio en una única suite (paquete integrado de aplicaciones) que está basada en los últimos estándares industriales. Oracle Developer Suite permite desarrollar rápidamente aplicaciones transaccionales de alta capacidad, que pueden ser desarrolladas en canales múltiples; incluyendo portales, servicios Web, dispositivos Wireless, y cualquiera de ellas puede ser extendida con las capacidades de la

inteligencia de negocio; incluyendo consultas "ad hoc" y análisis, informes Web de alta capacidad y análisis avanzado.

Las herramientas que se engloban en el conjunto Oracle Developer Suite son las siguientes:

- Oracle JDeveloper.
- Oracle Forms.
- Oracle Designer.
- Developer Suite Software Configuration Manager.
- Oracle Reports.
- Oracle Discover.
- Oracle Business Intelligence Beans.

Oracle Reports

Oracle Reports es un componente de Oracle Fusion Middleware que permite a las empresas la emisión de informes con una alta fidelidad. Proporciona al negocio un acceso inmediato a la información a todos los niveles, dentro y fuera de la organización de una forma escalable y segura.

Oracle Reports consta de una herramienta para el diseño y desarrollo de los informes que consta de una potente utilidad WYSIWYG (***W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et* que se traduce al castellano en "lo que ves es lo que obtienes") y un servidor de informes basado en JEE 5.0 con una arquitectura multitarea de acceso a las fuentes de datos para generar informes en cualquiera de los formatos más populares para web o en papel. Además permite la distribución de los informes a cualquier destino.

Oracle Forms

Oracle Forms es un componente de Oracle Fusion Middleware con una larga trayectoria dentro Oracle para el diseño y construcción de aplicaciones empresariales de forma rápida y eficiente.

Oracle mantiene el compromiso de desarrollo tecnológico para esta herramienta, así como su actualización como un componente más de las plataformas de Oracle.

Este compromiso con la tecnología de Forms le permite el aprovechamiento de la inversión realizada en aplicaciones de Forms de versiones anteriores, y la posibilidad de actualizarlas fácilmente a la nueva versión de Oracle Forms, así como tener la

posibilidad de construir a partir de ellas tecnologías web y servicios SOA (***S**ervice **O**riented **A**rchitectures* que se traduce al castellano en "Arquitectura Orientada a Servicios").

DESCARGA DEL SOFTWARE FORMS Y REPORTS

La descarga del software necesario y los procedimientos para instarlos son los mismos para las 3 casuísticas que se plantean a continuación:

- Un nuevo usuario de Oracle Forms y Reports que nunca antes instaló o configuró algún software de Oracle Forms y Reports.
- Un usuario que tiene instalado Oracle Forms y Reports versión 11g y quiere actualizarse a la versión 12c Release 12.2.1).
- Un usuario de OAS 10g (Oracle Application Server con alguna versión de Forms o Reports 10g) que quiere actualizarse a Oracle Forms y Reports 12c.

En todos los casos es necesario descargar e instalar los siguientes elementos:

- Oracle WebLogic Server.
- Oracle Forms and Reports 12c Release 12.2.1.

Tanto para la plataforma de Windows como para la de Linux, además es necesario tener instalado un JDK de Java.

Oracle WebLogic Server

La línea de productos de servidores de aplicaciones Oracle WebLogic es la plataforma más completa del mercado para desarrollar, implantar e integrar aplicaciones empresariales. Proporciona la base para la granja de aplicaciones (Application Grid), una arquitectura que permite a las empresas ofrecer un rendimiento superior al de la competencia pero que al mismo tiempo reduce los costes de explotación.

A la fecha de la redacción de este libro Oracle dispone como última versión de WebLogic: Oracle WebLogic Server 12c (12.2.1).

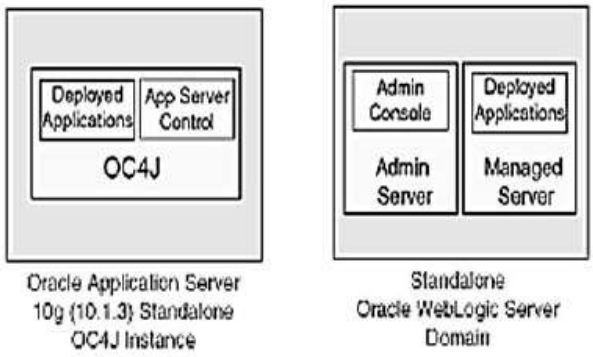
Desde la OTN de Oracle se puede descargar una versión de este producto siguiendo el enlace que se indica a continuación:
http://www.oracle.com/technetwork/middleware/weblogic/downloads/wls-main-097127.html

Descarga de Oracle Forms and Reports

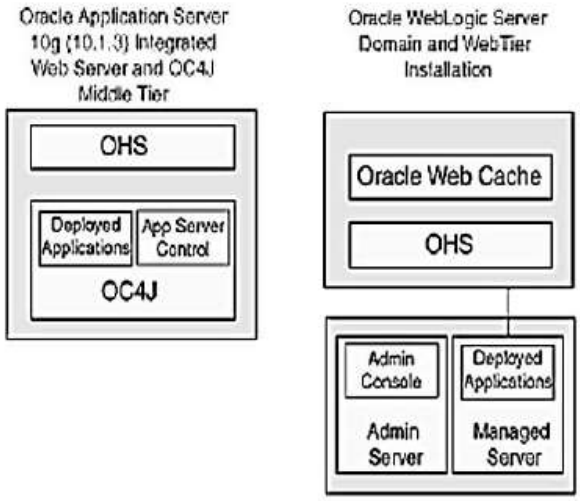
Desde la OTN de Oracle se puede descargar una versión de estos dos productos siguiendo el enlace que se indica a continuación:
http://www.oracle.com/technetwork/developer-tools/forms/downloads/index.html

COMPARATIVA DE ARQUITECTURAS 10g, 11g Y 12c

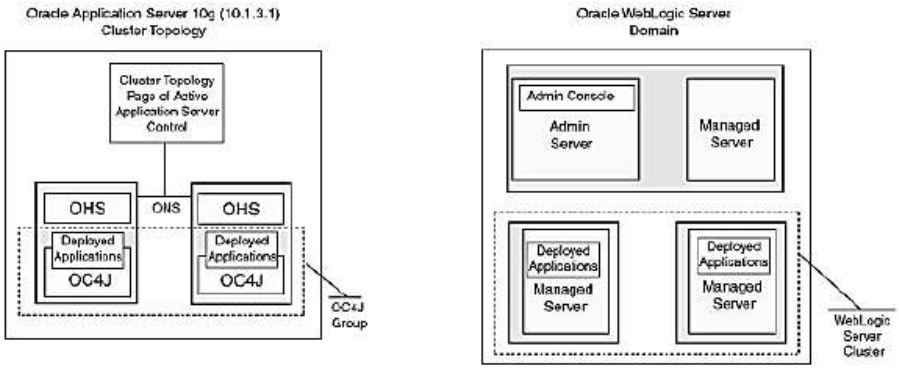
A continuación se muestra una gráfica comparativa de la arquitectura 10g con OC4J y la 11g con Weblogic (configuración "standalone"):



A continuación se muestra otra gráfica comparativa de la arquitectura 10g con OC4J y la 11g con Weblogic (configuración con integración en un servidor Web):



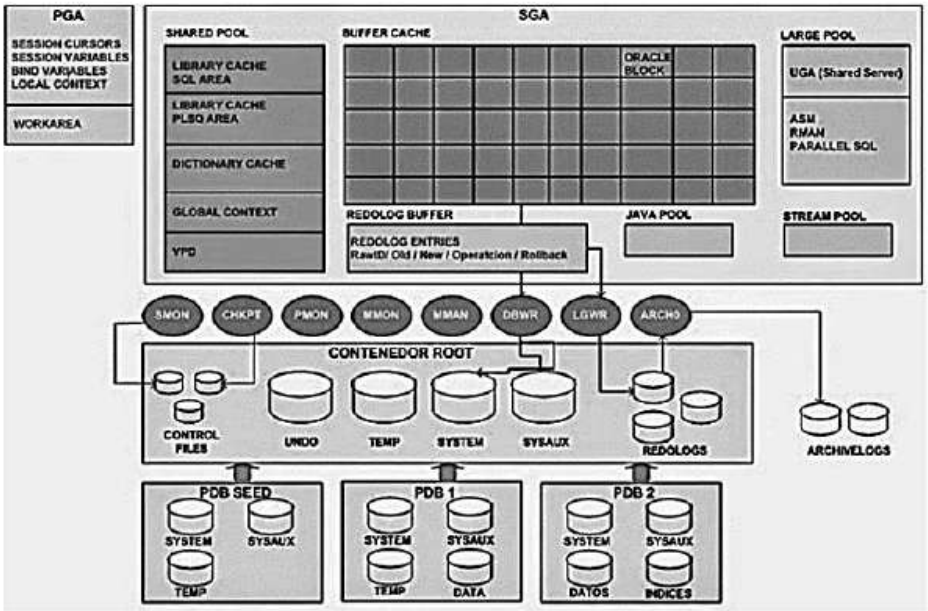
A continuación se muestra otra gráfica comparativa de la arquitectura 10g con OC4J y la 11g con Weblogic (configuración en un cluster):



Oracle 12c introduce una nueva arquitectura llamada Oracle MULTITENANT en la que se provee, a la base de datos, la capacidad de convertirse en un gran contenedor de bases de datos.

El contenedor es definido con el nombre de Multitenant Container Database (CDB) donde pueden ser incluidas desde 0 a más bases de datos llamadas Pluggable Databases (PDB).

En el siguiente gráfico se detalla cómo está compuesto el Multitenant Container Database (CDB).



Como se puede observar, el CDB está formado por una instancia (SGA + PGA), un grupo de procesos background, un contenedor Root y muchas bases de datos Pluggable (PDB).

Al tener una sola instancia, todos los PDBs comparten las mismas estructuras de memoria y, en consecuencia, el mismo archivo de parámetros spfile o pfile.

Si un cliente quiere consolidar muchas bases de datos en un solo servidor puede hacer uso de esta arquitectura y tener una sola instancia con muchas bases de datos de tipo Pluggable database.

Esto ayuda a optimizar el uso de la memoria debido a que se utiliza una gran instancia y un solo grupo de procesos background para todas las bases de datos Pluggable (PDB).

INSTALACIÓN DE WEBLOGIC EN WINDOWS 2

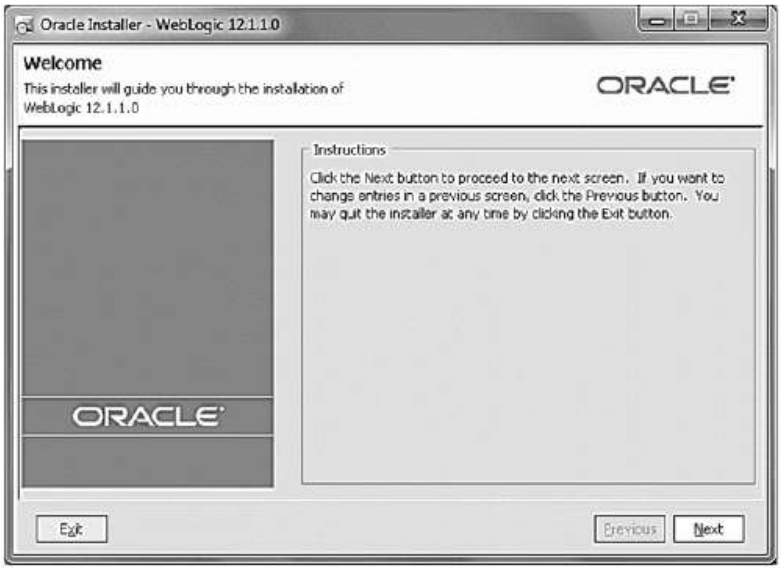
REQUERIMIENTOS MÍNIMOS

A continuación se especifican los requerimientos de hardware mínimos necesarios en un equipo con Microsoft Windows, para la instalación de Oracle WebLogic Server.

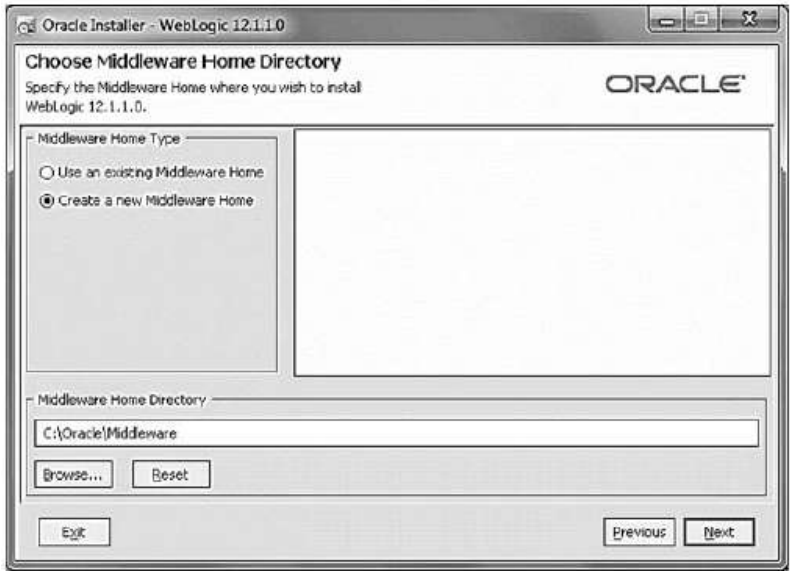
Componente	Requerimientos
Procesador	1 Ghz CPU
Disco duro	Para la instalación completa (incluyendo SDKs) se debe disponer de 3,9 GB libres en el disco duro (se incluye el espacio temporal de disco duro necesario para la instalación).
Memoria RAM	Mínimo 1 GB, aunque se recomienda 2 GB.
Paleta de colores y tarjeta gráfica	Para la instalación en modo gráfico se debe tener seleccionada una paleta gráfica de 8-bit de colores (256 colores).
JDK	La instalación del programa requiere que se tenga instalado en el equipo JRE (Java Runtime Environment).

INSTALACIÓN DEL SOFTWARE

A continuación se muestran de forma gráfica los pasos a realizar en la instalación de Oracle Weblogic en un equipo con Microsoft Windows.

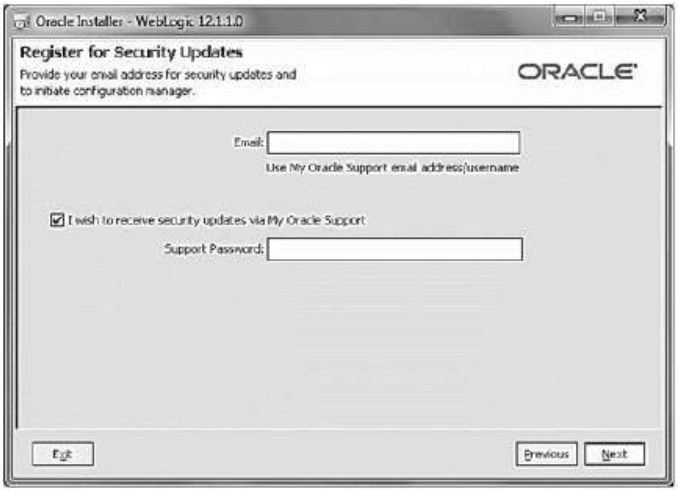


Para comenzar el proceso pulsamos el botón **Siguiente (Next).**



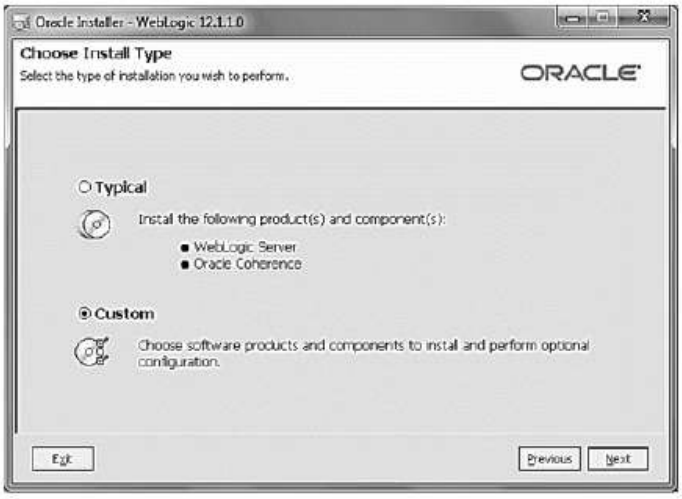
En esta pantalla tenemos que introducir la ruta donde se ubicará la instalación del Middleware (dejamos la ubicación por defecto **C:\Oracle\Middleware**, salvo que no haya espacio suficiente en la misma) y pulsaremos sobre el botón **Siguiente (Next)** para continuar.

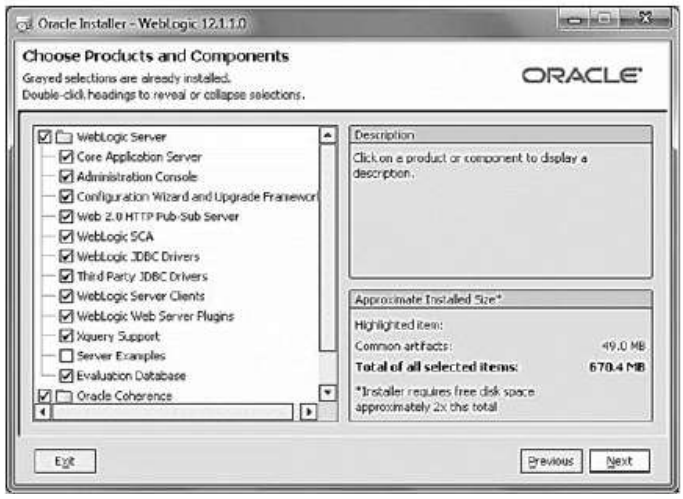
Si se quiere recibir soporte de Oracle, hay que marcar la casilla **"I wish to receive security updates via My Oracle Support"** y además introducir password del usuario que se tenga en Oracle Metalink. Igualmente si se quiere recibir comunicaciones de actualizaciones por e-mail, habrá que indicar la cuenta de correo electrónico en la casilla correspondiente.



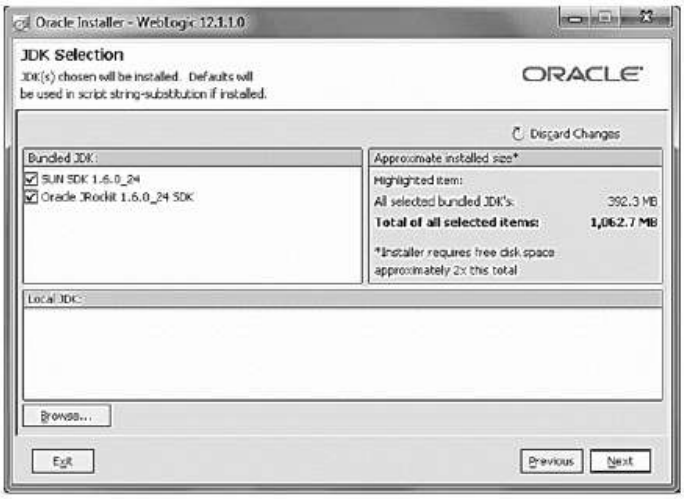
En cualquiera de los casos para continuar pulsaremos el botón **Siguiente (Next).**

Como se va a realizar una instalación de un nodo único, se selecciona la opción **"Custom (Personalizada)"** y pulsamos sobre el botón **Siguiente (Next)**.

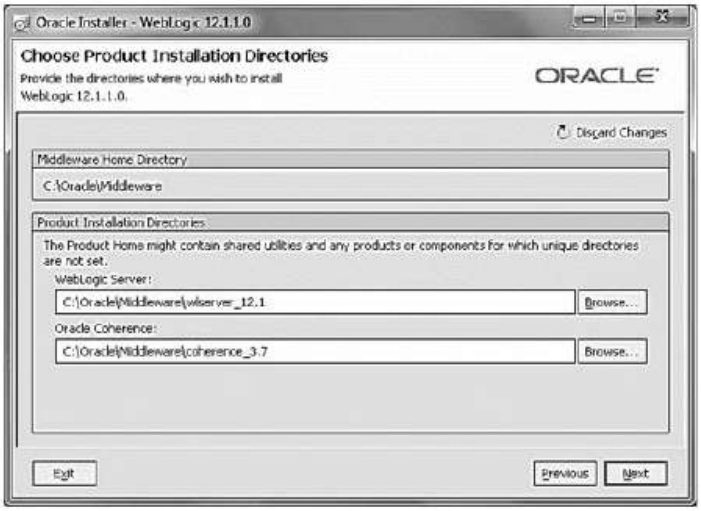




En esta pantalla nos aparece una lista con los productos disponibles en Weblogic Server para seleccionar aquellos que queremos instalar. Se ofrece en la siguiente pantalla una selección típica de instalación, aunque se podrán seleccionar o eliminar productos de la lista (para que no se instalen).

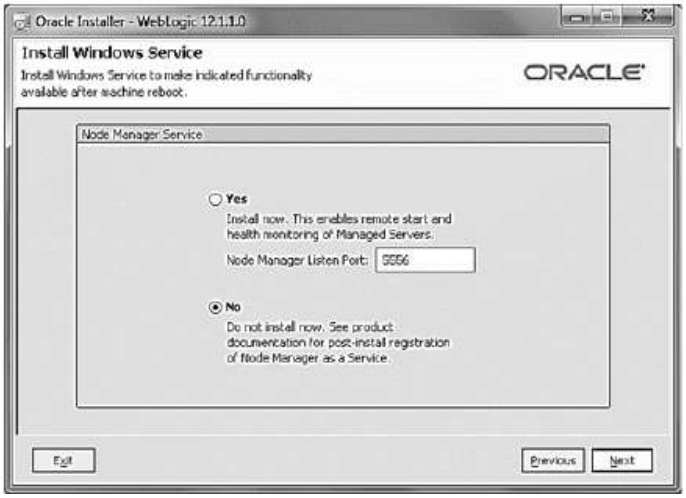


A continuación nos aparece una pantalla con la lista de plataformas JDKs. La lista que aparece en la misma diferirá dependiendo de la instalación específica que se esté utilizando. En esta pantalla se seleccionará el JDK o los JDKs que se quieran instalar. Además, también se puede seleccionar un JDK instalado de forma local. Una vez seleccionada la/s plataforma/s JDKs a utilizar con Weblogic pulsamos el botón **Siguiente (Next)**.



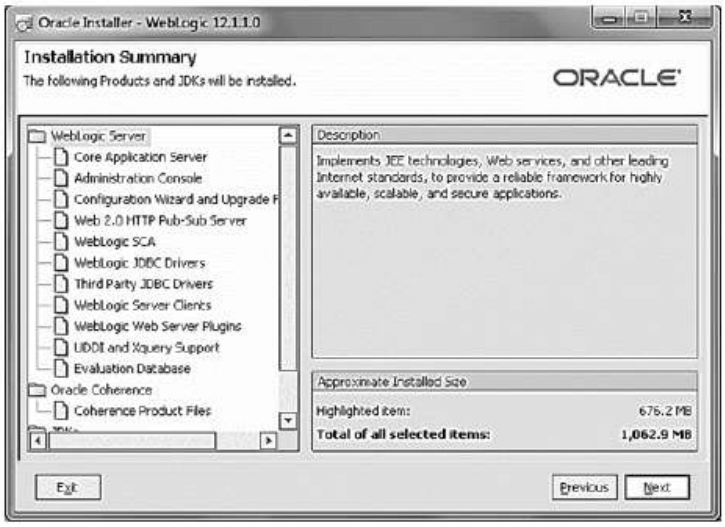
En esta pantalla, aceptamos por defecto los directorios para la instalación del servidor de Oracle Weblogic y pulsamos sobre el botón **Siguiente (Next)**.

En esta pantalla tenemos que seleccionar si se quieren instalar los servicios propios de gestión del nodo de Weblogic Server en la propia máquina de Windows. La gestión del nodo se utilizará para monitorizar, arrancar y parar los servicios instalación en el dominio de Weblogic. Si se selecciona la opción **Sí (Yes)**, que es la recomendada, hay que introducir el puerto por el que se escucharán las peticiones. Por defecto dicho puesto es el **5556**.



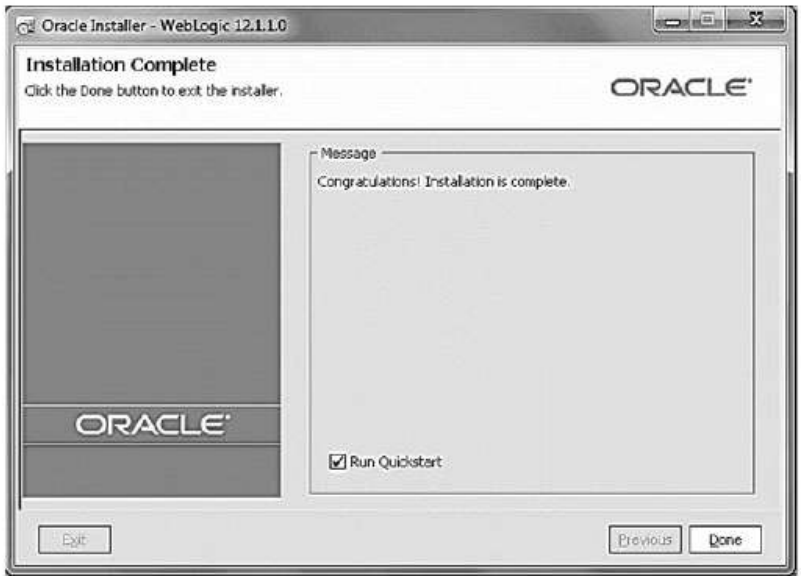
Seleccionamos la opción recomendada para crear un enlace (acceso desde Windows a un programa) para todos los usuarios del sistema operativo ("**Todos los Usuarios**" **–All Users–**) y pulsamos sobre el botón **Siguiente (Next).**

Revisamos el resumen de software pendiente de instalar y cuando estemos conformes pulsamos el botón **Siguiente (Next)**.



Esperamos hasta que se haya completado la instalación y pulsamos el botón **Siguiente (Next)**.

Para finalizar la instalación y salir de la misma, pulsamos sobre el botón **Listo (Done)**, marcando primero la casilla **Ejecutar Quickstart (Run Quickstart)**.



INSTALACIÓN DE FORMS Y REPORTS 12c EN WINDOWS 3

REQUERIMIENTOS MÍNIMOS

A continuación se especifican los requerimientos de hardware mínimos necesarios en un equipo con Microsoft Windows, para la instalación de Oracle Forms and Reports 12c.

Componente	Requerimientos
Procesador	Mínimo de 300 Mhz.
Disco duro	3600 MB
Memoria RAM	2 GB aunque se recomienda 4 GB.
Paleta de colores y tarjeta gráfica	Para la instalación en modo gráfico se debe tener seleccionada una paleta gráfica de 8-bit de colores (256 colores).
JDK	La instalación del programa requiere que se tenga instalado en el equipo JRE (Java Runtime Environment).
Base de datos	Se debe tener instalada ya una base de datos Oracle (recomendada 11g o 12c) y conocer la password del usuario SYS.

INSTALACIÓN DEL SOFTWARE

A continuación se muestran de forma gráfica los pasos a realizar en la instalación de Oracle Forms & Reports 12c en un equipo con Microsoft Windows.

Como paso previo habrá que comprobar que el fichero C:\windows\system32\drivers\etc\hosts contiene una entrada para el "localhost" y al menos otra con el nombre de la máquina donde se realiza la instalación. Un ejemplo sería el siguiente:

```
127.0.0.1       localhost
192.168.1.135   miequipwin98
```

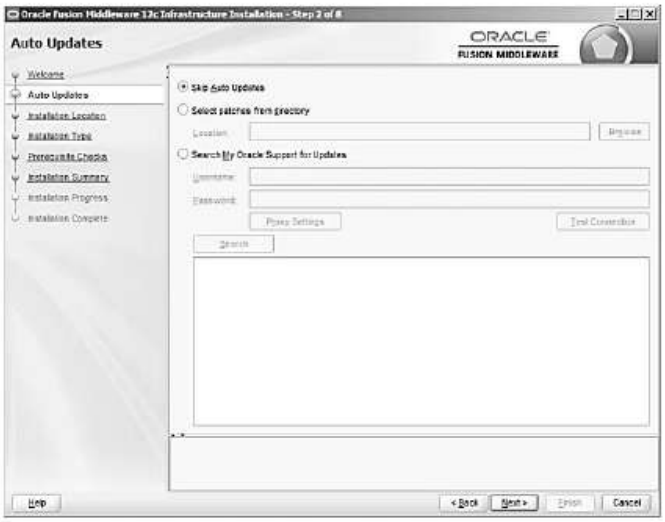
Además, será necesario disponer de una instalación JDK en el equipo. Si no dispone de ninguna versión, puede descargarla desde el siguiente enlace: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Instalación de Fusion Middleware

Se extraen los ficheros de instalación y se arranca el instalador automáticamente:

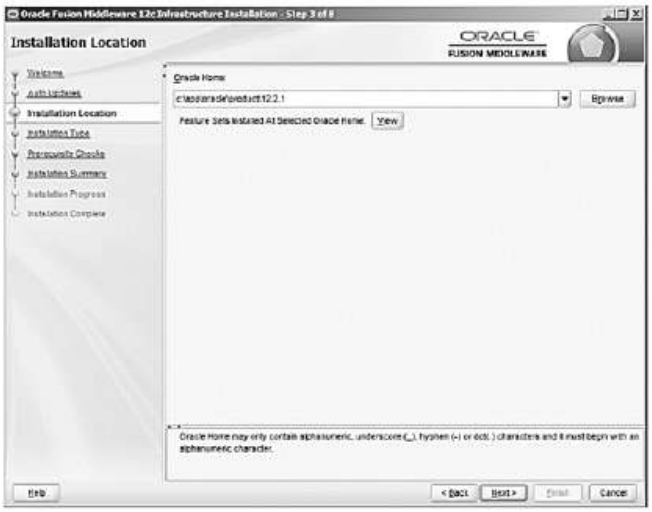
```
C:\media> unzip fmw_12.2.1.0.0_infrastructure_Disk1_1of1.zip

C:\media> %JAVA_HOME%\bin\java -jar
fmw_12.2.1.0.0_infrastructure.jar
```

Pulsamos sobre el botón **Siguiente (Next).**



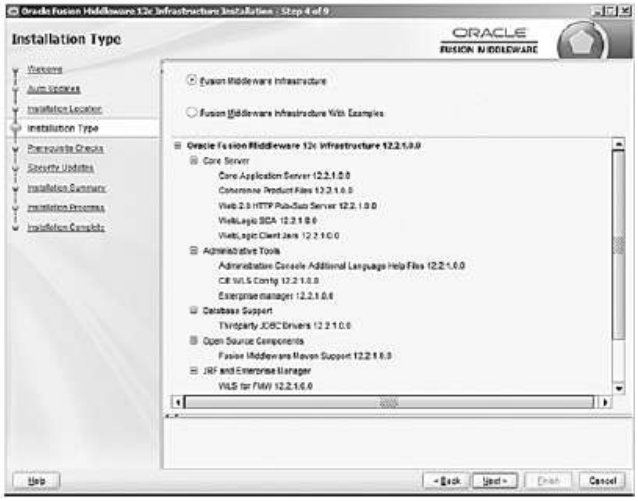
En esta pantalla seleccionamos la opción que queremos para las actualizaciones del software (para los propósitos de este libro, seleccionaremos la primera opción: **Omitir Actualizaciones de Software –Skip Auto Updates–**).

En esta pantalla se muestra el directorio donde se va a instalar el software. Por defecto la ruta es **C:\app\oracle\product\12.2.1.** A continuación pulsamos el botón **Siguiente (Next)**.



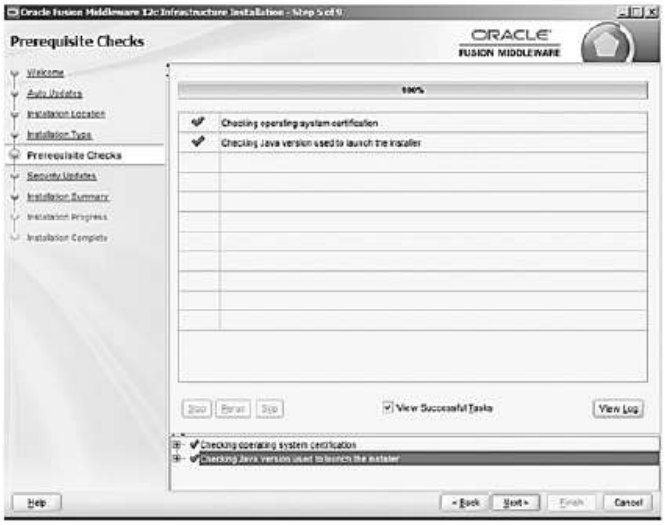
En esta pantalla aparece el tipo de instalación que se desea realizar. Tenemos dos opciones disponibles:

- La opción "**Fusion Middleware Infraestructure**", únicamente instala el producto.
- La opción "**Fusion Middleware Infraestructure With Examples**", instala el producto y una serie de ejemplos.



En nuestro caso seleccionamos la opción por defecto: **Fusion Middleware Infraestructure** y pulsamos el botón **Siguiente (Next).**

En esta pantalla el sistema analiza el ordenador donde se está instalando el producto

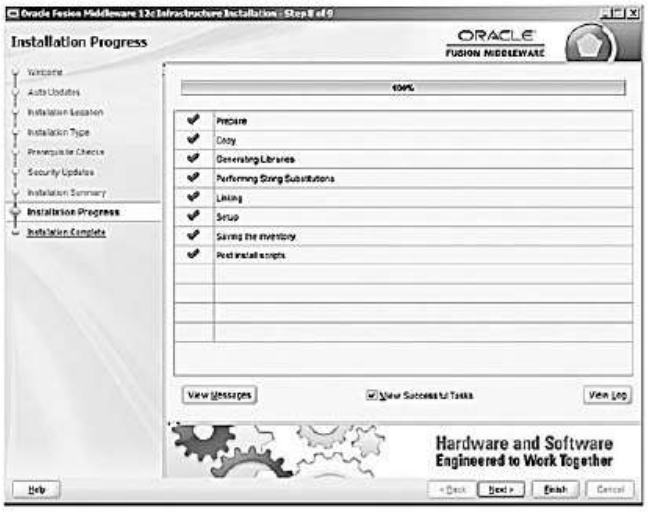


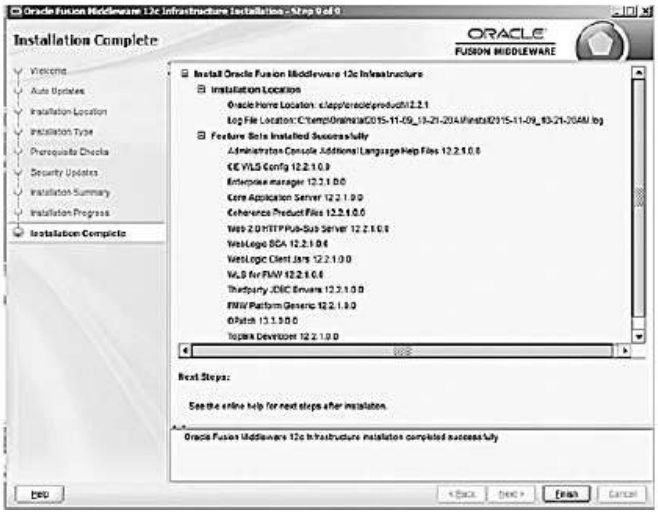
para comprobar que cumple todos los requisitos necesarios para llevar a cabo la instalación del software. Si falta algún requisito, se mostrará un breve error en la parte inferior de esta pantalla (zona de mensajes). Se puede intentar arreglar el error pulsando en **Reintentar.** Si queremos arreglar el error antes de continuar pulsaremos la opción **Cancelar.** Si se quiere ignorar el mismo o no ha habido ningún error y queremos continuar con la instalación se pulsará sobre el botón **Siguiente (Next).**

En esta pantalla se introducen los detalles de soporte de los que disponga el usuario en la contratación de los distintos productos de Oracle, o bien si no se dispone del servicio de soporte de Oracle, desmarcaremos la casilla **I wish to receive security updates via My Oracle Support**, y pulsaremos el botón **Siguiente (Next).** Seguidamente nos aparecerá una ventana emergente que nos avisa de no haber introducido ningún e-mail para recibir actualizaciones de seguridad. Pulsaremos el botón **Sí (Yes).**

En esta pantalla pulsamos el botón **Instalar (Install).**

Una vez completada la instalación, aparecerá esta pantalla con los procesos de la instalación. Si alguno de ellos hubiese dado error, se mostraría con una reseña en la parte izquierda. En caso contrario, todos aparecerán con el símbolo ✓. Para continuar el proceso pulsaremos la opción **Finalizar (Finish)**.



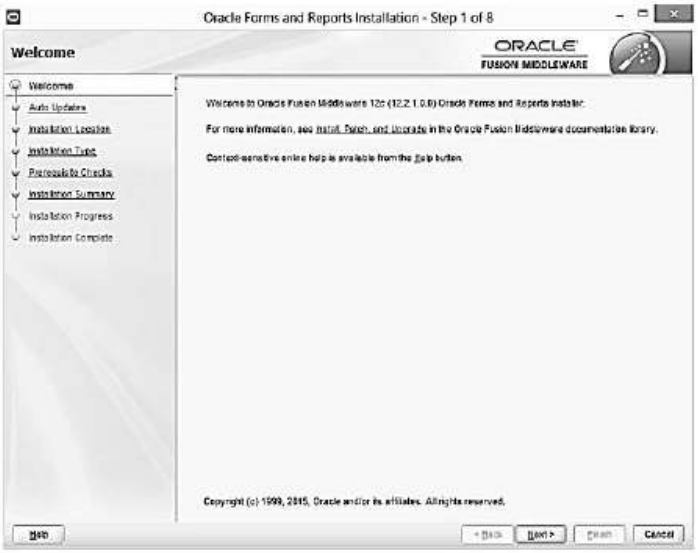


Una vez completada la instalación, se muestra esta pantalla de resumen de la instalación. Pulsaremos el botón **Finalizar (Finish)**.

Instalación de Forms and Reports 12c (12.2.1)

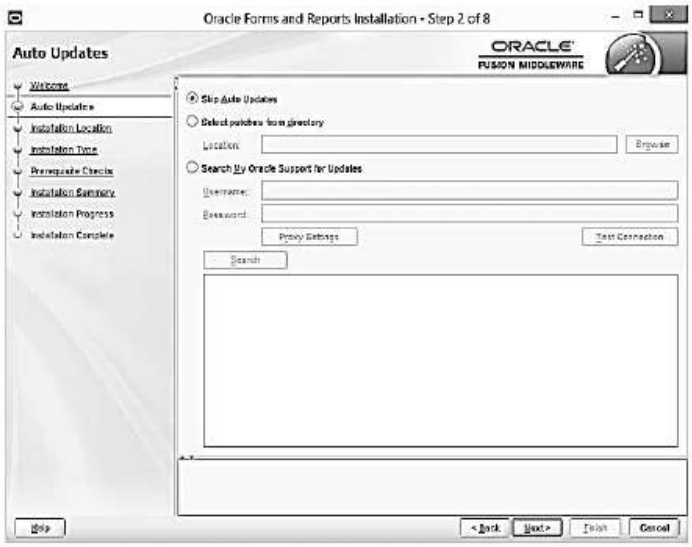
Se extraen los ficheros de instalación y se arranca el instalador automáticamente:

```
c:\media> unzip fmw_12.2.1.0.0_fr_linux64_Disk1_1of1.zip
c:\media> setup_fmw_12.2.1.0.0_fr_win64.exe
```



Pulsamos sobre el botón **Siguiente (Next).**

En esta pantalla seleccionamos la opción que queremos para las actualizaciones del software (para los propósitos de este libro, seleccionaremos la primera opción: **Omitir Actualizaciones de Software –Skip Auto Updates–**).



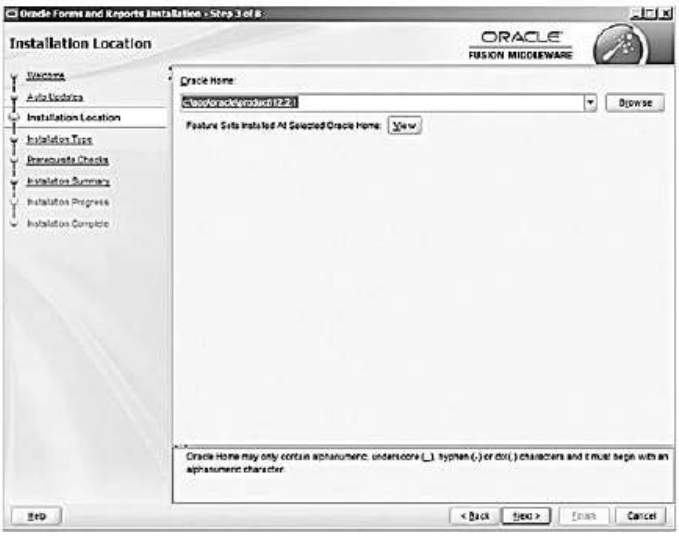
Disponemos de tres alternativas para realizar la actualización:

- No realizar actualizaciones de software (**Omitir Actualizaciones de Software –Skip Auto Updates–)**.
- Introducir las credenciales de una cuenta de Oracle para que se realicen las actualizaciones (**Buscar Actualizaciones en My Oracle Support)**. Una vez introducidas estas credenciales, si se quiere comprobar que son correctas y se tiene acceso a la cuenta de Oracle, se puede pulsar sobre el botón **Probar**

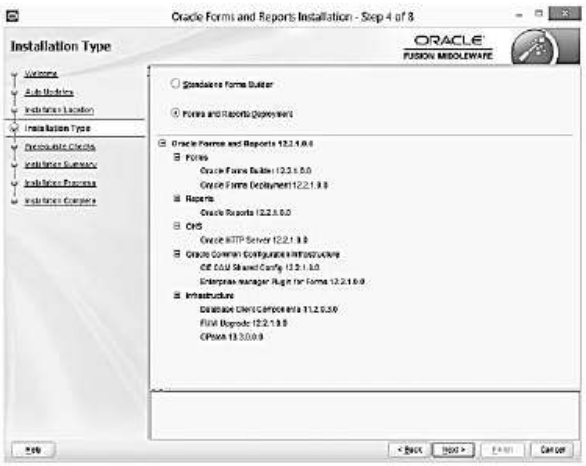
Conexión. De la misma forma, si fuese necesario modificar algún parámetro de la conexión a través de un proxy se puede realizar pulsando sobre el botón **Valores de Proxy**, en cuyo caso se mostrará la siguiente pantalla:

- Introducir un directorio local para las actualizaciones (**Buscar Actualizaciones en el Directorio Local)**. Si se encuentra alguna actualización en el directorio seleccionado, se mostrará el botón **Buscar Actualizaciones.**

Para continuar pulsamos el botón **Siguiente (Next)** y en caso de haber seleccionado una actualización, comenzará a realizarse la misma. Es posible que tras la actualización sea necesario reiniciar el sistema y la propia instalación, en este caso no se volverá a mostrar esta pantalla de actualización de software en el proceso de instalación.



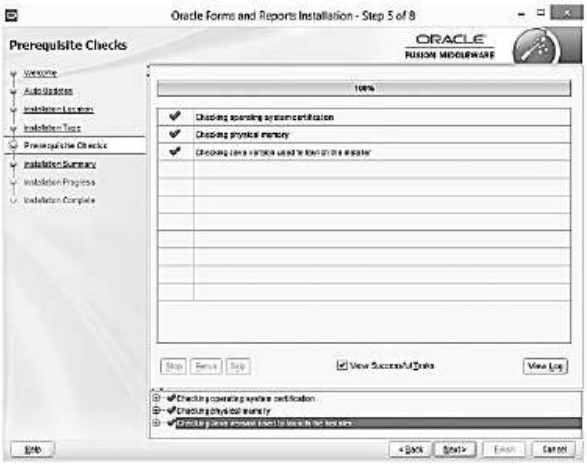
En esta pantalla se muestra el directorio donde se va a instalar el software. Por defecto la ruta es **C:\app\oracle\product\12.2.1.** A continuación pulsamos el botón **Siguiente (Next)**.



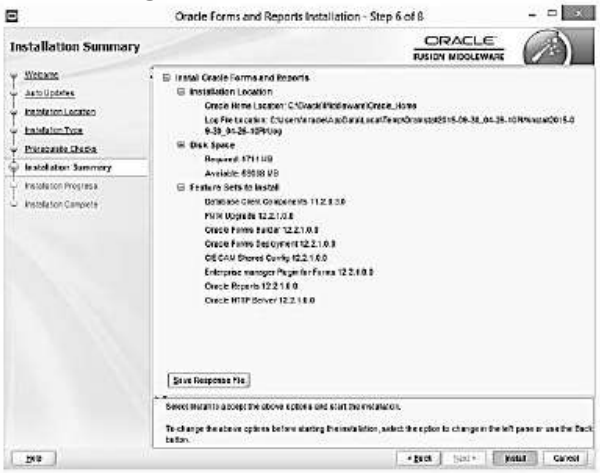
En esta pantalla aparece el tipo de instalación que se desea realizar. Tenemos dos opciones disponibles:

- La opción "**Standalone Forms Builder"** únicamente instala el producto y servicios de Forms.
- La opción "**Forms and Reports Deployment"** instala los productos y servicios de Forms y Reports.

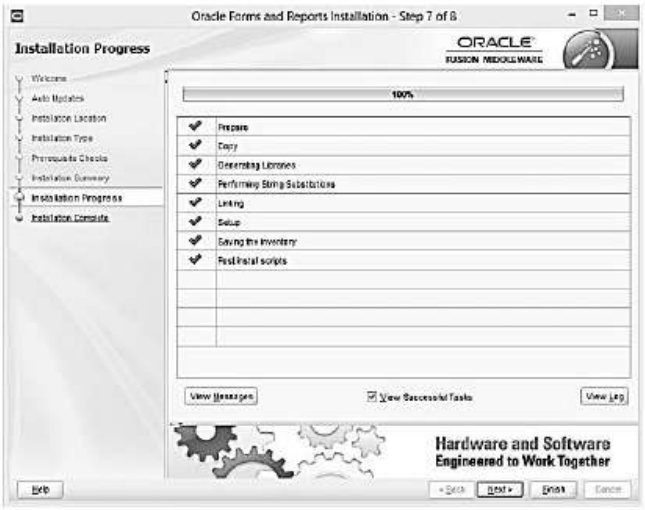
En nuestro caso seleccionamos la opción por defecto: **Standalone Forms Builder** y pulsamos el botón **Siguiente (Next).**



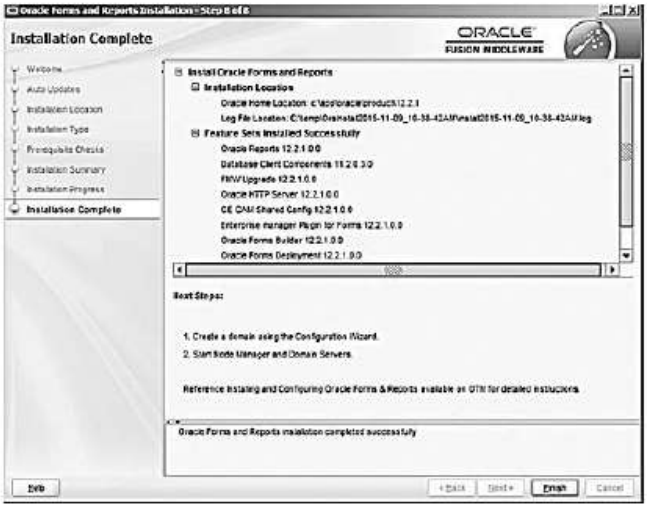
En esta pantalla el sistema analiza el ordenador donde se está instalando el producto para comprobar que cumple todos los requisitos necesarios para llevar a cabo la instalación del software. Si falta algún requisito, se mostrará un breve error en la parte inferior de esta pantalla (zona de mensajes). Se puede intentar arreglar el error pulsando en **Reintentar.** Si queremos arreglar el error antes de continuar, pulsaremos la opción **Cancelar.** Si se quiere ignorar el mismo o no ha habido ningún error y queremos continuar con la instalación, se pulsará sobre el botón **Siguiente (Next).**



En esta pantalla aparece el resumen de la instalación que se va a realizar y a continuación pulsamos el botón **Instalar (Install)**.



En esta pantalla aparece el progreso de la instalación. Al finalizar dicho proceso se muestran los resultados de los distintos pasos. Si alguno de ellos hubiese dado error, se mostraría con una reseña en la parte izquierda. En caso contrario, todos aparecerán con el símbolo ✓. Para continuar el proceso pulsaremos la opción **Finalizar (Finish)**.



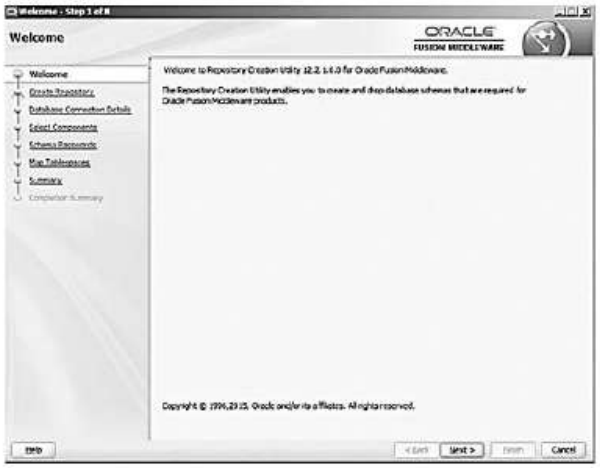
Al mostrarse esta pantalla, habremos concluido la instalación y pulsaremos el botón **Finalizar (Finish)**.

Ejecución de la utilidad RCU (Repository Creation Utility)

Antes de continuar con las siguientes tareas de instalación de Forms y Reports 12c, se recomienda la ejecución de la utilidad RCU (Repository Creation Utility) que se distribuye con el software de Oracle Fusion Middleware. Arrancaremos la utilidad con la siguiente instrucción:

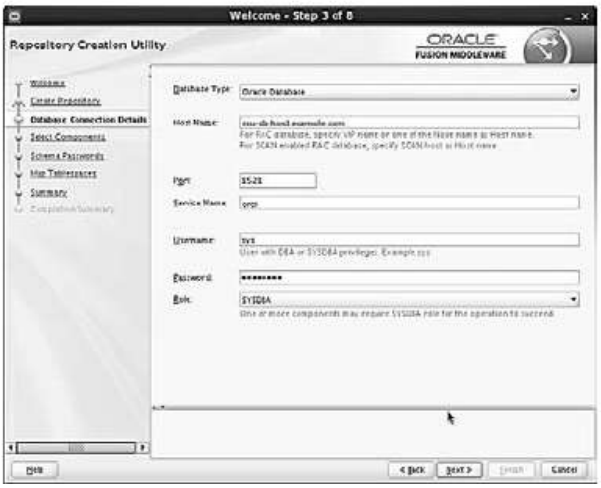
```
C:\app\oracle\product\12.2.1\oracle_common\bin\rcu.bat
```

A continuación se muestran los pasos que hay que seguir en dicha herramienta para su configuración e instalación.



En esta pantalla de bienvenida pulsaremos el botón **Next.**

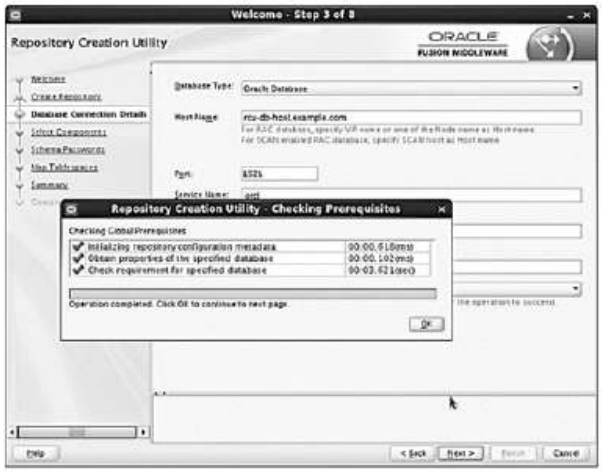
En esta pantalla se muestran las opciones para crear o borrar un repositorio. Si no se ha creado otro anteriormente, la opción por defecto es la que aparece marcada **Create Repository**, y dentro de la misma, también se mantendrá la subopción por defecto **System Load and Product Load.** A continuación pulsaremos el botón **Next.**



En esta pantalla se introducirán los detalles de conexión a la base de datos que se utilizará para la creación del repositorio. Los valores a introducir serán:

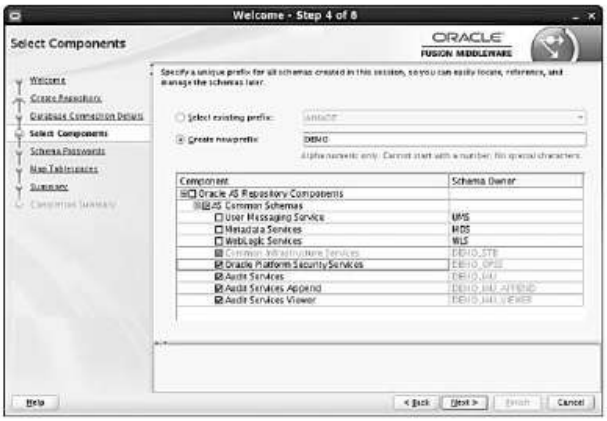
- Database Type: **Oracle Database.**
- Host Name: *nombre del equipo (o IP) donde esté instalada la base de datos.*
- Port: *el puerto de conexión a la base de datos (por defecto suele ser el 1521).*
- Service Name: *el nombre del servicio de conexión a la base de datos.*
- Username: **SYS**.
- Password: *la password que tenga el usuario SYS en la base de datos instalada.*
- Role: **SYSDBA**.

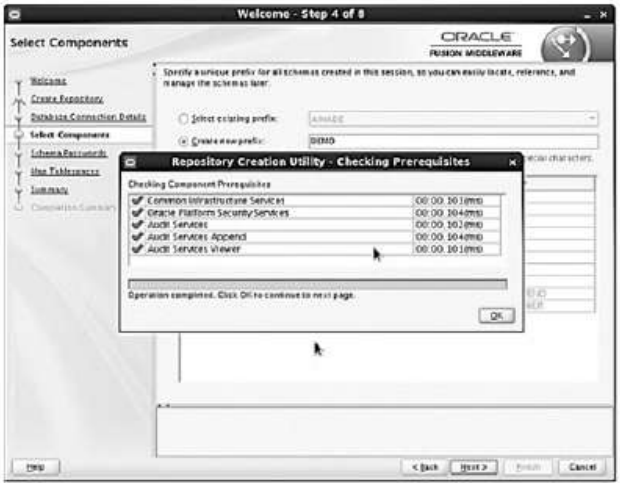
Después de introducir los datos se pulsará el botón **Next.**



En esta pantalla aparecen una serie de ventanas emergentes donde se muestra el progreso de chequeo de los prerrequisitos. Después se pulsará el botón **OK.**

En esta pantalla aparece la página de selección de componentes, partiendo de la primera opción que determina si se selecciona un prefijo para el esquema del repositorio o si se crea un nuevo prefijo. Por defecto la opción seleccionada es **Create new prefix** y el esquema prefijado es **DEMO**. A continuación debemos seleccionar los siguientes componentes: **Oracle Platform Security Services, Audit Services, Audit Services Append** y **Audit Services Viewer**. Después se pulsará el botón **Next.**

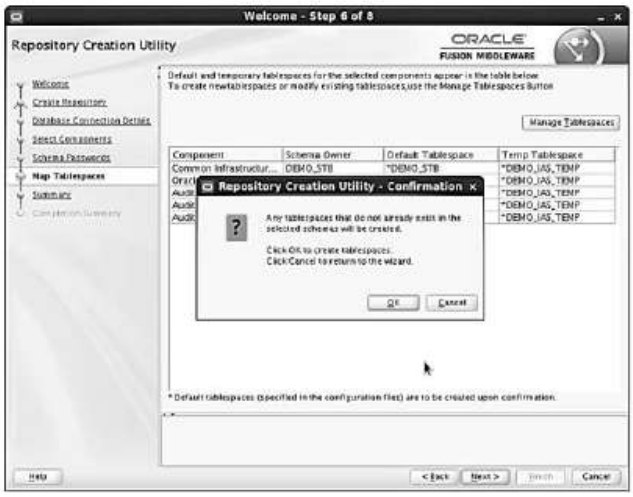




En esta pantalla aparecen una serie de ventanas emergentes donde se muestra el progreso de chequeo de los prerrequisitos. Después se pulsará el botón **OK.**

En esta pantalla se introduce la password para el esquema seleccionado (esquema DEMO). Mantendremos la opción **Use same passwords for all schemas**. Después se pulsará el botón **Next.**

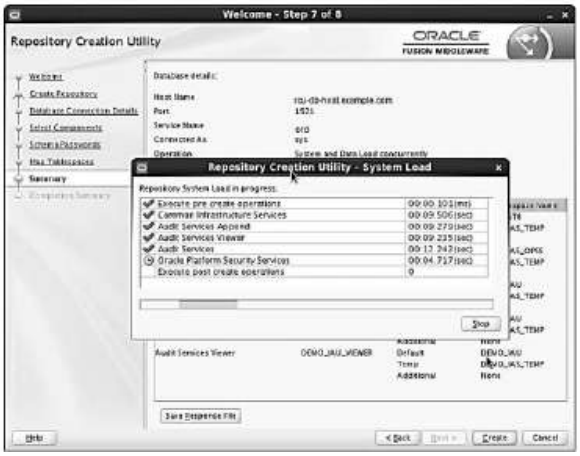
En esta pantalla se muestra el mapeo de los Tablespaces. Pulsaremos el botón **Next.**



En esta pantalla se muestra una ventana emergente de confirmación. Pulsaremos el botón **OK.**

En esta pantalla se muestra una ventana emergente con el progreso de creación de los nuevos tablespaces. Pulsaremos el botón **OK.**

En esta pantalla se muestra el resumen de las actuaciones y elecciones realizadas durante el proceso de instalación. Pulsaremos el botón **Create.**



En esta pantalla se muestra una ventana emergente con el progreso de creación del repositorio. Dejaremos que el proceso finalice sin errores y se cierre la pantalla.

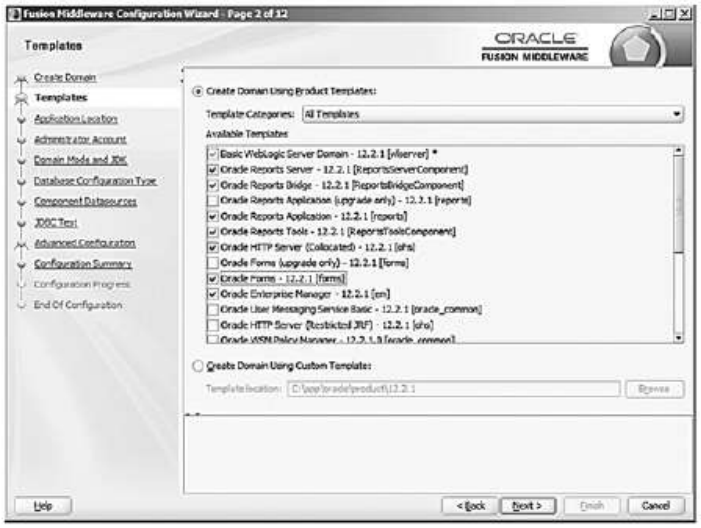
En esta pantalla pulsaremos **Close** para finalizar la instalación del RCU.

Creando el dominio

Finalizada la configuración e instalación del RCU indicada en el apartado anterior, tenemos que comenzar la configuración del dominio, para ello utilizaremos el asistente de configuración como se indica en los siguientes pasos:

```
C:\app\oracle\product\12.2.1\oracle_common\common\bin\config.cmd
```

Después de arrancar el asistente de configuración para configurar Oracle Forms, nos aparecerá esta pantalla para la creación o actualización de un dominio existente. Si es la primera instalación de Oracle Forms, debemos mantener la opción por defecto **Crear un nuevo dominio (Create a new domain)** y a continuación especificar la ruta deseada para la ubicación de dicho dominio (la instalación ya ofrece una ruta por defecto que podemos utilizar).

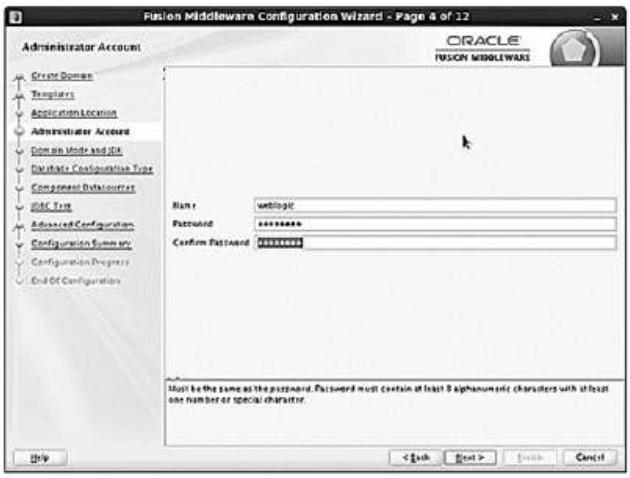


En esta pantalla se muestran las plantillas de los componentes que deseamos instalar. Para la configuración de Oracle Forms debemos de seleccionar los productos (solo los marcados con el símbolo ✓), para la instalación:

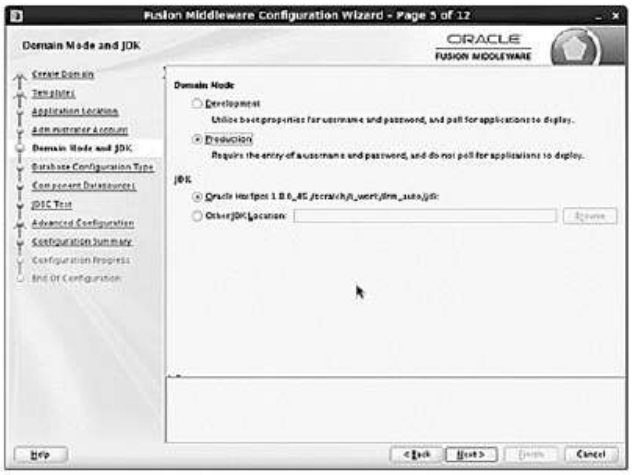
- *Basic Weblogic Server Domain*
- *Oracle Reports Server - 12.2.1 [ReportsServerComponent]*
- *Oracle Reports Bridge - 12.2.1 [ReportsServerComponente]*
- *Oracle Reports Application – 12.2.1 [reports]*
- *Oracle Reports Tools – 12.2.1 [ReportsToolsComponent]*
- *Oracle HTTP Server (Collocated) – 12.2.1 [ohs]*
- *Oracle Forms – 12.2.1 [forms]*
- *Oracle Enterprise Manager – 12.2.1 [em]*

A continuación pulsamos el botón **Next.**

En esta pantalla se muestra la ruta de instalación de la aplicación. Debemos mantener el valor que se ofrece por defecto y pulsar el botón **Next.**

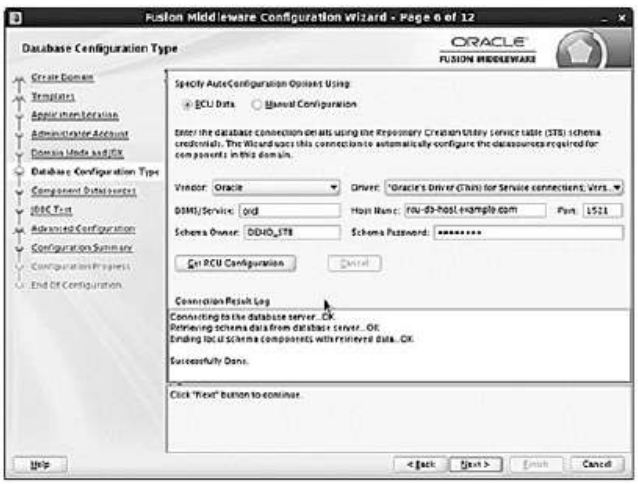


En esta pantalla se configura el usuario y la password que se utilizará en la administración del nuevo dominio que se creará. Por defecto el usuario que se ofrece es **weblogic**, debiendo introducir el instalador la password que desee para el mismo. Una vez realizada esta operación, se pulsará el botón **Next.**



En esta pantalla se configura en primer lugar el modo de funcionamiento del nuevo dominio, pudiéndose seleccionar dos posibles alternativas: *Development* o *Production*. Para los propósitos del libro seleccionaremos la opción **Production**.

Justo debajo se muestran las opciones de uso de la plataforma JDK necesaria para el funcionamiento de las aplicaciones. En nuestro caso mantendremos la opción que se ofrezca por defecto, salvo que deseemos utilizar otra versión de JDK previamente instalada. Seguidamente pulsaremos el botón **Next.**

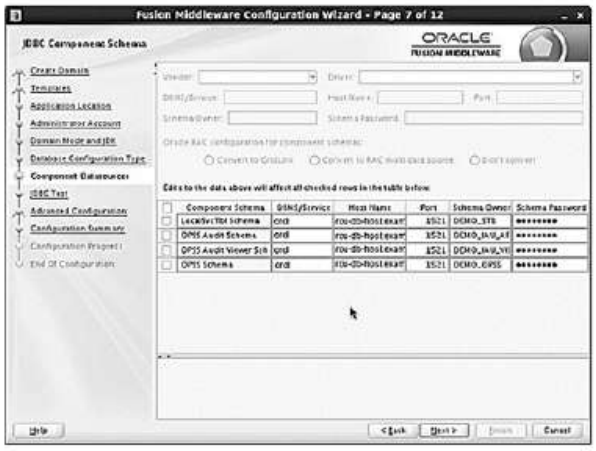


En esta pantalla se determina el tipo de configuración de la base de datos. En nuestro caso introduciremos la opción **RCU Data** con los siguientes valores:

- Vendor: **Oracle**
- Driver: **Oracle's Driver**
- DBMS/Service: *indicaremos el nombre del servicio de nuestra base de datos previamente instalada.*

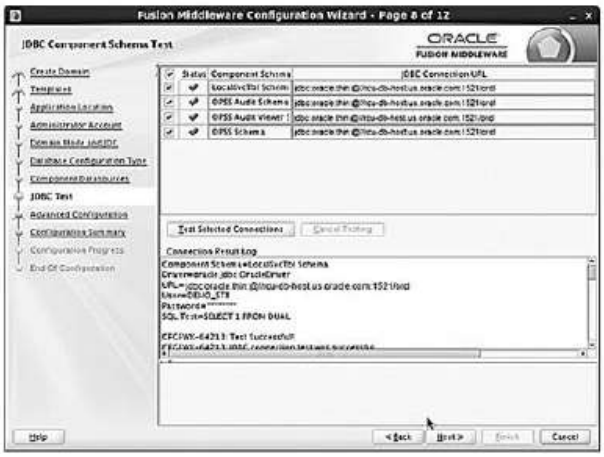
- Host Name: *indicaremos el nombre de la máquina (o IP) donde se encuentra instalada la base de datos.*
- Port*: indicaremos el nombre del puerto de conexión a la base de datos (por defecto suele ser el 1521).*
- Schema Owner: **DEMO_STB** *si mantuvimos el esquema por defecto de la instalación del RCU del apartado anterior. En caso de haber utilizado otro nombre de esquema en dicha instalación añadiremos al mismo el sufijo* **_STB.**
- Schema Password: *la password asociado al esquema indicado (el mismo que se indicó en la pantalla 5 de 8 de la instalación del RCU).*

A continuación pulsaremos el botón **Get RCU Configuration**, lo que debería generar una serie de mensajes en la ventana *Connection Result Log* que aparece justo debajo del botón. Si dichos mensajes devuelven el resultado OK, pulsaremos el botón **Next**.



En esta pantalla se muestran los componentes JDBC que van a ser instalados en la base de datos. Por defecto se asume que la password es equivalente en todos los esquemas mostrados. Si no fuese así habría que introducir cada una de las password de los esquemas en su casilla correspondiente (*Schema Password*).

Seguidamente pulsaremos el botón **Next** para continuar el proceso de instalación.

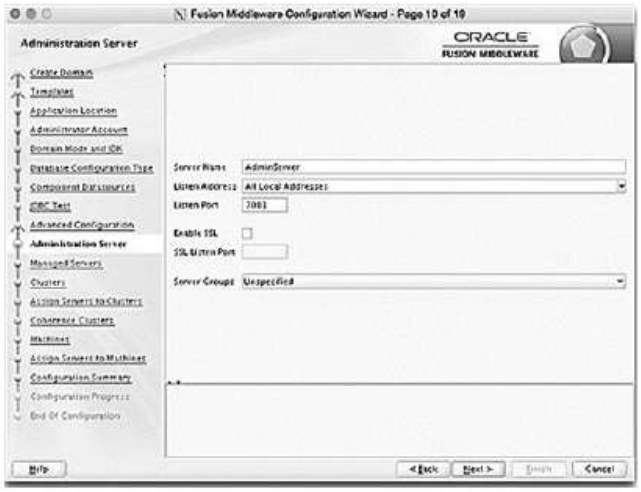


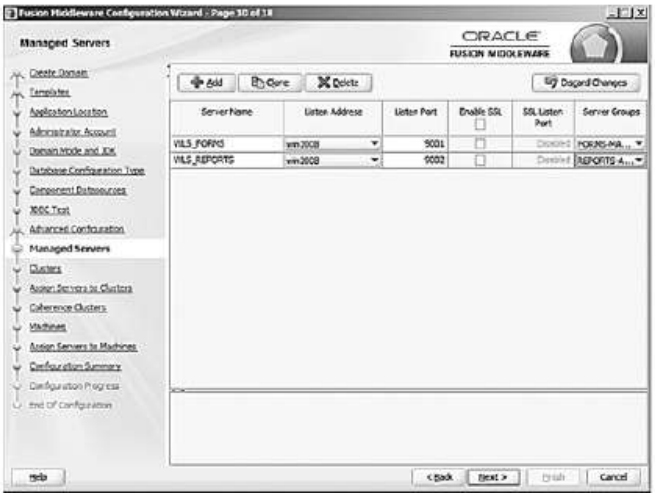
En esta pantalla se muestra el resultado de la instalación de los componentes JDBC. Si todos los componentes muestran un símbolo ✓ en la columna *Status,* entonces podremos continuar con la instalación pulsando el botón **Next.**

En esta pantalla se muestra la configuración avanzada que permite la instalación de la herramienta. Para nuestros propósitos, deberemos marcar los siguientes elementos: **System Components** y **Managed Servers, Clusters and Coherence.** A continuación pulsaremos el botón **Next.**

En esta pantalla se muestra la configuración de la herramienta Administrador del Servidor.

Los valores que se muestran por defecto en esta pantalla son los apropiados para la instalación, así que pulsaremos el botón **Next.**

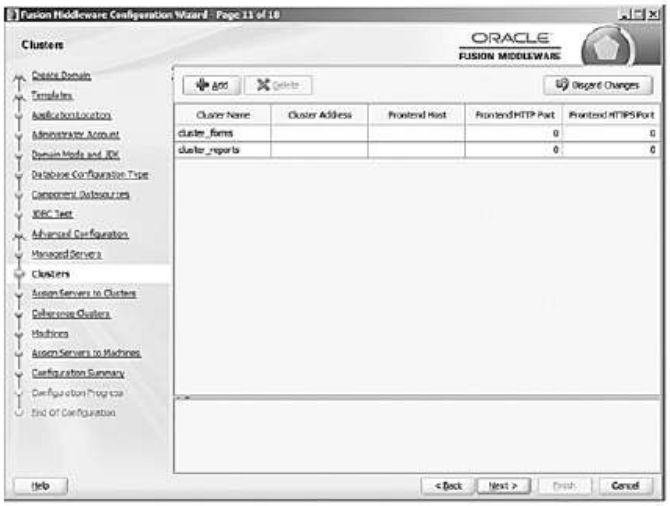




En esta pantalla se muestra el nombre del servidor de Forms y Reports que va a ser instalado.

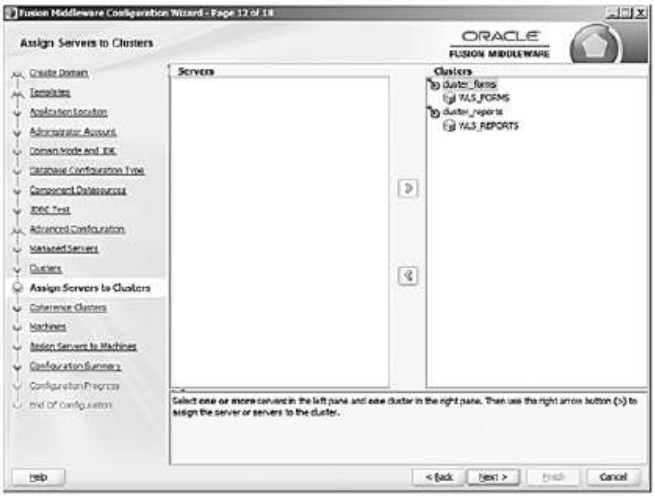
Debemos mantener los datos que se ofrecen por defecto y comprobar que el servidor de Forms pertenece al grupo de Servidores: **FORMS-MAN-SVR.**

Los nombres por defecto de los servidores serán **WLS_FORMS** (para el servidor de Forms) y **WLS_REPORTS** (para el servidor de Reports). Seguidamente pulsaremos el botón **Next.**



En esta pantalla aceptaremos los valores que se muestran por defecto para el cluster y pulsaremos el botón **Next.**

En esta pantalla aceptaremos los valores que se muestran por defecto para la asignación de servidores Forms y Reports al cluster y pulsaremos el botón **Next.**

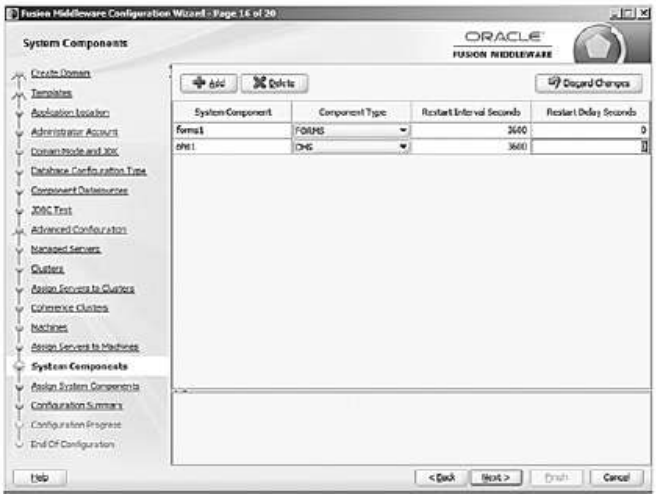


En esta pantalla aceptaremos los valores que se muestran por defecto para la configuración de Coherence y pulsaremos el botón **Next.**

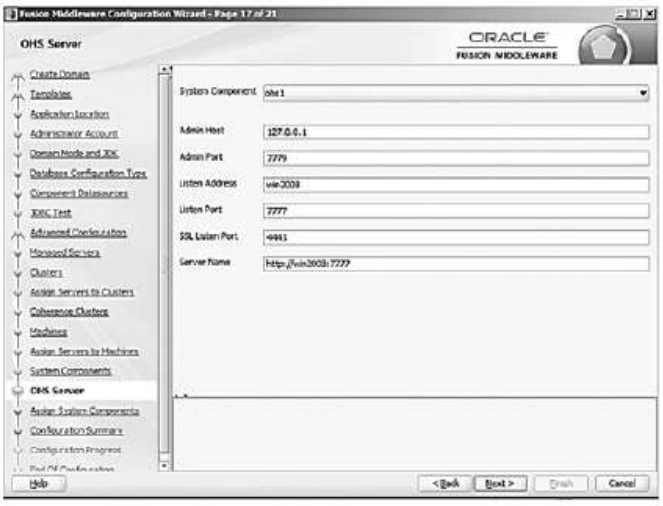
En esta pantalla se configurará la máquina o máquinas que gestionarán el dominio de Forms. Por defecto la máquina será aquella donde se está instalando el software: **localhost**, pero se podría modificar por otra e incluso añadir varias.

Finalizada esta tarea pulsaremos el botón **Next** para continuar la instalación.

En esta pantalla debemos mover desde la parte izquierda de la pantalla (Servers) a la parte derecha (Machines) el servidor de administración cuya máquina hemos configurado en la pantalla anterior para obtener una configuración similar a la mostrada en esta. Finalizada la tarea pulsaremos el botón **Next** para continuar la instalación.

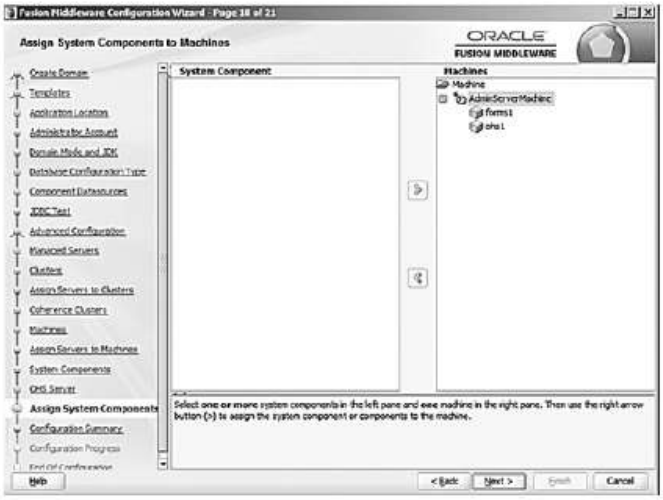


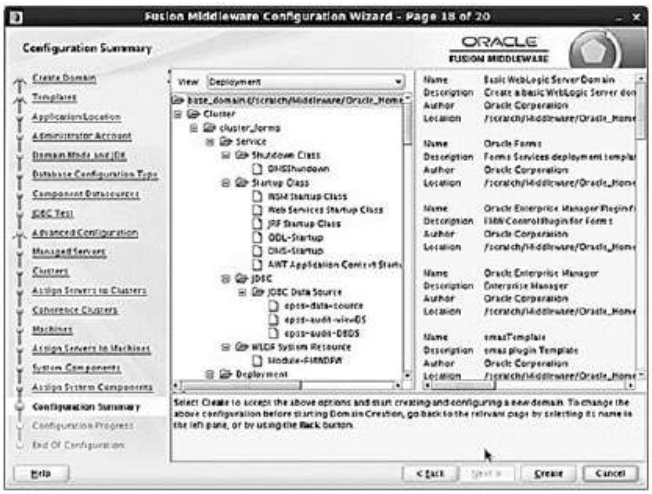
En esta pantalla aceptaremos los valores que se muestran por defecto para la configuración de los componentes del sistema y pulsaremos el botón **Next.**



En esta pantalla aceptaremos los valores que se muestran por defecto para la configuración del servidor OHS y pulsaremos el botón **Next.**

En esta pantalla aceptaremos los valores que se muestran por defecto para la asignación de componentes al sistema y pulsaremos el botón **Next.**





En esta pantalla se muestra el resumen del proceso de instalación con las opciones seleccionadas. Para llevarlo a la práctica, pulsaremos el botón **Create.**

En esta pantalla se muestra el progreso de instalación.

Cuando finalice el proceso de instalación se mostrará esta pantalla con la indicación del éxito o errores que se han producido durante la instalación. Si todo el proceso ha sido satisfactorio, pulsaremos el botón **Finish.**

Para completar la instalación es necesario que se arranque el administrador del servidor pulsando para ello sobre la URL que se mostrará en la pantalla bajo **Admin Server URL.**

Configuración adicional

Cuando se selecciona la opción **Production Mode** en la creación del dominio, será necesario crear un fichero de tipo "boot.properties", ajustando las credenciales de usuario (username) y password de conexión al dominio:

```
C:\>mkdir C:\app\oracle\config\domains\frs\servers\AdminServer\security
C:\>echo username=weblogic>
C:\app\oracle\config\domains\frs\servers\AdminServer\security\boot.properties
C:\>echo password=weblogic123>>
C:\app\oracle\config\domains\frs\servers\AdminServer\security\boot.properties

C:\>mkdir C:\app\oracle\config\domains\frs\servers\WLS_FORMS\security
C:\>echo username=weblogic>
C:\app\oracle\config\domains\frs\servers\WLS_FORMS\security\boot.properties
```

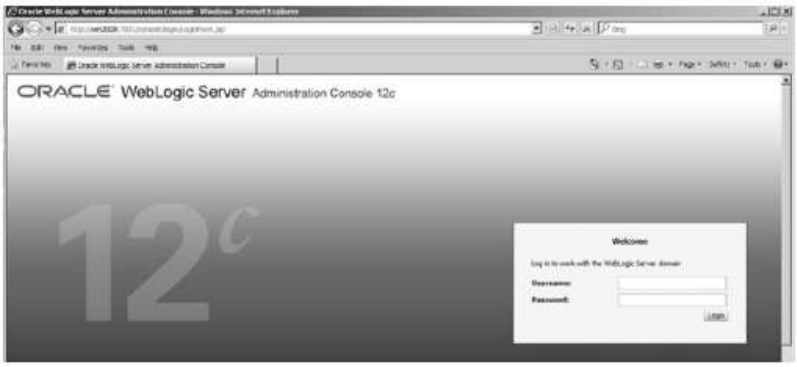
```
C:\>echo password=weblogic123>>
C:\app\oracle\config\domains\frs\servers\WLS_FORMS\security\boot.properties

C:\>mkdir C:\app\oracle\config\domains\frs\servers\WLS_REPORTS\security
C:\>echo username=weblogic>
C:\app\oracle\config\domains\frs\servers\WLS_REPORTS\security\boot.properties
C:\>echo password=weblogic123>>
C:\app\oracle\config\domains\frs\servers\WLS_REPORTS\security\boot.properties
```

PRIMER ACCESO A LA CONSOLA DEL ADMINISTRATION SERVER

En el directorio **c:\app\oracle\config\domains\frs** se encuentra un script que se puede utilizar para arrancar el servidor:

```
C:\app\oracle\config\domains\frs\startWebLogic.cmd
```

Una vez que el servidor está arrancado, se puede acceder a la consola de administración usando la URL: **http://hostname:port/console**. Esta dirección aparece en la última pantalla de la instalación de la sección anterior.



En esta pantalla introduciremos el usuario y password de instalación de Weblogic.

ARRANQUE Y PARADA DEL SERVIDOR WEBLOGIC

Para el arranque y la parada del servidor Weblogic se pueden utilizar los ficheros de comandos que se indican en esta sección.

Arranque de Weblogic

Para el arranque de Weblogic hay que arrancar el nodo, el dominio de weblogic y los servidores de gestión utilizando los siguientes ficheros de comandos:

```
# Start NodeManager
C:\app\oracle\config\domains\frs\bin\startNodeManager.cmd

# Start WebLogic Domain
C:\app\oracle\config\domains\frs\bin\startWebLogic.cmd
# Start the managed Servers
C:\app\oracle\config\domains\frs\bin\startManagedWebLogic.cmd
WLS_FORMS
C:\app\oracle\config\domains\frs\bin\startManagedWebLogic.cmd
WLS_REPORTS
```

Parada del Dominio de Weblogic

Para la parada de Weblogic hay que parar los servidores de gestión y después el dominio de Weblogic utilizando los siguientes ficheros de comandos:

```
# Stop the managed Servers
C:\app\oracle\config\domains\frs\bin\stopManagedWebLogic.cmd WLS_FORMS
C:\app\oracle\config\domains\frs\bin\stopManagedWebLogic.cmd
WLS_REPORTS

# Stop WebLogic Domain
C:\app\oracle\config\domains\frs\bin\stopWebLogic.cmd
```

CONFIGURACIÓN HTTP

Antes del arranque del servidor HTTP hay que comprobar que los ficheros de configuración de Forms y Reports contienen los datos del nombre del equipo (host) y puerto donde se manejarán ambos servicios.

En el caso de Forms debemos revisar que el fichero **c:\app\oracle\product\12.2.1\forms\templates\config\forms.conf** contiene la siguiente estructura:

```
<Location /forms>
       SetHandler weblogic-handler
       WebLogicHost win2008
       WebLogicPort 9001
</Location>
```

En el caso de Reports debemos revisar que el fichero **c:\app\oracle\product\12.2.1\forms\templates\config\reports_ohs.conf** contiene la siguiente estructura:

```
<Location /reports>
      SetHandler weblogic-handler
      WebLogicHost win2008
      WebLogicPort 9001
</Location>
```

Además, habrá que copiar los ficheros de configuración de la instancia "ohs1" sobre el directorio "moduleconf":

```
copy C:\app\oracle\product\12.2.1\forms\templates\config\forms.conf
C:\app\oracle\config\domains\frs\config\fmwconfig\components\OHS\instances\ohs1
\moduleconf

copy C:\app\oracle\product\12.2.1\reports\conf\reports_ohs.conf
C:\app\oracle\config\domains\frs\config\fmwconfig\components\OHS\instances\ohs1
\moduleconf
```

Para arrancar el servidor HTTP se debe usar la consola del Enterprise Manager como se indica a continuación:

- Logearse en la consola del Enterprise Manager.
- Pulsar en el icono "Target Navitacion".
- Expandir el nodo "HTTP Server"
- Pulsar en la instancia "ohs1".
- La página "ohs1" tiene un enlace para arranque "Start Up" y parada "Shut Down".

Si hay problemas en el arranque de la instancia, hay que intentar primero arrancar de nuevo el gestor del nodo y si el resultado sigue siendo infructuoso, entonces habrá que instalar los siguientes parches de Windows:

```
http://www.microsoft.com/en-us/download/confirmation.aspx?id=30679
http://www.microsoft.com/en-us/download/details.aspx?id=16771
http://www.microsoft.com/en-us/download/details.aspx?id=2092
```

APERTURA DE LA HERRAMIENTA FORMS BUILDER

Para abrir la herramienta de desarrollo de Forms: Forms Builder, se puede utilizar el siguiente fichero:

```
C:\app\oracle\product\12.2.1\bin\frmbld.exe
```

APERTURA DE LA HERRAMIENTA REPORTS BUILDER

Para abrir la herramienta de desarrollo de informes Reports: Reports Builder hay que seguir estos pasos una vez después de la instalación.

1. Ejecutar el fichero `wlst.cmd` desde el directorio `C:\app\oracle\product\12.2.1\oracle_common\common\bin.`

2. Conectarse al servidor de administración introduciendo las credenciales del usuario, password y nombre del equipo donde reside Weblogic:

```
connect("weblogic","weblogic_password","hostname:7001")
```

3. Ejecutar el siguiente comando:

```
createReportsToolsInstance(instanceName='reptools1',machine='Adm
inServerMachine')
```

4. Rearrancar WLS_REPORTS

Después de ejecutar estos pasos, cuando se quiera arrancar Reports Builder solo será necesario ejecutar el siguiente fichero de comandos sin necesidad de volver a realizar los pasos anteriores:

```
C:\app\oracle\config\domains\frs\reports\bin\rwbuilder.bat
```

CREAR UN REPORTS SERVER INDEPENDIENTE

Para crear un servidor Reports independiente (standalone) hay que realizar los siguientes pasos:

1. Ejecutar el fichero `wlst.cmd` desde el directorio `C:\app\oracle\product\12.2.1\oracle_common\common\bin.`

2. Conectarse al servidor de administración introduciendo las credenciales del usuario, password y nombre del equipo donde reside Weblogic:

```
connect("weblogic","weblogic_password","hostname:7001")
```

3. Ejecutar el siguiente comando:

```
createReportsServerInstance(instanceName='my_server1',machine='A
dminServerMachine')
```

4. En este punto, el servidor Reports independiente ya se puede utilizar.

ARRANQUE Y PARADA DE UN SERVIDOR DE REPORTS

Para el arranque y la parada de un servidor de Reports se pueden utilizar los ficheros de comandos que se indican en esta sección.

Arranque de un servidor independiente (standalone)

Para el arranque de un servidor Reports independiente (standalone) se siguen estos pasos:

```
# Nos situamos en el directorio raíz del dominio, sustituyendo
# "Domain home" por la ruta del dominio, por ejemplo
# c:\app\oracle\config|domains\frs\bin
cd "Domain home"\bin

# Arrancamos el servidor de Reports con el fichero de comandos
# sustituyendo "Reports Server Name" por nuestro nombre del
# servidor reports
startComponent.cmd "Reports Server Name"
```

Parada de un servidor independiente (standalone)

Para la parada de un servidor Reports independiente (standalone) se realizan los siguientes pasos:

```
# Nos situamos en el directorio raíz del dominio, sustituyendo
# "Domain home" por la ruta del dominio, por ejemplo
# c:\app\oracle\config|domains\frs\bin
cd "Domain home"\bin

# Paramos el servidor de Reports con el fichero de comandos
# sustituyendo "Reports Server Name" por nuestro nombre del
# servidor reports
stopComponent.cmd "Reports Server Name"
```

Arranque de un servidor no independiente

Para el arranque de un servidor Reports no independiente, se realizan los siguientes pasos:

```
# Ejecutar la URL siguiente sustituyendo "hostname" por el
# nombre del equipo donde se encuentre el servidor Reports.
http://hostname:9002/reports/rwservlet/startserver
```

Parada de un servidor no independiente

Para la parada de un servidor Reports no independiente, se siguen estos pasos:

```
# Ejecutar la URL siguiente sustituyendo "hostname" por el
# nombre del equipo donde se encuentre el servidor Reports.
http://hostname:9002/reports/rwservlet/stopserver
```

URLs DE INTERÉS

En este apartado se muestra una serie de URLs de interés para la gestión de los distintos servicios que se han instalado. En todas ellas se sustituirá "localhost" por el nombre de la máquina correspondiente a la instalación o la IP de la misma.

- Consola:

  ```
  http://localhost:7001/console
  ```

- Enterprise Manager:

  ```
  http://localhost:7001/em
  ```

- Forms:

  ```
  http://localhost:9001/forms/frmservlet
  http://localhost:7777/forms/frmservlet
  ```

- Reports:

  ```
  http://localhost:9002/reports/rwservlet
  http://localhost:7777/reports
  ```

WEB START

Oracle Forms se puede arrancar usando Java Web Start dependiendo de la compatibilidad con el navegador que se esté utilizando. En cualquier caso será necesario disponer en el equipo donde se lance de una instalación Java 8 JRE o superior.

Para arrancar Web Start se utilizará alguna de las siguientes URLs, sustituyendo "hostname" por el nombre de la máquina correspondiente:

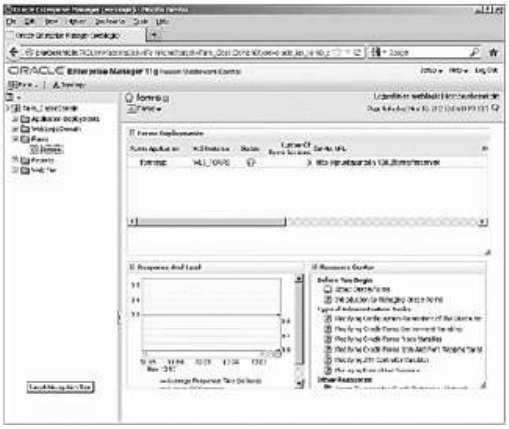
```
http://hostname:9001/forms/frmservlet?config=webstart
http://hostname:7777/forms/frmservlet?config=webstart
```

Dependiendo del navegador que se utilice, la URL puede provocar que se arranque directamente Web Start o que nos solicite almacenar un fichero ".jnlp" en el equipo. Una vez que se haya almacenado dicho fichero, habrá que realizar un doble clic sobre el mismo, y se arrancará Forms directamente en el PC como una aplicación Web Start sin utilizar el navegador.

GESTIÓN DE LOS FICHEROS DE CONFIGURACIÓN DEL ENTORNO

Para la gestión de los distintos ficheros de configuración del entorno de instalación, se utilizará la herramienta Enterprise Manager. Dentro de la misma, pulsaremos sobre la instancia de Oracle Forms Services que vayamos a configurar.

Para configurar cada uno de los ficheros hay que seleccionar la carpeta **Forms.** Una vez desplegado su contenido seleccionaremos el elemento **forms** contenido en ella.



Una vez realizado esto se nos mostrará el menú correspondiente de la pantalla **forms**, como se muestra en esta imagen.

Dentro de la opción **forms** se puede desplegar una serie de opciones como se muestra en esta imagen.

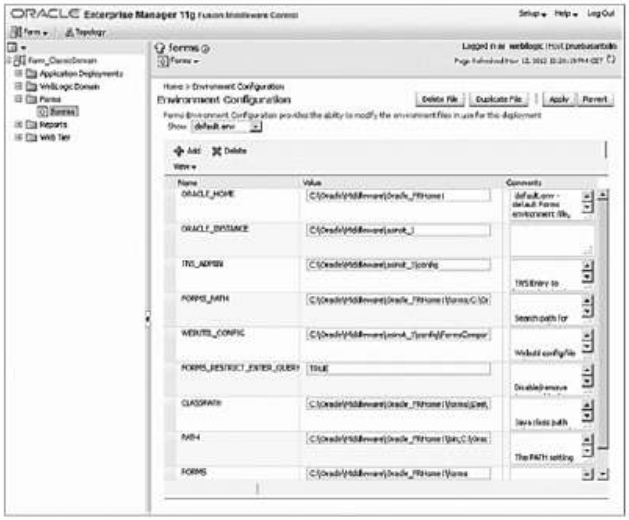
El significado de cada uno de estos elementos se explica a continuación:

Opción	Página que se visualiza
Home	Muestra la página raíz de Forms. Esta página muestra una lista de los desarrollos que se han realizado en Forms y sus detalles. Esta página también muestra las estadísticas de respuesta y carga, y un conjunto de enlaces útiles del Centro de Recursos.
Monitoring – Perfomance Summary	Muestra la página resumen de rendimiento. Esta página muestra un conjunto de gráficas de rendimiento que muestran los valores de las métricas específicas de rendimiento.
Monitoring – Servlet Log	Muestra la página de mensajes Log. Los componentes de la herramienta Oracle Fusion Middleware generan ficheros log que contienen mensajes que se graban al producirse cualquier tipo de evento.
JVM Controllers	Muestra la página de control de la Máquina Virtual de Java. Esta página se utiliza para gestionar el controlador JVM para la instancia de Forms.
Schedule Prestart	Gestiona la página de prearranque de la cola de Forms. Permite crear, borrar e importar y exportar una cola de Forms.
User Sessions	Muestra la página de las sesiones de los usuarios conectados. Esta página se utiliza para monitorizar y tracear las sesiones de los usuarios dentro de la instancia de Forms.
Web Configuration	Muestra la página de configuración Web. Esta página se utiliza para configurar los desarrollos de aplicaciones en Forms y la gestión de las secciones de configuración y parámetros en el fichero `formsweb.cfg`.
Trace Configuration	Muestra la página de configuración de la traza. Esta página se utiliza para gestionar la configuración utilizada para tracear las sesiones de los usuarios.

Fonts and Icons	Muestra la página del mapeo de las fuentes e iconos.
Mapping	Esta página se utiliza para cambiar, añadir o borrar parámetros en el fichero `Registry.dat`.
JVM Configuration	Muestra la página de configuración de JVM. Esta página se utiliza para modificar el controlador JVM.
Environment Configuration	Muestra la página de configuración de entorno. Esta página se utiliza para gestionar las variables del entorno "run time" de Forms. Configuran el fichero `default.env`.
Associate / Dissociate OID	Muestra la página para asociar o desasociar Oracle Internet Directory. Esta página se utiliza para asociar y desasociar un desarrollo con un equipo con Oracle Internet Directory para habilitar la funcionalidad del Single Sign-On de Oracle.
General Information	Muestra diversa información como la versión, instancia de Oracle, directorio raíz de Oracle, equipo, etc.

VARIABLES DE ENTORNO POR DEFECTO (DEFAULT.ENV)

Para la gestión de las variables de entorno, dentro de la herramienta Enterprise Manager seleccionaremos la carpeta **Forms.** Una vez desplegado su contenido seleccionaremos el elemento **forms** contenido en ella y elegimos la opción **Environment Configuration.**



Esta es la pantalla tipo que nos encontraríamos al seleccionar la opción `default.env` en la casilla **Show.**

A continuación se da una breve explicación de cada una de las variables de entorno que aparecen en esta pantalla.

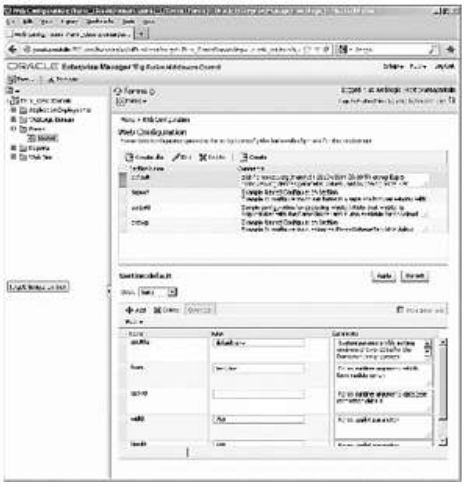
Variable	Descripción
ORACLE_HOME	Apunta al directorio base de la instalación de cualquier producto de Oracle.
ORACLE_INSTANCE	Contiene todos los ficheros de configuración, repositorios y logs de la instancia de Oracle.

TNS_ADMIN	Especifica la ruta de los ficheros TNS tales como TNSNAMES.ORA y SQLNET.ORA
FORMS_PATH	Especifica las rutas de las búsquedas de Oracle Forms cuando se ejecutan desde el menú de un formulario o librería.
WEBUTIL_CONFIG	Ruta de ubicación del fichero de configuración de la utilidad WEBUTIL de Oracle Forms.
FORMS_RESTRICT_ENTER_QUERY	Habilitar o borrar esta variable para usuarios finales que necesiten acceder a la funcionalidad de las aplicaciones únicamente en modo consulta.
CLASSPATH	Especifica la ruta de las clases de Java, que se necesitan para el proceso de debug de Forms.
PATH	Contiene los ejecutables de los productos de Oracle.
FORMS	Contiene los directorios para almacenar las trazas y los ficheros logs de Forms.

Para modificar cualquiera de los valores mostrados se accederá a la columna **Value** de la pantalla y se modificará la información contenida en ella. Para consolidar los cambios se pulsará sobre el botón **Apply.**

CONFIGURACIÓN WEB (FORMSWEB.CFG)

Para la configuración de los desarrollos de aplicaciones en Forms, dentro de la herramienta Enterprise Manager, seleccionaremos la carpeta **Forms.** Una vez desplegado su contenido seleccionaremos el elemento **forms** contenido en ella y elegimos la opción **Web Configuration.**



Al hacerlo se presenta una ventana como la mostrada. En esta pantalla, dentro de la sección **default** se muestra la cuadrícula **Show** con un desplegable donde se agrupan los parámetros de configuración. Estos se agrupan en las siguientes categorías:

- **Basic:** Parámetros de configuración básica.
- **Sso:** Parámetros de configuración del Single Sing-On.
- **Trace:** Parámetros de configuración de la traza.
- **Plugin:** Parámetros de configuración de los Plug-in.
- **Html:** Parámetros de configuración de la página HTML.
- **Applet:** Parámetros de configuración del Applet.
- **Advanced:** Parámetros de configuración avanzada.

Parámetros de configuración básica (basic)

Los parámetros de configuración básica controlan el funcionamiento del servlet de Forms. A continuación se describe la lista de estos parámetros:

Parámetro	Requerido/ Opcional	Valor del parámetro y descripción
envFile	Requerido	Especifica el nombre del fichero de configuración del entorno.El valor por defecto es default.env
form	Requerido	Especifica el nombre del módulo de formulario de mayor nivel que hay que arrancar.El valor por defecto es test.fmx.
userid	Opcional	Cadena de texto correspondiente a la conexión que se realiza con la base de datos.Por ejemplo: scott/tiger@ORADB.
width	Requerido	Especifica el ancho del applet del formulario medido en píxeles. El valor por defecto es 750.
height	Requerido	Especifica la altura del applet del formulario medido en píxeles. El valor por defecto es 600.

Parámetros de configuración del Single Sign-On (sso)

A continuación se describe la lista de los parámetros necesarios para la configuración del Single Sign-On de Oracle.

Parámetro	Requerido/ Opcional	Valor del parámetro y descripción
ssoDynamicResourceCreate	Opcional	Especifica si la creación de recursos dinámicos está habilitada si el recurso aún no ha sido creado en el OID. El valor por defecto es true.
ssoErrorUrl	Opcional	Especifica la URL de redireccionamiento si ssoDynamicResourceCreate tiene asignado el valor false.
ssoCancelUrl	Opcional	Especifica la URL de cancelación para la creación de recursos dinámicos de páginas DAS.
ssoMode	Opcional	Especifica si la URL está protegida. Se debe asignar valor true en la sección de especificación de una aplicación para habilitar Single Sign-On en la misma. El valor por defecto es false.
ssoProxyConnect	Opcional	Especifica si la sesión debería operar en un proxy de usuario soportado o no. Se debe asignar el valor yes para habilitarlo en una aplicación particular. El valor por defecto es no. Este parámetro es un subargumento de otros parámetros.

Parámetros de configuración de la traza (trace)

A continuación se describe la lista de los parámetros necesarios para la configuración de la traza.

Parámetro	Requerido/ Opcional	Valor del parámetro y descripción
debug	Opcional	Permite el arranque del modo debug. El valor por defecto es No.
host	Opcional	Especifica el equipo para realizar el debug de la sesión. Este parámetro solo se debe utilizar para propósitos de debug.
port	Opcional	El puerto que se usa para realizar el debug. Este valor identifica el puerto en el que los procesos de ingeniería del formulario están escuchando.
record	Opcional	Soporte de la traza y log. Este parámetro es un subargumento de otros.
tracegroup	Opcional	Soporte de la traza y log. Este parámetro es un subargumento de otros.
log	Opcional	Soporta el traceado y almacenamiento de logs. El valor de este parámetro si se indica es el nombre del fichero de traza donde se almacena el log. Este parámetro es un subargumento de otros.
allow_debug	Requerido	Este parámetro determina si se permite o no el proceso de debug. El valor por defecto es false.
EndUserMonitoringEnable	Opcional	Indica si la integración con la monitorización de usuario final es habilitada. El valor por defecto es false.
EndUserMonitoringURL	Opcional	Indica si se almacenan datos de la monitorización del usuario final.

Parámetros de configuración de los Plug-in (plugin)

A continuación se describe la lista de los parámetros necesarios para la configuración del Plug-in de Sun.

Parámetro	Requerido/ Opcional	Valor del parámetro y descripción
codebase	Requerido	Directorio virtual que se define para apuntar al directorio físico desde donde se descarga el applet JAR por defecto. El valor por defecto es /forms/java.
imageBase	Opcional	Indica dónde se encuentran almacenados los iconos. Los valores admitidos son: • codeBase, que indica que la ruta de búsqueda de los iconos es relativa al directorio que contiene las clases Java. Usa este valor si se almacenan los iconos en un fichero con extensión JAR (es lo recomendado). • documentBase, que indica la URL a la que apunta el fichero HTML. El valor por defecto es codeBase. Si no se especifica valor, entonces se utiliza el valor de documentBase.
archive	Opcional	Lista de archivos delimitados por comas que se utilizan o descargan por el cliente. Para cada fichero hay que incluir el nombre del fichero si el mismo está en el directorio codebase, o bien se incluirá la ruta virtual y el nombre del fichero. El valor por defecto es frmall.jar.
jpi_download_page	Requerido	Página de descarga del Plug-in de Java. El valor por defecto es: http://java.sun.com/products/archive/j2se/6u12/index.html.
jpi_classid	Requerido	El identificador de la clase del Plug-in de Java. El valor por defecto es: clsid:CAFEEFAC-0016-0000-0012-ABCDEFFEDCBA.
jpi_codebase	Requerido	La configuración del codebase del Plug-in de Java. El valor por defecto es: http://java.sun.com/update/1.6.0/jinstall-6-windows-i586.cab#Version=1,6,0,12.
jpi_mimetype	Requerido	Parámetro relativo a la versión del Plug-in de Java. El valor por defecto es: application/x-java-applet;jpi-version=1.6.0_12.

Parámetros de configuración de la página HTML (html)

A continuación se describe la lista de los parámetros necesarios para la configuración del entorno html.

Parámetro	Requerido / Opcional	Valor del parámetro y descripción
baseHTML	Requerido	El fichero base por defecto para HTML. El valor por defecto es base.htm.
baseHTMLjpi	Requerido	Ruta física del fichero HTML que contiene las etiquetas del Plug-in de Java. Se utiliza como el fichero baseHTML si el navegador del cliente no se encuentra bajo Windows y el navegador del cliente es tanto Firefox como Internet Explorer sin la configuración nativa de Internet Explorer. El valor por defecto es basejpi.htm.
pageTitle	Opcional	Título de la página HTML, atributos para la etiqueta <BODY> y <HTML> que se añaden antes y después del formulario. El valor por defecto es Oracle Fusion Middleware Forms Services.
HTMLbodyAttrs	Opcional	Atributos para la etiqueta <BODY> de la página HTML.
HTMLbeforeForm	Opcional	Contiene HTML que añade a la página, encima del área donde se visualiza la aplicación Form.
HTMLafterForm	Opcional	Contiene HTML que añade a la página, debajo del área donde se visualiza la aplicación Form.

Parámetros de configuración del Applet (applet)

Estos parámetros que se describen a continuación se especifican en el fichero baseHTML como valores para objetos o parámetros del applet. Describen, en su conjunto, aspectos visuales y apariencia del applet.

Parámetro	Requerido / Opcional	Valor del parámetro y descripción
width	Requerido	El ancho del marco del applet de Forms. El valor por defecto es 750.
height	Requerido	La altura del marco del applet de Forms. El valor por defecto es 600.
separateFrame	Opcional	Determina si el applet aparecerá dentro de una ventana separada de Windows. Los valores admitidos son: true o false. El valor por defecto es false.

splashScreen	Opcional	Especifica el fichero .GIF que debería aparecer antes de que se muestre el applet. Se asignará el valor NO para no mostrar esa imagen previa. Si se deja en blanco se utilizará la imagen por defecto. Para asignar un valor a este parámetro hay que incluir el nombre del fichero (por ejemplo myfile.gif) o la ruta virtual y el nombre del fichero (por ejemplo images/myfile.gif).
background	Opcional	Especifica el fichero .GIF que debería aparecer en el fondo de pantalla. Se asignará el valor NO cuando no se quiera tener un fondo de pantalla. Si se deja vacío se usa el fondo de pantalla por defecto.
lookAndFeel	Opcional	Determina el "look-and-feel" (conjunto de elementos que determinan el aspecto general o apariencia) de la aplicación. Los valores admitidos son: Oracle o Generic. El valor por defecto es Oracle.
colorScheme	Opcional	Determina el esquema de colores de la aplicación. Los valores admitidos son: Teal, Titanium, Red, Khaki, Blue, BLAF, SWAN, Olive o Purple. El valor por defecto es teal. **Nota:** colorScheme se ignora si se indica el valor Generic en el parámetro LookAndFeel.
logo	Opcional	Especifica el fichero .GIF que debería aparecer en la barra de menú del Forms. Se asignará el valor NO para no utilizar un logo. Si se deja vacío se utilizará el logo por defecto de Oracle.

Parámetros de configuración avanzada (advanced)

A continuación se muestra la relación de parámetros avanzados de configuración.

Parámetro	Requerido/ Opcional	Valor del parámetro y descripción
HTMLdelimiter	Opcional	Indica el símbolo que se utiliza para delimitar los parámetros de los ficheros base HTML. El valor por defecto es %.
escapeparams	Opcional	Si se saltan (omiten) ciertos caracteres especiales en los valores que se extraen de la URL de ejecución de parámetros. El valor por defecto es true.

digitSubstitution	Opcional	Nomenclatura a utilizar para los argumentos de arranque de Forms: BIDI digitSubstitution. El valor por defecto es context.
otherparams	Opcional	Cadena de otros parámetros para el arranque de Forms, que se agrupan como si fueran un único parámetro. El valor por defecto es: obr=%obr% record=%record% tracegroup=%tracegroup% log=%log% term=%term% ssoProxyConnect=%ssoProxyConnect%
obr	Opcional	Subparámetro de otherparams. El valor por defecto es no.
term	Opcional	Subparámetro de otherparams.
serverURL	Opcional	Direccionamiento virtual al Listener de Forms. El valor por defecto es /forms/lservlet.
allowAlertClipboard	Opcional	Parámetro del applet de Forms. El valor por defecto es true.
disableValidateClipboard	Opcional	Parámetro del applet de Forms. El valor por defecto es false.
highContrast	Opcional	Parámetro del applet de Forms. El valor por defecto es false.
guiMode	Opcional	Parámetro del applet de Forms. Los valores admitidos van del rango del 0 al 3. El valor por defecto es 0.
restrictedURLparams	Opcional	Parámetro del applet de Forms. El valor por defecto es: pageTitle,HTMLbodyAttrs,HTMLbeforeForm,HTMLafterForm,log.
formsMessageListener	Opcional	Parámetro del applet de Forms.
recordFileName	Opcional	Parámetro del applet de Forms.
serverApp	Opcional	Parámetro del applet de Forms. El valor por defecto es default.
Legacy_lifecycle	Opcional	Parámetro del applet para el plugin Java de Sun. El valor por defecto es false.
allowNewConnections	Opcional	Determina si se permite una nueva sesión de Forms. Este parámetro se utiliza también por la página raíz de Form en Fusion Middleware Control para mostrar el estado actual de Forms. El valor por defecto es true.
applet_name	Opcional	Configuración para la integración de JavaScript. Este es el nombre del applet de Forms que se puede utilizar para referirse a él desde el código de JavaScript.

enableJavascriptEvent	Opcional	El valor por defecto es `true`.
JavaScriptBlocksHeartBeat	Opcional	Configura variables que indicarán si el "heartbeat" se bloqueará cuando una llamada javascript sea de tipo bloqueo. El valor por defecto es `false`.

Modificaciones a realizar para el curso

Para llevar a cabo las prácticas y los supuestos propuestos en el curso es necesario realizar las siguientes modificaciones en el fichero **formsweb.cfg** bien directamente en el mismo, bien utilizando la opción **Web Configuration.**

`archive=frmall.jar,frmicons_curso.jar`.

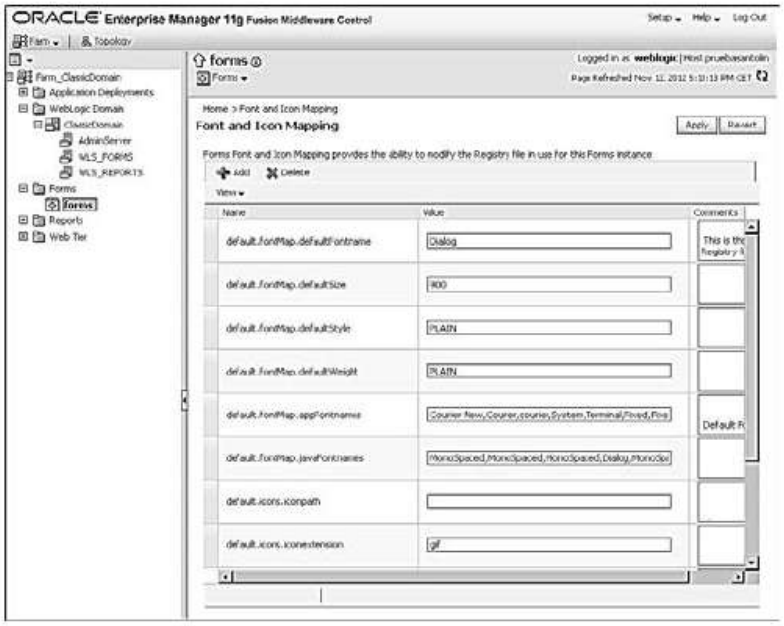
DESCARGA DE LA CARPETA ICONOS DESDE LA WEB DEL EDITOR E INSTALACIÓN:

Para poder visualizar correctamente los iconos que se van a utilizar durante el curso necesario que se descargue la carpeta ICONOS desde la web del editor. Una vez descargada se realizarán las siguientes operaciones:

- Sitúese en la carpeta de la instalación de Forms.
- Copie archivos descargados de la carpeta ICONOS en la carpeta anterior.
- Ejecute en Windows el programa REGEDIT.
- Sitúese en HKEY_LOCAL_MACHINE – SOFTWARE – ORACLE – KEY_OH111426769
- Edite el parámetro UI_ICON y asígnele la ruta de instalación de Forms.
- Añada un nuevo parámetro de tipo alfanumérico (Edición – Nuevo – Valor alfanumérico) con el nombre UI_ICON_EXTENSION y asóciele el siguiente valor: "gif": ".gif" "ico": ".ico"

FUENTES, ICONOS E IMÁGENES UTILIZADAS EN FORMS (REGISTRY.DAT)

Para la gestión de las fuentes, iconos e imágenes que se pueden utilizar en Forms hay que configurar el fichero `Registry.dat`. Dentro de la herramienta Enterprise Manager seleccionaremos la carpeta **Forms.** Una vez desplegado su contenido seleccionaremos el elemento **forms** contenido en ella y elegimos la opción **Font and Icon Mapping.** Al hacerlo se presenta una ventana como la mostrada en esta imagen.



Manejo de las fuentes de la aplicación

Usando Oracle Enterprise Manager se pueden cambiar las fuentes por defecto y la configuración de las fuentes en el fichero de `Registry.dat`. Todos los nombres de las fuentes corresponden con nombres de fuentes de Java. Cada uno de estos parámetros representa la propiedad por defecto que se usa cuando no se especifica ninguna otra. A continuación se muestra una lista con los valores por defecto de las fuentes.

Nombre de la fuente	Valor por defecto
`default.fontMap.defaultFontname`	`Dialog` Representa el nombre por defecto de la fuente Java.
`default.fontMap.defaultSize`	`900` Representa el tamaño por defecto de la fuente. El tamaño siempre se indicará como un múltiplo de 100 (por ejemplo una fuente de 10 puntos, se indicará en este parámetro como un tamaño 1000).
`default.fontMap.defaultStyle`	`PLAIN` Representa el estilo por defecto de la fuente. Los valores admitidos son `PLAIN` o **`BOLD.`**
`default.fontMap.defaultWeight`	`PLAIN` Representa el ancho por defecto de la fuente. Los valores admitidos son `PLAIN` o **`BOLD.`**

default.fontMap.appFontnames	Courier New,Courier,courier, System,Terminal,Fixedsys,Times, Times New Roman,MS Sans Serif, Arial Mapeo de las fuentes. Se indicará con una lista delimitada por comas. El número de entradas en la lista debería equivaler con la lista de nombres de fuentes Java (javaFontname). Los elementos de la lista están separados por comas y todos los caracteres se toman literalmente.
default.fontMap.javaFontnames	MonoSpaced, MonoSpaced, MonoSpaced,Dialog,MonoSpaced, Dialog,Dialog,Serif,Serif, Dialog,SansSerif Representa una lista delimitada por comas de los nombres de fuentes Java.

El mapeo de las fuentes entre los parámetros appFontnames y javaFontnames se refiere a cómo se convierte la fuente de Windows utilizada en el desarrollo de un formulario, cuando el mismo se presenta bajo el applet de Java. Por ejemplo siguiendo los valores por defecto de ambos parámetros, cuando se desarrolla un formulario con la fuente Courier, el mismo se presentará en el applet de Java con la fuente MonoSpaced.

Manejo de los iconos en una aplicación

Cuando se desarrollan aplicaciones en Oracle Forms, los ficheros de iconos que se utilizarán deben de estar en formato Web, tales como JPG o GIF (siendo este último el formato por defecto). Tal indicación se realiza en el parámetro default.icons.iconextension que por defecto tiene el valor gif.

Por defecto la ruta de búsqueda de los iconos se encuentra en el parámetro DocumentBase que muestra un directorio relativo. Para modificar la ruta de localización se debe asignar el parámetro imageBase a codebase en el fichero de control formsweb.cfg. Como alternativa a esto, también se puede cambiar el valor default.icons.iconpath en el fichero Registry.dat al siguiente directorio:$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config/forms/registry/oracle/forms/registry.

Si una aplicación utiliza muchas imágenes de iconos, es recomendable que se almacenen en un archivo Java y se asigne el valor `imageBase` al parámetro `codebase`. Los ficheros de iconos se pueden comprimir en un único archivo Java usando el comando Jar del kit de desarrollo de software Java (Java SDK).

Por ejemplo el comando `jar –cvf myico.jar *.gif` empaqueta todos los ficheros con extensión `.gif` en un archivo con el nombre `myico.jar`.

Con objeto de que Oracle Forms pueda acceder a los ficheros de iconos almacenados en un archivo Java, este tiene que estar almacenado en el directorio `forms/java`.

Modificaciones a realizar para el curso

Para llevar a cabo las prácticas y los supuestos propuestos en el curso es necesario realizar las siguientes modificaciones en el fichero **formsweb.cfg** bien directamente en el mismo, bien utilizando la opción **Web Configuration.**

```
default.icons.iconpath=/forms/java
```

SELECCIÓN DEL MODO DE EJECUCIÓN EN VENTANAS (INCRUSTRADO O NO)

Las aplicaciones que se desarrollan en Forms y se ejecutan en un navegador se pueden desplegar en modo INCRUSTRADO o modo NO incrustado.

En ambos casos, el explorador nos abre una ventana Windows donde se ejecuta la máquina virtual de Java (JVM) como se muestra en la imagen de la página siguiente.

Por tanto la diferencia entre los 2 modos de ejecución estriba en la apertura o no de una segunda ventana para la ejecución de la aplicación.

Ejecución en modo incrustado

En este modo el formulario se lanza en la misma ventana que posee la Máquina Virtual de Java (JVM). Además, se carga también en dicha página el applet de Java.

Para indicar este modo de ejecución hay que abrir el fichero de configuración `formsweb.cfg` y cambiar el parámetro `separateframe` al valor `false`. Si hemos seguido al detalle el proceso de instalación, este será el valor que tendrá el parámetro, lo que supone que el formulario se lanzará en modo incrustado.

Ejecución en modo no incrustado

En este modo el formulario se lanza en una segunda ventana, además de la que se abre para la Máquina Virtual de Java (JVM). Es en la segunda ventana donde se carga también el applet de Java JInitiator.

Para indicar este modo de ejecución hay que abrir el fichero de configuración `formsweb.cfg` y cambiar el parámetro `separateframe` al valor `true`.

Ventajas e inconvenientes de utilizar ambos modos

El modo incrustado presenta como ventaja que al ejecutarse todo en una única ventana, basta con cerrar la misma para que también salgamos de la aplicación. Pero presenta como inconveniente que se pierde espacio de visualización para la aplicación, dado que aparecen superpuestas las barras de navegación, menús y ventanas propias del explorador.

El modo no incrustado presenta como ventaja que al ejecutarse la aplicación en una ventana independiente, todo el marco de la misma se reserva para su visualización, de forma que desaparecen las barras de dirección, menús y navegación del navegador que se esté utilizando. Pero presenta como inconveniente que para cerrar la aplicación hay que cerrar las 2 ventanas abiertas o bien cerrar la ventana de la Máquina Virtual de Java (JVM) lo cual provocará también que se cierre la ventana donde se está ejecutando la aplicación.

INTRODUCCIÓN A ORACLE FORMS 4

ORACLE FORMS

Oracle Forms es un componente de Oracle Fusion Middleware que se utiliza para el desarrollo y despliegue de aplicaciones basadas en formularios. Estas aplicaciones proporcionan una interfaz de usuario que permite el acceso a bases de datos Oracle de un modo eficiente. Estas aplicaciones se pueden integrar con Java y servicios Web para obtener todas las ventajas que nos proporcionan las arquitecturas orientadas a servicios (SOA).

Oracle Forms incluye las siguientes herramientas.

- Oracle Forms Developer.
- Oracle Forms Services.

Oracle Forms Developer

Esta herramienta se utiliza para el desarrollo de los formularios que pueden acceder a bases de datos Oracle y presentar información de las mismas. Consta de una serie de utilidades como Oracle Forms Builder que permite el desarrollo mejorado de las aplicaciones.

El código fuente de los formularios se almacena en ficheros ***.fmb** y una vez compilados se convierten en ficheros ejecutables con extensión ***.fmx.**

Oracle Forms Services

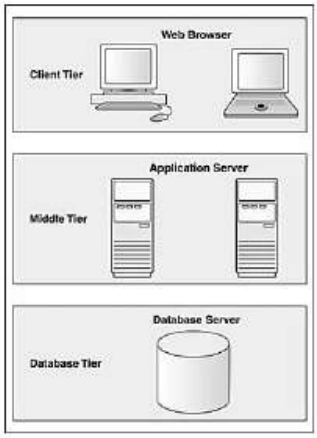
Es un entorno comprensivo para el despliegue de aplicaciones basadas en formularios. Las aplicaciones que se diseñan y desarrollan en Oracle Forms Developer posteriormente hay que desplegarlas en Oracle Fusion Middleware teniendo levantado el servicio de Oracle Forms Services.

ARQUITECTURA DE FUNCIONAMIENTO

La arquitectura de Oracle Forms Service se caracteriza por el uso de tres capas:

- La capa del cliente donde se encuentra el navegador por el que se accede a la aplicación desarrollada en Oracle Forms. Además del navegador, también son necesarios en el cliente el entorno de ejecución de Java (JRE) y el plug-in Java (JPI).
- La capa media donde se encuentra el servidor de aplicación.
- La capa de base de datos donde se encuentra almacenada la información que es accesible desde el formulario.

A continuación se muestra una imagen de esta arquitectura.



HERRAMIENTAS DE ORACLE FORMS

En algunos casos la instalación del software no crea en la lista de programas enlaces a las herramientas de Oracle Forms, en cambio sí lo hace con otras utilidades asociadas al software instalado: SQL Developer o SQL Plus. En estos casos resulta más cómodo crearse una serie de accesos directos en el escritorio a las distintas herramientas que componen Oracle Forms. A continuación se indica el nombre de los ejecutables y su significado para crearse un acceso directo.

Nombre del ejecutable	Descripción
`frmbld.exe`	Oracle Forms Builder. Permite desarrollar, compilar y ejecutar formularios.
`frmcmp.exe`	Oracle Forms Compiler. Permite compilar formularios desarrollados con la herramienta Forms Builder.

TIPOS DE FICHEROS GENERADOS DESDE FORMS

Oracle Forms es una herramienta que permite el manejo de diversos tipos de módulos y objetos que se almacenan en la base de datos. Para distinguir unos de otros se utilizan las siguientes extensiones de archivos:

- **.FMB**: módulos de formulario en código fuente.
- **.FMX**: módulos de formulario compilados y listos para ser ejecutados.
- **.PLL**: librerías de código PL/SQL en código fuente.
- **.PLX**: librerías de código PL/SQL compilado y listos para ser referenciados en los formularios que se ejecutan.
- **.MMB**: módulos de menú en código fuente.
- **.MMX**: módulos de menú compilados y listos para ser referenciados en los formularios que se ejecutan.
- **.OLB**: bibliotecas de objetos de Forms.

COMPONENTES DE FORMS BUILDER 5

INTRODUCCIÓN

Forms Builder consta de una serie de herramientas integradas para facilitar el diseño y la programación de los distintos tipos de módulos y objetos que se pueden manejar desde la misma.

A continuación se relacionan cada uno de estos componentes:

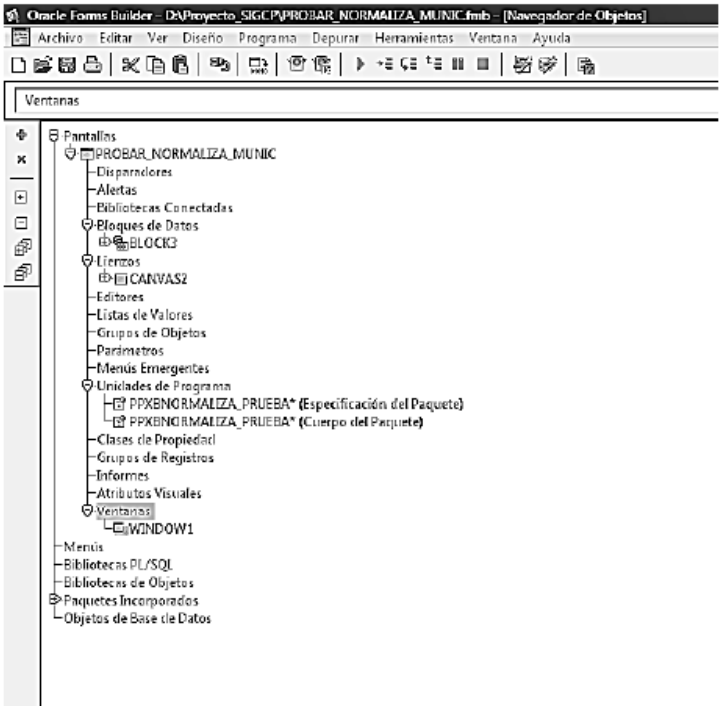
- **Navegador de Objetos**; es una herramienta que nos permite, de una forma visual, jerárquica e intuitiva, organizar los objetos que tiene un formulario.
- **Paleta de Propiedades;** es un conjunto de propiedades que posee cada objeto de un formulario que se muestran de forma visual y agrupadas por conceptos en este componente.
- **Editor de Diseño;** es una utilidad gráfica que permite diseñar el formulario utilizando todos los objetos gráficos de los que dispone Forms Builder.
- **Editor PL/SQL;** es una utilidad que permite incorporar código PL/SQL en la programación que se efectúa para cada formulario.
- **Editor de Menús;** es una utilidad gráfica que permite diseñar las barras de menús (horizontales) que pueden acoplarse a los formularios.
- **Consola de Depuración;** es una herramienta visual que nos permite realizar un seguimiento de la ejecución de un formulario, a fin de corregir los errores que se produzcan.

Navegador de Objetos

El navegador de objetos es una interfaz de exploración y edición jerárquica. Puede utilizar el navegador de objetos para ubicar y manipular objetos de aplicación, de forma rápida y sencilla. Entre sus funciones se encuentran:

- Una jerarquía representada por sangrado y nodos ampliables. Los nodos de nivel superior muestran tipos de módulos, objetos de base de datos y paquetes incorporados. Todos los demás nodos y los objetos que contienen se muestran con sangrado para indicar que pertenecen a dichos nodos de nivel superior.
- Campo de búsqueda e iconos que permiten realizar búsquedas hacia delante y hacia atrás, para cualquier nivel de nodo o para un elemento individual de un nodo.
- Los iconos de la barra de herramientas vertical replican las funciones comunes de los menús Editar y Ver.
- Un icono situado junto a cada objeto para indicar el tipo de objeto representado.

A continuación se muestra una imagen del navegador de objetos:

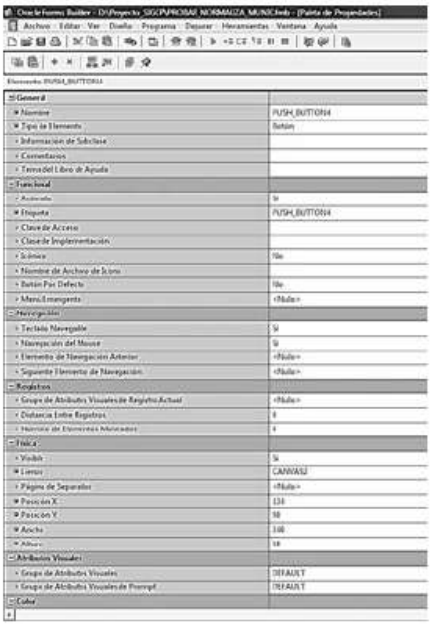


Paleta de propiedades

Todos los objetos de un módulo, incluido el propio módulo, tienen propiedades que se pueden ver y modificar en la paleta de propiedades. Entre sus funciones se encuentran:

- Copia y reutilización de propiedades de otro objeto.
- Campo Buscar e iconos, similar a los del navegador de objetos.

A continuación se muestra una imagen de la paleta de propiedades:

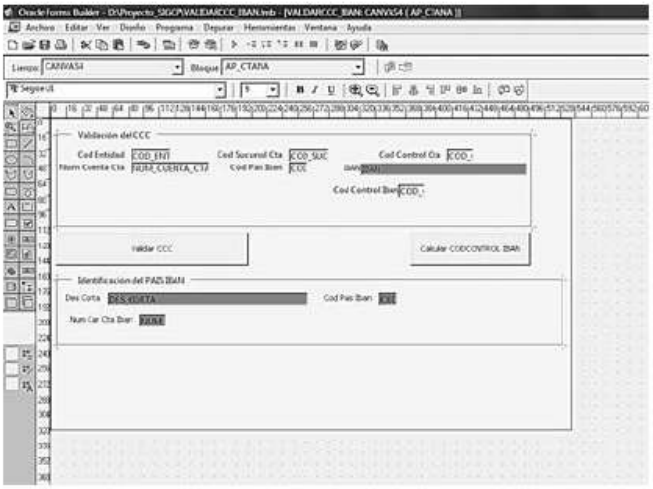


Editor de diseño

El editor de diseño (o modelo de diseño) es una utilidad de diseño gráfico para la creación y organización de elementos de interfaz y objetos gráficos en la aplicación. Entre sus funciones se encuentra:

- Diseño de formularios pudiendo utilizar la paleta de herramientas y la barra de herramientas, disponibles en el editor de diseño.
- Diseño del estilo, el color, el tamaño y la organización de los objetos visuales en la aplicación.
- Permite incluir objetos gráficos propios de Forms Builder, así como imágenes.
- Dispone de guías para una mejor ubicación de los objetos.

A continuación se muestra una imagen del editor de diseño:

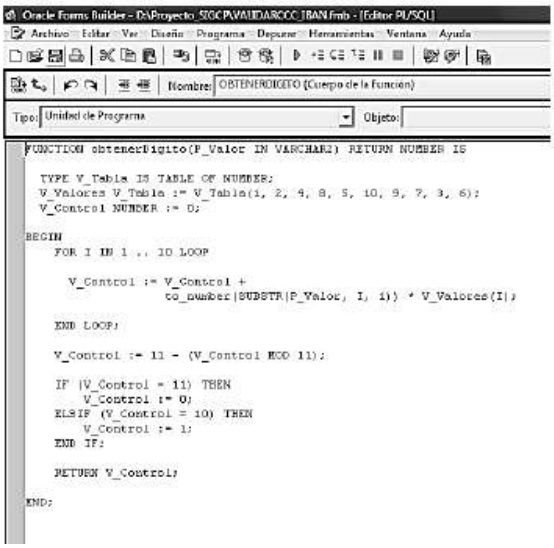


Editor PL/SQL

El editor PL/SQL permite incorporar objetos de código PL/SQL en la lógica de programación de los formularios. Entre los objetos de códigos se incluyen los siguientes:

- Disparadores de evento.
- Subprogramas (funciones y procedimientos).
- Comandos de opciones de menú.
- Código de inicio de menú.
- Paquetes.

En el propio editor PL/SQL, se puede introducir y compilar el código escrito en PL/SQL, así como en SQL. A continuación se muestra una imagen del editor:

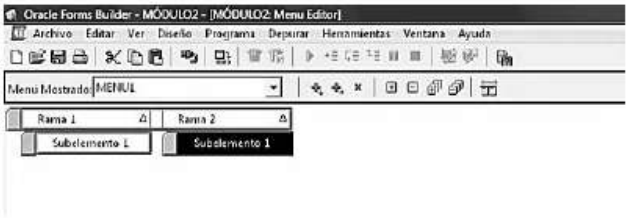


```
FUNCTION obtenerDigito(P_Valor IN VARCHAR2) RETURN NUMBER IS

  TYPE V_Tabla IS TABLE OF NUMBER;
  V_Valores V_Tabla := V_Tabla(1, 2, 4, 8, 5, 10, 9, 7, 3, 6);
  V_Control NUMBER := 0;

BEGIN
    FOR I IN 1 .. 10 LOOP

      V_Control := V_Control +
                   to_number(SUBSTR(P_Valor, I, 1)) * V_Valores(I);

    END LOOP;

    V_Control := 11 - (V_Control MOD 11);

    IF (V_Control = 11) THEN
        V_Control := 0;
    ELSIF (V_Control = 10) THEN
        V_Control := 1;
    END IF;

    RETURN V_Control;

END;
```

Editor de menús

Es una utilidad gráfica que permite diseñar las barras de menús horizontales que pueden acoplarse a nuestros formularios. Entre sus funciones se encuentran:

- Crear elementos del menú (hacia la izquierda o hacia abajo) de forma gráfica.
- Editor de PL/SQL integrado para dotar de funcionalidad a cada elemento del menú.
- Uso de elementos de menú predefinidos (elementos mágicos como Copiar, Cortar y Pegar).
- Uso de divisores (separadores) en los menús.
- Acceso rápido a los elementos del menú.



En esta pantalla se muestra una imagen del editor:

Consola de depuración

Es una herramienta visual que nos permite realizar un seguimiento de nuestro programa, a fin de corregir errores. Entre las funciones principales se encuentran:

- Creación de puntos de ruptura en el programa.
- Posibilidad de comprobación línea a línea, bloque a bloque o estructura a estructura.
- Visualización de los valores de cualquier objeto del programa en tiempo de ejecución.
- Posibilidad de cambio en el código del programa, solo para su depuración, sin que afecte al código original.

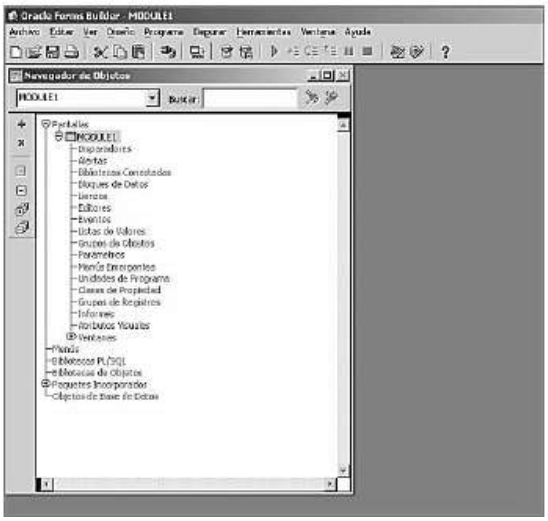
A continuación se muestra una imagen de la consola:

LA INTERFAZ DE FORMS BUILDER 6

INICIO DE FORMS BUILDER

Para iniciar Forms Builder, ejecute el fichero **frmbld.exe** desde el directorio de instalación.

Cuando se llama a Forms Builder, en primer lugar se muestra el recuadro de diálogo de *Bienvenida* donde nos facilita, mediante un asistente, la creación de ciertos objetos de Forms o abrir alguno ya existente. Esta pantalla puede ser deshabilitada para que no se muestre al comienzo de una sesión de Forms Builder desmarcando la casilla **Mostrar al Inicio.**



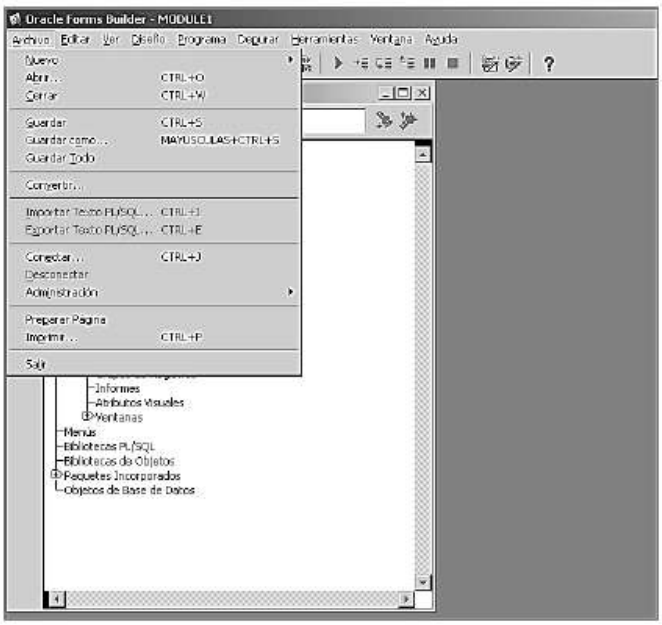
Si pulsamos el botón **Cancelar** para salir del recuadro de diálogo sin seleccionar ninguna de las opciones que nos muestra, aparecerá el navegador de objetos y un módulo (formulario) nuevo vacío.

Si se piensa crear un formulario que acceda a objetos de base de datos, necesita conectarse a una cuenta de usuario habilitada en la misma.

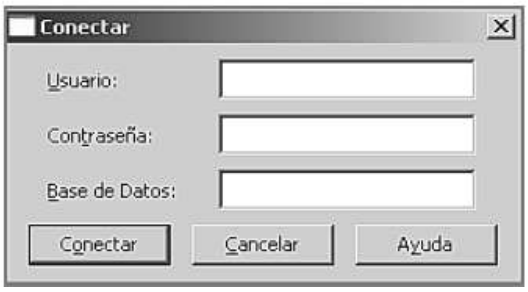
CÓMO CONECTARSE A UNA BASE DE DATOS ORACLE

Para poderse conectar a un usuario de base de datos Oracle desde Forms Builder hay que seguir estas indicaciones:

1. En la barra de menús de Forms Builder seleccionamos la opción **Archivo** y dentro de la misma elegimos **Conectar**, tal cual se muestra en la siguiente imagen:



2. A continuación se mostrará un cuadro de diálogo para introducir las credenciales de conexión a la base de datos. En el apartado **Base de Datos** tenemos que introducir uno de los alias que se hayan definido para conectarse a las bases de datos del sistema (los alias se encuentran relacionados en el fichero `tnsnames.ora` de la instalación de Oracle).



LA ESTRUCTURA DE LA BARRA DE MENÚS

El menú principal de Forms Builder contiene opciones que le permiten crear, modificar y gestionar los módulos y el resto de elementos que permiten su modificación dentro de la herramienta. Algunos de los elementos del menú tienen definidas teclas de acceso rápido para acceder a ellos.

Se compone de los siguientes elementos:

- **Archivo**; permite realizar tareas comunes con archivos como abrir, cerrar y guardar. Además permite conectarse a la base de datos, crear informes de objetos e imprimir la pantalla.
- **Editar;** permite realizar las tareas propias de cualquier ventana de Windows como son cortar, copiar, pegar y deshacer, así como operaciones propias del elemento seleccionado como crear y limpiar. Por último desde este apartado del menú se puede acceder a cambiar las preferencias del Forms Builder y realizar búsquedas.
- **Ver;** permite cambiar la vista en la ventana actual. Las opciones dentro de este menú varían dependiendo del contenido seleccionado.
- **Diseño;** solo se encuentra activa dentro del editor de diseño, y presenta las opciones comunes relacionadas con el diseño del formulario.
- **Programa;** incluye las opciones de compilación del código del módulo.
- **Depurar;** incluye las opciones de acceso a la consola de depuración.
- **Herramientas;** permite el acceso a los asistentes y componentes de Forms Builder.
- **Ventana;** nos muestra las características genéricas de cualquier ventana de Windows para su visualización (cascada, mosaico, minimizar, etc.).
- **Ayuda;** contiene la ayuda del programa.

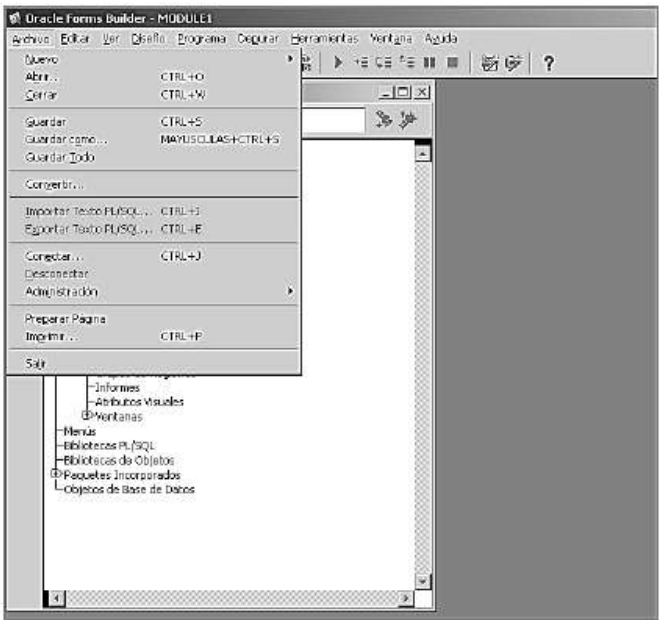
Archivo

Este menú se compone de las siguientes opciones:

- **Nuevo;** para crear un nuevo objeto compatible con Forms Builder: formulario, formulario de plantilla, menú, biblioteca PL/SQL o biblioteca de objetos.
- **Abrir;** para abrir un objeto existente que sea compatible con Forms Builder.
- **Cerrar;** para cerrar el objeto actual sin salir de Forms Builder.

- **Guardar;** para almacenar el objeto actual con el nombre asignado al mismo o si es la primera vez, se almacenará con el nombre que tenga el objeto seguido por la extensión que corresponda según el tipo.
- **Guardar como;** para almacenar el objeto actual con un nuevo nombre.
- **Guardar Todo;** para almacenar todos los objetos abiertos.
- **Capturar Diseño;** para capturar la información de un diseño en Oracle Designer.
- **Convertir;** para convertir módulos de binario a texto, y viceversa.
- **Importar Texto PL/SQL;** esta función se activa cuando nos encontramos dentro del editor PL/SQL y permite importar código PL/SQL de archivos almacenados.
- **Exportar Texto PL/SQL;** esta función se activa cuando nos encontramos dentro del editor PL/SQL y permite exportar el código PL/SQL visualizado a un archivo.
- **Conectar;** para conectarse a una base de datos.
- **Desconectar;** para desconectarse de la conexión abierta contra una base de datos.
- **Administración;** para generar informes y efectuar y liberar bloqueos.
- **Preparar Página;** para configurar la página antes de la impresión.
- **Imprimir;** para imprimir.
- **Salir;** para salir de Forms Builder.

A continuación se muestra una imagen del menú *Archivo*:



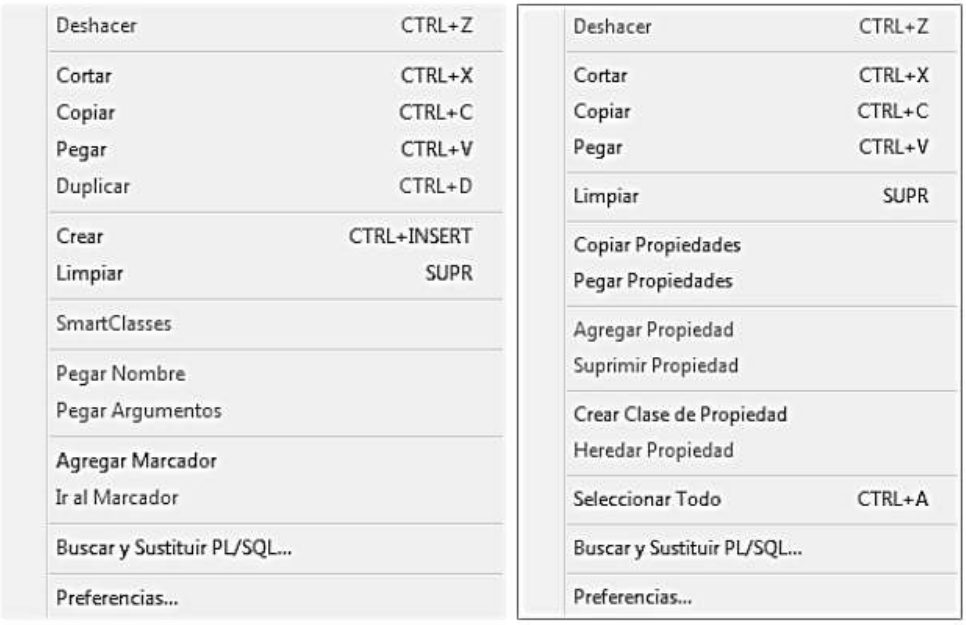
Editar

Este menú se compone de elementos que varían dependiendo del objeto que se haya seleccionado. A continuación se muestra una relación de todas las posibles opciones que pueden mostrarse:

- **Deshacer;** devuelve el objeto al estado inmediatamente anterior a la última modificación realizada.
- **Cortar;** corta un elemento seleccionado previamente.
- **Copiar;** copia un elemento que se ha seleccionado.
- **Pegar;** pega un elemento que se ha copiado o cortado previamente.
- **Duplicar;** permite duplicar un objeto con las mismas propiedades.
- **Crear;** permite crear un nuevo objeto en el navegador de objetos.
- **Limpiar;** permite borrar un elemento del formulario.
- **Suprimir;** borrar la parte del código seleccionado en el editor PL/SQL.
- **Sangrar;** permite realizar una sangría en el editor PL/SQL.
- **Eliminar Sangría;** elimina una sangría creada en el editor PL/SQL.
- **SmartClasses;** permite crear objetos de la librería de estándares o bien convertir un objeto en un estándar.
- **Seleccionar Todo;** permite seleccionar todos los elementos de un objeto en el editor de diseño.
- **Importar;** permite importar un elemento de tipo imagen o paleta de colores al formulario en el editor de diseño.
- **Exportar;** permite exportar una imagen o paleta de colores seleccionada en el editor de diseño.
- **Opciones de Gráfico;** permite modificar opciones gráficas de un elemento seleccionado en el editor de diseño.
- **Opciones de Diseño;** permite modificar aspectos del editor de diseño.
- **Copiar Propiedades;** permite copiar una o varias propiedades seleccionadas de un elemento dentro de la paleta de propiedades.
- **Pegar Propiedades;** permite pegar una o varias propiedades copiadas con la función anterior sobre un elemento, dentro de la paleta de propiedades.
- **Crear Clase de Propiedad;** crea una clase de propiedad a partir de una o varias propiedades seleccionadas en un elemento, dentro de la paleta de propiedades.
- **Heredar Propiedad;** hereda las propiedades de otro objeto.
- **Pegar Nombre;** esta función se activa cuando nos encontramos dentro del editor PL/SQL y permite importar código PL/SQL de archivos almacenados.

- **Pegar Argumentos;** esta función se activa cuando nos encontramos dentro del editor PL/SQL y permite exportar el código PL/SQL visualizado en pantalla sobre un archivo.
- **Agregar Marcador;** permite añadir un marcador a un elemento dentro del navegador de objetos.
- **Ir al Marcador;** se desplaza hasta el primer marcador señalado en el navegador de objetos.
- **Buscar y Sustituir;** permite realizar una búsqueda de un texto dentro del código visible actualmente en el editor PL/SQL.
- **Buscar y Sustituir PL/SQL;** permite realizar una búsqueda de un texto en todo el código PL/SQL que aparezca en el árbol de objetos de un formulario dentro del navegador de objetos.
- **Preferencias;** para cambiar diversas preferencias de la visualización, configuración de asistencias y arranque de Forms Builder.

A continuación se muestran diversas imágenes del menú *Editar*:



Ver

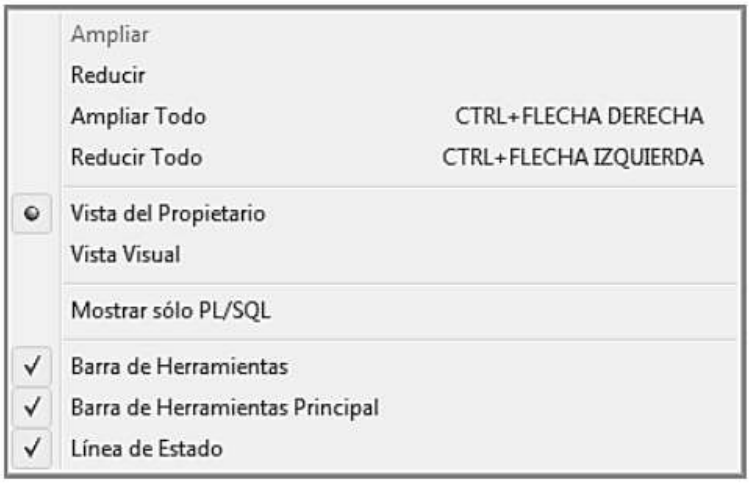
Este menú se compone de elementos que varían dependiendo del objeto que se haya seleccionado. A continuación se muestra una relación de todas las posibles opciones que pueden mostrarse:

- **Acercar;** permite hacer un zoom de aumento de la zona del editor de diseño en la que se encuentre posicionado el cursor.

- **Alejar;** permite hacer un zoom de decremento de la zona del editor de diseño en la que se encuentre posicionado el cursor.
- **Tamaño Normal;** devuelve al tamaño normal de visualización, el contenido del editor de diseño.
- **Ajustar a la Ventana;** ajusta al marco de la ventana la visualización completa del editor de diseño.
- **Mostrar Vista;** muestra/oculta la línea correspondiente al marco de la ventana donde se va a situar el lienzo que estamos pintando en el editor de diseño.
- **Mostrar Lienzo;** muestra/oculta el marco del lienzo o tapiz en el que estamos pintando los objetos dentro del editor de diseño.
- **Vistas Apiladas;** muestra de forma apilada los distintos lienzos que mantengamos abierto en el editor de diseño.
- **Reglas;** muestra/oculta las reglas verticales y horizontales para situar los objetos en el editor de diseño.
- **Guías de la Regla;** muestra/oculta las guías que se hayan creado sobre la regla del editor de diseño.
- **Cuadrícula;** muestra/oculta los puntos de visualización de la cuadrícula del editor de diseño.
- **Ajustar a Cuadrícula;** para ajustar el/los elementos seleccionados en el editor de diseño a la cuadrícula más cercana a ellos.
- **Personalizar Regla/Cuadrícula;** permite ajustar las opciones de visualización de la regla y cuadrícula del editor de diseño.
- **Paleta de Herramientas;** muestra/oculta la paleta de herramientas del editor de diseño.
- **Barra de Estado;** muestra/oculta la barra de estado.
- **Barra de Herramientas;** muestra/oculta la barra de herramientas icónica vertical que se muestra dependiendo del lugar donde nos encontremos en Forms Builder.
- **Barra de Herramientas Principal;** muestra/oculta la barra de herramientas principal.
- **Línea de Estado;** muestra/oculta todos los elementos de la línea de estado.
- **Ampliar;** para desplegar el contenido dependiente de un objeto del navegador (equivale a pulsar el símbolo + que haya en un objeto).
- **Reducir;** permite contraer el contenido dependiente de un objeto del navegador (equivale a pulsar el símbolo – que haya en un objeto).
- **Ampliar Todo;** permite desplegar el contenido de todos los objetos del navegador.
- **Reducir Todo;** permite contraer el contenido de todos los objetos del navegador.

- **Vista del Propietario;** es la opción por defecto de visualizar el navegador de objetos y permite visualizar todos los elementos del mismo.
- **Vista Visual;** únicamente se muestran los objetos relacionados con la visualización de un formulario: lienzos y ventanas.
- **Mostrar sólo PL/SQL;** únicamente muestra los objetos que poseen código PL/SQL asociado.

A continuación se muestran diversas imágenes del menú *Ver*:



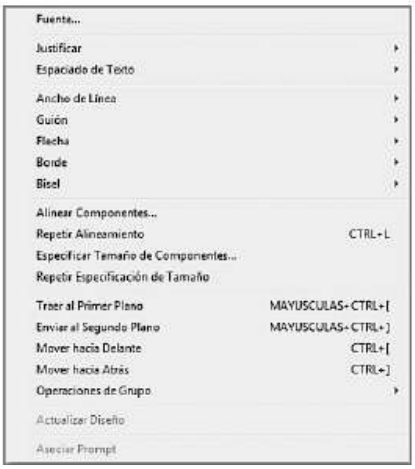
Diseño

Este menú se compone de los siguientes elementos que se habilitan cuando estamos trabajando en el editor de diseño:

- **Fuente;** muestra el cuadro de diálogo correspondiente a la fuente para modificar propiedades de la misma, correspondientes al/los elementos seleccionados.
- **Justificar;** permite especificar el tipo de justificación que se va a realizar sobre el contenido que se muestre en el elemento de tiempo de ejecución.
- **Espacio de texto;** permite cambiar las propiedades del espaciado del texto del contenido que se muestre en el elemento, en tiempo de ejecución.
- **Ancho de línea;** permite cambiar el grosor de la línea que bordea al elemento seleccionado.

- **Guion;** permite cambiar el tipo de línea correspondiente al borde del elemento seleccionado.
- **Flecha;** permite convertir la línea que bordea al elemento en una flecha.
- **Borde;** permite mostrar/ocultar las zonas del borde correspondiente a un elemento.
- **Bisel;** permite cambiar el bisel de un elemento seleccionado.
- **Alinear Componentes;** permite que varios elementos seleccionados se alineen en el editor de diseño, de una forma concreta.
- **Repetir Alineamiento;** permite repetir el último alineamiento efectuado sobre los objetos que estén seleccionados.
- **Repetir Especificación de Tamaño;** permite repetir la última especificación de tamaños sobre el/los objetos que estén seleccionados.
- **Traer al Primer Plano;** permite visualizar el elemento seleccionado por delante del resto (primer plano).
- **Enviar al Segundo Plano;** permite visualizar el elemento seleccionado por detrás del resto (segundo plano).
- **Mover hacia Delante;** mueve el elemento seleccionado una posición hacia delante.
- **Mover hacia Atrás;** mueve el elemento seleccionado una posición hacia atrás.
- **Operaciones de Grupo;** muestra las operaciones que se pueden llevar a cabo con un conjunto de elementos seleccionados a la vez.
- **Actualizar Diseño;** actualiza el diseño de un formulario con los últimos cambios efectuados que aún no se hayan refrescado en la pantalla.
- **Asociar Petición de Datos;** permite asociar una petición de datos de un elemento a otro.

A continuación se muestra una imagen del menú *Diseño*:

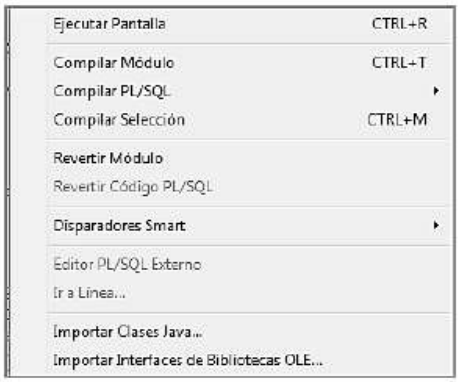


Programa

Este menú se compone de los siguientes elementos:

- **Ejecutar Pantalla;** permite ejecutar un formulario para probarlo en tiempo de ejecución.
- **Compilar Módulo;** compila un módulo entero.
- **Compilar PL/SQL;** compila el código PL/SQL que se esté visualizando en el momento de pulsar esta opción.
- **Compilar Selección;** compila únicamente la parte del código PL/SQL seleccionado.
- **Revertir Módulo;** vuelve al último estado del módulo almacenado antes de los cambios.
- **Revertir Código PL/SQL;** vuelve al último estado del código PL/SQL almacenado antes de los cambios.
- **Disparadores Smart;** muestra la lista de disparadores apropiados para el objeto seleccionado.
- **Editor PL/SQL Externo;** visualiza el código PL/SQL en el editor externo que se haya definido.
- **Ir a línea;** permite ir a una línea concreta del código PL/SQL visualizado.
- **Importar Clases Java;** muestra un cuadro de diálogo con las clases Java disponibles, para importar aquella/s que necesitemos.

A continuación se muestra una imagen del menú *Programa*:

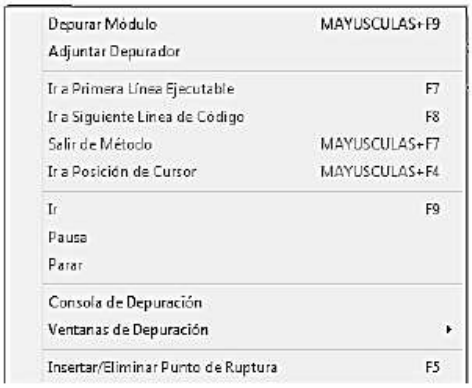


Depurar

Este menú se compone de los siguientes elementos, la mayoría de los cuales solo se encuentran activos en el modo de ejecución de pantalla con depuración.

- **Depurar Módulo;** lanza la ejecución del formulario en modo depuración.
- **Adjuntar Depurador;** permite depurar una aplicación que se encuentra en ejecución.
- **Ir a la Primera Línea Ejecutable;** se sitúa el puntero de ejecución de la depuración en la primera línea de código ejecutable dentro del programa.
- **Ir a la Siguiente Línea de Código;** el puntero de ejecución de la depuración se mueve a la siguiente línea de código del programa.
- **Salir de Método;** el puntero de ejecución de la depuración sale del método actual (bloque begin…end) y salta hasta la siguiente línea de código disponible en el programa.
- **Ir a Posición de Cursor;** el puntero de ejecución se sitúa en la posición donde se encuentra el cursor.
- **Ir;** ejecuta el módulo desde la posición actual del puntero de depuración.
- **Pausa;** hace una pausa en la ejecución de la depuración del formulario.
- **Parar;** detiene la ejecución del formulario y su depuración.
- **Consola de Depuración;** nos muestra la consola de depuración.
- **Ventanas de Depuración;** nos permite visualizar las distintas ventanas de depuración del formulario.
- **Insertar/Eliminar Punto de Ruptura;** permite insertar o eliminar un punto de ruptura dentro del código del formulario a fin de que la depuración se pare en dicho punto, para poder evaluar los resultados.

A continuación se muestra una imagen del menú *Depurar*:



Herramientas

Este menú se compone de los siguientes elementos:

- **Asistente de Bloques de Datos;** muestra el asistente para la creación de un bloque de datos.

- **Asistente de Diseño;** muestra el asistente para la creación de un diseño de un formulario (elementos del lienzo, disposición, textos informativos, etc.).
- **Asistente de Lista de Valores;** muestra el asistente para la creación de una lista de valores (LOV).
- **Editor de Diseño;** muestra el editor de diseño.
- **Editor PL/SQL;** muestra el editor PL/SQL.
- **Editor de Menús;** muestra el editor de menús.
- **Navegador de Objetos;** muestra el navegador de objetos.
- **Paleta de Propiedades;** muestra la paleta de propiedades de un elemento.
- **Paleta de Sintaxis;** muestra la paleta de sintaxis.
- **Biblioteca de Objetos;** muestra la biblioteca de objetos disponibles.
- **Reports Builder;** abre el diseñador de informes (Reports Builder).

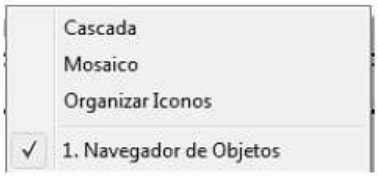
A continuación se muestra una imagen del menú *Herramientas*:

Ventana

Este menú se compone de los siguientes elementos propios de cualquier ventana de Windows:

- **Cascada;** muestra los distintos formularios abiertos en ventanas distintas, conformando una cascada.
- **Mosaico;** muestra los distintos formularios abiertos en ventanas distintas, conformando un mosaico.
- **Organizar Iconos;** muestra los distintos formularios abierto en forma de iconos organizados por nombre.

A continuación se muestra una imagen del menú *Ventana*:



Ayuda

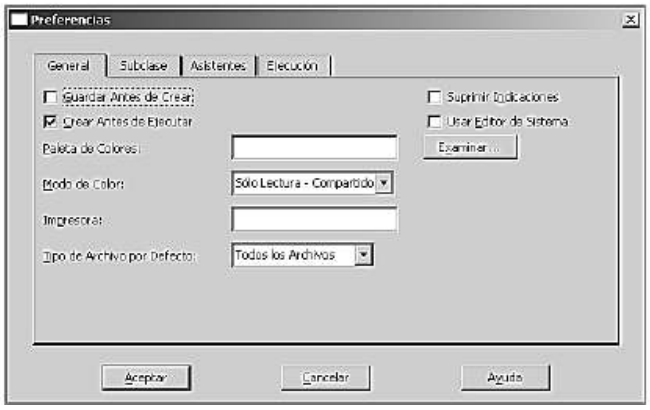
Este menú se compone de los siguientes elementos:

- **Ayuda en Pantalla;** muestra la ayuda de Forms Builder.
- **Forms en OTN;** muestra la información de ayuda de Forms Builder que existe en la web de Oracle Technology Network (OTN).
- **Acerca de Forms Builder;** muestra los créditos del programa.

A continuación se muestra una imagen del menú *Ayuda*:

PERSONALIZAR LA SESIÓN DE FORMS BUILDER

Para personalizar la sesión de Forms Builder tenemos que acceder a la opción **Preferencias** que se encuentra en la barra de menús **Editar**.



En esta pantalla se observa una imagen del cuadro de diálogo que se muestra al entrar en esta opción.

Las preferencias de Forms Builder se organizan en 4 solapas o separadores:

- **General;** permite determinar opciones para la ejecución de un formulario, seleccionar la paleta de colores e impresora por defecto, así como cambiar el tipo de editor que se mostrará.

- **Subclase;** determina cómo debe actuar Forms Builder cuando se guarde un formulario con objetos de tipo subclase en referencia a la ruta asociada al mismo.
- **Asistentes;** permite mostrar u ocultar la visualización de diversas páginas de bienvenida a los asistentes.
- **Ejecución;** determina el nombre de la máquina donde se ejecutan los formularios y el explorador web que se utilizará para abrirlos en modo ejecución.

General

Dentro de la solapa *General* se encuentran las siguientes opciones:

- **Guardar Antes de Crear;** para indicar a Forms Builder que guarde automáticamente el formulario cuando se invoca la compilación o ejecución del mismo.
- **Crear Antes de Ejecutar;** para indicar a Forms Builder que compile automáticamente el formulario antes de ejecutarlo.
- **Suprimir Indicaciones;** para eliminar los mensajes informativos (Hints) que aparecen en la barra de estado cuando se trabaja en Forms Builder.
- **Usar Editor de Sistema;** si está marcado, la información multilineal de un elemento de texto se mostrará en el editor por defecto que se tenga definido en el sistema operativo donde se esté ejecutando Forms Builder. Mientras que si no se marca esta opción, se mostrará en el editor por defecto de Forms.
- **Paleta de Colores;** determina la ruta y el nombre del fichero que contiene la paleta de colores que se utilizará por defecto en el diseño de los elementos de Forms.
- **Modo de Color;** determina la forma en la que se va a cargar la paleta de colores.
- **Impresora;** determina el nombre de la impresora por defecto que se utilizará en Forms Builder.

Subclase

Dentro de la solapa *Subclase* se encuentran las siguientes opciones:

- **Eliminar;** para indicar que la ruta del fichero del objeto original que se referencia como subclase sea eliminada.

- **Mantener;** para indicar que el objeto referenciado como subclase mantendrá la ruta.
- **Preguntar;** cada vez que se referencia una subclase se mostrará un cuadro de diálogo que nos preguntará si queremos mantener o eliminar la ruta de dicho objeto.

Asistentes

Dentro de la solapa *Asistentes* se encuentran las siguientes opciones:

- **Recuadro de Diálogo Bienvenido;** para mostrar (si se marca) u ocultar la visualización del cuadro de diálogo de bienvenida que aparece al arrancar Forms Builder.
- **Página de Bienvenida del Asistente de Bloque de Datos;** para mostrar (si se marca) u ocultar la visualización del cuadro de diálogo que aparece al crear un nuevo bloque de datos.
- **Página de Bienvenida del Asistente de Listas de Valores;** para mostrar (si se marca) u ocultar la visualización del cuadro de diálogo que aparece al crear una nueva lista de valores (LOV).
- **Página de Bienvenida de un Lienzo (Layout);** para mostrar (si se marca) u ocultar la visualización del cuadro de diálogo que aparece al crear un nuevo lienzo (layout).

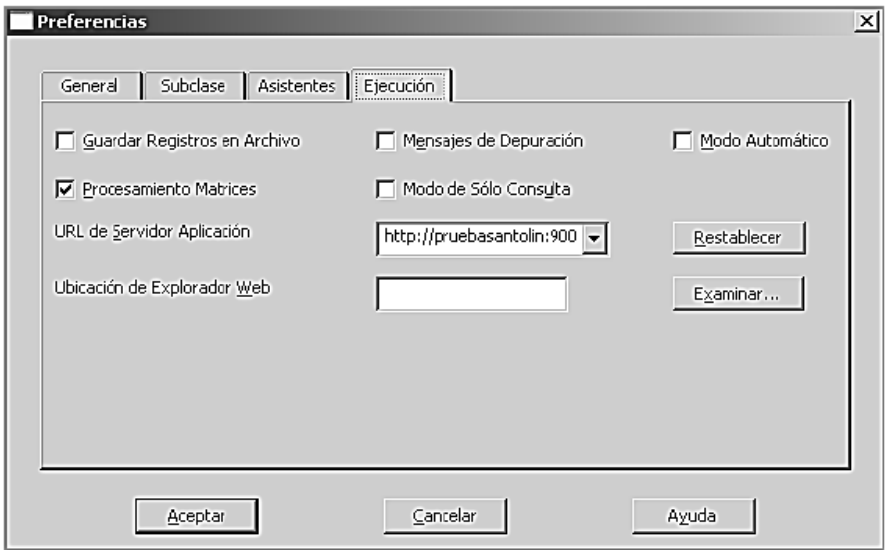
Ejecución

Dentro de la solapa *Ejecución* se encuentran las siguientes opciones:

- **Guardar Registros en Archivo;** para indicar a Forms Builder que guarde en un archivo físico aquellos registros que devuelva una consulta por encima del límite mínimo permitido más 3.
- **Mensajes de Depuración;** muestra los mensajes en tiempo de ejecución sobre el comportamiento de los triggers.
- **Procesamiento Matrices;** determina si Forms Builder procesa grupos de registros a la vez, para reducir el tráfico de red y aumentar el rendimiento.
- **Modo de Sólo Consulta;** fuerza que los formularios se ejecuten siempre en modo de solo consulta.
- **URL de Servidor Aplicación:** indica la ruta del equipo donde se ejecutan los formularios.

- **Ubicación de Explorador Web:** indica la ruta completa y el nombre del fichero ejecutable que arranca el explorador Web que se quiere utilizar para mostrar los formularios en modo ejecución.

A continuación se muestra una imagen con el contenido de esta solapa:



CREANDO EL PRIMER FORMULARIO ASISTIDO 7

INTRODUCCIÓN A LA PRÁCTICA

En este capítulo vamos a practicar la creación de nuestro primer formulario. Para ello vamos a utilizar los recursos que nos ofrece el asistente de Forms Builder.

Abrir Forms Builder y configurar bienvenida

El primer paso a realizar es la apertura del diseñador de formularios Forms Builder.

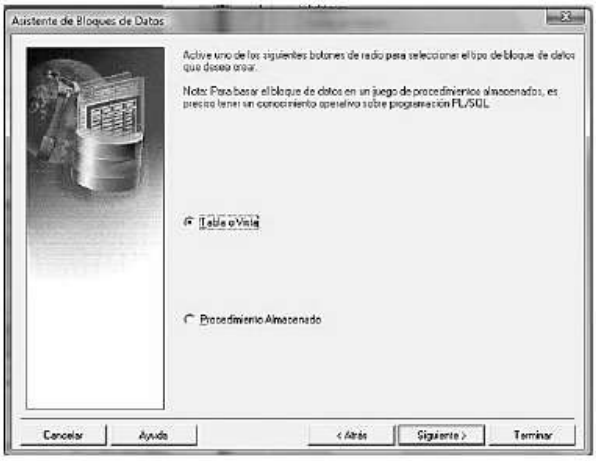
Una vez realizada esta tarea comprobaremos en el menú **Editar** – **Preferencias** – **Solapa Asistentes** que la opción **Recuadro de Diálogo Bienvenido** está marcada para que se nos muestre el cuadro de diálogo de bienvenida al abrir Forms Builder. Si no está marcada esta casilla, hay que hacerlo y cerrar el Forms Builder para posteriormente volverlo a abrir y que entonces se muestre el siguiente cuadro de diálogo.

En el cuadro de diálogo anterior mantendremos seleccionada la opción **Utilizar el Asistente de Bloque de Datos** y pulsaremos el botón **Aceptar** para comenzar el diseño del formulario.

Crear un bloque

El asistente de creación de un bloque de datos muestra en primer lugar una pantalla de bienvenida como la siguiente.

Únicamente tenemos que pulsar el botón **Siguiente** para avanzar en el proceso de creación del bloque.



A continuación se muestra el siguiente cuadro de diálogo del asistente.

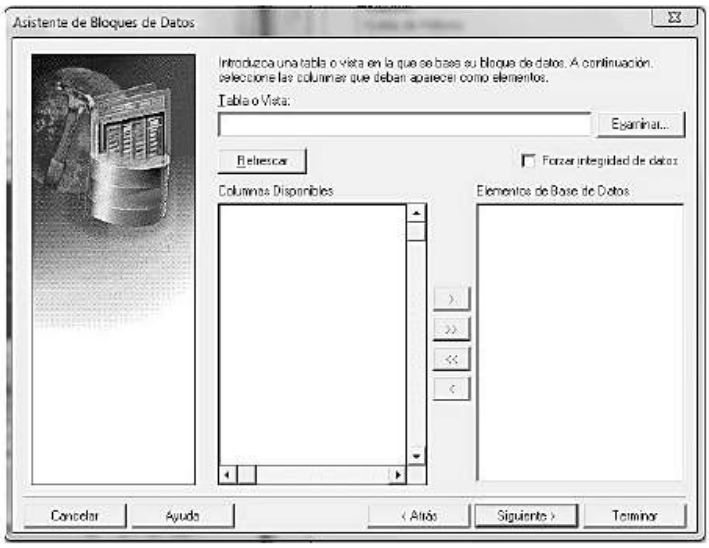
En esta pantalla tenemos que especificar el origen desde el que se obtiene la información para el bloque. Tenemos dos posibilidades:

- **Tabla;** para indicar que los datos se obtendrán del conjunto de valores de una tabla o vista definida en la base de datos a la que estemos conectados.
- **Procedimiento Almacenado;** para indicar que los datos se obtendrán de un procedimiento almacenado en base de datos que devuelve valores de tipo OUT o IN OUT.

Para nuestra primera práctica vamos a elegir la opción **Tabla** que es la más sencilla e intuitiva y pulsamos el botón **Siguiente.**

Conectarse a la base de datos

A continuación se nos muestra una nueva ventana para seleccionar una tabla o vista sobre la que queramos basar el formulario, como se muestra en la imagen de la página siguiente:



Para que se muestre la lista de tablas hay que pulsar el botón **Examinar** y se nos mostrará otra ventana donde nos solicitará las credenciales de conexión a la base de datos.

RECORDATORIO:

Para poder llevar a cabo las prácticas que se planean en este curso, es necesario tener instalada una base de datos Oracle en el mismo equipo donde se ha instalado Oracle Forms, o bien tener acceso a una base de datos Oracle remota (servidor). En cualquiera de los dos casos debemos disponer de un usuario de conexión con permisos suficientes para la creación de objetos (tablas, vistas, índices, procedimientos, paquetes, etc.).

En el libro "Oracle 11g SQL. Curso práctico de formación", publicado por el autor puede encontrar más información de cómo crear objetos de base de datos, otorgar permisos, etc.

Durante las prácticas que se van a realizar en este curso se accede a la información de un esquema de tablas correspondiente a un hospital. Para crear dicho esquema en la base de datos que tenga instalada en su equipo o a la que pueda acceder en remoto debe de ejecutar el script *script_bdhospital.sql* que se incluye junto a este libro.

Si no quiere introducir manualmente el código del script y prefiere ejecutarlo directamente, puede descargarse el fichero desde la página de la editorial RC Libros, donde encontrará instrucciones para su descarga. Una vez descargado hay que conectarse a la base de datos con un usuario administrador (tipo SYS o SYSTEM) y ejecutar el script para que se creen el esquema y el contenido (tablas, índices, datos, etc.).

Si se instala como base de datos local Oracle Express Edition habrá que realizar una modificación en el fichero **tnsnames.ora,** de la instalación de Forms & Reports 12c, con objeto de que se pueda acceder a la información de dicha base de datos. En primer lugar abrimos el fichero **tnsnames.ora** de la instalación de Oracle XE que se encuentra en la ruta por defecto **C:\oraclexe\app\oracle\product\11.2.0\server\network\ADMIN**. Dentro del mismo copiamos al portapapeles de Windows la sección *XE =….*, que tendrá un código similar al siguiente:

```
 XE =
 (DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP)(HOST = pruebasantolin)(PORT = 1521))
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = XE)
    ))
```

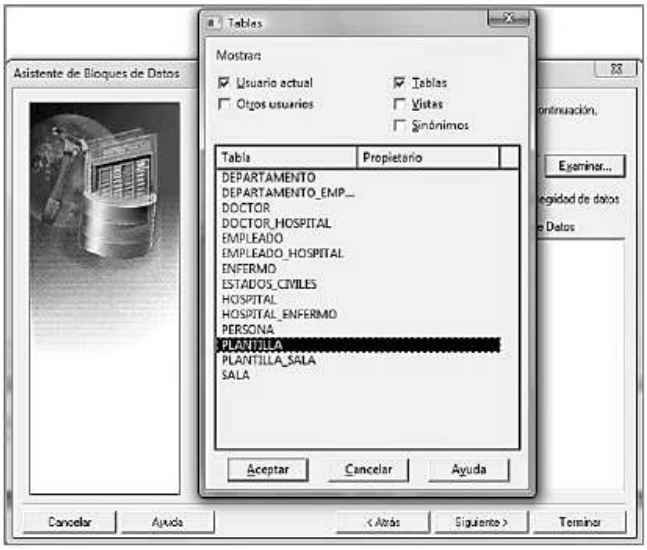
Ahora abrimos el fichero **tnsnames.ora** de la instalación de Oracle Forms & Reports 12c y pegamos el código copiado anteriormente. El contenido final fichero será similar al siguiente:

```
  ## sample tnsnames.ora file added by the Formsconfiguration
assistant.
  #
  #sample entry
  #<ALIAS> = (DESCRIPTION =(ADDRESS_LIST =
  #              (ADDRESS = (PROTOCOL = TCP)
  #              (HOST = <HOST>)(PORT = <PORT>))
  #              (CONNECT_DATA = (SID = <SID>))
  #
  #
   XE =
   (DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = pruebasantolin)(PORT = 1521))
  (CONNECT_DATA =
  (SERVER = DEDICATED)
  (SERVICE_NAME = XE)
      ))
```

Una vez llevadas a cabo estas operaciones, estamos en disposición de conectarnos a la base de datos hospital que se proporciona en el libro con las siguientes credenciales de conexión:

- Usuario: **PEPERF.**
- Contraseña: **PEPITO**.
- Base de datos: **XE**

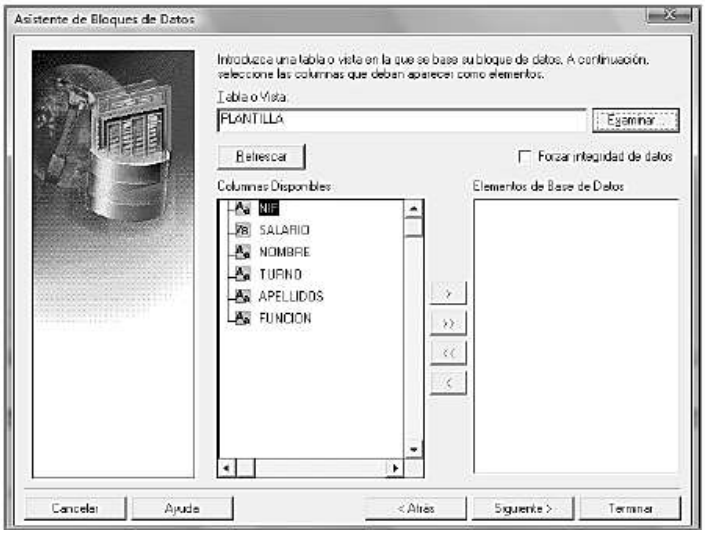
Selección de una tabla/vista



Tras introducir las credenciales se muestra un cuadro de diálogo con las tablas a las que posee acceso el usuario. Si además queremos mostrar las vistas y/o sinónimos disponibles hay que marcar las casillas correspondientes según se muestra en la siguiente imagen.

Para nuestro ejemplo, de la lista de tablas mostradas seleccionamos la tabla *PLANTILLA* y pulsamos el botón **Aceptar.**

Selección de las columnas de una tabla



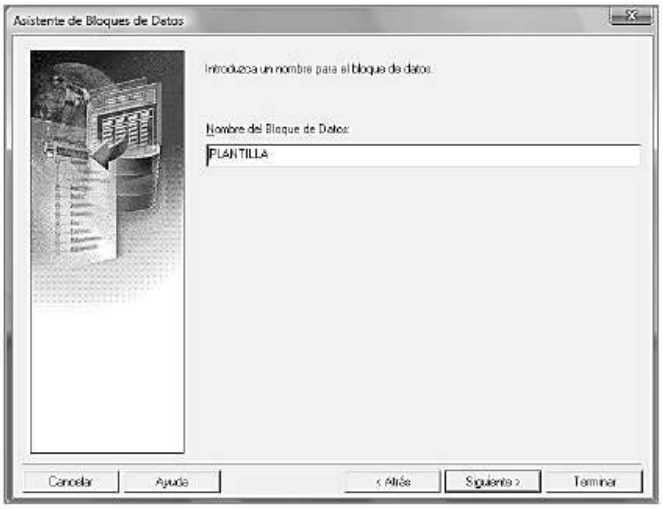
A continuación se recuperan las columnas que tiene la tabla en la zona del cuadro de diálogo denominada **Columnas disponibles** tal y como se muestra en la imagen de la página siguiente, donde tendremos que seleccionar cuál de ellas queremos mostrar en el formulario que estamos diseñando.

Para indicar las columnas que queremos que se vean hay que seleccionarlas en el apartado **Columnas Disponibles** y luego pulsar el botón **>** para trasladarlas al área **Elementos de Base de Datos**. En el supuesto de querer visualizar el contenido íntegro de la tabla se pulsará el botón **>>** para trasladarlas todas.

Si nos equivocamos en la selección pulsaremos el botón **<** para quitar una columna del área **Elementos de Base de Datos**, o bien pulsaremos **<<** para quitarlas todas.

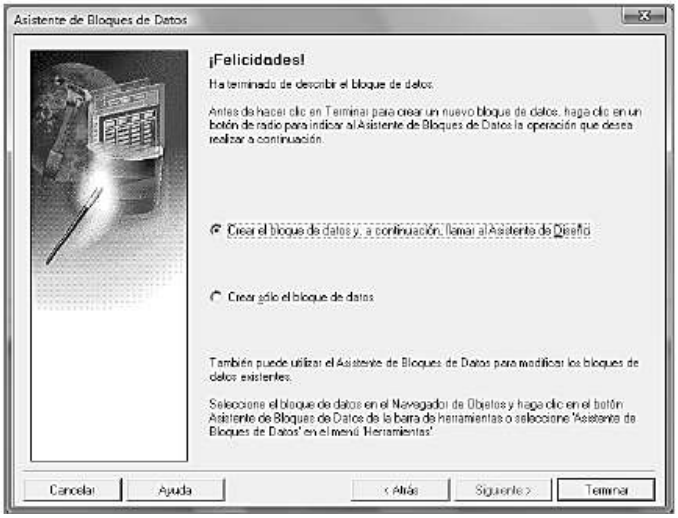
En nuestro caso práctico vamos a pulsar el botón **>>** para seleccionar todas las columnas de la tabla *PLANTILLA* para pintarlas en nuestro formulario y a continuación pulsamos el botón **Siguiente.**

Identificar el bloque con un nombre o alias



A continuación se presenta una nueva ventana en la que tenemos que identificar el bloque basado en la tabla seleccionada con un nombre. En nuestro caso se indicará como nombre de bloque *PLANTILLA* y pulsaremos el botón **Siguiente**.

Ejecución del Asistente de Diseño



Con el paso anterior se concluye la creación del bloque y a continuación comienza el diseño del lienzo (pantalla del formulario) donde se visualizará la información de las columnas seleccionadas en el bloque descrito en los pasos anteriores. Como comienzo del diseño del lienzo se muestra la siguiente pantalla.

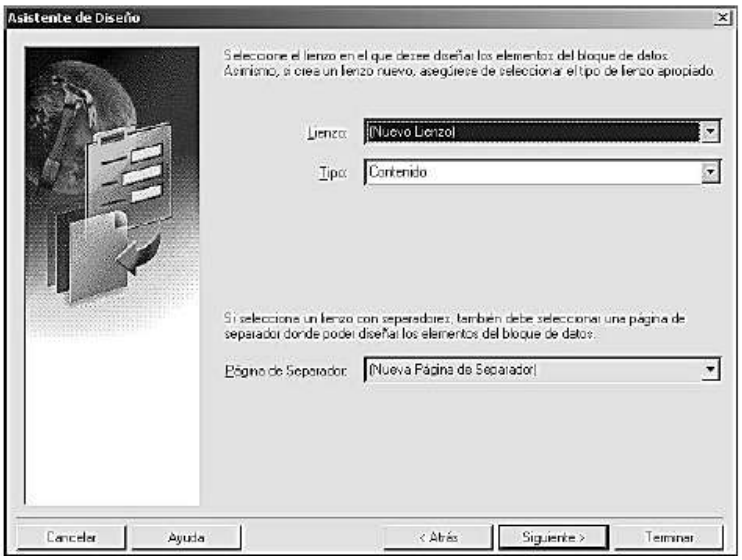
Dejamos marcada la opción por defecto de **Crear el bloque de datos y, a continuación llamar al Asistente de Diseño** para iniciar el diseño del lienzo. Si no quisiéramos diseñar el lienzo con el asistente y únicamente crear el bloque, tendríamos que seleccionar la segunda opción (*Crear sólo el bloque de datos*).

Pulsamos el botón **Terminar** y a continuación en la ventana de bienvenida al asistente de diseño pulsamos sobre el botón **Siguiente**.

Definición del tipo de lienzo

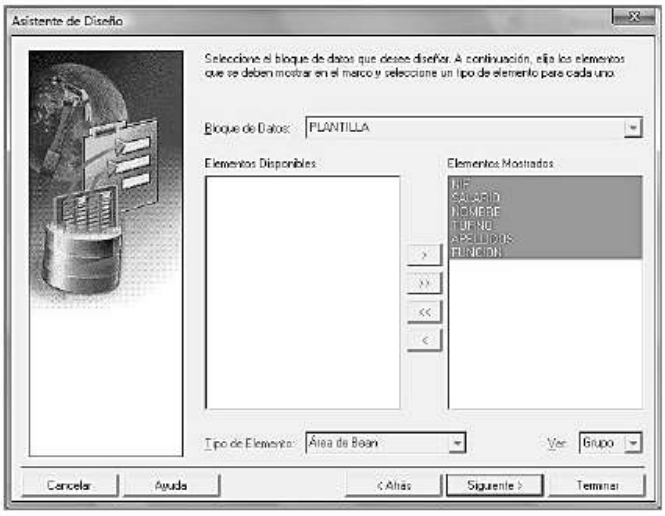
La primera fase del asistente de diseño de un lienzo nos solicita la definición del tipo de lienzo que se va a utilizar en el formulario. Existen 5 tipos de lienzos que se describen a continuación:

- **Contenido;** es el tipo de lienzo por defecto y se caracteriza porque el lienzo es un único tapiz liso.
- **Apilado;** define la creación de un lienzo que va a estar superpuesto a otro.
- **Separador;** define un lienzo con diversas solapas.
- **Barra de herramientas vertical;** define un lienzo vertical que se va a utilizar como barra de herramienta y que se adosará verticalmente a otro lienzo.
- **Barra de herramientas horizontal;** define un lienzo horizontal que se va a utilizar como barra de herramienta y que se adosará horizontalmente a otro lienzo.



En nuestro caso práctico vamos a dejar el tipo de lienzo por defecto (**Contenido**) y pulsamos el botón **Siguiente**.

Selección de elementos a visualizar en el lienzo

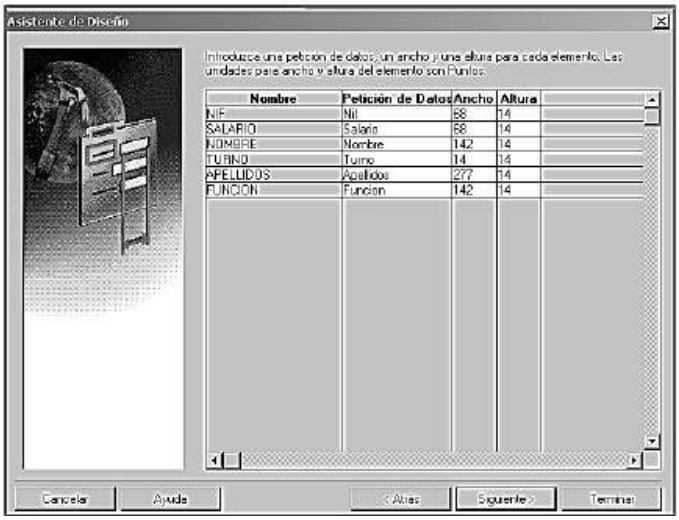


A continuación el sistema nos mostrará un nuevo cuadro de diálogo donde se muestran las columnas de las que dispone el bloque de datos (dentro del apartado *Elementos Disponibles*) para elegir aquellas que se quieren pintar en el lienzo.

Cada elemento que se quiera pintar en el lienzo hay que trasladarlo desde el apartado *Elementos Disponibles* al apartado *Elementos Mostrados*. Para hacerlo de forma individual hay que seleccionar primero la columna y luego pulsar el botón >. Si se quieren traspasar todas las columnas y pintarlas en el lienzo, hay que pulsar el botón >>.

En nuestro caso práctico pulsaremos el botón **>>** para mostrar todos los elementos del bloque y pulsaremos a continuación el botón **Siguiente**.

Definición de tamaños y Petición de Datos



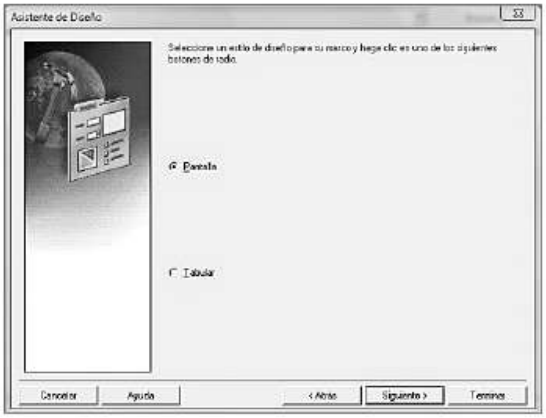
A continuación podemos definir en un nuevo cuadro de diálogo los tamaños (ancho y alto) de las celdas correspondientes a cada elemento, y la etiqueta (Petición de Datos) que acompañará a la celda de cada elemento.

En nuestro caso práctico vamos a aceptar como etiqueta (**Petición de Datos**) de los elementos, la misma que tiene el nombre de cada elemento y en cuanto a los tamaños, los que nos ofrece el propio sistema. Para terminar la definición de las columnas visualizadas pulsaremos el botón **Siguiente**.

Disposición de los elementos

A continuación el asistente nos ofrece dos posibilidades para mostrar los elementos:

- **Pantalla;** los elementos se posicionan intentando ocupar toda la pantalla y normalmente se colocan uno debajo de otro. En este caso solo se muestra un registro cada vez.
- **Tabular;** los elementos se posicionan en columnas (como una hoja de cálculo) y se muestran varios registros a la vez en el mismo formulario.



En nuestro caso práctico vamos a seleccionar la opción por defecto **Pantalla** y pulsaremos el botón **Siguiente**.

Título del marco y número de registros

La última fase del diseño nos solicita en primer lugar un título para el marco (frame) que engloba a todos los elementos que se van a mostrar, y en segundo lugar el número de registros que queremos que se visualicen a la vez en el mismo formulario. Por último en esta misma ventana tenemos que indicar la distancia entre dichos registros, y si se muestra o no una barra de desplazamiento vertical para movernos entre los diferentes registros mostrados.

En nuestro caso práctico seleccionamos para esta pantalla el **Título del Marco** *Relación de personas de la tabla PLANTILLA*, como número de **Registros Mostrados** indicamos *1* y como **Distancia entre Registros** indicamos el valor *0*, lo que significa

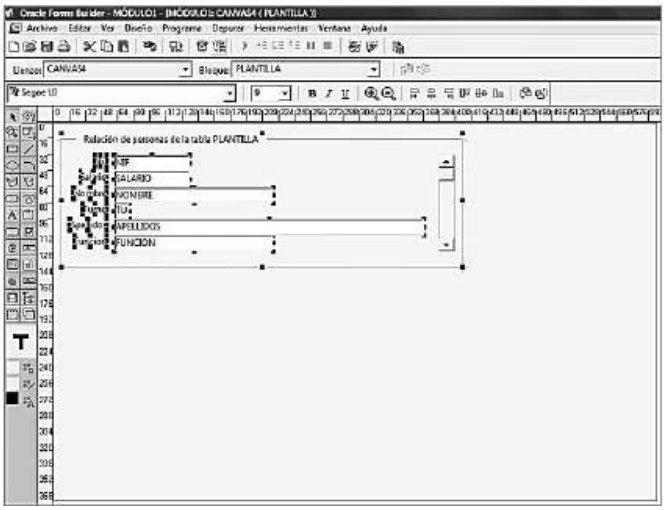
que entre las columnas visualizadas se aplica la distancia por defecto. Por último indicaremos que queremos **Mostrar Barra de Desplazamiento** en nuestro formulario.

Para terminar pulsamos el botón **Siguiente**, y a continuación se mostrará una nueva ventana (la última) con la felicitación por el proceso de creación del formulario.

Por último pulsamos el botón **Terminar**.

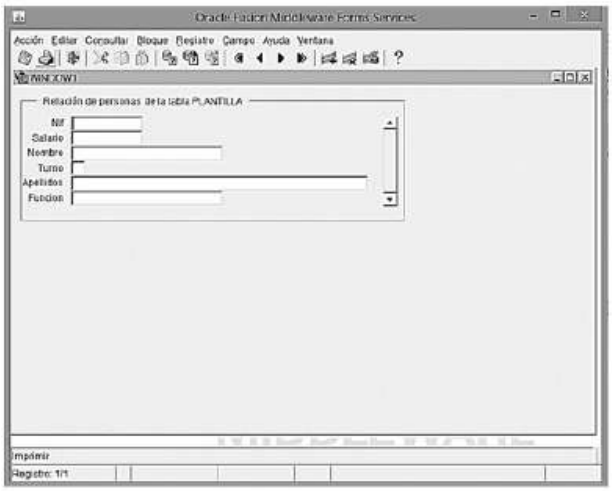
Trabajando en el editor de diseño

Cuando termina la fase asistida de creación de un formulario se presenta el editor de diseño donde se muestra la configuración de las columnas que el sistema ha determinado más adecuada según la configuración de los valores introducidos en los distintos asistentes.



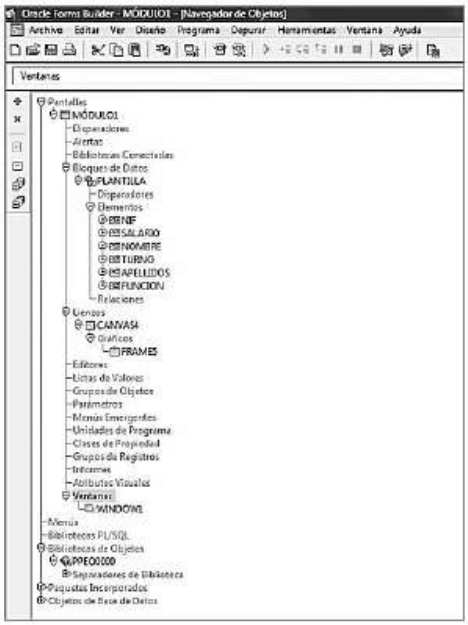
En esta pantalla podemos realizar libremente las modificaciones que se consideren oportunas para ajustar la visualización de las columnas en el lienzo.

Ejecución del formulario



Si queremos probar la ejecución del formulario hay que pulsar sobre el botón  . A continuación se muestra la imagen del formulario que se muestra en tiempo de ejecución.

Elementos del navegador de objetos



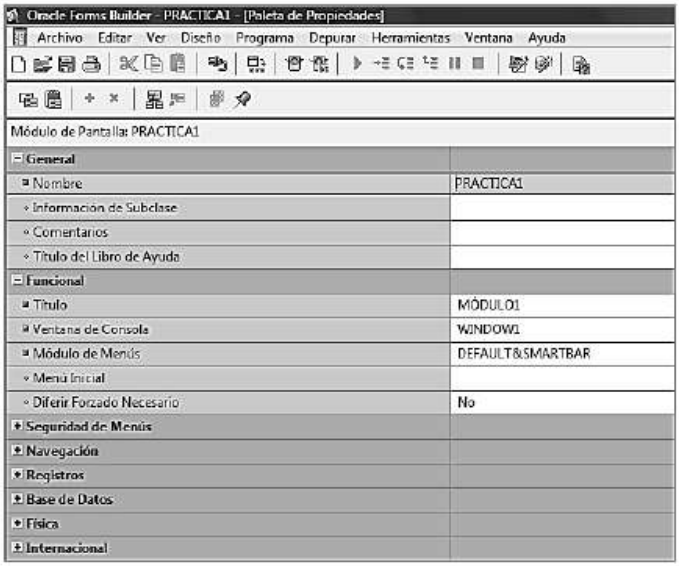
Si cerramos la ventana de ejecución del formulario y volvemos a Forms Builder para situarnos en el navegador de objetos, comprobaremos cómo se han creado los diversos elementos a través de los asistentes. Para ir al navegador de objetos desde el menú **Herramientas** elegimos **Navegador de Objetos**.

Es importante fijarse que el nombre del módulo que se asigna al formulario es **MODULO1**. Este nombre se asigna por defecto cuando se crea un nuevo formulario, pero es conveniente cambiarlo para asignarle el que cada uno considere necesario a fin de ser almacenado, con la restricción de que no puede tener más de 30 caracteres.

Cambiar el nombre del formulario

Para cambiar el nombre del formulario (módulos) marcamos en el navegador de objetos el nombre del mismo (en nuestro caso práctico *MODULO1*) y a continuación

pulsamos en la barra de menús **Herramientas,** para seleccionar dentro de la misma la opción **Paleta de Propiedades,** lo que hará que se muestre la siguiente pantalla:



Nos posicionamos en la propiedad **Nombre** y cambiamos *MODULO1* por el nuevo nombre (en nuestro caso práctico introduciremos *PRACTICA1*).

Guardar un formulario

A continuación guardaremos el formulario (módulo) en un fichero fuente con extensión **.FMB**.

Si queremos que el formulario se almacene en un fichero con el mismo nombre que tiene en el navegador de objetos, pulsaremos sobre la barra de menú **Archivo** y a continuación seleccionaremos la opción **Guardar.**

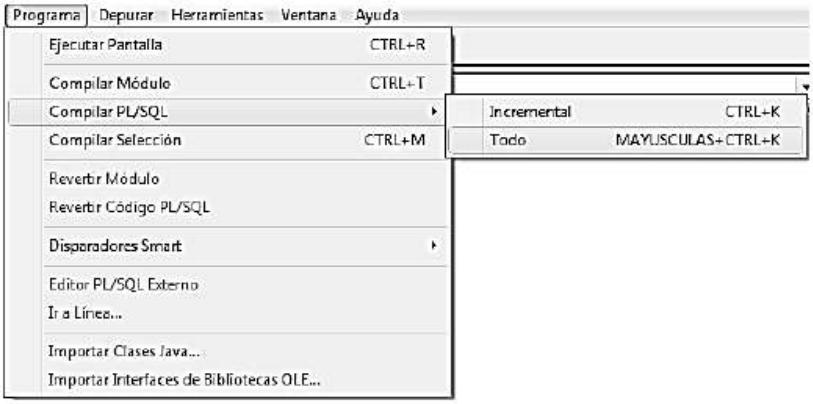
Si quisiéramos guardar el formulario con otro nombre diferente al que tiene en el navegador de objetos tendríamos que seleccionar la opción **Guardar Como** dentro de la barra de menú **Archivo** y se nos presentaría un cuadro de diálogo para introducir el nuevo nombre.

En nuestro caso práctico vamos a guardar el formulario con el mismo nombre que tiene en el navegador por lo tanto seleccionaremos la opción **Guardar** y se generará un fichero con el nombre **PRACTICA1.FMB.**

Compilar el formulario

Como continuación a nuestra práctica, en este punto vamos a compilar el formulario (módulo) para determinar si no hay ningún error de programación en el

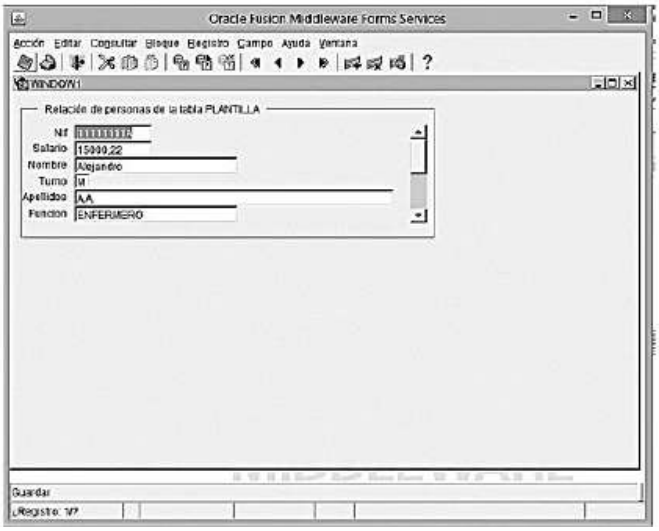
mismo. Para ello pulsaremos sobre la barra de menú en **Programa** y a continuación seleccionaremos la opción **Compilar PL/SQL – Todo**.



Generar el fichero ejecutable (.fmx)

Para generar un fichero ejecutable del formulario hay que seleccionar en la barra de menú **Programa**, la opción **Compilar Módulo**. Esto hará que se cree un fichero con el mismo nombre del fuente pero con extensión .FMX. En nuestro caso práctico, al hacerlo se genera el fichero **PRACTICA1.FMX.**

Pruebas de funcionamiento



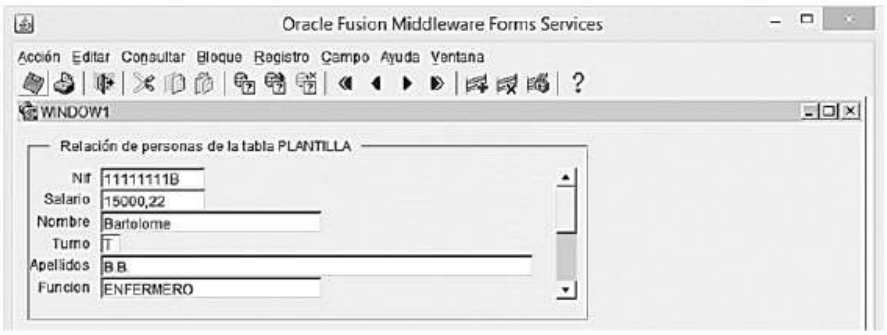
Por último comprobaremos la funcionalidad del formulario ejecutándolo. Para ello pulsamos sobre el botón  que aparece en la barra de herramientas de Forms Builder. La primera prueba de funcionalidad será la ejecución de una consulta no restringida.

Para ello en el formulario en ejecución pulsamos sobre la barra de menú en **Consultar** y seleccionamos la opción **Ejecutar**, lo que provocará que se muestre la información mostrada en la pantalla anterior.

Para navegar entre los distintos registros que se pueden mostrar en el formulario al acceder a la tabla de base de datos sobre la que están vinculados, hay 3 alternativas posibles:

- Utilizar la barra de menús *Registro* con las opciones de navegación que se incluyen en la misma.
- Utilizar la barra de navegación icónica que se muestra en la barra de herramientas del formulario: 
- Utilizar la barra de desplazamiento entre registros que se encuentra adosada a la derecha del frame donde se visualizan estos:

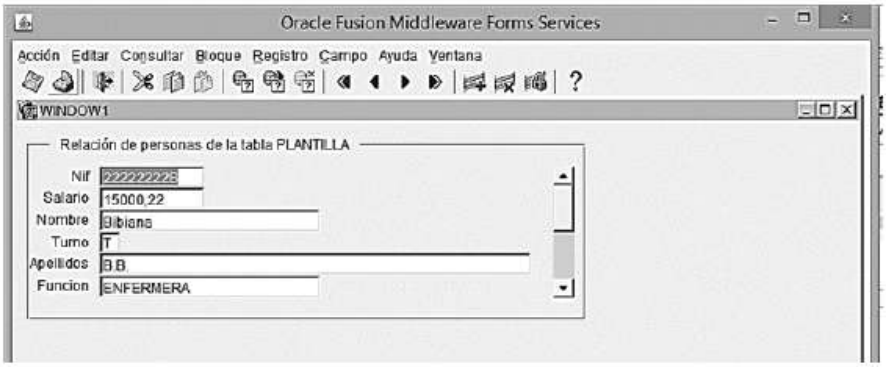
También podemos ejecutar consultas restringidas en las que indicamos criterios de búsqueda (filtro) sobre los distintos campos del formulario, para los que queremos obtener los resultados.



Para comenzar una consulta restringida en primer lugar tenemos que pulsar sobre la barra de menús **Consultar** y seleccionar la opción **Introducir**, lo que provoca que se borre la información de los campos del formulario para que podamos introducir los criterios de búsqueda necesarios. En nuestro caso práctico introduciremos en el campo *Turno* el valor **T**.

Para ejecutar la consulta con este criterio de búsqueda volveremos a seleccionar la barra de menús **Consultar** y seleccionaremos la opción **Ejecutar** lo que provocará que se devuelvan los resultados equivalentes, como se muestra en la pantalla anterior.

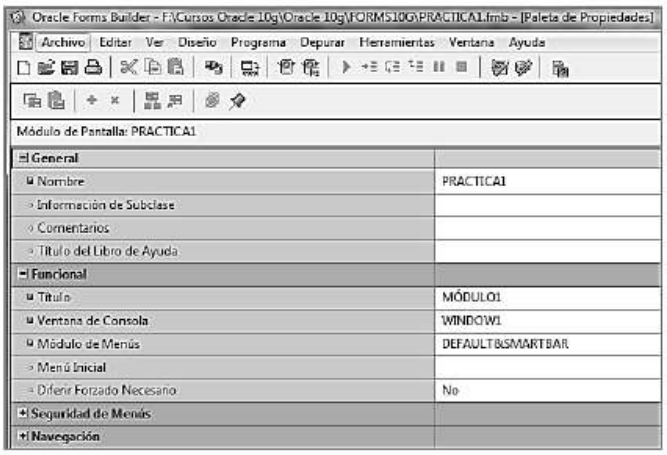
La última forma de consultar que vamos a probar en este caso práctico es la consulta restringida introduciendo patrones de texto, que equivale a realizar una consulta donde en la parte WHERE se introduce un comparador de tipo LIKE.



Al igual que en la consulta anterior seleccionamos en la barra de menús **Consultar** y a continuación la opción **Introducir**, para a continuación posicionarnos en el campo *Nombre* e introducir el texto **%b%** que significa que queremos buscar todos los nombres que contengan la letra **b** en cualquier posición.

Para ejecutar la consulta y obtener los resultados seleccionaremos en la barra de menús **Consultar** la opción **Ejecutar**, que nos devolverá la información que se muestra en la pantalla anterior.

LA BARRA SMARTBAR EN TIEMPO DE EJECUCIÓN



Cuando queremos ejecutar un formulario con el menú por defecto y la barra icónica por defecto (smartbar) que proporciona Forms, tenemos que incluir la propiedad **Default&Smartbar** en el apartado **Módulos de Menús** dentro de las propiedades de un formulario (módulo), como se muestra en la imagen de la página siguiente.

La barra Smartbar presenta el siguiente conjunto de herramientas icónica:

El significado de cada uno de los iconos es el siguiente:


Guarda un formulario con el nombre que posea en la carpeta destino por defecto.


Imprime un formulario.


Cierra y sale del formulario.


Cortar.


Copiar.


Pegar.


Introducir consulta.


Ejecutar consulta.


Cancelar consulta.


Primer / Anterior / Siguiente / Último registro.


Insertar en la b.d. un registro nuevo.


Borrar un registro existente de la b.d.


Bloquear un registro existente en la b.d.


Muestra la ayuda.

PROPIEDADES 8

INTRODUCCIÓN

Todos los objetos que se crean en Forms Builder tienen propiedades que determinan el comportamiento del mismo.

Cuando se crea un objeto dentro de Forms Builder se le asignan automáticamente una serie de valores por defecto a sus distintas propiedades.

Cualquier propiedad de un elemento de Forms se puede cambiar en la paleta de propiedades.

Cada elemento del Forms tiene diversos tipos de propiedades. Las propiedades se manipulan de distinta manera dependiendo del tipo de propiedad. A continuación se muestra un resumen de controles utilizados en la paleta de propiedades y una imagen de los mismos:

Control de propiedad	Descripción
Campo de texto	Se muestra cuando la propiedad actual se puede definir introduciendo un valor de texto. Para valores de texto mayores aparece también un botón icónico que permite abrir un editor de texto. A continuación se muestra un ejemplo de este tipo de control de propiedad. 
Lista emergente	Se da cuando se permite un juego de valores fijos (como SÍ o NO) para la propiedad. Haga clic en la flecha que se muestra a la derecha de la casilla para abrir la lista de valores y seleccionar uno de ellos. A continuación se

	muestra un ejemplo de este tipo de control de propiedad: 
Ventana LOV	Las listas de valores se dan cuando hay disponible una lista grande de valores posibles. Haga clic en el botón icónico que aparece en la columna del valor de propiedad para llamar a la lista de valores.
Botón MORE	Aparece cuando se necesitan valores más complejos que requieren de la apertura de un cuadro de diálogo adicional para introducirlos. A continuación se muestra un ejemplo de este tipo de control de propiedad: 

ICONOS DE LA PALETA DE PROPIEDADES

Cada propiedad de la paleta de propiedades tiene un icono a su izquierda. A continuación se muestra un resumen de dichos iconos y su descripción:

Icono	Descripción
Círculo	Especifica que el valor de la propiedad es el valor por defecto.
Cuadrado	Especifica que el valor de la propiedad ha cambiado de su valor por defecto a otro que le ha asignado el usuario.
Flecha	Especifica que el valor de la propiedad es heredada.
Fecha con una cruz	Especifica que el valor de la propiedad era heredada pero se ha sustituido por otro valor que asignó el usuario.

Una vez que se activa la paleta de propiedades la ventana permanece abierta hasta que se cierre. La ventana muestra automáticamente las propiedades de cada objeto visitado en el editor de diseño o en el navegador de objetos. Esto se debe a que por defecto, la lista de propiedades de la paleta de propiedades se sincroniza cuando se selecciona un objeto.

LA PROPIEDAD "NOMBRE"

Esta propiedad es común a todos los objetos que se pueden crear en Forms Builder y permite nominar a cada uno de ellos con un nombre.

Es necesario recordar que cualquier nombre de objeto de Forms Builder se ajusta a las reglas comunes de denominación de los elementos de base de datos de Oracle, es decir:

- El máximo de caracteres permitidos para el nombre es de 30 caracteres.
- El nombre debe comenzar obligatoriamente por una letra.
- El nombre puede incluir después de la primera letra los siguientes caracteres:
    - Letras.
    - Números.
    - El símbolo dólar ($).
    - El carácter de subrayado (_).
    - El símbolo almohadilla (#).
- El nombre no es sensible a las mayúsculas y minúsculas.

A continuación se muestra una imagen de la propiedad nombre:

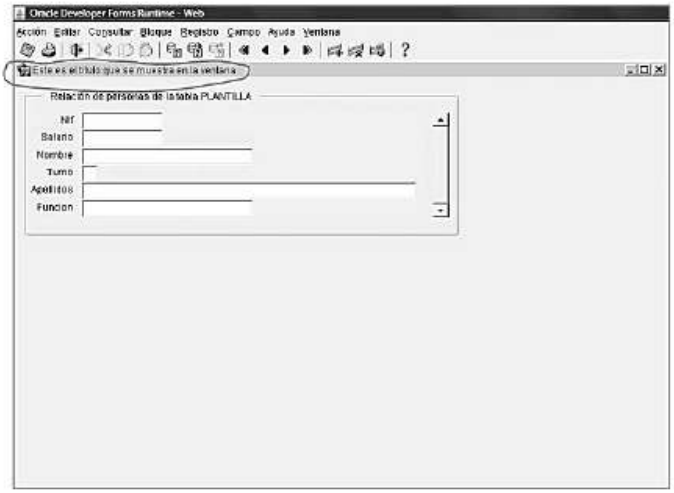
PROPIEDADES BÁSICAS DE LAS VENTANAS

Los objetos de tipo ventana dentro de Forms Builder presentan las siguientes propiedades básicas:

- Título de la ventana y título minimizado.
- Ancho y alto.
- Barras de desplazamiento.
- Color de fondo.
- Cierre, movimiento, maximización y minimización permitidos.

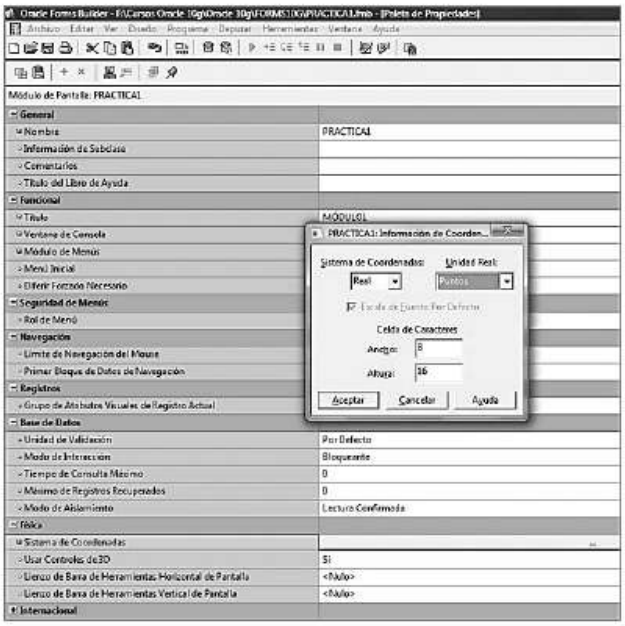
Título de la ventana y título minimizado

El título de la ventana (*propiedad* **Título***)* permite indicar el texto que queremos que se muestre como parte del marco que recubre la ventana. A continuación se muestra una imagen de una ventana con un título.



El **Título Minimizado** de una ventana permite que se muestre un texto debajo del icono que se forma al minimizarla.

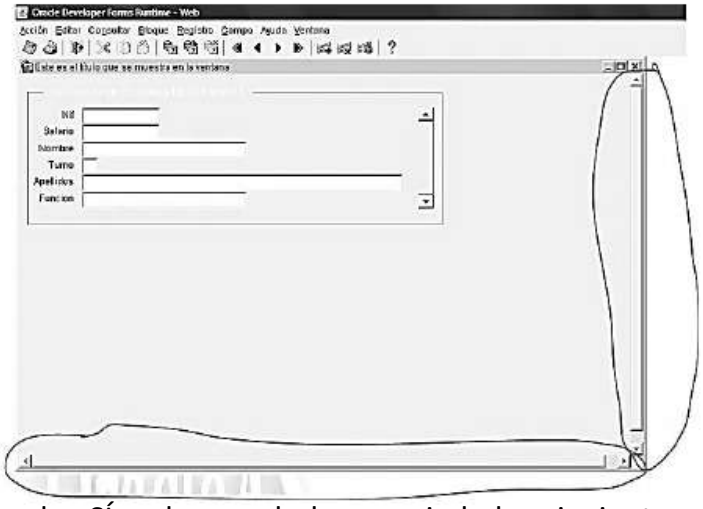
Ancho y alto



Entre las propiedades físicas de las ventanas (y en general del resto de elementos que se pueden visualizar), se encuentran el **Ancho** y la **Altura**. Estas propiedades permiten especificar el tamaño de una ventana.

Los valores correspondientes a estas propiedades se pueden especificar en centímetros, píxeles, pulgadas, puntos y valores en coma decimal, dependiendo de la unidad de medida seleccionada para el formulario. Para especificar la unidad de medida, tenemos que entrar en las propiedades del Módulo y pulsar sobre la propiedad **Sistema de Coordenadas**.

Barras de desplazamiento



Las barras de desplazamiento de una ventana permiten desplazar la visión del formulario de forma horizontal o de forma vertical, en el caso de que el mismo sea mayor al tamaño especificado para la ventana.

No obstante para que aparezcan físicamente las barras de desplazamiento dentro de la ventana, hay que indicar con el valor SÍ cada una de las propiedades siguientes: **Mostrar Barra de Desplazamiento Horizontal** y **Mostrar Barra de Desplazamiento Vertical.**

Color de fondo

Permite especificar un color para el fondo de la ventana, seleccionado dentro de los disponibles en la paleta de colores que se haya seleccionado.

Cierre, movimiento, maximización y minimización

Como cualquier ventana definida en el entorno Windows, con las ventanas de Forms podemos ejecutar las siguientes operaciones:

- Cerrar la ventana.
- Mover la ventana (desplazarla de su posición x, y inicial de apertura).

- Maximizar la ventana.
- Minimizar la ventana.
- Cambiar el tamaño de la ventana (redimensionamiento).

Pero además en Forms podemos programar el control sobre estas operaciones con el fin de permitir o prohibir realizar alguna de las mismas con la ventana en cuestión, en tiempo de ejecución del formulario.

Funcional	
Título	
Lienzo Primario	<Nulo>
Lienzo de Barra de Herramientas Horizontal	<Nulo>
Lienzo de Barra de Herramientas Vertical	<Nulo>
Estilo de Ventana	Documento
Modal	No
Ocultar al Salir	No
Cierre Permitido	Sí
Movimiento Permitido	Sí
Cambio de Tamaño Permitido	Sí
Maximización Permitida	Sí
Minimización Permitida	Sí
Título Minimizado	
Nombre de Archivo de Icono	
Menú Heredar	Sí

En esta imagen se muestra dónde se encuentran estas propiedades.

PROPIEDADES BÁSICAS DE LOS LIENZOS

Un lienzo es el espacio dentro de una ventana sobre la que se pintan los distintos elementos gráficos (casillas, botones, líneas, etc.). Toda ventana tiene como mínimo un lienzo establecido.

Todo lienzo presenta las siguientes propiedades básicas:

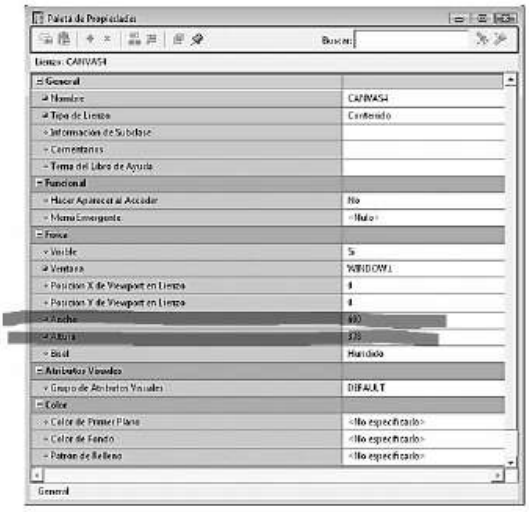
- Ancho y altura.
- Visible.
- Ventana contenedora.
- Color de fondo.
- El Viewport (posición x, y).

RECORDATORIO:

Todo formulario debe contener como mínimo los siguientes elementos: una ventana, un lienzo, un bloque y un elemento navegable perteneciente al bloque.

De no poseer estos elementos mínimos, no se podrá ejecutar el formulario.

Ancho y altura de un lienzo



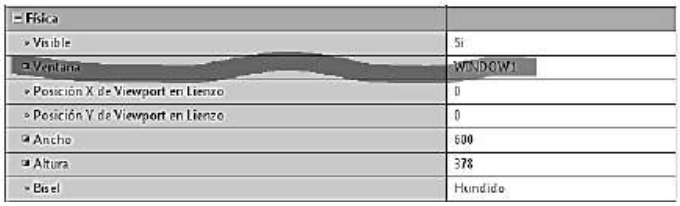
Estas propiedades actúan de la misma forma que con las ventanas, permiten definir el ancho y alto de un lienzo dentro de una ventana, especificando el tamaño en la unidad de medida que se haya especificado en el formulario.

Propiedad visible de un lienzo

Esta propiedad permite especificar si se muestra o se oculta la visión de un lienzo.

Ventana contenedora de un lienzo

Al igual que toda ventana requiere de un lienzo para que se visualice algo dentro de ella, todo lienzo debe de tener una ventana asociada donde se vaya a visualizar el contenido pintado en el lienzo.



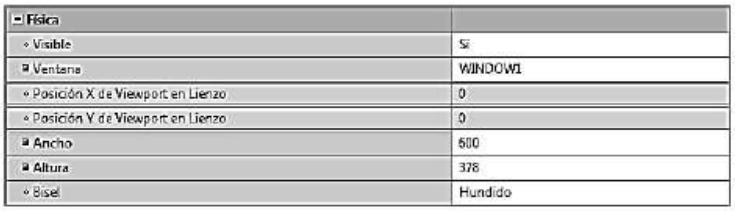
La propiedad que indica la ventana sobre la que se visualizará el lienzo (ventana contenedora) es **Ventana**.

Color de fondo de un lienzo

Esta propiedad funciona de la misma forma que en las ventanas, permitiendo dar un color de fondo al lienzo donde se están pintando los elementos a visualizar.

El viewport

Determina la porción del lienzo que va a ser visible, es decir, aunque un lienzo tenga un tamaño concreto, el viewport puede llegar a ser inferior si queremos que sólo se muestre una parte del mismo.



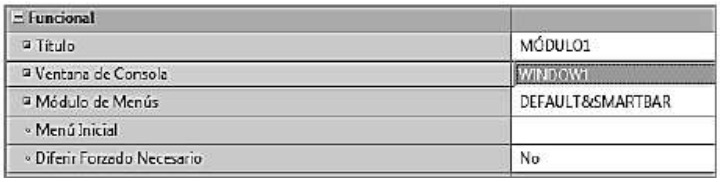
Cuando las coordenadas del viewport (x,y) tienen valor 0 (que es el valor por defecto), significa que el lienzo se muestra entero. En caso contrario, las coordenadas (x,y) indicarán la posición de comienzo de la visualización del viewport respecto a la esquina superior izquierda del lienzo.

PROPIEDADES BÁSICAS DE LOS MÓDULOS

Los módulos o formularios dentro de Forms Builder presentan las siguientes propiedades básicas:

- Ventana de Consola.
- Módulo de Menús.
- Sistema de Coordenadas.

Ventana de consola



Esta propiedad especifica cuál es la ventana por defecto que se asocia al módulo al abrirse. Además, implícitamente indica que se mostrará la barra de estado en el formulario (cuando se haya especificado un nombre de ventana).

En algunas aplicaciones resulta necesario que no aparezca la barra de estado, en cuyo caso no indicaremos ningún nombre de ventana en el módulo. Esto no es óbice para que al abrir el formulario se abra una ventana (aquella que tenga asociada el lienzo donde se muestran los elementos del primer bloque navegable).

Módulo de menús

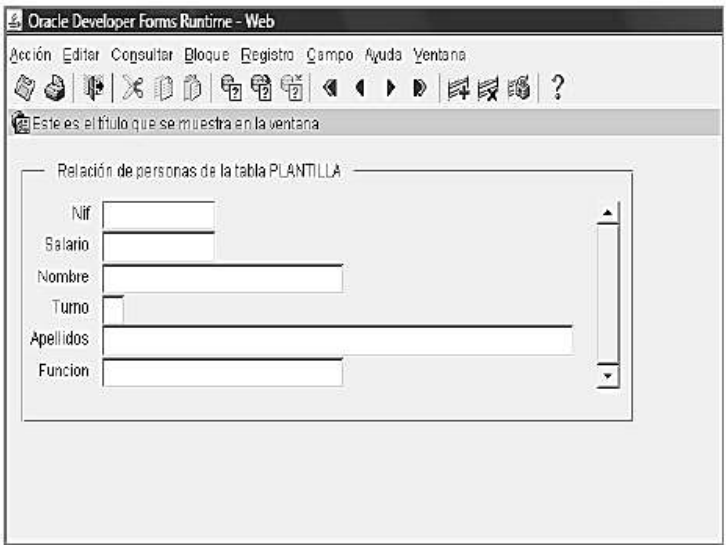
Funcional	
Título	MÓDULO1
Ventana de Consola	WINDOW1
Módulo de Menús	DEFAULT&SMARTBAR
Menú Inicial	
Diferir Forzado Necesario	No

Esta propiedad especifica cuál es el menú que queremos mostrar en nuestra ventana de la aplicación.

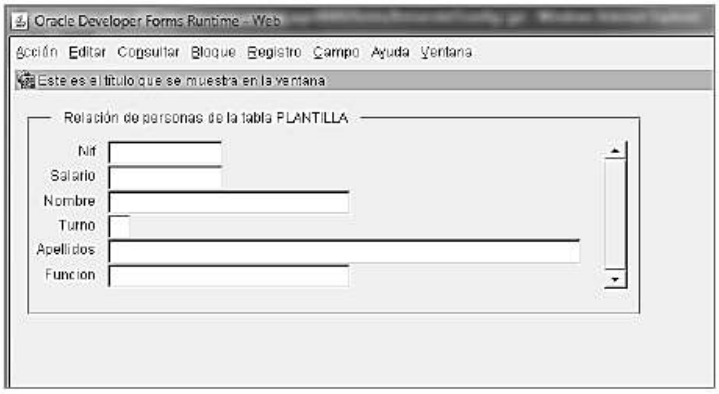
Por defecto todo nuevo formulario se le asocia como menú **DEFAULT&SMARTBAR** que significa que se mostrará el menú por defecto y la barra icónica de herramientas horizontal.

Pero además de esta opción, en esta propiedad se puede especificar lo siguiente:

- **DEFAULT&SMARTBAR**
- **DEFAULT** (solo muestra el menú por defecto).
- ***nombre_fichero.mmx*** (un módulo de menú compilado)
- Ningún valor.

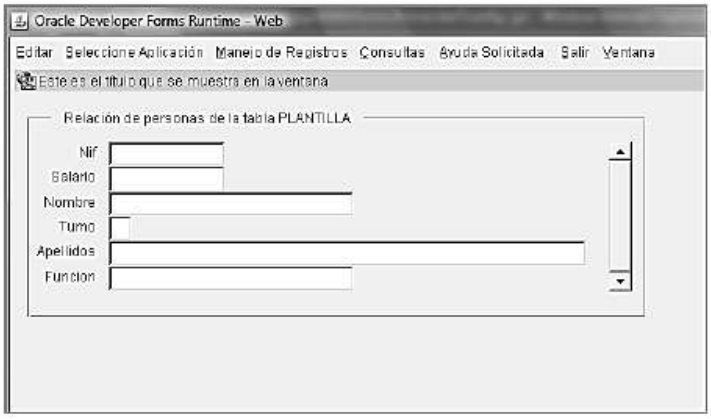


En esta pantalla se muestra un ejemplo de ejecución de un formulario con la propiedad *Módulo de Menús* con valor DEFAULT&SMARTBAR.

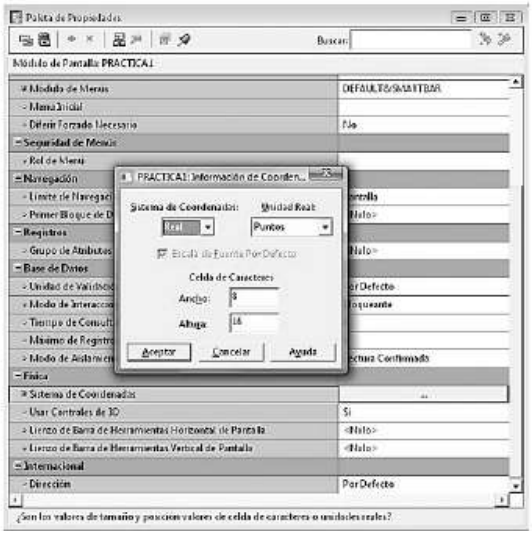


Si cambiamos en el mismo ejemplo la propiedad *Módulo de Menús* y ahora le asociamos el valor DEFAULT, el resultado de ejecutar el formulario sería el siguiente.

Si volvemos a cambiar el mismo ejemplo la propiedad *Módulo de Menús* y ahora le asociamos el valor *menú_personalizado.mmx*, que indicaría que queremos visualizar un menú que se ha diseñado con Forms Builder, y que se almacenó con el nombre *menú_personalizado.mmx*, el resultado de ejecutar el formulario sería el siguiente.



Sistema de coordenadas



Esta propiedad especifica la unidad de medida que se utilizará para el diseño del formulario. Afecta principalmente a los dimensionamientos (ancho y alto) de los elementos o al posicionamiento dentro de un lienzo (x e y).

Disponemos de dos tipos de sistemas de coordenadas para elegir el diseño del formulario:

- Modo **Carácter**.
- Modo **Real.**

En el modo carácter no se puede indicar ningún tipo de unidad de medida y en el modo real se pueden especificar los siguientes:

- Píxeles.
- Centímetros.
- Pulgadas.
- Puntos.
- Coma decimal.

PROPIEDADES BÁSICAS DE LOS BLOQUES

Los bloques dentro de Forms Builder presentan las siguientes propiedades básicas que se agrupan en cinco categorías:

- Navegación.
- Registros.
- Base de datos.
- Barra de desplazamiento.
- Color.

Sección navegación

En esta sección de las propiedades de un bloque se determina la forma en la que se va a navegar dentro del bloque y a saltar desde un bloque a otro.

Incluye las siguientes propiedades:

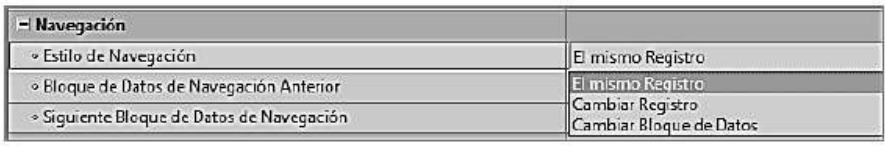
- Estilo de navegación.
- Bloque de datos de navegación anterior.
- Siguiente bloque de datos de navegación.

ESTILO DE NAVEGACIÓN

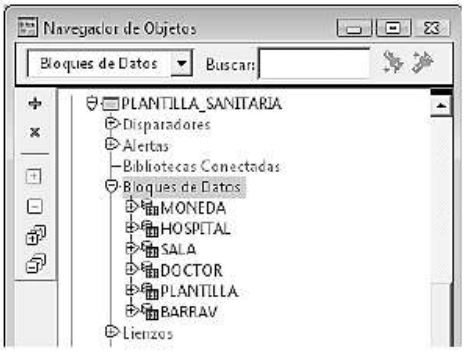
Normalmente cuando se navega después del último elemento en un registro, Forms devuelve el cursor al primer elemento del mismo registro. Con esta propiedad se puede cambiar este comportamiento para navegar al siguiente registro o bloque de datos. Los valores válidos son:

- **El mismo registro** (valor por defecto). En este caso cuando se llegue al último elemento del registro, el cursor se posicionará otra vez en el primer elemento del mismo registro.

- **Cambiar registro**. En este caso cuando se llegue al último elemento del registro, el cursor saltará automáticamente el primer elemento del siguiente registro.
- **Cambiar Bloque de Datos**. En este caso cuando se llegue al último elemento del registro, el cursor saltará automáticamente al primer elemento navegable del siguiente bloque.



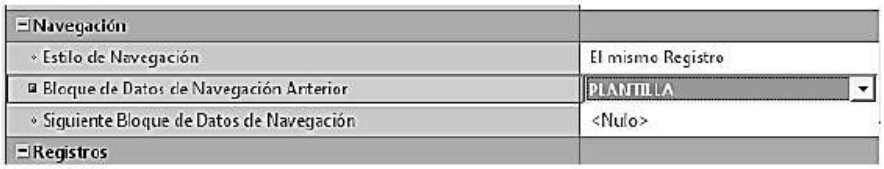
BLOQUE DE DATOS DE NAVEGACIÓN ANTERIOR



La navegación por defecto dentro de los bloques de un formulario se realiza en el orden en el que están creados dentro del navegador de objetos.

Si tomamos el ejemplo que se visualiza aquí, la navegación entre bloques por defecto sería la siguiente: MONEDA – HOSPITAL – SALA – DOCTOR – PLANTILLA - BARRAV – MONEDA. Donde el bloque anterior y posterior de navegación del bloque HOSPITAL sería MONEDA y SALA, respectivamente.

La propiedad **Bloque de Datos de Navegación Anterior** permite modificar esta navegación por defecto para indicar cuál es el bloque anterior al que se quiere navegar desde el actual.



SIGUIENTE BLOQUE DE DATOS DE NAVEGACIÓN

Es la misma propiedad que la anterior, pero en este caso en vez de indicar el bloque de navegación anterior se refiere al siguiente bloque al que se navegará después de salir del actual.

Si no se indica nada en la propiedad (*<Nulo>*), por defecto el bloque siguiente al que se navegará será el que aparezca en orden en el árbol del navegador de objetos.

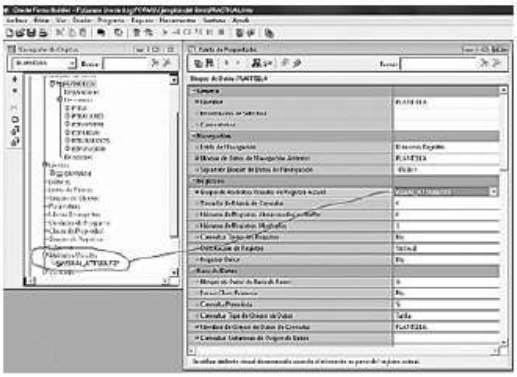
Sección registros

En esta sección de las propiedades de un bloque se determinan las características de visualización de los registros de un bloque: número de registros, distancia entre ellos, atributos visuales, etc.

Incluye las siguientes propiedades:

- Grupo de atributos visuales de registro actual.
- Tamaño de matriz de consulta.
- Número de registros almacenados en buffer.
- Número de registros mostrados.
- Consultar todos los registros.
- Orientación de registro.

GRUPO DE ATRIBUTOS VISUALES DE REGISTRO ACTUAL



Dentro de los registros que se visualizan en un bloque es posible resaltar aquel en el que se encuentra el cursor en tiempo de ejecución. Para ello hay que seleccionar uno de los grupos de atributos visuales que se hayan definido en el formulario y que aparezcan en la lista.

TAMAÑO DE MATRIZ DE CONSULTA

Especifica el máximo número de registros que Forms debe recuperar a la vez de la base de datos.

Cuanto más pequeño es el valor de esta propiedad se consigue un tiempo de respuesta más rápido en la recuperación de los registros; sin embargo, un valor mayor, supone menos llamadas a la base de datos para obtener registros, lo que reduce el tiempo de procesamiento global del formulario.

Esta propiedad asume por defecto el valor de la propiedad **Número de Registros Mostrados** cuando se indica un valor 0 para la misma.

NÚMERO DE REGISTROS ALMACENADOS EN BUFFER

Especifica el número de registros almacenados en la memoria del sistema durante una consulta de un bloque.

El valor por defecto es 0. En este caso se asume el valor del número de registros mostrados + una constante de 3.

NÚMERO DE REGISTROS MOSTRADOS

Es el número de registros que se quieren mostrar a la vez en el bloque de datos.

CONSULTAR TODOS LOS REGISTROS

Especifica si todos los registros equivalentes al criterio de búsqueda se deberían recuperar en el bloque de datos cuando se ejecuta la consulta en tiempo de ejecución del formulario.

Si se indica el valor SÍ todos los registros equivalentes a la consulta serán recuperados en el bloque. En cambio si se indica el valor NO se recuperará el número de registros indicados en la propiedad *Tamaño de Matriz de Consulta*.

ORIENTACIÓN DE REGISTROS

Determina la orientación de los registros en el bloque: horizontal o vertical. Cuando se asigna un valor a esta propiedad, automáticamente Forms Builder reajusta la disposición visual de los campos del registro en el bloque, de acuerdo a la orientación seleccionada.

Sección base de datos

En esta sección de las propiedades de un bloque se determinan las características de acceso y consulta a la/s tablas de base de datos ligadas al bloque.

Incluye las siguientes propiedades básicas:

- Bloque de datos de base de datos.
- Forzar clave primaria.
- Consultar tipo origen de base de datos.
- Nombre de origen de datos de consulta.
- Cláusula WHERE.

- Cláusula ORDER BY.
- Inserción, actualización y supresión permitida.
- Solo actualizar columnas cambiadas.

BLOQUE DE DATOS DE BASE DATOS

Determina si el bloque de datos está basado en un objeto de base de datos (si la propiedad tiene un valor SÍ), o si es un bloque de control no basado en un objeto de base de datos (si la propiedad tiene un valor NO).

FORZAR CLAVE PRIMARIA

Determina si Forms Builder se va a encargar de comprobar la unicidad de un registro antes de insertarlo o actualizarlo en la tabla de la base de datos, para evitar tener filas duplicadas en la misma (en cuyo caso se indicará *SÍ* en la propiedad).

CONSULTAR TIPO ORIGEN DE BASE DE DATOS

Determina el tipo de origen desde el que se va a consultar la información del bloque de datos.

Los valores posibles de esta propiedad son:

- **Ninguno**. Cuando se trata de un bloque de control sin acceso a base de datos.
- **Tabla** (valor por defecto). Cuando se trata de un bloque de datos basado en una tabla de la base de datos.
- **Procedimiento**. Cuando se trata de un bloque de datos basado en un procedimiento de base de datos que devuelve un recordset.
- **Disparadores de transacciones**. Cuando se trata de un bloque de datos basado en un disparador (trigger).
- **Consulta de cláusula FROM.** Cuando se trata de un bloque de datos basado en una consulta.

NOMBRE DE ORIGEN DE DATOS DE CONSULTA

Especifica el nombre del origen de datos de la consulta en la que se basa el bloque de datos. Esta propiedad solo se utiliza si el tipo de origen de datos de consulta es *Tabla, Consulta de cláusula FROM* o *Procedimiento*.

CLÁUSULA WHERE

Especifica una condición/filtro SQL que se va a ejecutar en la consulta de recuperación de información sobre el bloque de datos.

CLÁUSULA ORDER BY

Especifica un método de ordenación por defecto para los registros mostrados a partir de una consulta.

INSERCIÓN, ACTUALIZACIÓN Y SUPRESIÓN PERMITIDA

Controla si se pueden realizar las operaciones de inserción, actualización o borrado de registros a través del bloque de datos.

SOLO ACTUALIZAR COLUMNAS CAMBIADAS

Cuando esta propiedad se define a valor *SÍ*, solo los datos que se hayan actualizado en la ejecución del formulario, utilizando el bloque en cuestión, serán los que se escriban en las columnas de base de datos asociadas al bloque. Se puede ahorrar tráfico de red si el usuario que utiliza el formulario actualiza o inserta registros normalmente con solo una o dos columnas actualizadas. Por defecto este valor de la propiedad está definido a *NO*, por tanto cuando se realiza una grabación de la información del bloque en la tabla, se produce un UPDATE de todas las columnas con independencia de que se haya modificado o no la información en el bloque.

Sección barra de desplazamiento

En esta sección de las propiedades de un bloque se determinan las características para mostrar o no una barra de desplazamiento asociada al bloque para podernos mover entre sus diversos registros.

Incluye las siguientes propiedades básicas:

- Mostrar barra de desplazamiento.
- Lienzo de barra de desplazamiento.
- Posición X, Y de barra de desplazamiento.
- Ancho y longitud de barra de desplazamiento.

MOSTRAR BARRA DE DESPLAZAMIENTO

Especifica si Forms Builder debe crear una barra de desplazamiento para el bloque de datos correspondiente (en cuyo caso se indicará *SI* como valor de la propiedad), o bien si no se quiere mostrar la misma se indicará un valor *No.*

LIENZO DE BARRA DE DESPLAZAMIENTO

Especifica el nombre del lienzo en el que se mostrará la barra de desplazamiento del bloque.

POSICIONES X E Y DE BARRA DE DESPLAZAMIENTO

Especifica la posición en pantalla mediante las coordenadas X e Y en la que se mostrará la barra de desplazamiento en el lienzo. El valor por defecto para ambas coordenadas es 0.

ANCHO Y LONGITUD DE BARRA DE DESPLAZAMIENTO

Especifica el ancho y la altura de la barra de desplazamiento cuando se visualiza en el lienzo.

Sección color

En esta sección se controla el colorido del primer plano (fuentes) y fondo del lienzo donde se ubica el bloque.

Incluye las siguientes propiedades básicas:

- Color de Primer Plano.
- Color de Fondo.

COLOR DE PRIMER PLANO

Especifica el color que se va utilizar en el primer plano de los elementos del bloque. El primer plano corresponde con el color de las fuentes.

COLOR DE FONDO

Especifica el color del fondo del marco del bloque que se posiciona en el lienzo.

MANEJO DE VARIAS PALETAS DE PROPIEDADES

Puede ser que durante el diseño de un formulario necesite visualizar las propiedades de un objeto con varias paletas, o bien quiera comparar las propiedades de 2 objetos en paletas de propiedades diferentes que se visualicen a la vez. Ambas opciones están soportadas en Forms Builder.

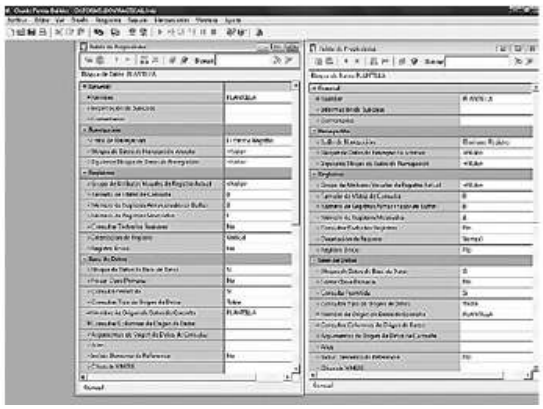
Vamos a diferenciar los dos métodos de visualización múltiple de paletas de propiedades para determinar cómo realizar cada uno de los pasos necesarios en cada caso:

- Visualizar más de una paleta de propiedades para el mismo objeto.
- Visualizar más de una paleta de propiedades para más de un objeto diferente.

Visualizar más de una paleta de propiedades para el mismo objeto

Para mostrar las propiedades de un objeto en varias paletas de propiedades realice los siguientes pasos:

- Abra una paleta de propiedades para el objeto en cuestión. Desde el menú **Herramientas**, seleccione la opción **Paleta de Propiedades**.
- Minimice o reajuste la ventana de propiedades para que no ocupe toda la pantalla.
- Pulse la tecla *[Shift]* y haga clic dos veces en el icono del objeto en el navegador de objetos, para el que tiene ya visualizada sus propiedades.
- Minimice o reajuste la nueva ventana de propiedades para que no ocupe toda la pantalla.

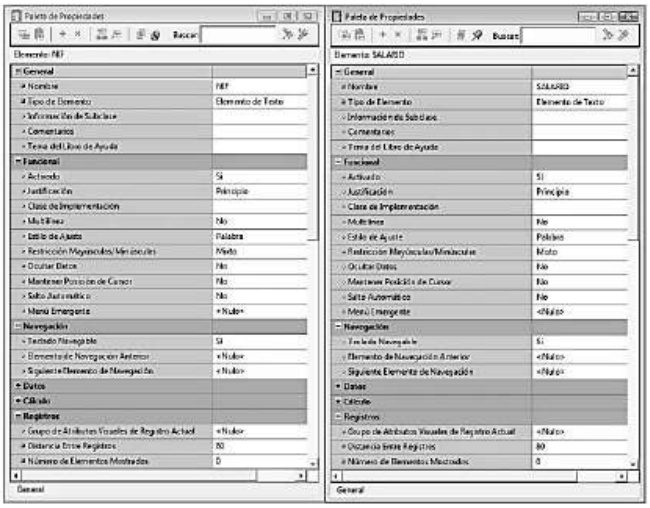


Tenga en cuenta que cuando si vuelve a abrir por segunda vez una paleta de propiedades del mismo objeto, esta se posiciona encima de la anterior, por lo que deberá desplazar de posición una de ellas para ver ambas a la vez. En esta imagen se puede ver una posible distribución en pantalla de 2 paletas de propiedades abiertas para el mismo elemento.

Visualizar más de una paleta de propiedades para más de un objeto diferente

Para mostrar las propiedades de más de un objeto diferente simultáneamente realice los siguientes pasos:

- Abra una paleta de propiedades para el primer objeto. Desde el menú **Herramientas**, seleccione la opción **Paleta de Propiedades**.
- Pulse sobre el icono  **Adjuntar.**
- Minimice o reajuste la ventana de propiedades para que no ocupe toda la pantalla.
- Vuelva al navegador de objetos y seleccione el segundo objeto para el que quiere ver sus propiedades, y repita el primer paso para abrir la paleta de propiedades.
- Minimice o reajuste la nueva ventana de propiedades para que no ocupe toda la pantalla.



En esta imagen se puede ver una posible distribución en pantalla de 2 paletas de propiedades abiertas para objetos diferentes.

DEFINICIÓN DE PROPIEDADES PARA VARIOS OBJETOS SIMULTÁNEAMENTE

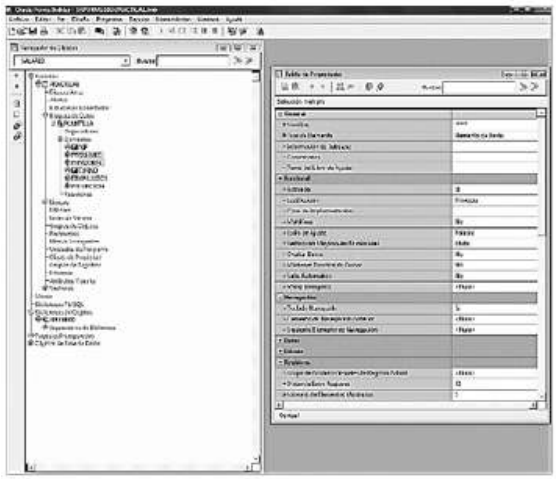
Puede ver y modificar las propiedades de varios objetos simultáneamente tanto si son objetos del mismo tipo como si son de diferente tipo sin necesidad de mostrar más de una paleta de propiedades.

En primer lugar tiene que seleccionar los objetos en el **navegador de objetos** y a continuación mostrar una combinación de las propiedades en la **paleta de propiedades**. La combinación puede ser:

Intersección Muestra el juego de propiedades que son comunes a los objetos seleccionados.

Unión Muestra el juego de propiedades mezcla de ambos objetos. Las propiedades comunes y las que diferencian a los objetos.

Cuando hay valores diferentes para una propiedad en los objetos seleccionados, se mostrará ***** en el valor de la propiedad. Esto cambia a un valor definitivo una vez que se introduce el nuevo valor en la paleta de propiedades. Este nuevo valor se aplicará a todos los objetos seleccionados para los que la propiedad sea compatible.



En esta imagen del Forms, se puede ver cómo se han seleccionado 2 objetos del navegador para mostrar en una misma paleta sus propiedades con la combinación unión.

COPIAR PROPIEDADES

Puede copiar las propiedades y valores de la paleta de propiedades a un buffer de memoria, de forma que se puede aplicar (pegar) a otros objetos en la sesión de diseño que está abierta.

Para copiar las propiedades realice los siguientes pasos:

- Abra una paleta de propiedades para el objeto. Desde el menú **Herramientas**, seleccione la opción **Paleta de Propiedades**.
- Visualice y seleccione las propiedades que desee copiar:

  - Para copiar una única propiedad basta con tener seleccionada la misma.
  - Para copiar todos las propiedades del objeto seleccione en el menú **Editar** la opción **Seleccionar Todo**.
  - Para copiar varias propiedades de un objeto, debe mantener pulsada la tecla *[Ctrl]* a medida que va haciendo clic con el ratón en cada una de las propiedades.
- Haga clic en el icono de copia de las  propiedades.
- Cierre la paleta de propiedades del objeto.
- Desde el navegador de objetos seleccione un nuevo elemento y abra su paleta de propiedades.
- Haga clic con el ratón en cualquiera de sus propiedades.
- Haga clic en el icono de pegar las  propiedades. El objeto seleccionado recibe los valores de todas las propiedades copiadas que sean compatibles por el tipo de objeto.

RECORDATORIO:

Cuando se copian propiedades entre objetos, tanto si son del mismo tipo como si son diferentes, únicamente se copian aquellas propiedades compatibles con ambos objetos, desechándose el resto.

CREANDO UN FORMULARIO MAESTRO-DETALLE 9

INTRODUCCIÓN

En este capítulo aprenderemos de forma práctica a crear un formulario maestro-detalle.

Este tipo de formularios se crean cuando existen dos o más tablas vinculadas en la base de datos. Este vínculo puede existir mediante restricciones de integridad del tipo FOREIGN KEY, o mediante semántica del contexto de la información que almacenan que nos permita realizar la unión. Es decir, las tablas vinculadas por la semántica contienen columnas equivalentes con el mismo tipo y significado semántico.

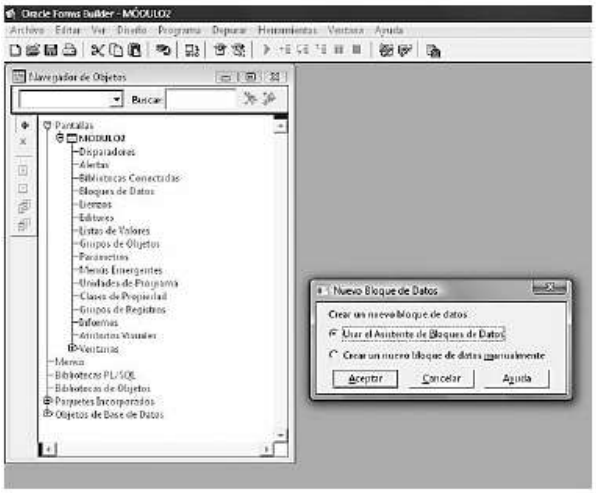
A continuación se van a describir los pasos necesarios para completar esta práctica guiada.

Apertura de Forms Builder

El primer paso a realizar es la apertura del diseñador de formularios Forms Builder. Una vez abierto, creamos un nuevo formulario desde el menú **Archivo**, seleccionando la opción **Nuevo** y a continuación **Pantalla**.

En esta imagen se muestran estos pasos.

Creación de un bloque maestro



Nos situamos dentro del navegador de objetos en Bloque de Datos y pulsamos sobre el botón + para añadir un nuevo bloque. Vamos a utilizar al asistente de bloque de datos para crearlo.

Selección del origen de datos

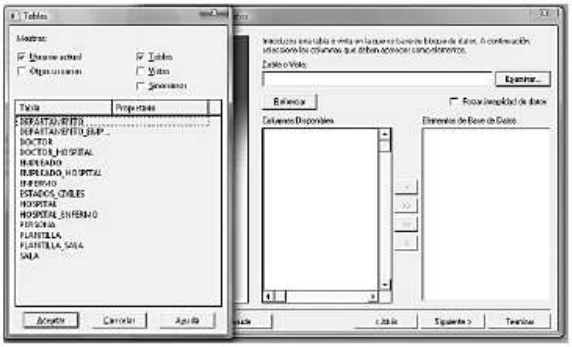
A continuación vamos a seleccionar como origen de datos para el bloque la opción **Tabla o Vista**.

Y pulsamos el botón **Siguiente**.

Conexión a una base de datos y selección de la tabla maestro

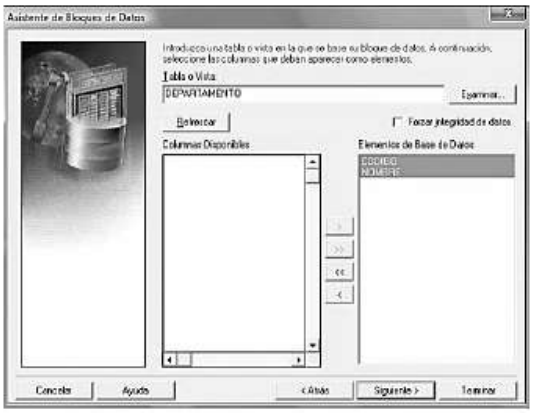
Para seleccionar la primera tabla en la que vamos a basar el formulario hay que pulsar el botón **Examinar** y a continuación el sistema nos solicita una conexión a la base de datos.

Las credenciales de conexión tienen que ser **PEPERF** (como usuario) y **PEPITO** (como contraseña). En el supuesto de haber instalado una base de datos local Oracle XE, en la casilla base de datos indicaremos **XE**, en cualquier otro caso, indicaremos el nombre de la conexión remota a otra base de datos que se haya definido en el fichero **tnsnames.ora**.



Pulsamos el botón **Conectar** y se nos presenta una lista con las tablas disponibles para el usuario **PEPERF**. De dicha lista seleccionamos la tabla *DEPARTAMENTO*.

Selección de columnas para el bloque

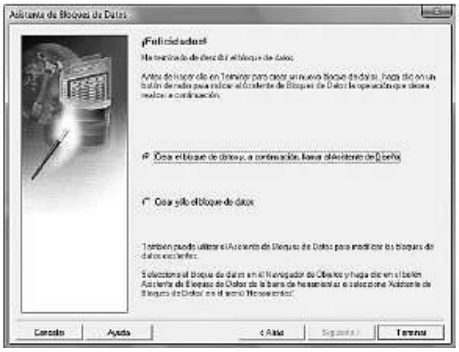


De la tabla *DEPARTAMENTO* vamos a seleccionar todas las columnas para tratarlas en el bloque de datos, para ello pulsamos sobre el botón **>>**.

En esta imagen se muestra el resultado que tenemos que obtener.

A continuación pulsamos el botón **Siguiente**.

Ejecución del asistente de diseño



Concluida la selección de columnas de la tabla que queremos tratar en el bloque, tenemos que ejecutar el asistente de diseño para dar formato a la visualización del bloque dentro del formulario. Para ello pulsaremos sobre la opción **Crear bloque de datos y a continuación, llamar al Asistente de Diseño.** A continuación se muestra una imagen de la pantalla que aparece en esta fase.

Pulsamos el botón **Terminar** para continuar.

Selección del tipo de lienzo

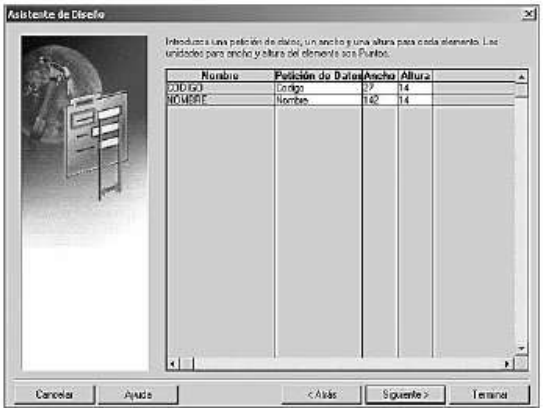
En esta fase seleccionamos el tipo de lienzo sobre el que queremos mostrar la información del bloque.

Seleccionamos como tipo de lienzo *Contenido* y a continuación pulsamos el botón **Siguiente** para continuar.

Selección de las columnas a pintar en el lienzo

A continuación tenemos que indicar las columnas del bloque que queremos visualizar en el lienzo. Al igual que ocurrió en el diseño del bloque vamos a visualizar todas las columnas del bloque en el lienzo y para ello pulsaremos sobre el botón **>>**. El resultado obtenido será el que se muestra en la siguiente imagen.

Propiedades de las columnas a pintar en el lienzo



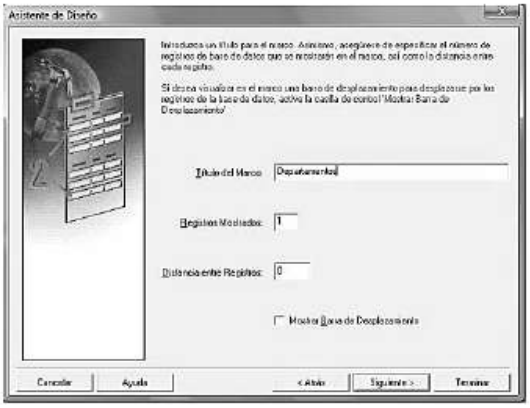
Una vez pulsado el botón **Siguiente** en la ventana anterior, tenemos que indicar las propiedades de visualización de cada una de las columnas, es decir: título a mostrar para cada una (*Petición de Datos*) y ancho/alto de las celdas sobre las que se mostrará o introducirá información.

Para continuar pulsamos el botón **Siguiente.**

Estilo de diseño

Definidas las propiedades de las columnas a visualizar, tenemos que indicar el formato de presentación de estas columnas en el lienzo. En la pantalla que se muestra a continuación pulsaremos sobre la opción **Pantalla**.

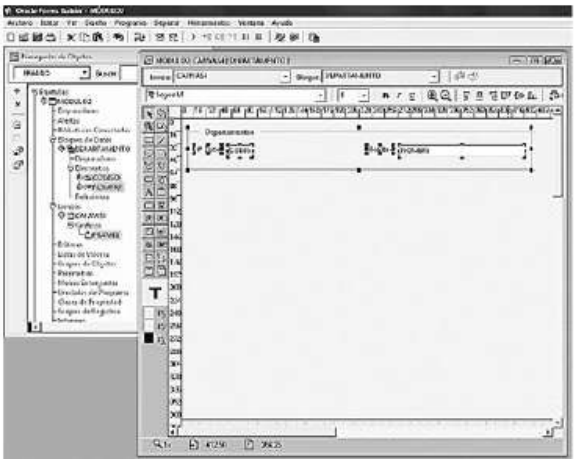
Título del marco del bloque y atributos de registro



A continuación tenemos que indicar el título que vamos a darle al marco que bordeará a los elementos del bloque, así como el número de registros (filas) que queremos visualizar simultáneamente en el mismo.

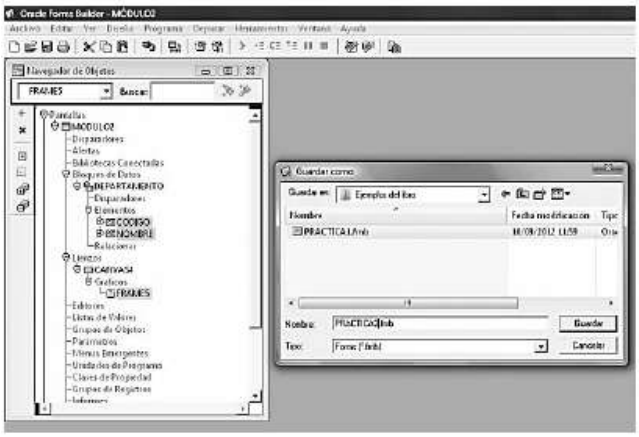
Una vez introducido como *Título del Marco* el valor **Departamentos** y como *Registros Mostrados* el valor **1**, pulsamos el botón **Siguiente** para continuar.

Finalizando la creación del bloque maestro



Al terminar el asistente de diseño nos aparecerá la ventana del editor con la representación en el tapiz de las columnas del bloque, como se muestra en la siguiente imagen.

Cerramos la ventana del editor de diseño y nos quedamos únicamente con la ventana del navegador de objetos abierta.



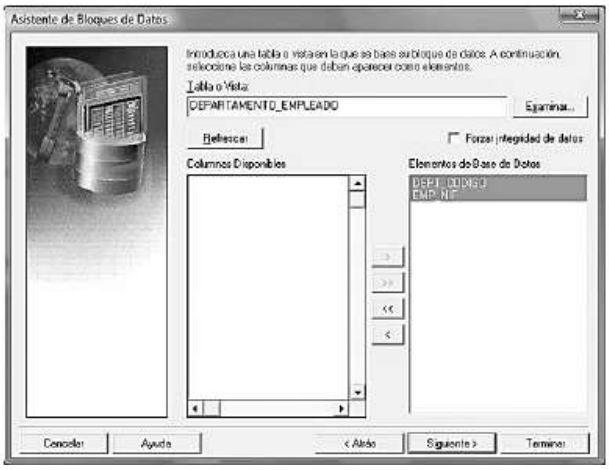
Sobre el menú pulsamos en **Archivo** y elegimos la opción **Guardar** para poder almacenar ya el formulario con el nombre *PRACTICA2.FMB* con el fin de no perder los cambios efectuados hasta este momento. En la página siguiente se muestra una imagen de este proceso.

Comenzar la creación del bloque detalle

En este punto vamos a comenzar la creación del bloque que actuará como detalle. Para ello nos situamos dentro del navegador de objetos en la sección **Bloque de Datos** y dentro de esta estructura marcamos el bloque DEPARTAMENTO. A continuación pulsamos sobre  para crear un nuevo bloque asistido.

A continuación pulsamos sobre el botón **Aceptar** para seguir el proceso.

Selección de columnas del bloque detalle



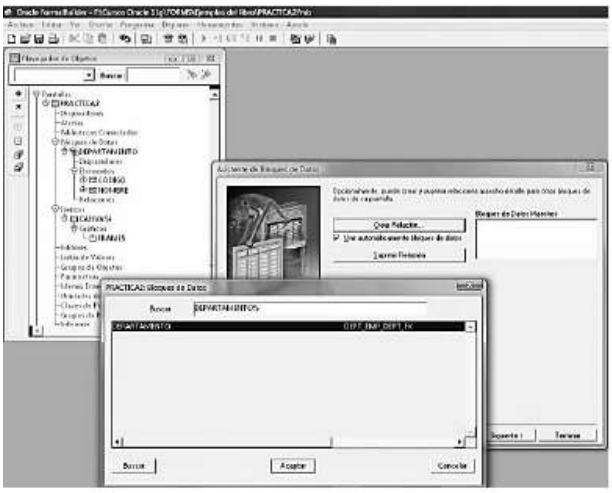
Como origen de datos para rellenar el bloque detalle volvemos a seleccionar **Tabla/Vista** y pulsamos el botón **Siguiente.**

A continuación pulsamos el botón **Examinar** para seleccionar la tabla *DEPARTAMENTO_EMPLEADO* como fuente de información del bloque y trasladamos todas sus columnas con el botón **>>** a la cuadrícula **Elementos de Base de Datos**. El resultado que se obtendrá será el que se muestra en la siguiente imagen.

Creación de la relación maestro-detalle

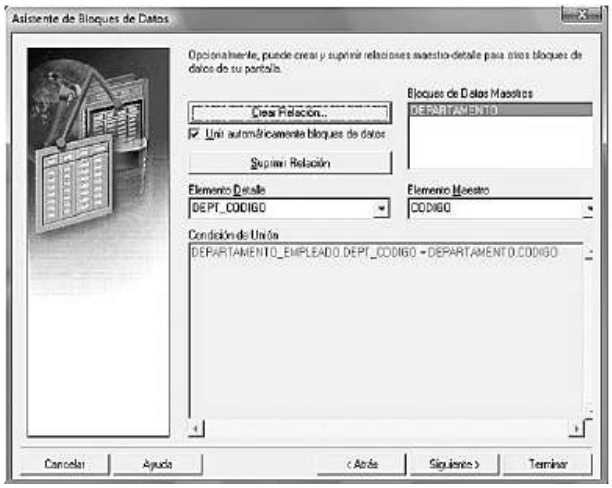
Cuando ya existe un bloque de datos y creamos uno nuevo, Forms asume que cualquiera de los creados puede ser un bloque maestro, mientras que el nuevo puede ser detalle. Si además existe una relación FOREIGN KEY entre ambas tablas, creada en la base de datos, podremos dejar marcada la opción **Unir automáticamente bloque de datos** y pulsar el botón **Crear relación**.

En el supuesto de que no exista una relación mediante el uso de FOREIGN KEY, también se puede crear el maestro y el detalle mediante una condición de unión entre columnas de ambas tablas. Para hacer esto último tendríamos que desmarcar la opción **Unir automáticamente bloque de datos**, y a continuación pulsaríamos el botón **Crear relación** para indicar que columnas de ambas tablas están relacionadas.



En nuestro caso práctico sí que existe una restricción de tipo FOREIGN KEY entre ambas tablas en la base de datos, por lo que marcamos la casilla **Unir automáticamente bloque de datos**, y pulsamos el botón **Crear relación** lo que hace que se presente la siguiente pantalla.

En esta pantalla Forms nos muestra la relación que existe entre la tabla detalle *DEPARTAMENTO_EMPLEADO* y la tabla maestro *DEPARTAMENTO* a nivel de restricción de base de datos. En nuestro caso práctico existe una restricción de tipo FOREIGN KEY, denominada *DEPT_EMP_DEPT_FK,* entre ambas tablas.

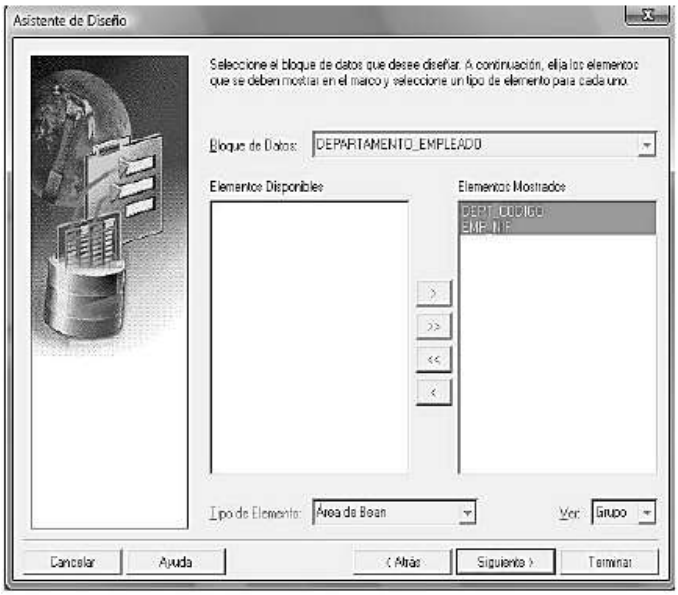


Pulsamos el botón **Aceptar** para continuar, y se nos muestra esta pantalla donde Forms rellena automáticamente la información de los campos que unen a ambas tablas, utilizando la información de la base de datos almacenada en la restricción *DEPT_EMP_DEPT_FK.*

A continuación pulsamos el botón **Siguiente** para proseguir con el proceso de diseño.

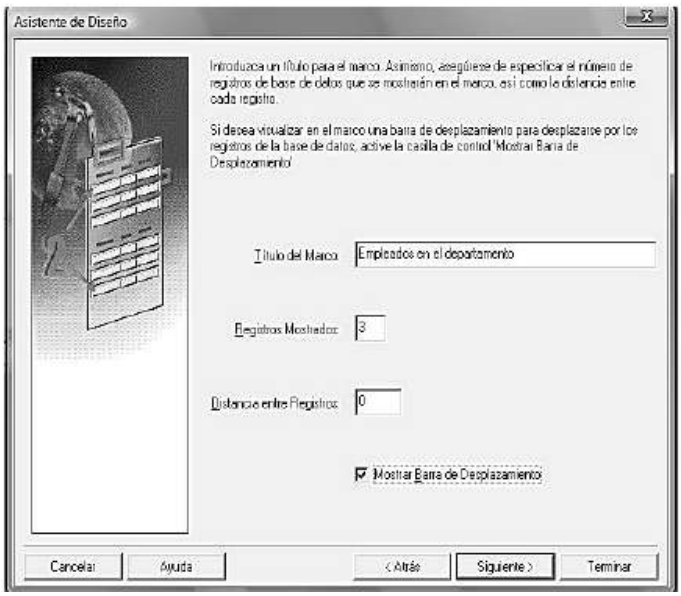
Diseño de los campos del bloque detalle en el lienzo

Finalizado el asistente de bloque de datos, seleccionamos la opción del asistente de diseño para pintar las columnas del detalle en el lienzo.



Aceptamos el nombre del lienzo propuesto para pintar el detalle (que es el mismo que el maestro), y marcamos para visualizar en el lienzo todas las columnas del bloque, lo que nos llevará a una pantalla como la siguiente:

A continuación pulsamos **Siguiente** (dos veces) para aceptar también las propiedades de título y tamaño de las celdas que nos propone Forms, hasta llegar a la pantalla donde tenemos que seleccionar el tipo de diseño del marco, en cuyo caso elegiremos **Pantalla** y volveremos a pulsar el botón **Siguiente** para llegar hasta la pantalla del asistente, donde nos solicita que introduzca el título para el marco que bordeará los elementos del bloque de detalle, así como el número de registros que queremos visualizar en dicho bloque.



En nuestro caso práctico, debemos introducir los valores que se indican en esta pantalla.

Pulsamos el botón **Siguiente** en esta pantalla y **Terminar** en la siguiente para concluir el proceso de diseño asistido.

Ajustes manuales de las propiedades del detalle

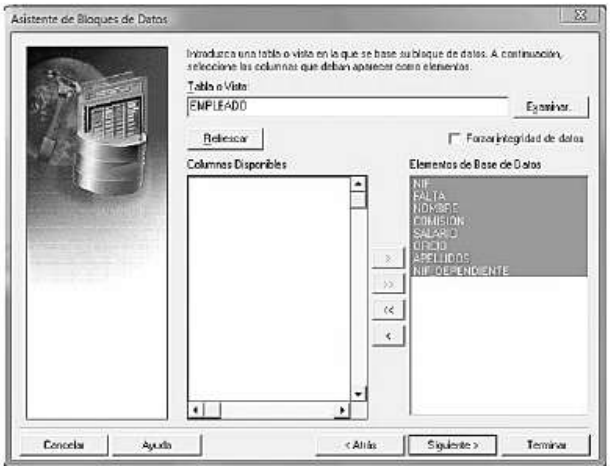
Finalizado el asistente de diseño se nos presenta el editor del lienzo donde tenemos marcados todos los elementos del bloque "*Empleados en el departamento*". Pulsamos sobre una zona del editor libre para que se desmarquen los elementos y a continuación manteniendo la tecla CTRL pulsada seleccionamos la columna *Dept Codigo y Emp Nif.* A continuación vamos a editar sus propiedades seleccionando el menú **Herramientas** y a continuación la opción **Paleta de Propiedades**.

Dentro de la Paleta de Propiedades nos situamos en la sección **Petición de Datos** e indicamos los siguientes valores para las propiedades de las columnas seleccionadas:

- Justificación de Petición de Datos: *Centro*
- Borde de Anexo de Petición de Datos: *Arriba*
- Alineamiento de Petición de Datos: *Centro*

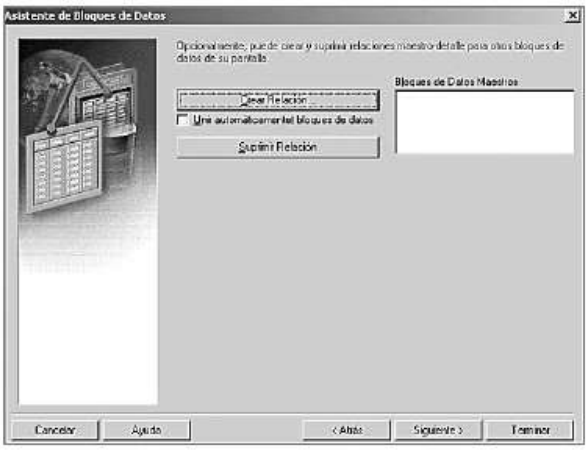
Creación de un nuevo bloque de detalle

A continuación vamos a crear un nuevo bloque de detalle. Para ello repetiremos los pasos realizados en la creación del bloque de detalle anterior pero en este caso seleccionaremos como tabla origen del bloque, la tabla EMPLEADO en la que seleccionaremos todas sus columnas para tratarlas dentro del bloque.



El resultado que tenemos que obtener hasta llegar a este punto es el que se muestra en esta imagen.

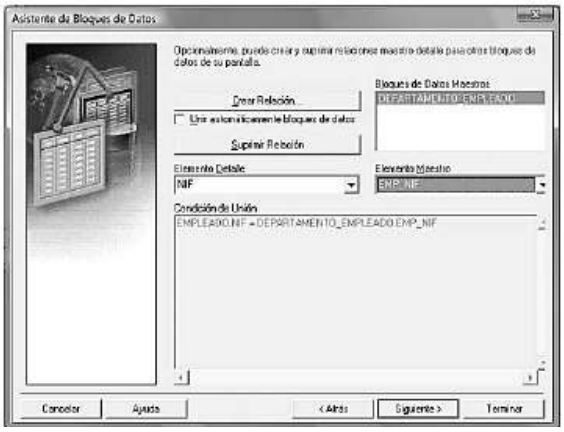
Definiendo una relación maestro-detalle manualmente



En este punto vamos a crear una relación manual maestro-detalle mediante una condición de unión entre la columna **NIF** de la tabla **EMPLEADO** y la columna **EMP_NIF** de la tabla **DEPARTAMENTO_EMPLEADO**, para ello pulsaremos el botón **Siguiente** en la ventana anterior y a continuación desmarcaremos la opción **Unir automáticamente los bloques de datos** en la siguiente imagen:

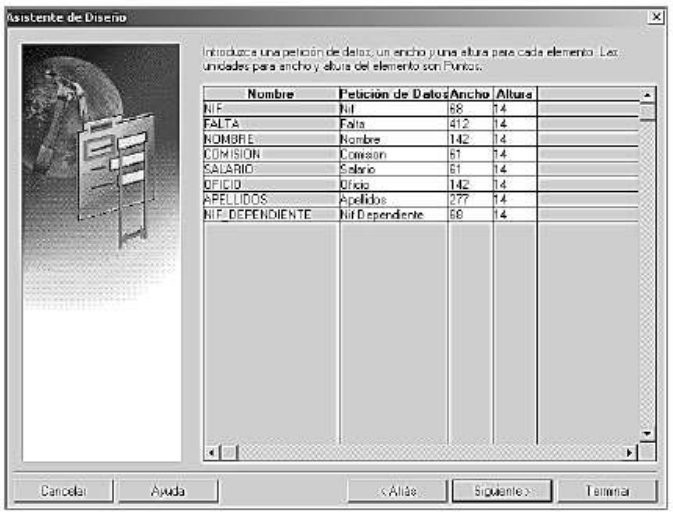
A continuación pulsaremos el botón **Crear Relación** para obtener el cuadro de diálogo que se indica a continuación.

Seguidamente marcaremos sobre la opción **Basada en una condición de unión** y pulsaremos sobre el botón **Aceptar,** lo que nos presentará una pantalla para que seleccionemos la tabla que actuará como maestra.

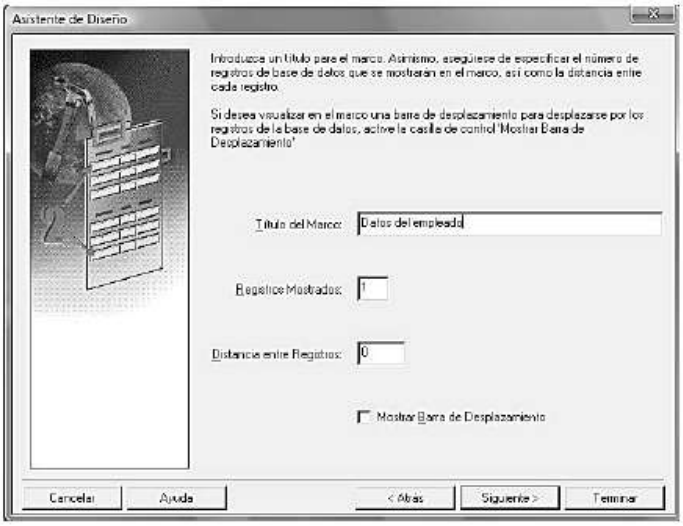


Seleccionaremos la tabla *DEPARTAMENTO_EMPLEADO* y pulsaremos sobre el botón **Aceptar**, lo que nos devolverá al cuadro de diálogo para la creación de una relación donde tendremos que indicar qué columna forma el nexo de unión (relación) entre el bloque detalle (en el que nos encontramos actualmente) y el bloque maestro seleccionado. Para continuar el proceso pulsaremos el botón **Siguiente** y aceptaremos la llamada al asistente de diseño para pintar los elementos del nuevo bloque en el lienzo.

Dando formato al bloque en el lienzo



Dentro del editor de diseño vamos a seleccionar todos los campos para visualizar en el mismo lienzo que se han pintado el resto de bloques, aceptaremos como Petición de Datos y tamaño de los campos los que nos ofrece Forms por defecto.



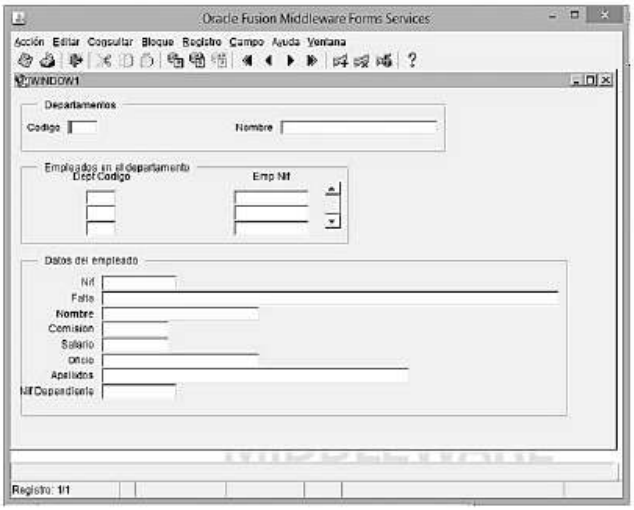
A continuación pulsaremos el botón **Siguiente** y seleccionaremos como formato de presentación de los campos **Pantalla,** para pasar al cuadro de diálogo donde tenemos que definir el título del marco que recubre al bloque y el número de registros que vamos a presentar en el mismo, tal cual se indica en la imagen de la página siguiente.

Finalizando y guardando el formulario

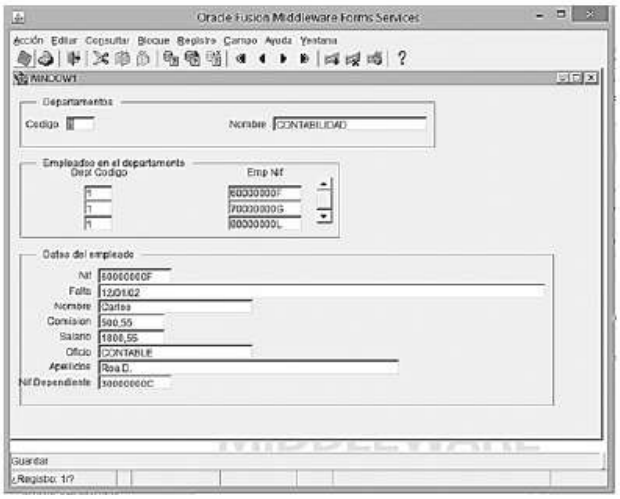
Completamos la creación del formulario pulsando **Siguiente** y después **Terminar** en las ventanas que aparecen. Una vez que Forms nos devuelve el Editor de Diseño con el formato de presentación de todos los elementos del formulario vamos a proceder a guardar todos los cambios efectuados. Para ello pulsamos sobre el menú **Archivo** y a continuación pulsamos en **Guardar.** Como el formulario ya tenía nombre al haberse guardado en los pasos anteriores, únicamente se actualiza el contenido del fichero con la nueva información.

Compilando y ejecutando el formulario

A continuación vamos a compilar todo el formulario para comprobar que no hay ningún error y obtener el fichero ejecutable **PRACTICA2.FMX** necesario antes de que procedamos a ejecutarlo.

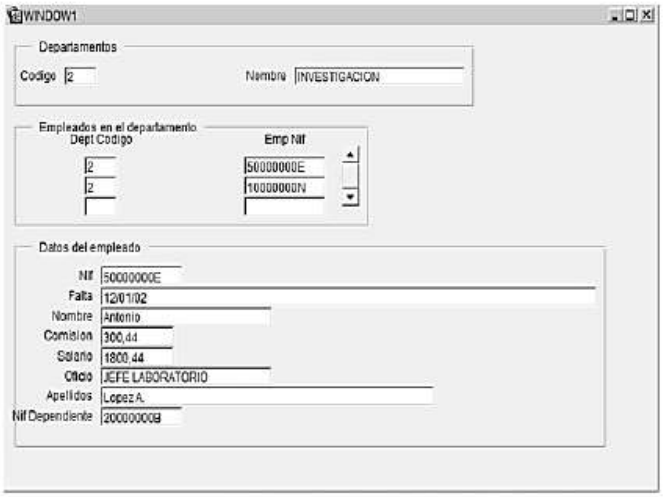


En primer lugar compilaremos todo el formulario y para ello pulsaremos sobre el menú **Programa** y a continuación sobre la opción **Compilar PL/SQL** y dentro de esta la opción **Todo.** A continuación pasaremos a generar el fichero ejecutable **PRACTICA2.FMX** y para ello pulsaremos sobre el menú **Programa** y a continuación sobre la opción **Compilar Módulo**. Terminados los puntos anteriores el formulario ya se puede ejecutar y para ello pulsaremos sobre el menú **Programa** y a continuación sobre la opción **Ejecutar pantalla,** lo que nos presentará esta imagen.



A continuación vamos a realizar unas pruebas de funcionamiento del formulario para ver cómo se comportan los distintos bloques detalle con respecto a su maestro. Para ello nos situamos en el primer bloque (*Departamentos)* sobre el campo *Codigo* y en el menú **Consultar** pulsamos la opción **Ejecutar**, lo que nos devolverá esta pantalla.

Una vez obtenida esta información vamos a navegar a un registro siguiente dentro del bloque *Departamentos* para ver cómo la información de los bloques detalle varía automáticamente dependiendo del valor que tiene el bloque maestro correspondiente.



Así que sin salir del campo *Codigo,* seleccionamos en el menú **Registro** la opción **Siguiente,** lo que nos mostrará la información que se aparece a continuación.

Si ahora nos posicionamos con el cursor sobre el campo *Emp Nif* del bloque *Empleados en el departamento* y cambiamos entre los registros del mismo, por ejemplo saltamos del empleado con nif 10000000N al empleado con nif 50000000E, veremos que la información del bloque detalle *Datos del empleado* también varía automáticamente.

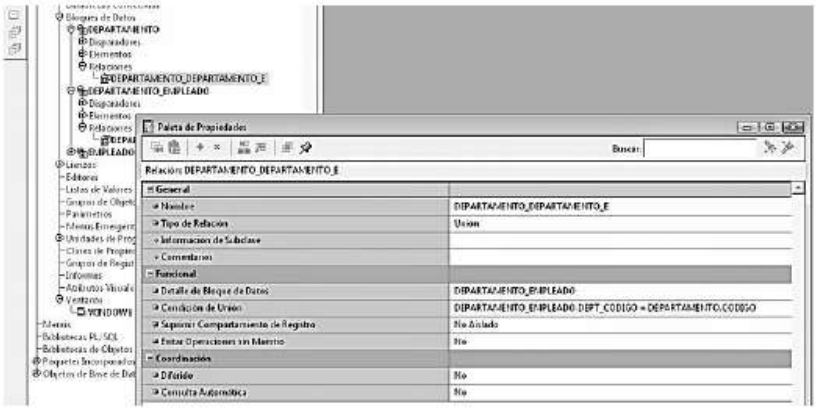
PROPIEDADES DEL NAVEGADOR DE OBJETOS LIGADAS A BLOQUES MAESTRO-DETALLE

Una vez terminadas las pruebas de funcionalidad del formulario, cerramos las ventanas abiertas de la ejecución del mismo y volvemos al Forms Builder dentro del navegador de objetos, para observar los nuevos elementos que se han creado como consecuencia de los bloques maestro-detalle y sus relaciones.

Distinguiremos entre los elementos que se han creado a nivel de los bloques maestro-detalle, y los elementos creados a nivel de unidades de programa.

Propiedades maestro-detalle a nivel de bloque

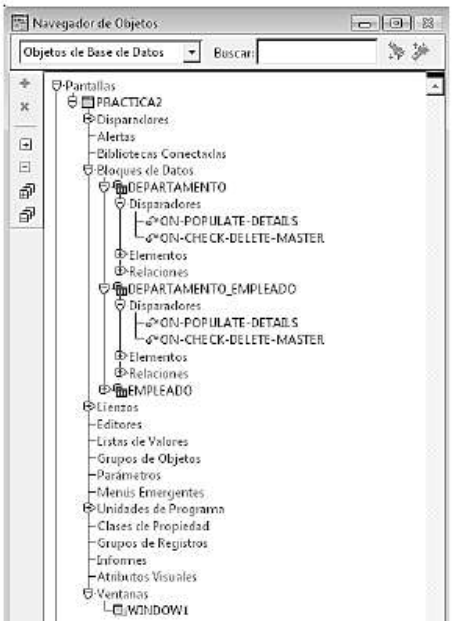
Continuando con el formulario **PRACTICA2.FMX** que hemos creado en este capítulo observamos a nivel de los dos bloques que actúan como maestro: *DEPARTAMENTO* y *DEPARTAMENTO_EMPLEADO* (este último también actúa como detalle), que se han creado objetos de tipo **Relación** donde se encuentran definidas las condiciones de unión entre las columnas de los bloques maestro y detalle.



Por ejemplo, si abrimos la paleta de propiedades de la relación *DEPARTAMENTO_DEPARTAMENTO_E,* asociada al bloque *DEPARTAMENTO*, obtendremos la siguiente pantalla. Para abrir la paleta de propiedades hay que hacer doble clic sobre el icono que se encuentra a la izquierda del nombre de la relación.

Dentro de las propiedades de una relación distinguiremos las siguientes como básicas:

- **Tipo de Relación**; que nos indica el método por el que se relacionan los bloques maestro y detalle: *Unión* o *Ref*. El método habitual es *Unión* cuando se realiza entre tablas de base de datos.
- **Detalle de Bloque de Datos**; indica el nombre del bloque del formulario que va a actuar como detalle del bloque maestro en el que nos encontramos.
- **Condición de Unión**; determina la/s columa/s que forman la relación entre los bloques maestro y detalle.

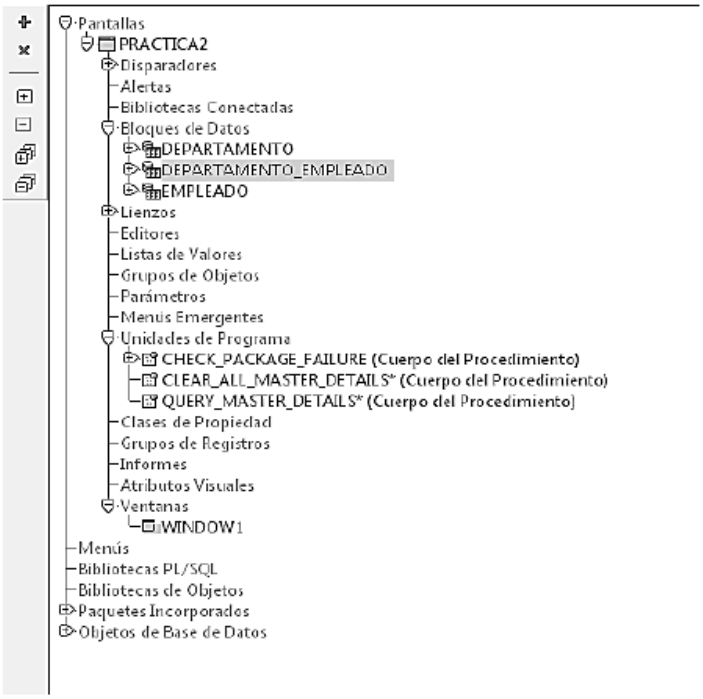


Además de los nuevos objetos de tipo relación, también se crean nuevos objetos de tipo **Disparadores** en el bloque maestro, como se puede observar en la imagen de la siguiente página.

El disparador *ON-POPULATE-DETAILS* de un bloque maestro, controla cómo se deben rellenar las columnas de la tabla detalle cuando se navega entre los elementos del maestro, para ello utiliza una programación en código PL/SQL. Se puede observar el código programado en nuestro caso práctico pulsando doble clic sobre el icono asociado a la izquierda del nombre del disparador.

El disparador *ON-CHECK-DELETE-MASTER* ejerce la misión de control referencial entre los datos de ambos bloques, para impedir, por ejemplo, el borrado de un elemento del maestro si tiene datos del detalle referenciados. Este control al igual que en el caso anterior se realiza mediante programación en código PL/SQL.

Propiedades maestro-detalle a nivel de unidad de programa



Continuando con el formulario **PRACTICA2.FMX,** que hemos creado en este capítulo, observamos que a nivel del objeto **Unidades de Programa** se crean los siguientes elementos como consecuencia de la creación de bloques maestro-detalle: *CHECK_PACKAGE_FAILURE, CLEAR_ALL_MASTER_DETAILS* y *QUERY_ALL_MASTER*_DETAILS.

El procedimiento *CHECK_PACKAGE_FAILURE* controla mediante programación PL/SQL si existe algún error dentro del formulario en tiempo de ejecución, en cuyo caso aborta la ejecución del mismo. Para abrir el código hay que pulsar doble clic en el icono que se encuentra a la izquierda del nombre del paquete.

El procedimiento *CLEAR_ALL_MASTER_DETAILS* se utiliza para limpiar de la pantalla del formulario los campos del bloque maestro y su bloque detalle correspondiente.

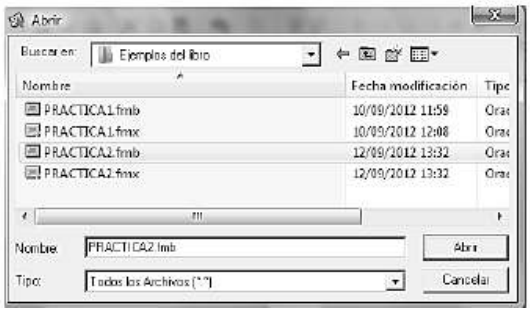
El procedimiento *QUERY_MASTER_DETAILS* se utiliza para lanzar a ejecución las consultas dentro de los bloques maestro-detalle y comprobar el correcto funcionamiento del formulario.

CREANDO UN BLOQUE MANUAL. COPIANDO PROPIEDADES 10

INTRODUCCIÓN

Hasta llegar a este capítulo, hemos aprendido a crear formularios utilizando siempre los asistentes que nos proporciona Forms, pero en algunas situaciones no podemos crear un bloque con los mismos. Por ejemplo, el caso típico es la necesidad de crear un bloque que no está enlazado a ninguna tabla de la base de datos, es decir, un bloque de control. En este capítulo aprenderemos de forma práctica a crear un bloque de forma manual y a copiar propiedades.

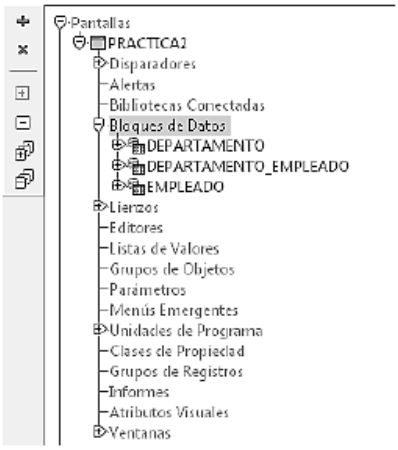
Apertura de un módulo existente



El primer paso a realizar es la apertura del diseñador de formularios Forms Builder. Una vez abierto vamos a abrir el formulario que hemos creado en el capítulo anterior. Para ello desde el menú **Archivo** seleccionamos la opción **Abrir** y se nos muestra un cuadro de diálogo donde debemos navegar dentro de nuestro sistema hasta la ruta en la que se encuentra almacenado el archivo *PRACTICA2.FMB.*

Una vez seleccionado el fichero *PRACTICA2.FMB* pulsamos el botón **Abrir**.

Desplegando elementos del navegador de objetos



Cuando abrimos un formulario, por defecto se muestra el navegador de objetos con todos los elementos del mismo contraídos. Para ver la relación de elementos de cualquiera de los objetos del mismo, hay que pulsar sobre el icono que se encuentra a la izquierda del nombre de cada objeto.

En nuestro caso práctico vamos a pulsar sobre el icono citado del objeto **Bloques de Datos,** para mostrar todos los elementos de los que dispone, lo que provocará que obtengamos la siguiente pantalla.

Crear un bloque manual

Para crear y añadir un nuevo bloque al formulario *PRACTICA2,* en primer lugar seleccionaremos dentro de la lista de bloques existentes el que se denomina *EMPLEADO,* pulsando un solo clic sobre el nombre del bloque. A continuación pulsaremos el símbolo en la barra vertical icónica. Al hacerlo nos aparece un nuevo cuadro de diálogo preguntándonos si queremos utilizar el asistente para crear un bloque o si lo vamos a crear manualmente.

En nuestro caso práctico marcaremos sobre la opción **Crear un nuevo bloque de datos manualmente** y a continuación pulsaremos sobre el botón **Aceptar**.

Copiar propiedades de otro bloque

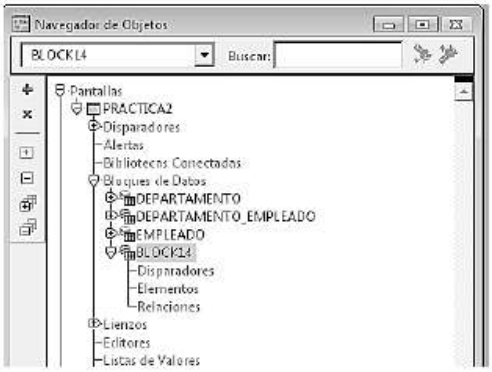
Para facilitar la creación del bloque de forma manual, vamos a aprovechar algunas propiedades de otro de los bloques ya creados. En este caso vamos a abrir la paleta de propiedades del bloque *DEPARTAMENTO.* Una vez seleccionado desde el menú **Herramientas,** pulsamos la opción **Paleta de Propiedades**.

Una vez abierta la paleta de propiedades del bloque *DEPARTAMENTO* seleccionamos todas las propiedades del bloque salvo el nombre del mismo, para ellos seguimos estos pasos:

- En el menú **Editar** pulsamos la opción **Seleccionar todo.**
- A continuación pulsamos la tecla *[CTRL]* y sin soltarla damos un solo clic con el ratón sobre la propiedad **Nombre** para quitarla de la selección.
- Por último, de nuevo en el menú **Editar,** pulsamos la opción **Copiar propiedades**.

Pegar las propiedades copiadas en el bloque manual

Después de copiar las propiedades, cerramos la paleta de propiedades del bloque *DEPARTAMENTO* y volvemos al navegador de objetos.



Dentro del navegador de objetos seleccionaremos el bloque que hemos creado manualmente en los pasos anteriores. Dicho bloque tendrá como nombre el que le asocia por defecto Forms Builder: *BLOCK* + un número secuencial.

A continuación abrimos la paleta de propiedades del bloque manual (desde el menú **Herramientas** pulsamos la opción **Paleta de Propiedades)**, para poder pegar las propiedades copiadas anteriormente. Para ello vamos a seguir estos pasos:

- Damos un clic con el ratón en cualquiera de las propiedades (como por ejemplo la propiedad *Nombre*).
- En el menú **Editar** pulsamos sobre la opción **Pegar propiedades.**

Guardando la práctica con un nombre diferente

Dado que la práctica la hemos comenzado abriendo un formulario ya existente (*PRACTICA2.FMB*), si no queremos que al guardar los cambios se sobrescriba la información de dicho formulario, tenemos que guardar esta práctica con otro nombre. Para ello vamos al menú **Archivo** y seleccionamos la opción **Guardar Como**. En el cuadro de diálogo que se presenta, escribimos dentro del apartado **Nombre,** el nuevo nombre del formulario: *PRACTICA3.FMB* y pulsamos sobre el botón **Guardar.**

Para mantener la coherencia entre el nombre del fichero y el nombre del módulo (formulario), nos situamos en el nombre del módulo (que seguirá teniendo

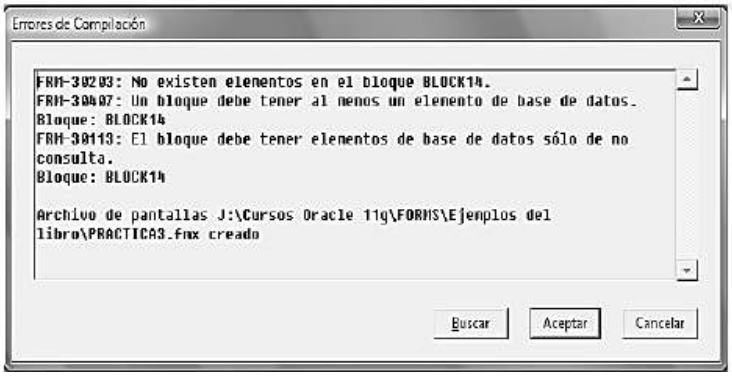
PRACTICA2) y abrimos la paleta de propiedades donde cambiaremos la propiedad **Nombre** para que ponga *PRACTICA3***.**

Conexión a B.D., compilación y visualización de errores

Una vez almacenado el módulo, lo vamos a compilar, pero antes de hacerlo tenemos que conectarnos a la base de datos, debido a que hay bloques de datos ligados a tablas de la misma. Para ello desde el menú **Archivo** elegimos la opción **Conectar** e introducimos las credenciales indicadas en los capítulos anteriores.

Realizada la conexión, podemos proceder a compilar el formulario, para ello desde el menú **Programa** seleccionaremos la opción **Compilar PL/SQL** y dentro de esta la opción **Todo**.

A continuación y dado que el formulario no tiene errores, procedemos a generar el fichero ejecutable **PRACTICA3.FMX**, para ello desde el menú **Programa** seleccionamos la opción **Compilar Módulo**. Al ejecutar esta fase Forms Builder nos devuelve el siguiente error:



Este error se produce porque hemos creado un nuevo bloque (el manual), que no contiene ningún elemento de navegación, de hecho no tiene ningún elemento. Este error no obstante no ha impedido la generación del fichero ejecutable *PRACTICA3.FMX,* porque este tipo de error se considera un *Warning* o advertencia. Si se hubiese producido un error en la fase de compilación, entonces no se podría generar el fichero ejecutable hasta solucionarlo.

CREANDO UNA RELACIÓN MANUAL ENTRE BLOQUES 11

INTRODUCCIÓN

De la misma forma que podemos crear bloques de forma asistida y manual, también podemos crear relaciones entre los mismos de forma asistida o manual. Hasta el momento solo habíamos creados relaciones utilizando el asistente. No obstante, en algunos casos se hace necesario crear una relación manual entre los bloques para provocar la sincronización de datos entre ellos.

En este capítulo aprenderemos de forma práctica a crear una relación manual.

Apertura de un módulo existente

El primer paso a realizar es la apertura del diseñador de formularios Forms Builder. Una vez abierto vamos a abrir el formulario *PRACTICA2.FMB* que hemos creado en capítulos anteriores. Para ello desde el menú **Archivo** seleccionamos la opción **Abrir,** y se nos muestra un cuadro de diálogo donde debemos navegar dentro de nuestro sistema hasta la ruta en la que se encuentra almacenado el archivo *PRACTICA2.FMB.*

Una vez seleccionado el fichero *PRACTICA2.FMB* pulsamos el botón **Abrir**.

Desplegando elementos del navegador de objetos

A continuación desplegamos los elementos del objeto **Bloques de Datos** y una vez realizada esta operación, desplegamos los elementos del bloque de datos **DEPARTAMENTO,** y por último desplegamos los elementos del objeto **Relaciones.**

Eliminando una relación

A continuación vamos a eliminar la relación *DEPARTAMENTO_DEPARTAMENTO_E* del bloque de datos *DEPARTAMENTO*, para ello seleccionamos la relación y pulsamos el icono  que aparece en la barra icónica vertical. Por último confirmamos la operación pulsando el botón **Sí.**

RECORDATORIO:

Al igual que se crean una serie de disparadores y unidades de programa cuando se crea una relación con los asistentes, también se eliminan estos objetos cuando se elimina la relación del formulario.

Guardando la práctica con un nombre diferente

Dado que la práctica la hemos comenzado abriendo un formulario ya existente (*PRACTICA2.FMB*), si no queremos que al guardar los cambios se sobrescriba la información de dicho formulario, tenemos que guardar esta práctica con otro nombre. Para ello vamos al menú **Archivo** y seleccionamos la opción **Guardar Como**. En el cuadro de diálogo que se presenta, escribimos dentro del apartado **Nombre** el nuevo nombre del formulario: *PRACTICA4.FMB* y pulsamos sobre el botón **Guardar.**

Igualmente cambiamos las propiedades del nombre del módulo para que también se denomine **PRACTICA4**.

Creando la relación de forma manual

A continuación vamos a crear la relación de forma manual, para ello nos situamos en el objeto **Relaciones** del bloque de datos *DEPARTAMENTO* y pulsamos el icono  que se encuentra en la barra icónica vertical, lo que provocará que se presente un cuadro de diálogo en el que se nos solicitan los datos de la relación, como se muestra en la siguiente imagen.

A continuación rellenamos las propiedades necesarias para crear la relación, de acuerdo a la definición que se da de cada una de ellas seguidamente:

- **Bloque Maestro;** es el bloque donde se está creando la relación manual y actúa como maestro de la relación.
- **Bloque Detalle;** es el bloque con el que se completa la relación y actúa como detalle de la misma. En nuestro caso práctico, seleccionaremos (pulsando el botón **Seleccionar**) como bloque detalle: *DEPARTAMENTO_EMPLEADO.*
- **Supresiones en Bloques Maestro;** esta propiedad controla el borrado en los elementos de la relación maestro-detalle. Hay que indicar una de las siguientes opciones:
  - ***En Cascada;*** indica que cuando se suprima un registro del bloque maestro, automáticamente se eliminarán todos los registros del bloque detalle cuyo valor de unión (clave de la relación) coincida.
  - ***Aislado;*** indica que solo se suprimirá el registro del bloque maestro aunque existan registros vinculados por la clave de unión en el bloque detalle.
  - ***No Aislado;*** indica que se impedirá borrar registros del bloque maestro cuando existan registros en el bloque detalle coincidentes por la clave de unión. En nuestro caso práctico, vamos a seleccionar esta opción.
- **Coordinación;** esta propiedad determina cómo se visualizarán los registros del bloque detalle cuando se consulta un registro en el bloque maestro, y existen datos coincidentes por la clave de unión en el bloque detalle. Hay que indicar una de las siguientes opciones:
  - ***Diferido con consulta automática;*** indica que se pospone el procesamiento y visualización de las consultas del bloque detalle

hasta que el cursor se posicione físicamente dentro de alguno de los campos visualizados para el bloque detalle.

- ***Diferido sin consulta automática;*** permite la introducción de criterios de consulta adicionales en los elementos del bloque detalle, antes de realizar la consulta.
- ***Consulta automática;*** es la opción por defecto y consiste en que cada vez que se consulta un nuevo registro en el bloque maestro se presenta automáticamente la información relacionada en el bloque detalle.
- ***Evitar operaciones sin maestro;*** esta opción asegura que los elementos del bloque detalle no se pueden consultar o utilizar para insertar registros en la base de datos, cuando no se muestra antes un registro equivalente (por clave de unión) en el bloque maestro.

En nuestro caso práctico vamos a marcar la opción *Consulta automática* y *Evitar operaciones sin maestro***.**

- **Condición de unión;** esta propiedad es donde se define el/los campo/s del bloque maestro y detalle que se encuentran relacionados. En nuestro caso práctico vamos a definir la siguiente condición de unión entre los bloques maestro y detalle: *DEPARTAMENTO.CODIGO = DEPARTAMENTO_EMPLEADO.DEPT_CODIGO*

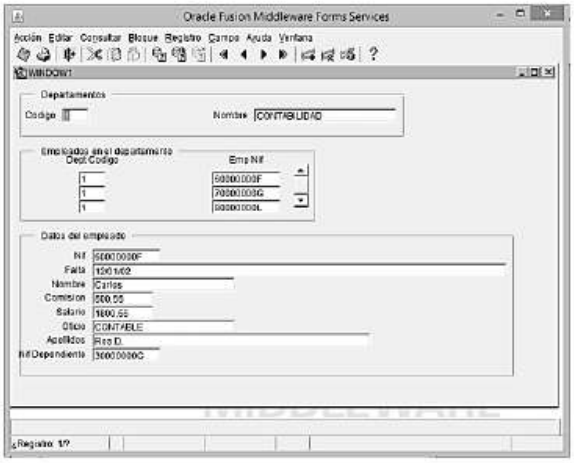
El resultado de la creación manual de la relación debería reflejar la información que se muestra en esta imagen.

Pulsamos el botón **Aceptar** para completar la creación de la relación y como vemos en el navegador de objetos (bloque *DEPARTAMENTO*) se vuelven a crear los disparadores que controlan la relación maestro-detalle.

Guardar, compilar, generar, ejecutar y probar el formulario

Para terminar guardamos los cambios del formulario desde el menú **Archivo** seleccionando la opción **Guardar.** Compilamos todo el módulo desde el menú

Programa, seleccionando la opción **Compilar PL/SQL,** y a continuación la opción **Todo.** Generamos el fichero ejecutable desde el menú **Programa** seleccionando la opción **Compilar Módulo,** y por último lo ejecutamos desde el menú **Programa** seleccionando la opción **Ejecutar Pantalla.**



Como prueba de funcionalidad del formulario vamos a realizar una consulta para mostrar el primer registro del bloque *Departamentos*, para ello dentro del menú **Consultar** seleccionamos la opción **Ejecutar**, lo que muestra esta imagen.

A continuación nos situamos en cualquier de los campos del bloque maestro *Departamentos*: Codigo o Nombre y pulsamos sobre el menú **Registro** en la opción **Eliminar**. Al hacerlo comprobaremos que en la parte inferior de la pantalla (barra de estado), aparece un mensaje con la siguiente información: "***No se puede suprimir el registro maestro al existir registros detalle coincidentes***", que es lo que queríamos conseguir con las propiedades que hemos indicado en la relación.

Cambiando las propiedades de una relación

En este punto vamos a cambiar las propiedades de la relación manual *DEPARTAMENTO_DEPARTAMENTO_E* del bloque *DEPARTAMENTO* que hemos creado en los puntos anteriores.

En primer lugar seleccionamos la relación *DEPARTAMENTO_DEPARTAMENTO_E* en el navegador de objetos, y abrimos la paleta de propiedades desde el menú **Herramientas,** seleccionando la opción **Paleta de Propiedades**.

Dentro de las propiedades de la relación vamos a cambiar la propiedad **Suprimir Comportamiento de Registro,** y ahora le asignamos el valor **Aislado.**

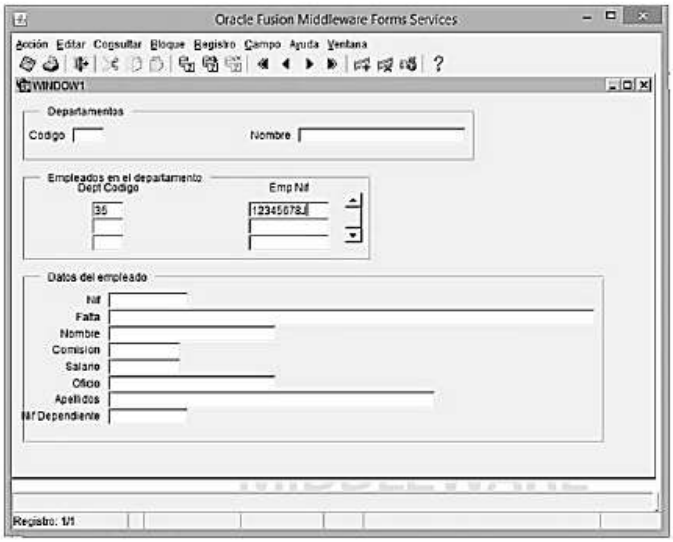
Después de este cambio cerramos la paleta de propiedades y volvemos al navegador de objetos, dentro del objeto **Disparadores** del bloque de datos *DEPARTAMENTO*. Al desplegar

este objeto vemos que ha desaparecido uno de los dos triggers que había anteriormente y ya solo queda el disparador *ON-POPULATE-DETAILS*.

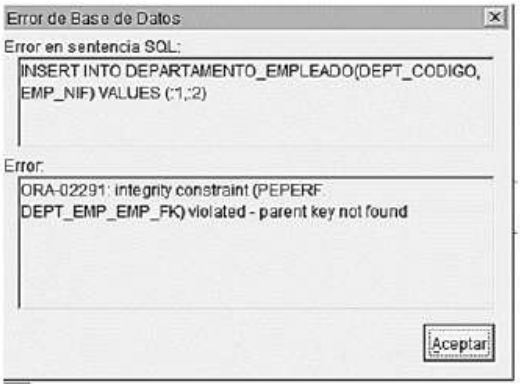
A continuación cambiamos la propiedad **Evitar Operaciones sin Maestro** de la relación *DEPARTAMENTO_DEPARTAMENTO_E* del bloque *DEPARTAMENTO* para asignarle ahora el valor **No.**

Comprobando la funcionalidad del formulario

Después de realizar los cambios indicados en las propiedades de la relación *DEPARTAMENTO_DEPARTAMENTO_E* vamos a ejecutar el formulario para comprobar la funcionalidad del mismo. Para ello seguimos los pasos indicados anteriormente para compilar, y ejecutar el formulario.



Una vez dentro del formulario, nos situamos con el cursor en el campo **DEPT CODIGO** del bloque *Empleados en el departamento* e introducimos como valor para dicho campo: **35** y continuación en el campo **NIF** introduciremos el valor **12345678J,** dejando en blanco el resto de los campos de los bloques *Departamentos* y *Datos del empleado*, como se muestra en la imagen siguiente:



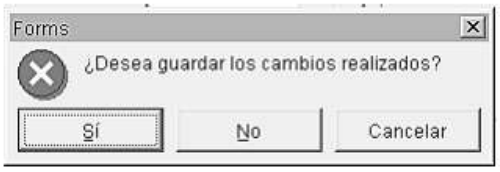
A continuación pulsamos el botón de guardar  y al hacerlo el sistema nos devuelve un error "***FRM-40508 (Error de ORACLE: No se ha podido insertar (INSERT) un registro)"*** en la barra de estado del formulario. Si queremos ver ampliada la información del error dentro del menú **Ayuda** pulsamos la opción **Mostrar error** y aparece este cuadro de diálogo.

La interpretación de este error es sencilla, en la parte superior (sección *Error en sentencia SQL*) se muestra el comando que ha intentado ejecutar Forms Builder para llevar a cabo la operación de guardado, en este caso es una inserción sobre la tabla *DEPARTAMENTO_EMPLEADO,* a la que está vinculado el bloque donde hemos introducido información. En la parte inferior (sección *Error*) se muestra una descripción del error, que en nuestro caso viene a decir que se está intentando insertar información en una tabla detalle vinculada con una maestra en la que la clave de unión de la misma (el código de departamento) no existe en la tabla maestra, es decir, el departamento 35 no existe en la tabla departamentos, por lo que no se pueden dar de alta empleados en el mismo.

Aunque la restricción no existiese definida a nivel de base de datos, igualmente hemos visto que la podemos crear a nivel de Forms Builder para evitar este tipo de inconsistencia al introducir información en el formulario.

Finalizando la práctica

Para terminar con la práctica cerramos la ventana de ejecución del formulario. Al hacerlo y dado que teníamos un error en los datos a introducir en la base de datos el sistema nos devuelve el siguiente mensaje de alerta:



Pulsamos sobre el botón **No** y al volver a Forms Builder guardamos los cambios efectuados y salimos de la herramienta.

PALETA DE HERRAMIENTAS DEL EDITOR DE DISEÑO 12

INTRODUCCIÓN

La paleta de herramientas del Editor de Diseño abarca una gran cantidad de utilidades que permiten crear objetos gráficos y elementos propios del formulario en un lienzo/tapiz. En este capítulo vamos a mostrar cada una de ellas, agrupadas en las siguientes categorías:

- Utilidades de selección y redimensionamiento de objetos.
- Utilidades para crear objetos gráficos en 2D.
- Utilidad para escribir textos.
- Utilidad para crear marcos.
- Utilidad para crear botones.
- Utilidad para crear Check Box.
- Utilidad para crear Botones de Radio.
- Utilidad para crear Elementos de texto.
- Utilidad para crear Elementos mostrados.
- Utilidad para crear contenedores de Elementos de Imagen.
- Utilidad para crear contenedores de Elementos de Gráfico.
- Utilidad para crear contenedores de Elementos Bean.
- Utilidad para crear Árboles Jerárquicos.
- Utilidad para crear Lienzo con Separadores y Lienzos Apilados.
- Utilidades para cambiar el color de fondo, de frente (la letra) y de las líneas.
- Utilidades para alinear objetos dentro del tapiz.
- Utilidades para cambiar propiedades de la fuente.

- Utilidades para realizar Zoom sobre el diseño.
- Herramientas para colocar objetos delante (traer al frente) o detrás de otros (enviar al fondo).
- Herramientas para conocer el nombre del lienzo y del bloque asociado a los objetos pintados.
- Asistente para el diseño de objetos y de bloques.

Utilidades de selección y redimensionamiento de objetos

Los elementos que nos encontramos en esta paleta son los que se indican a continuación:

Imagen	Utilidad	Descripción
	Seleccionar un objeto.	Permite señalar cualquier elemento de los visualizados en el tapiz.
	Rotar un objeto.	Permite rotar un elemento del tapiz a un ángulo diferente de visualización del objeto.
	Ampliar el zoom.	Permite aumentar el zoom de visualización de los elementos del tapiz.
	Remodelar un objeto.	Permite cambiar la perspectiva (altura y profundidad de visualización de un elemento del tapiz).

Utilidades para crear objetos gráficos en 2D

Los elementos que nos encontramos en esta paleta son los que se indican a continuación:

Imagen	Utilidad	Descripción
	Rectángulo.	Crear un rectángulo con las aristas rectas.
	Línea.	Crea una línea recta.
	Círculo.	Crea un círculo.
	Arco.	Crea un arco.
	Polígono	Crea un polígono de líneas rectas.
	Polilínea.	Crea una polilínea (múltiples líneas rectas continuas sin cerrarse entre ellas).
	Rectángulo redondeado.	Crea un rectángulo con las aristas redondeadas.
	Mano alzada.	Permite pintar elementos a mano alzada.

Utilidades para dibujar controles en pantalla

Los elementos que nos encontramos en esta paleta son los que se indican a continuación:

Imagen	Utilidad	Descripción
	Elemento de texto.	Permite crear un elemento de texto (**Text Item**). Un elemento de texto permite realizar modificaciones y consultas sobre su contenido.
	Elemento mostrado.	Permite crear un elemento de visualización de texto (**Display Item**).
	Botón de radio.	Permite crear un botón de radio (**Radio Button**) que se integra en un grupo de botones.
	Casilla de control.	Permite crear una casilla de control (**Check Box**).
	Botones.	Permite crear un botón (**Push Button**).
	Elemento de Lista.	Permite crear una lista (**List Item**).
	Árbol jerárquico.	Permite crear un árbol de elementos (**Hierarchical Tree**).
	Texto.	Permite crear un texto (**Text**).
	Marco.	Permite crear un marco (**Frame**).
	Lienzo con separadores.	Permite crear un lienzo con separadores (**Tab Canvas**).
	Lienzo apilado.	Permite crear un lienzo apilado (**Stacked**).
	Área bean.	Permite crear elementos bean (**Area Bean**).
	Imagen.	Permite crear un elemento de imagen (**Image**).

Utilidades para el cambio de color

Los elementos que nos encontramos en esta paleta son los que se indican a continuación:

Imagen	Utilidad	Descripción
	Color de relleno (fondo).	Permite seleccionar el color del relleno (fondo) del elemento.
	Color de línea.	Permite seleccionar el color de la línea del elemento.
	Color de texto (frente).	Permite seleccionar el color de texto o del frente de un elemento.

Utilidades para alinear objetos dentro del tapiz

Los elementos que nos encontramos en esta paleta son los que se indican a continuación:

Imagen	Utilidad	Descripción
	Alinear a la izquierda.	Permite alinear varios objetos al margen izquierdo de aquel elemento que se encuentre más hacia la izquierda.
	Alinear al centro verticalmente.	Permite alinear varios objetos al centro vertical de los elementos seleccionados.
	Alinear a la derecha.	Permite alinear varios objetos al margen derecho de aquel elemento que se encuentre más hacia la derecha.
	Alinear arriba.	Permite alinear varios objetos al margen de arriba de aquel elemento que se encuentre más arriba.
	Alinear al centro horizontalmente.	Permite alinear varios objetos al centro horizontal de varios elementos seleccionados.
	Alinear abajo.	Permite alinear varios objetos al margen de debajo de aquel elemento que se encuentre más abajo.

Utilidades para cambiar propiedades de la fuente

Los elementos que nos encontramos en esta paleta son los que se indican a continuación:

Imagen	Utilidad	Descripción
	Tipo de fuente.	Permite cambiar el tipo de fuente asociado a los textos de los elementos del lienzo.
	Tamaño de la fuente.	Permite cambiar el tamaño de la fuente asociada a la fuente de los textos.
	Propiedades de la fuente.	Permite cambiar aspectos de la fuente: negrita, cursiva y subrayada respectivamente.

Utilidades para hacer zoom en el diseño

Los elementos que nos encontramos en esta paleta son los que se indican a continuación:

Imagen	Utilidad	Descripción
	Ampliar zoom.	Permite ampliar el zoom.
	Reducir zoom	Permite reducir el zoom.

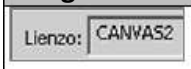
Utilidades para colocar objetos delante o detrás de otros

Los elementos que nos encontramos en esta paleta son los que se indican a continuación:

Imagen	Utilidad	Descripción
	Traer al primer plano.	Permite llevar un elemento que se encuentra en segundo plano (detrás) hacia el primer plano (delante).
	Enviar al segundo plano.	Permite llevar un elemento que se encuentra en primer plano (delante) hacia el segundo plano (detrás).

Utilidades informativas

Los elementos que nos encontramos en esta paleta son los que se indican a continuación:

Imagen	Utilidad	Descripción
	Nombre del lienzo.	Nos informa del nombre del lienzo en el que estamos pintando los objetos.
	Nombre del bloque.	Informa del bloque en el que se están creando los objetos dentro del lienzo.

Utilidades para invocar asistentes

Los elementos que nos encontramos en esta paleta son los que se indican a continuación:

Imagen	Utilidad	Descripción
	Asistente de diseño.	Ejecuta el asistente para el diseño de los elementos de un bloque en un lienzo.
	Asistente de bloque.	Lanza el asistente para la creación de un bloque.

13 ELEMENTOS DE TEXTO

INTRODUCCIÓN

El tipo de objeto por defecto en un formulario de Oracle Forms Builder es el **elemento de texto (Text Item)**. Hasta el momento hemos visto en los capítulos anteriores cómo crear un nuevo bloque de datos basado en una tabla, donde se creaban de forma automática elementos de texto para cada columna seleccionada de dicha tabla. En este capítulo se mostrará la forma de personalizar los elementos de texto para cambiar su aspecto y su comportamiento. Además comprobaremos las propiedades asociadas a los mismos.

¿QUÉ ES UN ELEMENTO DE TEXTO?

Un elemento de texto o Text Item es un objeto del interfaz de un formulario mediante el que se pueden realizar operaciones de consulta, inserción, actualización o supresión de datos. Normalmente un elemento de texto corresponde a una columna de una tabla de la base de datos.

Cuando se crea un elemento en el editor de diseño, por defecto se crea como un elemento de texto.

Todo elemento del editor de diseño posee una propiedad denominada **Tipo de Elemento**, la cual define las propiedades que se van a visualizar en la paleta de propiedades.

MODIFICACIÓN DE ASPECTOS VISUALES DE UN ELEMENTO DE TEXTO

Las propiedades de un elemento de texto, como el resto de objetos de Forms, se dividen en grupos. En esta sección vamos a tratar los siguientes grupos de propiedades que afectan a los aspectos visuales de un elemento de texto:

- General.
- Física.
- Registros.
- Atributos Visuales.
- Color.
- Fuente.
- Petición de Datos.

Grupo de propiedades General

Este grupo de propiedades de un elemento de texto consta de la siguiente propiedad que afecta a su aspecto visual:

- **Tipo de elemento;** determina el tipo de objeto que se va a representar en el lienzo. En el caso concreto que estamos tratando en este capítulo, aparecerá *Elemento de Texto*.

Grupo de propiedades Física

Este grupo de propiedades de un elemento de texto consta de las siguientes propiedades que afectan a su aspecto visual:

- **Visible;** determina si se visualiza o no el elemento en tiempo de ejecución.
- **Lienzo;** determina el nombre del lienzo en el que se mostrará el elemento. Si se deja vacía esta propiedad, se dice que el elemento es de "lienzo nulo", y no se visualizará en tiempo de ejecución, ni tampoco se mostrará en el editor de diseño.
- **Posición X, Y;** define las coordenadas X e Y de la posición del elemento dentro del lienzo.
- **Ancho y Alto;** define el ancho y el alto del elemento.
- **Bisel;** controla el aspecto del marco (bisel o recuadro) que recubre al elemento.

Grupo de propiedades Registros

Este grupo de propiedades de un elemento de texto consta de las siguientes propiedades que afectan a su aspecto visual:

- **Grupo de Atributos Visuales del Registro Actual;** especifica el nombre del atributo visual (conjunto de propiedades visuales) que se va a utilizar cuando el elemento forma parte del registro actual donde se encuentra el cursor en tiempo de ejecución.
- **Distancia entre registros;** determina la cantidad de espacio entre los registros de un bloque multirregistro.
- **Número de elementos mostrados;** especifica el número de registros mostrados para el elemento cuando está en un bloque multirregistro.

Grupo de propiedades Atributos Visuales

Este grupo de propiedades de un elemento de texto consta de las siguientes propiedades que afectan a su aspecto visual:

- **Grupo de Atributos Visuales;** especifica el nombre del atributo visual (conjunto de propiedades visuales) que se va a utilizar sobre el elemento.
- **Grupo de Atributos Visuales de Petición de Datos;** especifica el nombre del atributo visual (conjunto de propiedades visuales) que se va a utilizar sobre la Petición de Datos (leyenda que aparece adjunta al elemento).

Grupo de propiedades Color

Este grupo de propiedades de un elemento de texto consta de las siguientes propiedades que afectan a su aspecto visual:

- **Color de Primer Plano;** especifica el nombre del color que se utiliza para rellenar el primer plano (texto que se indica/visualiza dentro del elemento).
- **Color de Fondo;** especifica el nombre del color con el que se rellenará el fondo del elemento.
- **Patrón de Relleno;** especifica el nombre del tipo de patrón que se utiliza para rellenar el fondo de un objeto.

Grupo de propiedades Fuente

Este grupo de propiedades de un elemento de texto consta de las siguientes propiedades que afectan a su aspecto visual:

- **Nombre de Fuente;** especifica el nombre de la fuente que se va a utilizar para mostrar/insertar texto dentro del elemento.
- **Tamaño de Fuente;** especifica el tamaño de la fuente (en puntos) que se va a utilizar.
- **Grosor de Fuente;** especifica el patrón de relleno de la fuente, pudiéndose elegir entre uno de los siguientes:
    - Muy Clara.
    - Extra Clara.
    - Clara.
    - Semiclara.
    - Media.
    - Seminegrita.
    - Negrita.
    - Extranegrita.
    - Muy negrita.
- **Estilo de Fuente;** especifica el comportamiento de la fuente, pudiéndose elegir entre uno de los siguientes:
    - Normal.
    - Cursiva.
    - Oblicuo.
    - Subrayado.
    - Contorno.
    - Sombra.
    - Invertido.
    - Tachado.
    - Parpadeo.
- **Espaciado de Fuente;** especifica cómo se van a separar las letras del texto que se muestren para la fuente seleccionada, pudiéndose elegir entre uno de los siguientes:
    - Muy Denso.
    - Extradenso.
    - Denso.
    - Semidenso.
    - Normal.
    - Semiexpandido.
    - Expandido.

  - Extra Expandido.
  - Muy Expandido.

Grupo de propiedades Petición de Datos

Este grupo de propiedades de un elemento de texto consta de las siguientes propiedades que afectan a su aspecto visual:

- **Petición de Datos;** especifica el texto o leyenda que se mostrará asociado al campo.
- **Estilo de Visualización del Petición de Datos;** especifica cómo se va a visualizar u ocultar la Petición de Datos (leyenda adjunta al elemento de texto) en el registro. Los valores permitidos son:
  - Oculto.
  - Primer Registro.
  - Todos los Registros.
- **Justificación de Petición de Datos;** especifica el modo en que se justificará el texto introducido como Petición de Datos. Los valores permitidos son:
  - Izquierda.
  - Derecha.
  - Centro.
  - Principio.
  - Fin.
- **Offset de Anexo de Petición de Datos;** especifica la distancia entre el elemento de texto y su Petición de Datos.

CONTROL DE LOS DATOS DE UN ELEMENTO DE TEXTO

Las propiedades que permiten controlar el contenido (los datos) de un elemento de texto, se agrupan dentro del grupo de propiedades **Datos,** dentro del navegador de objetos.

Las propiedades que nos encontramos en este grupo son las siguientes:

- Tipo de Dato.
- Semántica de Longitud de Datos.
- Longitud Máxima.
- Valor Inicial.

- Necesario.
- Máscara de Formato.
- Menor Valor Permitido.
- Mayor Valor Permitido.
- Copiar Valor de Elemento.
- Sincronizar con elemento.

Propiedad Tipo de Dato

Esta propiedad permite indicar el tipo de dato que queremos representar en el formulario. Los tipos disponibles para esta propiedad son los siguientes:

- **Char;** permite representar valores de tipo texto alfanuméricos.
- **Number;** permite representar valores numéricos con y sin decimales.
- **Date;** permite representar valores de tipo fecha.
- **Alfa;** permite representar valores alfabéticos (no admite números ni símbolos especiales).
- **Entero;** permite representar valores numéricos sin decimales.
- **Datetime;** permite representar valores de tipo fecha y hora.
- **Long;** permite representar valores de tipo texto de alta capacidad (hasta 65.534 bytes).
- **Rnumber;** representa valores numéricos en coma flotante.
- **Jdate;** representa una fecha en el formato juliano.
- **Edate;** representa una fecha en formato europeo.
- **Time;** representa valores de tipo hora.
- **Rinteger;** representa valores numéricos enteros en coma flotante.
- **Money;** representa valores con o sin signo que representan importes monetarios.
- **RMoney;** representa valores monetarios en coma flotante.
- **Objeto REF;** encapsula una referencia a un objeto.

Propiedad Semántica de Longitud de Datos

Esta propiedad se utiliza para especificar, en los tipos de dato de texto, cómo se almacenará la información, pudiendo elegir entre BYTE o CHAR. En el resto de tipos de datos se pondrá el valor NULL.

Propiedad Longitud Máxima

Especifica la longitud máxima del valor de datos que se puede almacenar en un elemento.

Propiedad Valor Inicial

Esta propiedad permite indicar un valor por defecto al elemento cuando se crea un registro. El valor que se indique puede ser de tres tipos:

- Un valor estático concreto. Por ejemplo: 340, HOLA, etc.
- Una variable del sistema.
- Un valor de una variable global.
- Un elemento de un bloque del formulario.
- Un valor que se derive de una secuencia.
- Un parámetro del formulario.

En cualquiera de los casos, el valor deberá ser compatible con el tipo de dato.

Si se especifica algún valor en las propiedades *Menor Permitido* y *Mayor Permitido*, el valor de la propiedad *Valor Inicial,* deberá estar incluido en el rango que forman ambas propiedades.

VARIABLES DEL SISTEMA

Una variable de sistema recupera información interna del equipo donde se ejecuta el formulario o de la base de datos, ajustadas a un formato predefinido.

Las variables del sistema que recuperan información de tipo fecha/hora del equipo, donde se ejecuta la aplicación, son las siguientes:

Variable	Formato
$$DATE$$	DD-MES-AA
$$DATETIME$$	DD-MES-AAAA hh:mi[:ss]
$$TIME$$	Hh:mi[:ss]

Las variables del sistema que recuperan información de tipo fecha/hora de la base de datos, a la que se conecta la aplicación, son las siguientes:

Variable	Formato
$$DBDATE$$	DD-MES-AA
$$DBDATETIME$$	DD-MES-AAAA hh:mi[:ss]
$$DBTIME$$	Hh:mi[:ss]

VARIABLES GLOBALES

Una variable global se crea en cualquier parte del código PL/SQL del formulario cuando se le asigna un valor.

La definición de una variable global se realiza anteponiendo el descriptor *:GLOBAL*.

Por ejemplo:

```
:GLOBAL.CODIGO_PRODUCTO := 10;
```

Para asignar la variable global del ejemplo a la propiedad de Valor Inicial, lo haríamos de la siguiente forma:

PARÁMETROS DEL FORMULARIO

Los parámetros de un formulario se definen en la sección *Parámetros* dentro del navegador de objetos de Forms Builder, y se pueden utilizar en cualquier parte del formulario (tanto a nivel de propiedad de un elemento como a nivel de código PL/SQL). Por tanto en este sentido son equivalentes a las variables globales.

La invocación de un parámetro se realiza anteponiendo el descriptor *:PARAMETER*.

Por ejemplo:

```
:PARAMETER.CODIGO_PRODUCTO := 10;
```

ELEMENTOS DE BLOQUE

Son aquellos que se definen en la sección *Bloques* dentro del navegador de objetos de Forms Builder.

Para invocar a un elemento de un bloque se realiza anteponiendo el nombre del bloque al que pertenece el elemento.

Por ejemplo:

```
:BLOQUE1.CODIGO_PRODUCTO := 10;
```

SECUENCIAS

El valor inicial puede hacer referencia a una secuencia en la base de datos. Para hacerlo habrá que indicarlo de la siguiente forma:

:SEQUENCE.<nombre_secuencia>.NEXTVAL

Por ejemplo:

```
:SEQUENCE.COD_PROVINCIAL.NEXTVAL
```

Propiedad Necesario

Impide que el elemento en cuestión se quede sin contenido cuando se estén introduciendo datos en el formulario.

Cuando se crea un bloque de datos basado en una tabla, todos aquellas columnas que tengan NOT NULL en las propiedades de base de datos, se convierten en elementos de Forms con la propiedad *Necesario.*

Propiedad Máscara de Formato

Permite especificar cualquier máscara de formato que sea válida para el tipo de datos indicado en el elemento. Los formatos permitidos se deben de ajustar a la sintaxis SQL.

A continuación se indica los modificadores de formato que se pueden utilizar con tipos de datos numéricos:

- 9 : para indicar un dígito. Si no aparece se sustituye por un espacio.
- 0 : para indicar un dígito. Si no aparece se sustituye por un cero.
- D : para indicar el separador decimal.
- G : para indicar el separador de miles.

A continuación se indican los modificadores de formato que se pueden utilizar con tipos de datos fecha u hora:

- AM: para indicar si la hora se encuentra antes (AM) o después de las 12 (PM) del mediodía.

- BC: para indicar si la fecha corresponde a un periodo Anterior (AC) o posterior (BC) al nacimiento de Cristo.
- d: para indicar el día de la semana (1 – 7).
- dd: para indicar el día del mes (1-31).
- ddd: para indicar el día del año (1-366).
- DAY: para indicar el nombre del día (lunes – domingo).
- dy: para indicar el nombre del día de forma abreviada.
- hh: para indicar la hora en formato de 2 posiciones (1-12).
- hh12: para indicar la hora en formato 12 hs.
- hh24: para indicar la hora en formato 24 hs.
- iw: para indicar la semana dentro del año.
- mi: para indicar los minutos (0 – 59).
- mm: para indicar el mes (1 – 12).
- MON: para indicar el nombre del mes en formato abreviado (ENE-DIC).
- MONTH: para indicar el nombre del mes (ENERO-DICIEMBRE).
- q: para indicar el número correspondiente al cuatrimestre (1,2,3,4).
- rm: para indicar el mes en números romanos.
- ss: para indicar los segundos (0 – 59).
- SCC: para indicar el siglo.
- ww: para obtener la semana del año.
- w: para obtener la semana del mes.
- y: para indicar una posición para el año (admite desde y hasta yyyy).
- year: para obtener la fecha traducida a texto en letras.

Con un formato de fecha *dd/mm/aa*, los valores admitidos para introducir en el formulario serían: 10/12/00, 10 12 00, 10-DEC-00, 10-DIC-00 o 101200. Se puede introducir cualquier carácter para representar la barra inclinada (/) del formato de fecha, dado que los caracteres embebidos en la máscara de formato solo se utilizan con fines de representación de la información en pantalla y no se almacenan en la base de datos.

Propiedades Menor y Mayor Permitido

Permiten especificar un rango de valores (mínimo y máximo) aceptados para el tipo de dato del elemento.

Propiedad Copiar Valor de Elemento

Permite la posibilidad de que el valor del elemento no se introduzca directamente en el formulario sino que se copie desde otro elemento. Se puede especificar un elemento de cualquier bloque de los definidos en el formulario indicando: *NOMBRE_BLOQUE.NOMBRE_ELEMENTO*

Propiedad Sincronizar con Elemento

Especifica el nombre del elemento del cual debemos derivar su valor sobre el elemento actual, para sincronizar ambos elementos de forma que sean un duplicado uno del otro.

Cuando se encuentre en ejecución el formulario y se cambie el valor de uno de los elementos, también se cambia automáticamente el valor del elemento sincronizado.

La sincronización solo permite introducir elementos del mismo bloque.

CONTROL DE LA NAVEGACIÓN DE UN ELEMENTO DE TEXTO

Las propiedades del grupo *Navegación* de un elemento de texto se utilizan para controlar el comportamiento en la navegación desde/hacia el elemento.

Las propiedades que nos encontramos en este grupo son las siguientes:

- Teclado Navegable.
- Elemento de Navegación Anterior y Siguiente.

Propiedad Teclado Navegable

Determina si se puede navegar al elemento que posee esta propiedad, durante la navegación por defecto con las teclas de función o las opciones de menú, y centrarse en dicho elemento.

Cuando la propiedad se define a valor *No*, Forms Builder salta el elemento e introduce el siguiente elemento navegable que exista en el formulario, dentro de la secuencia de navegación por defecto.

Propiedad Elemento de Navegación Anterior y Siguiente

Determina cuál es el elemento anterior y siguiente al que se va a navegar desde el actual.

CONTROL DE LAS PROPIEDADES DE BASE DE DATOS DE UN ELEMENTO DE TEXTO

Las propiedades del grupo *Base de Datos* de un elemento de texto permiten definir los mecanismos para interactuar con la columna de la tabla de la base de datos con la que se relaciona el elemento de texto del formulario.

Las propiedades que nos encontramos en este grupo son las siguientes:

- Elemento de base de datos.
- Nombre de columna.
- Clave primaria.
- Consulta permitida.
- Inserción permitida.
- Actualización permitida.
- Actualizar solo si es nulo.
- Consulta no sensible a mayúsculas/minúsculas.
- Longitud de la consulta.

Propiedad Elemento de base de datos

Indica si el elemento de texto va a interactuar con una columna de base de datos.

Propiedad Nombre de columna

Indica el nombre de la columna de la tabla con la que interactúa.

Propiedad Clave primaria

Indica si el elemento de texto forma parte de la clave primaria de la tabla a la que hace referencia. Si se indica el valor *Si* implica que el contenido de este elemento, en tiempo de ejecución, no puede quedarse vacío dentro del formulario.

Propiedad Consulta permitida

Controla que se permiten realizar operaciones de consulta sobre el elemento de texto.

Propiedad Inserción permitida

Controla que se permiten realizar operaciones de inserción de nuevos registros utilizando el elemento de texto.

Propiedad Actualización permitida

Controla que se permiten realizar modificaciones de la información contenida en el elemento de texto.

Propiedad Actualizar solo si es nulo

Controla si se permite modificar la información del campo cuando su contenido únicamente sea nulo.

Propiedad Consulta no sensible a mayúsculas/minúsculas

Controla si se reconocen las mayúsculas y minúsculas introducidas en el elemento de texto cuando se utiliza en el procesamiento de consultas.

Propiedad Longitud de la consulta

Especifica la longitud máxima del criterio de búsqueda que se introduce en el elemento de texto cuando se va a realizar una consulta.

CONTROL FUNCIONAL DE UN ELEMENTO DE TEXTO

Se puede modificar o mejorar la forma en que los elementos de texto interactúan con su columna correspondiente de base de datos, definiendo las propiedades del grupo *Funcional*.

Las propiedades que nos encontramos en este grupo son las siguientes:

- Activado.
- Justificación.
- Multilínea.
- Estilo de ajuste de texto.
- Restricción mayúsculas / minúsculas.
- Ocultar datos.
- Salto automático.

Propiedad Activado

Controla si se bloquea el acceso al contenido del elemento de texto en tiempo de ejecución.

Propiedad Justificación

Indica cómo se justifica el contenido del elemento del texto en tiempo de ejecución. Las posibilidades de justificación son las siguientes:

- Izquierda.
- Derecha.
- Centro.
- Principio.
- Fin.

Propiedad Multilínea

Determina si el elemento de texto se visualiza como un cuadro que permite introducir varias líneas de texto (con una barra de desplazamiento) o no.

Propiedad Estilo de ajuste de texto

Esta propiedad se utiliza en combinación de la propiedad *Multilínea,* cuando el elemento de texto admite varias líneas de texto. En este caso, esta propiedad indica cómo se va a visualizar la información dentro del elemento de texto cuando una línea de texto excede del ancho de la casilla que se ha pintado para el mismo.

Propiedad Restricción mayúsculas / minúsculas

Esta propiedad se utiliza para indicar cómo queremos que se represente la información dentro del elemento de texto. Las posibilidades son:

- **Superior**; para escribirlo todo en mayúsculas.
- **Inferior**; para escribirlo todo en minúsculas.
- **Mixto**; para admitir mayúsculas y minúsculas.

Propiedad Ocultar datos

Esta propiedad oculta los caracteres del elemento de texto que se introducen en tiempo de ejecución (sustituye cada letra introducida por un asterisco). Normalmente se utiliza para crear campos con protección de escritura como por ejemplo una contraseña.

Propiedad Salto automático

Esta propiedad permite controlar que cuando se llene el tamaño máximo del elemento de texto, el cursor salte al siguiente elemento navegable de forma automática.

CONTROL DE LA AYUDA DE UN ELEMENTO DE TEXTO

Para mostrar información al usuario sobre el contenido a introducir en el campo correspondiente se puede utilizar el grupo *Ayuda*.

Las propiedades que nos encontramos en este grupo son las siguientes:

- Indicación.
- Mostrar indicio automáticamente.
- Ayuda de burbuja.
- Grupo de Atributos Visuales de la ayuda de burbuja.

Propiedad Indicación

Esta propiedad permite escribir un texto de ayuda específico para el elemento de texto. La información de ayuda que se introduce en esta propiedad, se visualiza en la

línea de mensajes del formulario ejecutado cuando el foco de entrada (cursor) se encuentra dentro del elemento.

Propiedad Mostrar indicio automáticamente

Esta propiedad determina si la propiedad anterior (*Indicación*) se mostrará automáticamente.

Si se defina a valor *No*, el texto de ayuda solo se visualizará cuando el usuario pulse sobre la barra de herramientas *Ayuda* del formulario en ejecución.

Propiedad Ayuda de burbuja

Esta propiedad permite escribir un texto de ayuda complementario al de la propiedad *Indicación,* que se mostrará en un recuadro pequeño debajo del elemento de texto, cuando el ratón pasa por encima de él.

En este caso no es necesario que el foco (cursor) esté dentro del elemento para mostrarse la ayuda.

Propiedad Grupo de Atributos Visuales de la ayuda de burbuja

Esta propiedad permite especificar el nombre del atributo visual (combinación de propiedades de color) que se utiliza para la ayuda de burbuja.

CREANDO ELEMENTOS DE TEXTO 14

INTRODUCCIÓN

En este capítulo vamos a poner en práctica todos los conocimientos adquiridos en el capítulo anterior, con objeto de crear elementos de texto en un formulario utilizando el Editor de Diseño.

Creando un nuevo formulario

Como primer paso vamos a crear un nuevo formulario. Para ello abrimos Forms Builder y a continuación desde la barra de menú **Archivo** seleccionamos **Nuevo** y por último **Pantalla.**

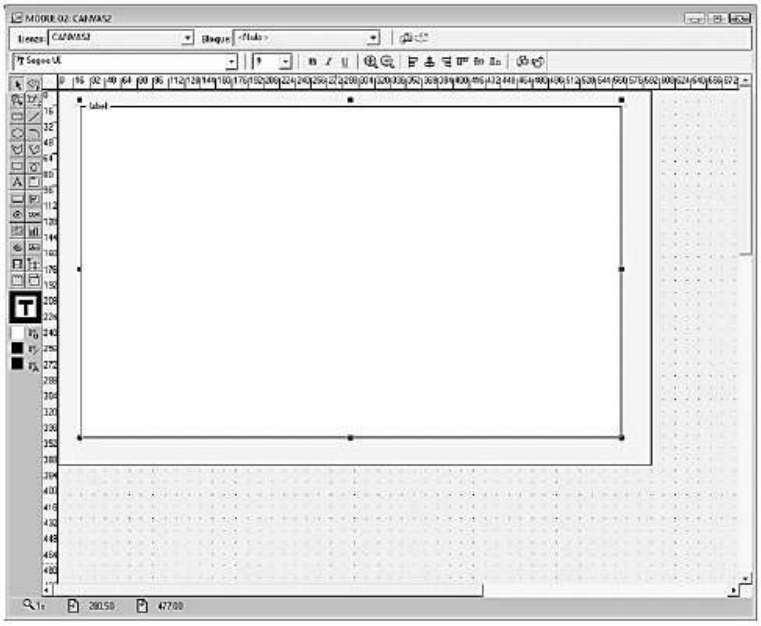
Apertura del Editor de Diseño

Seguidamente abrimos el editor de diseño desde la barra de menú **Herramientas** seleccionando la opción **Editor de Diseño.**

Crear un marco (frame)

Desde el editor de diseño lo primero que vamos a crear manualmente es un marco o frame. Para ello nos situamos en la paleta de herramientas icónica y pulsamos el botón  .

A continuación marcamos un punto de comienzo dentro del lienzo que se muestra en el editor de diseño, y lo desplazamos en diagonal hasta el extremo contrario para abrir el marco.



Tenemos que diseñar un marco con una longitud aproximada a la que se muestra en esta imagen.

Podemos partir del diseño desde la posición 570 horizontal y 351 vertical.

Cambiar el título del marco

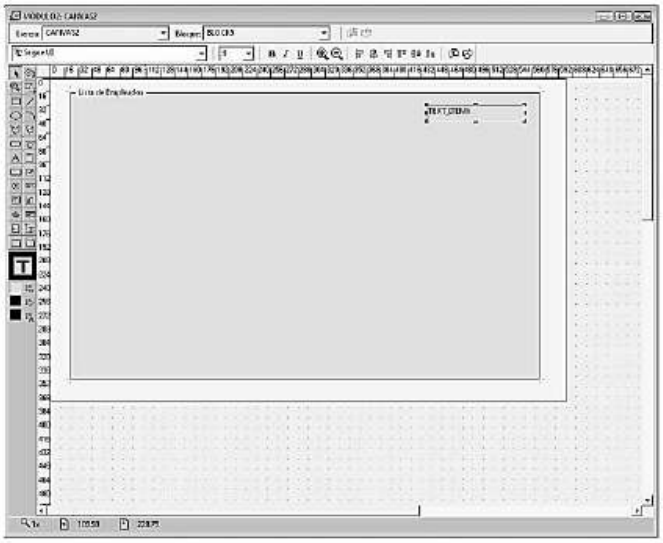
Hacemos un clic con el ratón sobre el marco, y pulsamos sobre la barra de menú **Herramientas** para seleccionar la **Paleta de Propiedades**, que nos abre las propiedades del elemento.

Dentro de la paleta de propiedades elegimos **Título de Marco** y escribimos *Lista de Empleados*.

Cambiar el color del fondo del marco

Sin abandonar la paleta de propiedades del marco, vamos a cambiar el color de fondo (relleno) del frame que acabamos de pintar en el lienzo. Para ello nos situamos en la propiedad **Color de Fondo,** y a continuación introducimos como valor el siguiente: *r100g88b75*.

Creando un elemento de texto en el editor de diseño



Cerramos la paleta de propiedades y volvemos al editor de diseño. A continuación seleccionamos el icono dentro de la barra de herramientas icónica y pulsamos en un punto del marco, arrastrándolo en diagonal para conseguir obtener una imagen como la que se muestra en esta imagen.

MANEJO DE COLORES Y NOTACIÓN EN FORMS

Para hacer referencia a los colores dentro de las propiedades de Forms se pueden utilizar 2 técnicas:

- Indicar el nombre del color en inglés.
- Indicar el nombre del color en codificación RGB (Red Green Blue).

Notación del color en inglés

Cuando queremos hacer uso de un color sólido sin mezclas, podemos nominarlo con su sintaxis inglesa. Por ejemplo: *white* (blanco), *blue* (azul), *red* (rojo), *green* (verde), etc.

Cuando queremos hacer uso de un color sólido en el que queramos atenuar u oscurecer su intensidad, podemos nominarlo con sus sintaxis inglesa antepuesto de las palabras *light* (claro) o *dark* (oscuro). Por ejemplo: *lightblue* (azul claro), *darkblue* (azul oscuro), etc.

Notación del color en formato RGB

Cuando queremos hacer uso de un color de la paleta de colores que tiene disponible el Forms y que normalmente se obtiene por la mezcla de varios de ellos, tenemos que indicarlo mediante notación RGB.

La notación RGB es la que nomina los colores bajo la combinación de los 3 colores básicos: Red (rojo), Green (verde) y Blue (azul). Para estos tres colores podemos indicar la intensidad de uso de cada uno, utilizando valores desde el 0 (no se hace uso del color) hasta el 255 (máxima cantidad del color).

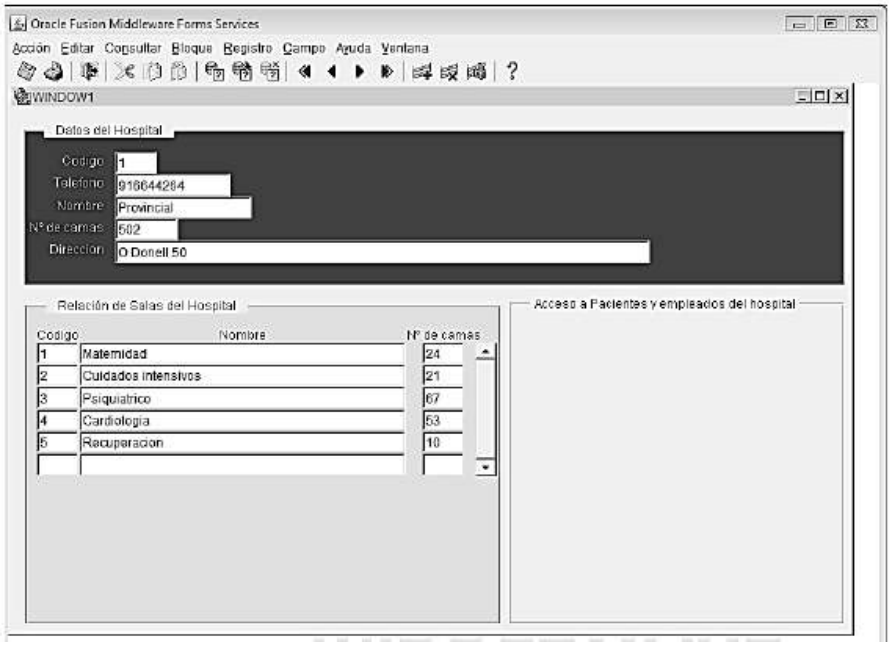
Por tanto, si por ejemplo quisiéramos indicar como color a utilizar el azul a su máxima intensidad lo haríamos de la siguiente forma en notación RGB: *r0g0b255.*

Otros ejemplos en notación RGB podrían ser los siguientes: *r0g0b0* (equivale al blanco), *r255g255b255* (equivale al negro), etc.

SUPUESTO PRÁCTICO 1

MÓDULO HOSPITAL-SALA

En este supuesto práctico se deberá diseñar un módulo (formulario) de Forms denominado **HOSPITAL_SALA** que tenga un diseño similar al siguiente:



Las características que debe incluir el formulario son las siguientes:

- Este módulo accederá a las tablas HOSPITAL y SALA.
- Se creará un bloque basado en tabla para cada una de las anteriores.

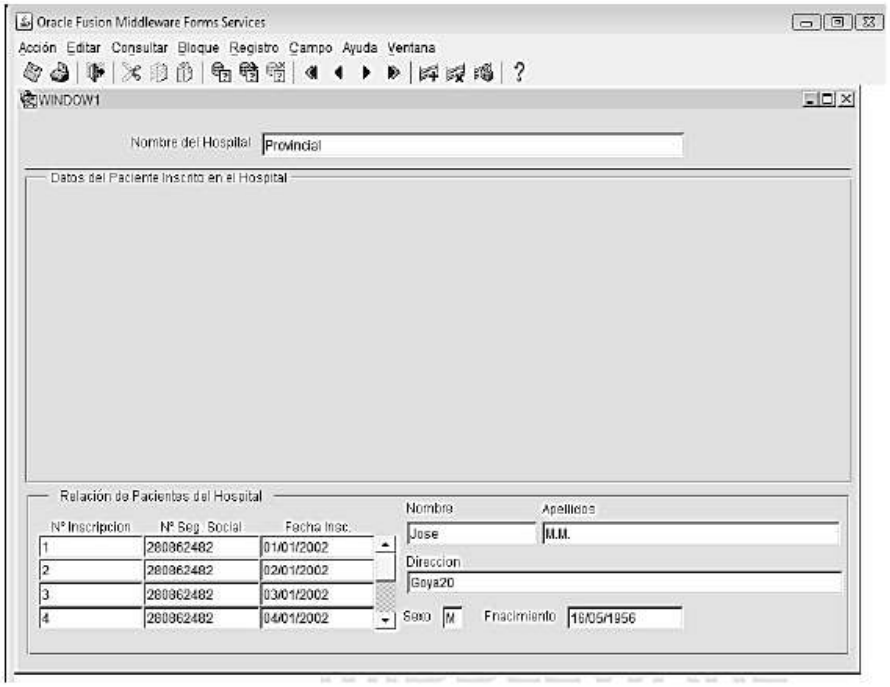
- Se debe crear una relación maestro-detalle entre los bloques de las tablas HOSPITAL y SALA *(SALA.HOSP_CODIGO = HOSPITAL.CODIGO)*.
- La relación entre los bloques será **No aislada** y deberán **Evitar operaciones sin Maestro** y permitir **Consultas automáticas**.
- Del bloque HOSPITAL se mostrarán todos los campos.
- Del bloque SALA se mostrarán todos los campos menos el correspondiente al código de hospital (que no se mostrará en pantalla, pero sí se recuperará en el bloque).
- En el bloque HOSPITAL únicamente se mostrará un registro.
- En el bloque SALA se mostrarán 6 registros (**6 salas**) a la vez, para el hospital que se haya seleccionado.
- Se diseñará un frame que recubra los elementos del bloque HOSPITAL y que tenga un color de fondo **r0g0b50** y el título **Datos del Hospital**.
- Se diseñará un frame que recubra los elementos del bloque SALA y que tenga un color de fondo **r75g100b100** y el título **Relación de Salas del Hospital**.
- Se modificará el literal asociado a los campos **NUMCAMAS** de ambos bloques, para que muestren: **Nº de camas.**

MÓDULO ENFERMOS-HOSPITAL

En este supuesto práctico se deberá diseñar un módulo (formulario) de Forms denominado **ENFERMOS_HOSPITAL** que tenga un diseño similar al que se muestra en la página siguiente y que posea las siguientes características:

- Este módulo accederá a las tablas HOSPITAL, HOSPITAL_ENFERMO y ENFERMO.
- Se creará un bloque basado en tabla para cada una de las anteriores.
- Se debe crear una relación maestro-detalle entre los bloques de las tablas HOSPITAL y HOSPITAL_ENFERMO *(HOSPITAL_ENFERMO.HOSP_CODIGO = HOSPITAL.CODIGO)*, así como entre HOSPITAL_ENFERMO y ENFERMO *(ENFERMO.NUMSEGSOCIAL = HOSPITAL_ENFERMO.ENF_NUMSEGSOCIAL)*.
- La relación entre los bloques será **No aislada** y deberán **Evitar operaciones sin Maestro** y permitir **Consultas automáticas**.
- Del bloque HOSPITAL se mostrará únicamente el campo **NOMBRE**.
- Del bloque HOSPITAL_ENFERMO se mostrarán todos los campos menos el código de hospital.

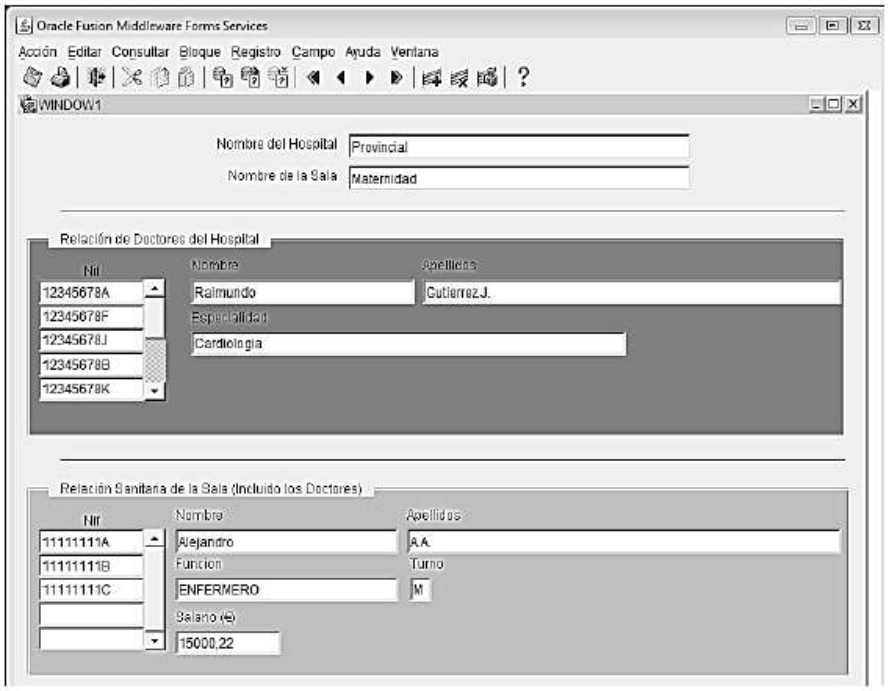
- Del bloque ENFERMO se mostrarán todos los campos excepto el campo NUMSEGSOCIAL.
- En el bloque HOSPITAL únicamente se mostrará un registro.
- En el bloque HOSPITAL_ENFERMO se mostrarán 4 registros (**4 inscripciones**) a la vez, para el hospital que se haya seleccionado.
- En el bloque ENFERMO únicamente se mostrará un registro.
- Se diseñará un frame que recubra los elementos de los bloques HOSPITAL_ENFERMO y ENFERMO, que tenga un color de fondo **r100g75b75** y el título **Relación de Pacientes del Hospital**.
- Se modificarán los literales asociados a los campos del bloque HOSPITAL_ENFERMO para que muestren respectivamente: **Nº Inscrip, Nº Seg. Social.** y **Fecha Insc.**



MÓDULO PLANTILLA-SANITARIA

En este supuesto práctico se deberá diseñar un módulo (formulario) de Forms denominado **PLANTILLA_SANITARIA** que tenga un diseño similar al que se muestra en la página siguiente y que posea las siguientes características:

- Este módulo accederá a las tablas HOSPITAL, SALA, PLANTILLA_SALA, PLANTILLA, DOCTOR_HOSPITAL y DOCTOR
- Se creará un bloque basado en tabla para cada una de las anteriores.
- Se debe crear una relación maestro-detalle entre los bloques de las tablas siguientes: HOSPITAL y SALA *(SALA.HOSP_CODIGO = HOSPITAL.CODIGO)*, SALA y PLANTILLA_SALA *(PLANTILLA_SALA.SALA_HOSP_CODIGO = SALA.HOSP_CODIGO AND PLANTILLA_SALA.SALA_CODIGO = SALA.CODIGO)*, PLANTILLA_SALA y PLANTILLA *(PLANTILLA.NIF = PLANTILLA_SALA.PLAN_NIF)*, HOSPITAL y DOCTOR_HOSPITAL *(DOCTOR_HOSPITAL.HOSP_CODIGO = HOSPITAL.CODIGO)* y por último entre DOCTOR_HOSPITAL y DOCTOR *(DOCTOR.NIF = DOCTOR_HOSPITAL.DOC_NIF)*.
- Las relaciones entre los bloques serán todas **No aislada** y deberán **Evitar operaciones sin Maestro** y permitir **Consultas automáticas**.
- Del bloque HOSPITAL se mostrará únicamente el campo **NOMBRE**.
- Del bloque SALA se mostrará únicamente el campo **NOMBRE.**
- Del bloque PLANTILLA_SALA se mostrará el campo **PLAN_NIF.**
- Del bloque PLANTILLA se mostrarán todos los campos menos el NIF.
- Del bloque DOCTOR_HOSPITAL se mostrará el campo **DOC_NIF**.
- Del bloque DOCTOR se mostrarán todos los campos menos el NIF.
- En el bloque DOCTOR_HOSPITAL se mostrarán 5 registros (**5 nifs de doctores**) a la vez, para el hospital que se haya seleccionado.
- En el bloque DOCTOR únicamente se mostrará 1 registro.
- En el bloque PLANTILLA_SALA se mostrarán 5 registros (**5 nifs de miembros de la plantilla**) a la vez, para el hospital y sala que se haya seleccionado.
- En el bloque PLANTILLA únicamente se mostrará 1 registro.
- Se diseñará un frame que recubra los elementos del bloque PLANTILLA que tenga un color de fondo **r75g75b75** y el título **Plantilla Sanitaria de la Sala (Incluido los Doctores)**.
- Se diseñará un frame que recubra los elementos del bloque DOCTOR que tenga un color de fondo **r25g75b50** y el título **Relación de Doctores del Hospital**.
- Se modificará el título del campo SALARIO para que se muestre **Salario (€).**

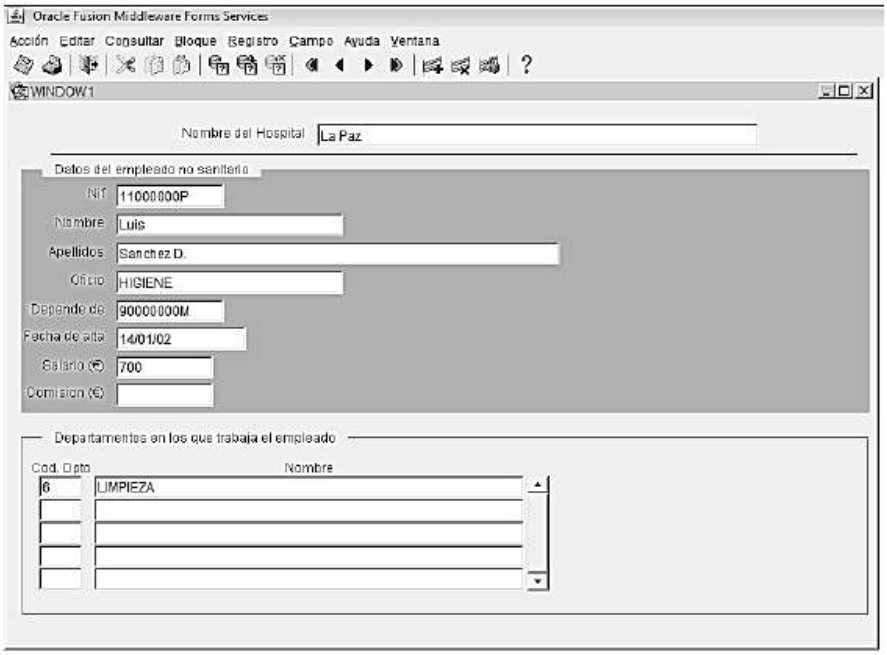


MÓDULO PLANTILLA-NO-SANITARIA

En este supuesto práctico se deberá diseñar un módulo (formulario) de Forms denominado **PLANTILLA_NO_SANITARIA** que tenga un diseño similar al que se muestra en la página siguiente y que posea las siguientes características:

- Este módulo accederá a las tablas HOSPITAL, EMPLEADO_HOSPITAL, EMPLEADO y DEPARTAMENTO.
- Se creará un bloque basado en tabla para cada una de las anteriores.
- Se debe crear una relación maestro-detalle entre los bloques de las tablas siguientes: HOSPITAL y EMPLEADO_HOSPITAL *(EMPLEADO_HOSPITAL.HOSP_CODIGO = HOSPITAL.CODIGO),* entre EMPLEADO_HOSPITAL y EMPLEADO *(EMPLEADO.NIF = EMPLEADO_HOSPITAL.EMP_NIF)*, entre DEPARTAMENTO_EMPLEADO y EMPLEADO (*DEPARTAMENTO_EMPLEADO.EMP_NIF = EMPLEADO.NIF),* y por último entre DEPARTAMENTO y DEPARTAMENTO_EMPLEADO (*DEPARTAMENTO.CODIGO = DEPARTAMENTO_EMPLEADO.DEPT_CODIGO).*
- Las relaciones entre los bloques serán todas **No aislada** y deberán **Evitar operaciones sin Maestro** y permitir **Consultas automáticas**.
- Del bloque HOSPITAL se mostrará únicamente el campo **NOMBRE**.
- Del bloque EMPLEADO_HOSPITAL se mostrará únicamente el campo **EMP_NIF.**

- Del bloque EMPLEADO se mostrarán todos los campos menos el Nif.
- Del bloque DEPARTAMENTO_EMPLEADO se mostrará el campo **DEPT_CODIGO**
- Del bloque DEPARTAMENTO sólo se mostrará el campo **NOMBRE**.
- En el bloque EMPLEADO y en el bloque EMPLEADO_HOSPITAL se mostrará 1 registro.
- En el bloque DEPARTAMENTO y en el bloque DEPARTAMENTO_EMPLEADO se mostrarán 5 registros.
- Se diseñará un frame que recubra los elementos del bloque EMPLEADO y DEPARTAMENTO_EMPLEADO que tenga un color de fondo **r50g88b75** y el título **Datos del empleado no sanitario**.
- Se diseñará un frame que recubra los elementos del bloque DEPARTAMENTO y DEPARTAMENTO_EMPLEADO que tenga un color de fondo **r100g75b75** y el título **Departamentos en los que trabaja el empleado**.
- Se modificará el título de los siguientes campos:
  - EMP_NIF: **Nif**
  - NIF_DEPEDIENTE: **Depende de**
  - FALTA: **Fecha de alta**
  - SALARIO: **Salario (€)**
  - COMISION: **Comision (€)**
  - DEPT_CODIGO: **Cod. Dpto**



LISTA DE VALORES (LOV) 15

INTRODUCCIÓN

Dentro del Forms Builder se pueden crear listas de valores para facilitar al usuario la selección de una información a introducir en un elemento de texto y mejorar el funcionamiento de una aplicación.

Una lista de valores (LOV) es un objeto de un formulario que abre su propia ventana cuando se activa en tiempo de ejecución. Se define a nivel de formulario, lo cual significa que puede utilizarse para soportar elementos de texto en cualquier bloque del formulario.

Cuando se despliega la ventana emergente con los valores de la lista, el usuario puede seleccionar uno de ellos para trasportarlo hasta el elemento de texto del cual depende.

Una lista de valores puede incluir en la ventana emergente varias columnas de información.

El usuario puede reducir las líneas mostradas en la lista mediante técnicas de reducción simples automáticas o mediante cadenas de búsqueda.

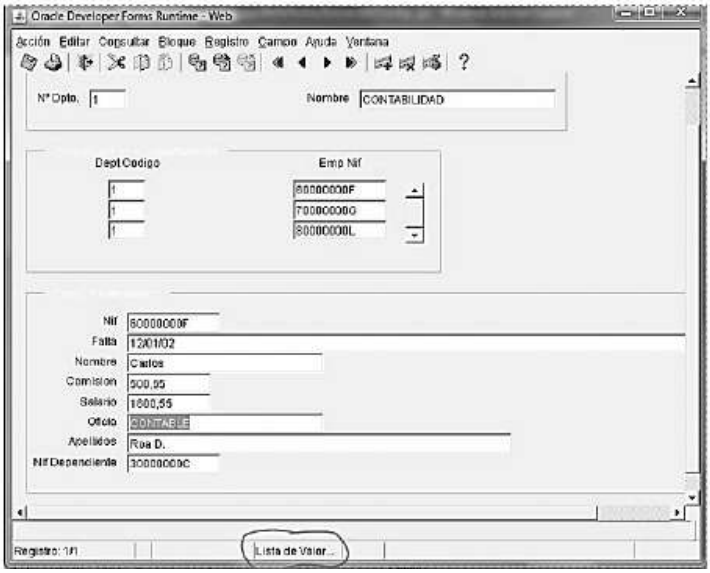
Cada línea de una lista de valores puede presentar diversos valores de campo, con cabeceras de columna encima. Puede diseñar su propia lista de valores para recuperar alguno o todos los valores mostrados.

Características de una LOV

Las listas de valores tienen las siguientes características:

- **Dinámicas;** los valores que se muestran en una lista cambian dinámicamente durante la ejecución del formulario sincronizadas con los cambios que se produzcan en los datos de origen que se consultan en la lista.
- **Independientes;** las listas de valores se pueden invocar tanto desde un elemento de texto del cual depende, como directamente a través de programación.
- **Flexibles;** se puede reutilizar la misma lista de valores para soportar varios elementos de texto.
- **Eficientes;** se pueden diseñar listas de valores que utilicen datos ya cargados en pantalla en lugar de acceder a la base de datos para cada invocación que se haga de la lista.

Cómo utilizar una LOV en tiempo de ejecución

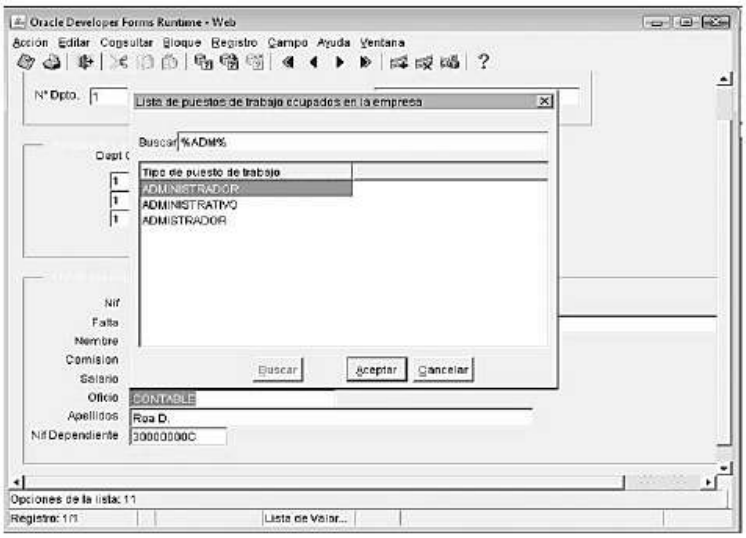


Cuando un elemento de texto tiene una lista de valores asociada, en la línea de estado del formulario en ejecución se muestra el indicador "***Lista de Valor***" mientras el cursor permanece sobre el elemento.

Para que se muestre la lista habrá que pulsar sobre el menú **Editar** y seleccionar la opción **Mostrar Lista.**

Para seleccionar un registro de los resultados de la lista hay que hacer un clic con el ratón sobre aquella que se necesite.

Si no se quiere mostrar el contenido completo de la lista de valores y se quieren filtrar los valores devueltos, se pueden utilizar patrones dentro del apartado **Buscar** del cuadro de visualización de resultados de la lista, tal y como se muestra en la imagen siguiente:



En el ejemplo mostrado se ha filtrado la información de la lista de forma que solo retorne aquellos elementos que contengan el texto "ADM" en cualquier posición (para ello se usa el símbolo % delante y detrás de la cadena de texto).

Objetos asociados a una lista de valores

Cuando se crea una lista de valores hay que tener en cuenta los siguientes objetos que participan en la misma:

- **Grupo de registros;** es un objeto de Forms Builder que se utiliza para almacenar la matriz de valores presentados por una lista de valores. El grupo de registros se puede crear antes de la lista de valores o bien crearla a la vez que la LOV (si se basa en una consulta contra tablas de la base de datos).
- **Lista de valores (LOV);** corresponde con el nombre de las columnas que se van a mostrar del grupo de registros asociado así como su tamaño y posible asociación con elementos de texto de los bloques del formulario. Además incluye características para la visualización de la lista: ancho/alto del cuadro emergente que muestra la lista, posición de visualización dentro del formulario, colorido, etc.).
- **Elementos de texto;** corresponde con el/los elemento/s de texto que se asocian a una lista de valores.

Grupo de registros

Un grupo de registros es una estructura de columna/s y fila/s almacenadas en la memoria del formulario ejecutado, y es similar a la estructura de una tabla de base de datos. Retiene registros que pueden reutilizar otras aplicaciones de Oracle Forms y Oracle Reports, reduciendo así el acceso repetido a datos externos.

Los grupos de registros se pueden diseñar para contener valores estáticos. Como opción se pueden rellenar mediante programación en tiempo de ejecución, o más habitualmente se pueden rellenar mediante una consulta SQL.

Los grupos de registros presentan las siguientes características:

- Los datos se presentan mediante listas de valores.
- Contienen datos que pueden ser dinámicos o estáticos dependiendo del diseño que se haga.
- Contienen datos que se pueden utilizar tanto en Oracle Forms, como en Oracle Reports.
- Al ser objetos independientes se pueden utilizar por más de una lista de valores distinta.

Propiedades de una LOV

En esta sección vamos a tratar las propiedades que definen a una lista de valores.

POSICION X, Y

Estas dos propiedades definen las coordenadas de pantalla para la apertura de la ventana emergente que muestra la lista de valores. Las coordenadas se deberán especificar en la misma unidad en la que se definen las coordenadas del formulario.

ANCHO, ALTURA

Estas dos propiedades definen el tamaño de la ventana emergente que muestra la lista de valores.

PROPIEDADES DE CORRESPONDENCIA DE COLUMNAS

En estas propiedades se especifican las relaciones entre las columnas del *Grupo de Registros* que muestra la LOV y aquellos elementos de texto de los bloques del formulario donde se recibirán los datos seleccionados en la LOV.

FILTRAR ANTES DE MOSTRAR

Determina si se debe mostrar un recuadro de diálogo para que el usuario introduzca un valor de búsqueda antes de llamar a la lista y recuperar los resultados.

El valor que se introduzca en este cuadro de diálogo únicamente se utilizará como restricción adicional en la primera columna de la consulta de la LOV.

VISUALIZACIÓN AUTOMÁTICA

Determina si se debe visualizar automáticamente la lista de valores cuando el usuario va a introducir información (en tiempo de ejecución) en el elemento de texto que tiene asociada la LOV.

SELECCIÓN AUTOMÁTICA

Determina si automáticamente Forms debe retornar en tiempo de ejecución el contenido de la lista sobre el/los elemento/s de texto asociados cuando la LOV solo retorna una fila en los resultados.

SALTO AUTOMÁTICO

Determina si el cursor (en tiempo de ejecución) tiene que saltar automáticamente al siguiente elemento navegable cuando el usuario selecciona un valor de la LOV y el contenido se retorna a los elementos de texto asociados.

REFRESCAMIENTO AUTOMÁTICO

Si esta propiedad tiene el valor *Sí* indica que cada vez que se abra la LOV se ejecuta la consulta asociada al grupo de registros. En cambio cuando el valor de la propiedad es *No,* la consulta asociada al grupo de registros solo se ejecutará una vez cuando se visualice por primera vez la lista de valores.

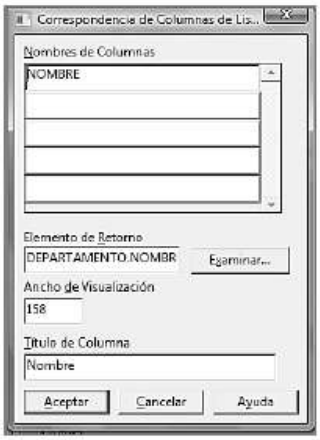
POSICIÓN AUTOMÁTICA

Determina si Forms coloca automáticamente la ventana emergente de la lista de valores cerca del elemento de texto que la invoca, en tiempo de ejecución. Si esta propiedad se pone a valor *Sí*, las propiedades *Posición X* y *Posición Y* no se tienen en cuenta.

ANCHO DE COLUMNA AUTOMÁTICO

Determina si Forms define automáticamente el ancho de cada una de las columnas que se muestren en la LOV. Forms ajusta el ancho de la columna al tamaño más grande de los que se obtengan del contenido de la columna y del título del encabezado.

Propiedad de Correspondencia de Columnas



Al pulsar sobre la propiedad *Correspondencia de Columnas* dentro de las propiedades de una LOV, se muestra este cuadro de diálogo.

El significado y contenido a introducir en cada uno de los elementos del cuadro de diálogo es el siguiente:

- **Nombres de Columnas;** permite seleccionar las columnas del grupo de registros que queremos visualizar en la LOV.
- **Elementos de Retorno;** especifica el nombre de un parámetro, variable global o elemento de texto de un bloque del formulario al que vamos a devolver el resultado de la LOV. No es necesario introducir un elemento de retorno para cada nombre de columna. En el caso de que no exista elemento de retorno para un nombre de columna, su contenido se visualizará en el cuadro de diálogo de la LOV pero no se llevará a ningún elemento del formulario.
- **Ancho de Visualización;** especifica el ancho de visualización de la columna dentro del cuadro de diálogo de la LOV.
- **Título de Columna;** especifica el título o cabecera para la columna.

Propiedades de un elemento de texto asociado a una LOV

En esta sección vamos a describir las propiedades de un elemento de texto que está vinculado a una LOV. Estas propiedades se agrupan con el nombre *Lista de Valores* dentro de la paleta de propiedades del elemento de texto.

- **Lista de Valores;** en esta propiedad se indicará el nombre de la LOV con la que se vincula el elemento de texto.
- **Listar Posición X, Y;** en estas dos propiedades especificaremos las coordenadas X e Y de pantalla en la que queremos mostrar la lista de valores. De introducirse valores en estas propiedades prevalecen sobre las propiedades *Posición X, Posición Y* definidas en la LOV.
- **Validad desde la Lista;** especifica la obligatoriedad de que el usuario introduzca en el elemento de texto un valor que exista entre los que retorna la LOV.

CREANDO UNA LOV 16

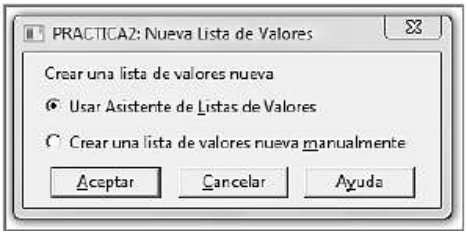
INTRODUCCIÓN

En este capítulo vamos a poner en práctica todos los conocimientos adquiridos en el capítulo anterior con objeto de crear una lista de valores (LOV) en un formulario.

Apertura de Forms Builder y formulario

Como primer paso abriremos Forms Builder y una vez hecho esto desde la barra de menús **Archivo** seleccionaremos **Abrir** el fichero **PRACTICA2.FMB** que guardamos en capítulos anteriores.

Añadir una LOV con el asistente



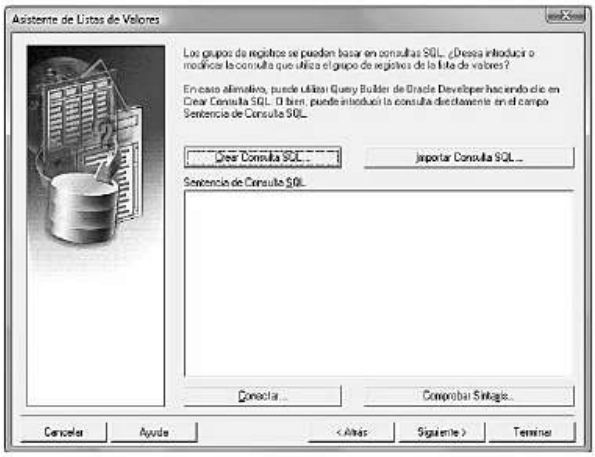
Seguidamente vamos a crear una lista de valores (LOV) de forma asistida, para ello dentro del navegador de objetos nos posicionamos en el grupo **Listas de Valores** y pulsamos sobre el botón , lo que provocará que se muestre este cuadro de diálogo:

Seleccionaremos la opción **Usar Asistente de Listas de Valores** y pulsaremos el botón **Aceptar**.

Creando un grupo de registros

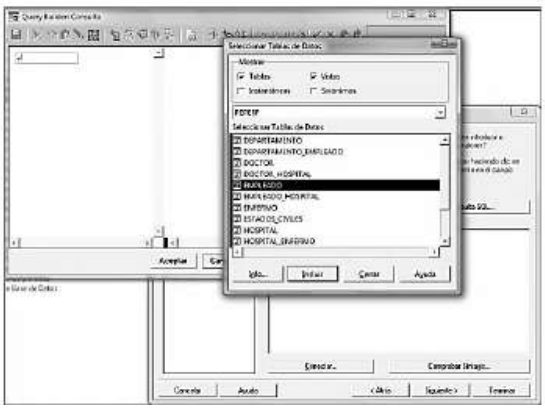
A continuación se nos muestra esta pantalla para determinar el mecanismo de creación del grupo de registros (nuevo o utilizando uno existente).

Seleccionaremos la opción **Nuevo Grupo de Registros Basado en una Consulta** y pulsaremos el botón **Siguiente,** lo que mostrará la siguiente pantalla.



En esta pantalla tenemos que indicar el mecanismo de recuperación de información del grupo de registros, teniendo dos alternativas: crear una nueva consulta SQL, o bien importar una que se haya creado anteriormente.

En nuestro caso pulsaremos sobre el botón **Crear Consulta SQL**, dado que vamos a diseñar una nueva consulta, y si nos solicita las credenciales de conexión a base de datos, indicaremos las expuestas en los capítulos anteriores de este libro.

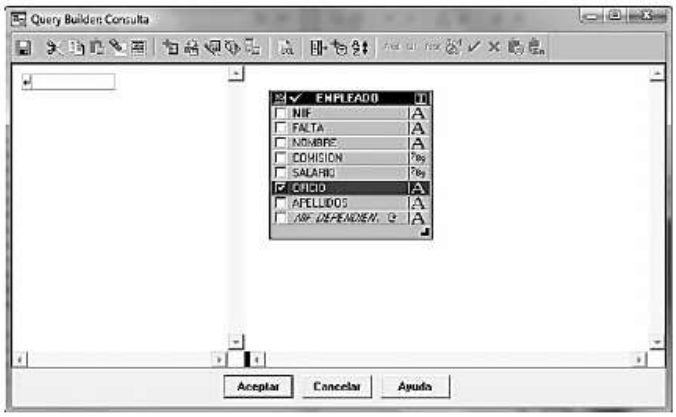


En la nueva pantalla que se nos muestra se presenta la utilidad Query Builder de Forms que permite el diseño de consultas SQL de manera asistida.

En esta práctica vamos a seleccionar la tabla **EMPLEADO** y a continuación pulsaremos el botón **Incluir,** para finalizar pulsamos el botón **Cerrar.**

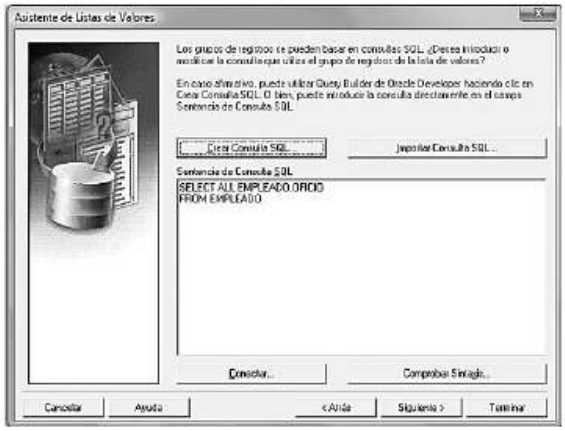
Perfilando una consulta en Query Builder

Una vez seleccionada una tabla en Query Builder es necesario indicar cuál/es son las columnas que vamos a visualizar en la consulta, y si tienen algún criterio específico para su visualización (equivaldría a la sección WHERE de una SQL).



En nuestra práctica vamos a marcar para visualizar la columna **OFICIO** y a continuación pulsaremos el botón **Aceptar,** tal como se muestra en la imagen de la siguiente página.

Observando el código generado desde Query Builder



Al salir del asistente para la creación de consultas de Forms (Query Builder) volvemos al asistente de creación de un grupo de registros, pero se nos muestra la consulta que ha generado Query Builder a partir de las indicaciones que le hemos dado.

Cambiando manualmente la consulta

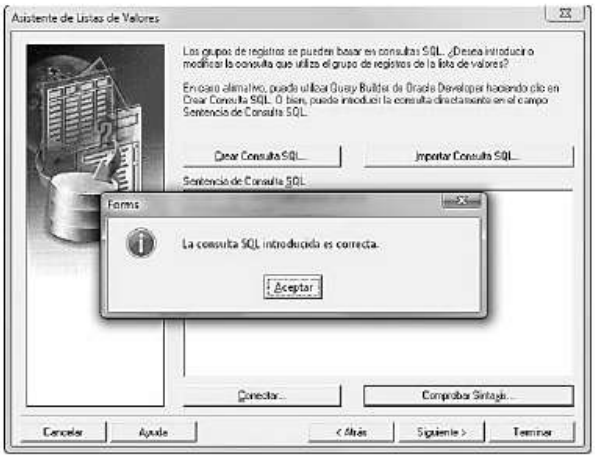
La consulta mostrada en la imagen anterior puede ser manipulada manualmente por el usuario a su discreción para completarla o ampliarla.

En nuestro caso vamos a cambiar la consulta mostrada por la siguiente:

```
SELECT DISTINCT EMPLEADO.OFICIO
FROM EMPLEADO
ORDER BY EMPLEADO.OFICIO
```

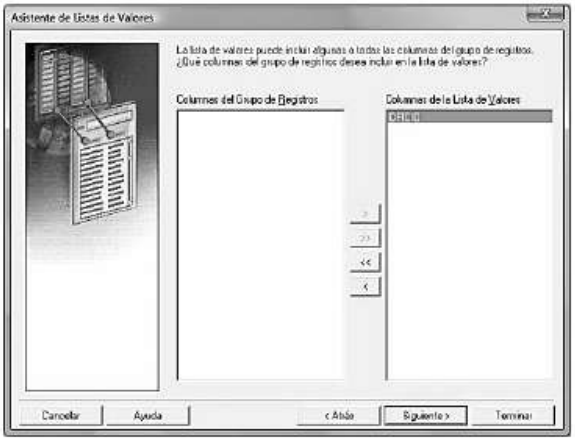
Comprobando la sintaxis de una consulta

Cuando se termina de escribir/modificar una consulta, se puede comprobar que no hay ningún error sintáctico, de acuerdo a las reglas del lenguaje SQL de Oracle, pulsando sobre el botón **Comprobar Sintaxis**. Al hacer esto, Forms nos devolverá una alerta indicando si hay o no errores.



En nuestro caso al pulsar sobre el botón **Comprobar Sintaxis** vemos que no hay errores sintácticos, por lo que pulsamos el botón **Aceptar** de la alerta y a continuación sobre el botón **Siguiente** del asistente para continuar el proceso de creación de la LOV.

Seleccionando columnas a visualizar en el LOV

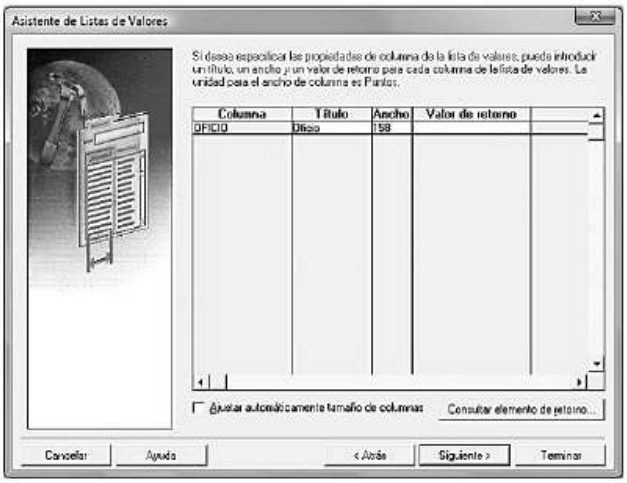


En esta fase del diseño tenemos que seleccionar de la lista de **Columnas del Grupo de Registros** aquellas que queremos visualizar en la LOV. Para ello iremos seleccionando cada columna y pulsando sobre el botón **>** para trasladarlas a la sección de visualización de la LOV, o bien pulsaremos el botón **>>** para trasladarlas todas. Esta última opción es la que vamos a utilizar en esta práctica para obtener la siguiente imagen:

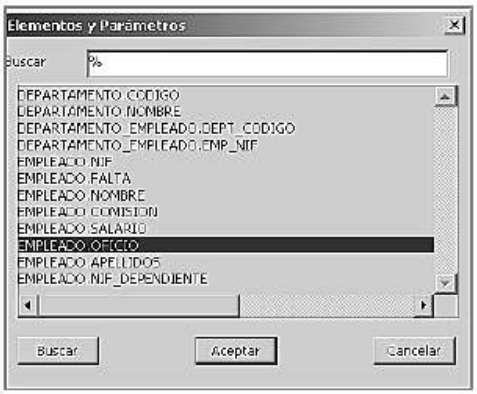
A continuación pulsamos el botón **Siguiente**.

Seleccionando elementos de retorno de la LOV

Como complemento a la visualización de columnas de la LOV al menos tenemos que indicar un elemento de retorno donde se llevará la información mostrada.

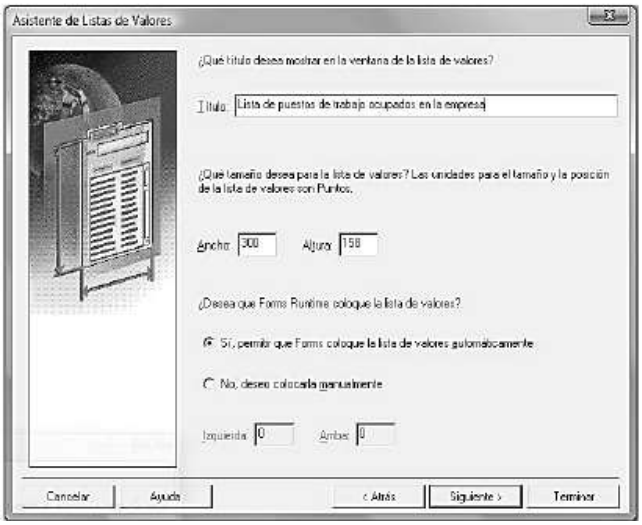


Hacemos un clic sobre el cuadro en blanco **Valor de retorno** y a continuación pulsamos sobre el botón **Consultar elemento de retorno**, lo que nos abre un nuevo cuadro de diálogo como el que se muestra aquí, donde aparecen todos los elementos de texto que existen en el formulario, para elegir uno de ellos.



En nuestra práctica seleccionamos el elemento **EMPLEADO.OFICIO** y pulsamos el botón **Aceptar**. Una vez retornados al asistente de la LOV, pulsamos el botón **Siguiente.**

Definiendo el título y tamaño de la LOV



En esta fase se muestra una ventana donde tenemos que indicar el título con el que queremos que se muestre el cuadro de diálogo de la LOV y el tamaño de la misma como se muestra en esta imagen.

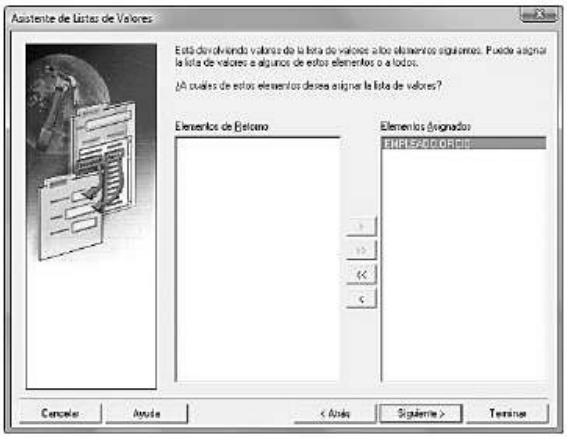
En nuestra práctica indicaremos como título: *Lista de puestos de trabajo ocupados en la empresa,* y como tamaño **300** de ancho y **158** de altura. A continuación pulsamos el botón **Siguiente.**

Definiendo el nº de registros a mostrar en una LOV

En esta fase tenemos que indicar el número de registros que queremos mostrar a la vez en el cuadro de diálogo de la LOV, así como el método de refresco de la lista, y si vamos a permitir al usuario filtrar los datos que se visualicen en ella.

En nuestra práctica seleccionaremos **10** filas a mostrar, y marcaremos la opción **Refrescar los datos del grupo de registros antes de mostrar la lista de valores**. A continuación pulsaremos el botón **Siguiente.**

Asociando la LOV a un elemento del formulario



Por último en esta fase podemos asociar la LOV a uno o varios de los elementos seleccionados como retorno en los pasos anteriores, para que cuando el cursor se encuentre dentro de ellos (en tiempo de ejecución) se visualice en la barra de estado la opción de abrir una lista de valores.

En nuestra práctica pulsaremos el botón **>>** para trasladar todos los elementos de retorno a la sección **Elementos Asignados** y pulsaremos el botón **Terminar** para cerrar el asistente de creación de la LOV.

Explorando elementos creados en el navegador

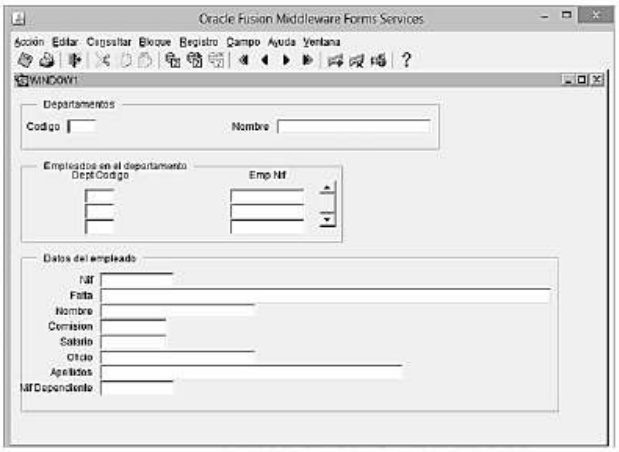
Cuando finalizamos el diseño asistido de la LOV retornamos al navegador de objetos del formulario donde aparecen los nuevos elementos que se crean. Si nos fijamos en el navegador de objetos, se habrán creado 2 nuevos elementos: una lista de valores y un grupo de registros (ambos con el mismo nombre).

Salvar, compilar, generar y ejecutar el formulario

A continuación vamos a salvar la práctica que llevamos diseñando hasta el momento. Para ello desde la barra de menús **Archivo** seleccionamos la opción **Guardar Como,** y a continuación asignamos como nombre a la práctica el siguiente: **PRACTICA5.FMB**.

Para mantener la coherencia con el nombre del archivo, también cambiamos las propiedades del módulo para que se llame **PRACTICA5.**

A continuación compilamos el código del formulario desde la barra de menús **Programa** y desde aquí seleccionamos **Compilar PL/SQL,** para seguidamente pulsar **Todo**.



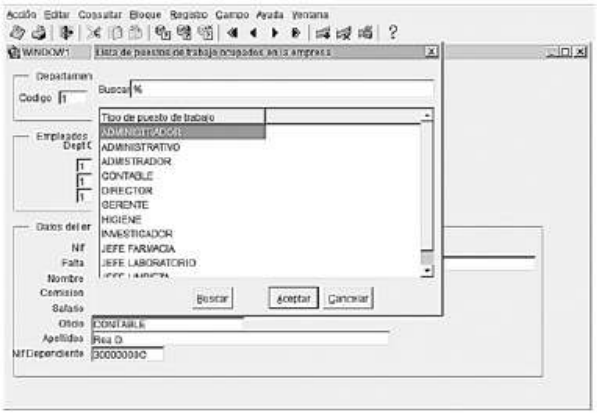
Una vez compilado generamos el ejecutable del formulario desde la barra de menús **Programa** y desde aquí seleccionamos **Compilar Módulo**.

Por último ejecutamos el formulario desde la barra de menús **Programa** y seleccionamos **Ejecutar Pantalla** para que se muestre una imagen como la siguiente.

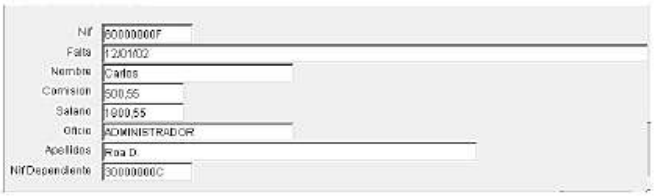
Pruebas de funcionalidad

Con el formulario ejecutado vamos a realizar una serie de pruebas de funcionalidad del mismo, siguiendo estos pasos:

- Ejecutamos una consulta sin restricciones (filtros), para ello desde la barra de menú **Consultar** elegimos **Introducir**. Y a continuación desde el mismo menú **Consultar** elegimos **Ejecutar**.
- Una vez visualizados los datos nos situamos dentro del campo **Oficio** de los *Datos del empleado* y pulsamos sobre el menú **Editar** la opción **Mostrar Lista** para ver la lista de valores creada en la práctica, mostrándose la imagen siguiente:



- Una vez visualizada la lista de valores seleccionamos el oficio **ADMINISTRADOR** y pulsamos **Aceptar**, lo que provoca que se actualice el contenido del campo **Oficio.**



Concluidas estas pruebas cerramos el formulario en ejecución sin guardar los cambios y volvemos al Navegador de Objetos.

Revisando las propiedades de los elementos de la LOV

Dentro del navegador de objetos nos situamos en el grupo *Lista de Valores*, desplegamos el contenido y seleccionamos el único elemento que se muestra. Una vez realizado esto abrimos la paleta de propiedades.

Podemos observar que la propiedad **Grupo de Registros** contiene el nombre del elemento correspondiente en el que se encuentra la consulta para mostrar la información de la LOV.

Pulsamos después en la propiedad **Propiedades de Correspondencia de Columnas** donde se muestran la correlación entre los elementos del grupo de registros y los elementos de los bloques de datos del formulario.

Dentro del cuadro de diálogo anterior vamos a cambiar el título de la columna para ponerle "*Tipo de puesto de trabajo*" y en el ancho de visualización de la misma indicaremos el valor *150.* Seguidamente pulsamos el botón **Aceptar** para volver a la paleta de propiedades.

Dentro de la propiedad **Ancho** cambiamos el valor a *320* y a la propiedad **Altura** le indicamos el valor *250*.

A continuación nos situamos nuevamente en el navegador de objetos y desplegamos el grupo *Grupos de Registros* del cual seleccionamos también el único elemento que se muestra. Una vez realizado esto abrimos la paleta de propiedades.

Podemos observar cómo en la propiedad **Consulta de Grupo de Registros** se muestra la consulta SQL que se ha diseñado durante la práctica con el asistente Query Builder de Forms. Si fuese necesario, desde esta propiedad también se podría cambiar manualmente la consulta. De hacer modificaciones manuales en la misma, no dispondremos de un corrector sintáctico como ocurre dentro del asistente, pero igualmente Forms no permitirá insertar una consulta que contenga errores. Si los hubiese mostraría el mensaje de la imagen adjunta.

Para comprobar la funcionalidad de estas modificaciones efectuadas en las propiedades compilamos el código, lo guardamos, generamos el ejecutable y ejecutamos el formulario. Con el formulario ejecutado vamos a realizar una serie de pruebas de funcionalidad del mismo, siguiendo estos pasos:

- Ejecutamos una consulta sin restricciones (filtros), para ello desde la barra de menú **Consultar** elegimos **Introducir**. Y a continuación desde el mismo menú **Consultar** elegimos **Ejecutar**.
- Una vez visualizados los datos nos situamos dentro del campo **Oficio** de los *Datos del empleado* y pulsamos sobre el menú **Editar** la opción **Mostrar Lista** para ver cómo han afectado los cambios en las propiedades de la lista de valores:

Finalizadas las pruebas cerramos la ventana de ejecución del formulario y al volver al navegador de objetos guardamos los cambios del formulario **PRACTICA5.FMB.**

ELEMENTOS DE ENTRADA 17

INTRODUCCIÓN

Aparte de los elementos de texto que hemos visto en capítulos anteriores, dentro de Forms existen otros elementos que aceptan entrada de datos por parte del usuario. Entre estos tipos de elementos se incluyen los siguientes:

- Casillas de control (Check Box).
- Elementos de Listas (List Item).
- Grupo de botones de radio (Radio Groups).

La característica principal de un elemento de entrada es la posibilidad de interactuar con la base de datos mediante alguna de estas operaciones:

- Insertar valores.
- Actualizar valores existentes.
- Suprimir valores existentes.
- Consultar valores existentes.

Además de estas funciones predefinidas para los elementos de entrada, se puede ampliar la funcionalidad mediante la creación de disparadores y unidades de programa PL/SQL.

CASILLAS DE CONTROL (CHECK BOX)

Una casilla de control o Check Box es un objeto de un formulario que consta de dos estados que indican si un valor determinado está activado o no activado.

Se pueden utilizar las casillas de control para mejorar la interfaz de usuario convirtiendo elementos existentes. Aunque una casilla de control está limitada a dos estados, no se limita solo a dos valores. En una casilla de control hay que especificar los siguientes valores:

- El valor para representar el estado *Activado*.
- El valor para representar el estado *No Activado*.
- Cómo se procesan o representan otros valores.

Uso de un Check Box en tiempo de ejecución

En tiempo de ejecución se puede operar con las casillas de control de la siguiente forma:

- Definiendo un valor sobre la propia casilla (activándola o no con un clic del ratón).
- Modificándose automáticamente el valor de una casilla de control mediante programación PL/SQL incluida en el formulario.
- También es posible utilizar una casilla de control en el modo *Introducir Consulta*:
    - Consulte los valores activados haciendo clic una o más veces hasta que el elemento esté activado.
    - Consulte los valores no activados haciendo clic una o más veces hasta que el elemento no esté activado.
    - Para ignorar los valores de la casilla de control en el modo *Introducir Consulta* no haga clic sobre la casilla dejando el valor mostrado de inicio.

Propiedades del Check Box

A continuación se relacionan las propiedades que permiten modificar el aspecto y el comportamiento de una casilla de control.

SECCIÓN DATOS

Dentro de esta sección se incluyen las siguientes propiedades de una casilla de control:

- **Tipo de Datos;** debe ser compatible con los valores especificados en las propiedades que empiezan con la palabra *Valor*.
- **Valor Inicial;** valor de inicio con el que se mostrará la casilla de control (activada o no).

SECCIÓN FUNCIONAL

Dentro de esta sección se incluyen las siguientes propiedades de una casilla de control:

- **Etiqueta;** texto que se muestra junto a la casilla de control para dar título a la misma dentro del formulario.
- **Clave de acceso;** combinación de teclas que se pueden utilizar para navegar hasta este elemento, así como para activarlo o desactivarlo.
- **Valor al Activar;** valor que representa el estado activado de la casilla de control.
- **Valor cuando No está Activado;** valor que representa el estado No activado de la casilla de control.
- **Correspondencia de Casillas de Control de Otros Valores;** determina cómo se van a procesar otros posibles valores que se recuperen de la base de datos para la casilla de control.

SECCIÓN NAVEGACIÓN

Dentro de esta sección se incluyen las siguientes propiedades de una casilla de control:

- **Navegación del Mouse;** determina si Forms puede navegar hasta el elemento y mover el foco de entrada cuando el usuario hace un clic con el ratón sobre el mismo. El valor por defecto de esta propiedad es *Sí.*

Tratamiento de otros valores

Si la columna de la tabla origen a la que está vinculada la casilla de control acepta otros valores, la casilla de control los debe tener en cuenta.

Se puede asignar otros valores al estado activado o no activado mediante la propiedad **Correspondencia de Casillas de Control de Otros Valores**. Como opción, puede elegir no aceptar los demás valores con el valor *No Permitido*.

Tratamiento de valores nulos

Si la columna de la tabla origen a la que está vinculada la casilla de control acepta valores nulos, los puede tratar de una de las siguientes formas:

- Defina la propiedad C**orrespondencia de Casillas de Control de Otros Valores**.
- Defina el estado *Activado* o N*o Activado* para representar un valor nulo (deje el valor de alguna de estas propiedades en blanco).

MODIFICACIÓN EN LA BASE DE DATOS:

Para poder seguir las prácticas propuestas en este curso debe de efectuar una alteración en la tabla EMPLEADO de la base de datos, según se especifica a continuación:

- Ejecute SQL*PLUS o cualquier otra herramienta de conexión a la base de datos e intérprete de comandos SQL.
- Conéctese con las credenciales PEPERP/PEPITO (usuario y contraseña respectivamente).
- Ejecute la siguiente sentencia SQL:

```
ALTER TABLE EMPLEADO
ADD (DESEMPLEADO CHAR(1) CONSTRAINT
CK_EMP_DESEMPLEADO CHECK (DESEMPLEADO IN ('S','N'))
);
```

ELEMENTOS DE LISTA (LIST BOX)

Un elemento de lista o List Box es un objeto de un formulario que muestra un conjunto de opciones predefinido, cada una de ellas correspondiente a un valor de datos específico. Los elementos de lista se utilizan en tiempo de ejecución para seleccionar un único valor. Los elementos o las opciones de lista se excluyen mutuamente, por tanto solo se puede elegir uno a la vez.

Estilos de un List Box

Un List Box se puede representar en un formulario con tres estilos diferentes:

- Lista emergente.
- Lista de texto.
- Recuadro combinado.

A nivel de reducción de espacio en el diseño de un formulario hay que decir que la lista emergente y el recuadro combinado son los que menos ocupan.

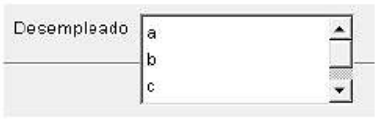
Por el contrario en cuanto a funcionalidad de visualización de resultados para el usuario, la lista de texto es la más eficiente al mostrar varios valores a la vez.

LISTA EMERGENTE

Se muestra como un elemento de texto pero con una flecha selectora de los elementos de la lista tal y como se muestra en esta imagen.

Cuando se pulsa sobre la flecha se muestran todos los elementos contenidos en la lista.

LISTA DE TEXTO



Se muestra como un elemento de texto (sin flecha selectora) con los elementos de la lista tal y como se muestra en esta imagen.

Cuando el área de visualización de la lista de texto no es lo suficientemente grande para mostrar todos los elementos, automáticamente Forms presenta en tiempo de ejecución una barra de desplazamiento para movernos entre sus elementos.

RECUADRO COMBINADO

Se muestra igual que una lista emergente pero con la diferencia que un recuadro combinado se permite añadir registros a la lista por parte del usuario.

Propiedades de un List Box

A continuación se relacionan las propiedades que permiten modificar el aspecto y el comportamiento de un List Box.

SECCIÓN FUNCIONAL

Dentro de esta sección se incluyen las siguientes propiedades de un List Box:

- **Elementos en Lista;** define los valores que se permitirán seleccionar en la lista así como el título que se mostrará en tiempo de ejecución para cada uno de ellos. Al entrar en esta propiedad se muestra este cuadro de diálogo.

  En el apartado *Elementos de Lista* se introducirá el título que queremos que se muestre en tiempo de ejecución para cada elemento de la lista (este texto es el que verá y seleccionará el usuario).

  En el apartado *Valor de Elemento de Lista* se introducirá el valor real que se asocia a cada elemento de la lista (este será el valor que se almacenará / leerá de la base de datos).

- **Estilo de Lista;** permite seleccionar entre uno de los tres estilos de lista que se han indicado anteriormente: *Lista emergente (Poplist), Lista de texto (Tlist) o Recuadro combinado (Combo Box).*
- **Correspondencia de otros valores;** determina de qué forma se van a procesar otros valores que puedan llegar al elemento de lista que no hayan sido contemplados en la propiedad *Elementos en Lista*.

SECCIÓN NAVEGACIÓN

Dentro de esta sección se incluyen las siguientes propiedades de un List Box:

- **Navegación del Mouse;** permite que el usuario navegue con el ratón hasta el elemento, en tiempo de ejecución.

Valores nulos en un List Box

Si la columna de la tabla origen a la que está vinculada un elemento de lista acepta valores nulos, Forms Builder creará una pseudoopción en la lista para representar el valor nulo.

En cualquiera de los tres estilos de un List Box se mostrará el campo en blanco cuando se obtenga un valor nulo para el elemento de la lista.

Si la propiedad *Necesario* de un elemento de lista tiene valor *No*, se pueden dar las siguientes situaciones respecto a los valores nulos:

- Una lista emergente muestra un elemento en blanco para representar un valor nulo.
- El usuario puede omitir seleccionar un valor concreto del elemento de la lista o puede limpiar el contenido de la misma. De esta manera se está seleccionando el valor NULL para la lista.
- Un recuadro combinado no muestra elementos en blanco, por lo que el usuario deberá seleccionar uno de los valores mostrados y luego suprimir el contenido, de esta forma se consigue obtener el valor NULL.

Manejo de otros valores en un List Box

Si la columna de la tabla origen a la que está vinculada un elemento de lista acepta valores distintos a los asociados en la propiedad *Elementos en Lista*, se debe especificar en Forms Builder cómo se desean tratar estos valores en List Box. Para ello se pueden realizar las siguientes acciones:

- No permitir valores no definidos en la propiedad *Elementos en Lista.* Para ello se dejará en blanco la propiedad *Correspondencia de Otros Valores.*
- Permitir otros valores. Para ello será necesario introducir el valor permitido en la propiedad *Correspondencia de Otros Valores.*

BOTONES DE RADIO (RADIO BUTTON)

Un botón de radio solo se puede entender en el contexto de un grupo de botones de radio, que es el elemento que aglutina todos los botones dentro del abanico de opciones que se quiera definir, cada uno de los cuales con un valor distinto.

En un grupo de botones de radio, sus valores, y por tanto sus botones de radio correspondientes, se excluyen mutuamente.

Los usos y ventajas de los grupos de botones de radio son los siguientes:

- Ofrecen una alternativa de elección entre dos o más valores estáticos.
- Ofrecen una alternativa a elementos de lista con dos o tres opciones.

Propiedades de un grupo de Botones de Radio

Se pueden definir las siguientes propiedades más importantes de un elemento de tipo grupo de botones de radio.

SECCIÓN DATOS

Dentro de esta sección se incluyen las siguientes propiedades de un grupo de botones de radio:

- **Tipo de Datos;** define los tipos de valores que se admiten en los distintos botones de radio. Tanto la propiedad *Correspondencia de Otros Valores* como cada valor de los botones de radio, deben de ser compatibles con el tipo definido aquí.
- **Valor Inicial;** permite definir el valor inicial del grupo. En la práctica equivale a indicar cuál es el botón de radio que queremos que aparezca señalado cuando entremos en el grupo.

SECCIÓN FUNCIONAL

Dentro de esta sección se incluyen las siguientes propiedades de un grupo de botones de radio.

- **Correspondencia de Otros Valores;** esta propiedad permite procesar otros valores distintos a los especificados en cada botón de radio.

SECCIÓN NAVEGACIÓN

Dentro de esta sección se incluyen las siguientes propiedades de un grupo de botones de radio.

- **Correspondencia de Otros Valores;** determina si Forms navega hasta el elemento cuando el usuario activa el grupo de opciones de radio con el ratón.

Propiedades de un Botón de Radio

Se pueden definir las siguientes propiedades más importantes de un elemento de tipo botón de radio.

SECCIÓN FUNCIONAL

Dentro de esta sección se incluyen las siguientes propiedades de un botón de radio.

- **Etiqueta;** define el texto que aparece adyacente al botón de radio.
- **Clave de Acceso;** permite especificar las combinaciones de teclas que se pueden utilizar para navegar hasta el botón de radio y manipularlo.
- **Valor de Botón de Radio;** determina el valor del botón de radio de acuerdo al tipo seleccionado en el grupo de botones. El grupo de botones de radio asumirá este valor de botón cuando esté seleccionado.

MODIFICACIÓN EN LA BASE DE DATOS:

Para poder seguir las prácticas propuestas en este curso debe de efectuar una alteración en la tabla EMPLEADO de la base de datos, según se especifica a continuación:

- Ejecute SQL*PLUS o cualquier otra herramienta de conexión a la base de datos e intérprete de comandos SQL.
- Conéctese con las credenciales PEPERP/PEPITO (usuario y contraseña respectivamente).
- Ejecute la siguiente sentencia SQL:

```
ALTER TABLE EMPLEADO
ADD (NIVEL_ESTUDIOS  NUMBER(1));
```

Manejo de otros valores de un grupo de Botones de Radio

Si la columna de la tabla origen a la que está vinculada el grupo de botones de radio acepta valores distintos a los asociados a cada uno de los botones de un grupo radio, se debe utilizar uno de los siguientes métodos para especificar el comportamiento del grupo radio:

- Ignorar otros valores, dejando en blanco la propiedad *Correspondencia de Otros Valores* del grupo de botones de radio.
- Asociando los otros valores a uno de los botones de radio existentes. Para ello habrá que indicar le nombre del botón para el que se asocian estos valores en la propiedad *Correspondencia de Otros Valores.*

Valores nulos de un grupo de Botones de Radio

Un grupo de botones de radio puede tratar un valor nulo como válido siempre y cuando la columna de la tabla origen a la que está vinculada, lo admita. El mecanismo para realizarlo es a través de una de las siguientes formas:

- Utilizando la propiedad *Correspondencia de Otros Valores* para forzar implícitamente el valor nulo en un botón de radio.
- Asignando el valor nulo a uno de los botones de radio. Para hacer esto hay que dejar en blanco la propiedad *Valor de Botón de Radio* del botón correspondiente.

CREANDO UN CHECK BOX 18

INTRODUCCIÓN

En este capítulo vamos a poner en práctica los conocimientos adquiridos en el capítulo anterior sobre las casillas de control (Check Box).

Apertura de Forms Builder y formulario

Como primer paso abriremos Forms Builder y una vez hecho esto desde la barra de menús **Archivo** seleccionaremos **Abrir** el fichero **PRACTICA5.FMB** que guardamos en capítulos anteriores.

Crear un bloque con el asistente

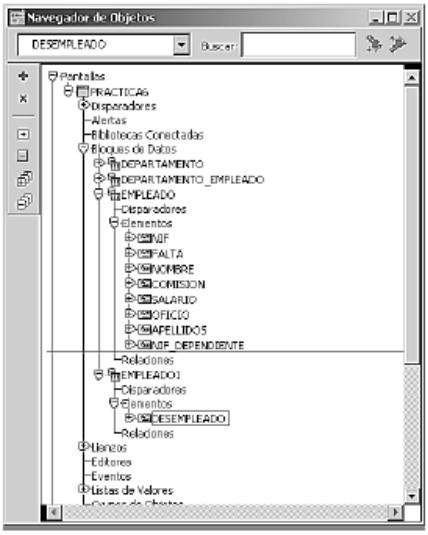
Seguidamente nos situamos dentro del navegador de objetos en el bloque EMPLEADO y pulsamos el botón  , para añadir un bloque nuevo utilizando el asistente de bloques. Dentro del asistente tenemos que seleccionar como origen de datos la tabla EMPLEADO. De todas las columnas disponibles de la tabla, únicamente seleccionaremos el campo DESEMPLEADO para manejar en el bloque.

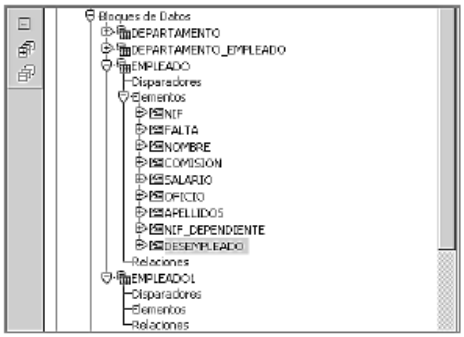
A continuación pulsamos el botón **Siguiente** sucesivamente hasta llegar a la pantalla en la que se solicita crear solo el bloque o crear el bloque y llamar al asistente de diseño. En esta pantalla seleccionaremos la opción **Crear sólo el bloque de datos** y pulsaremos el botón **Terminar.**

Movimientos de elementos entre bloques

Seguidamente desplegamos los bloques de datos EMPLEADO y EMPLEADO1 en el navegador de objetos.

A continuación nos situamos en el bloque *EMPLEADO1* dentro del elemento *DESEMPLEADO* y hacemos un clic con el botón izquierdo del ratón. Sin soltar el clic del ratón movemos el campo *DESEMPLEADO* desde el bloque *EMPLEADO1* al bloque *EMPLEADO* situándolo justo por debajo del campo *NIF_DEPENDIENTE*, tal cual se muestra en la secuencia de imágenes siguiente.





Cambiando las propiedades de un elemento

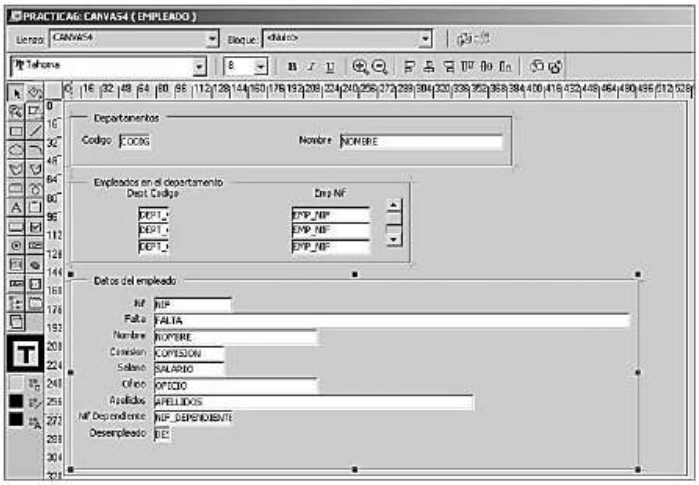
A continuación vamos a modificar las propiedades del elemento *DESEMPLEADO* del bloque *EMPLEADO* y para ello dentro del menú **Herramientas** pulsamos sobre **Paleta de Propiedades.**

Dentro de la paleta de propiedades nos situamos en la sección **Física** y cambiamos la propiedad **Lienzo** de <Nulo> (que es el valor que tiene) a *CANVAS4.*

Borrando un bloque y guardando el formulario

Cerramos la paleta de propiedades y nos situamos en el bloque *EMPLEADO1*. A continuación pulsamos el botón  para eliminarlo. Y realizada esta operación, almacenamos el formulario con el nombre **PRACTICA6.FMB** modificando previamente también el nombre del formulario.

Redimensionando un marco contenedor de elementos



A continuación dentro del menú **Herramientas** pulsamos en la opción **Editor de Diseño** para poder redimensionar el marco *Datos del empleado* a fin de incluir el campo *DESEMPLEADO* dentro del mismo. Al ampliar dicho marco hacia abajo, automáticamente se presenta el campo *DESEMPLEADO* como se muestra en esta imagen.

Modificando el tipo de elemento

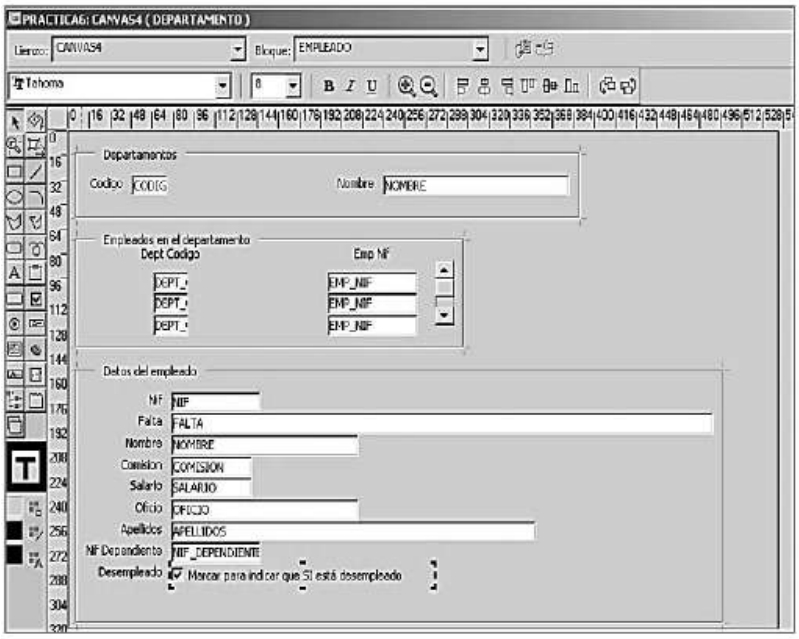
Sin abandonar el editor de diseño nos situamos en el campo *DESEMPLEADO* (dentro de la casilla que se utiliza para introducir datos) y dentro del menú **Herramientas** pulsamos en la opción **Paleta de Propiedades.**

Dentro de la paleta de propiedades nos situamos en la sección **General,** y en la propiedad **Tipo de Elemento** cambiamos el tipo actual por el de **Casilla de Control.**

Al cambiar el tipo de elemento, las propiedades del campo *DESEMPLEADO* también variarán para adaptarse al formato de la casilla de control, por lo que tenemos que dar los valores correspondientes a las mismas según se indica a continuación:

- Propiedad *Etiqueta:* introduciremos el texto **Marcar para indicar que SÍ está desempleado.**
- Propiedad *Valor al Activar:* indicamos el valor **S.**
- Propiedad *Valor cuando No Está Activado*: indicamos el valor **N.**
- Propiedad *Correspondencia de Casillas de Control de Otros Valores:* indicamos la opción **Desactivado.**

Redimensionando una casilla de control

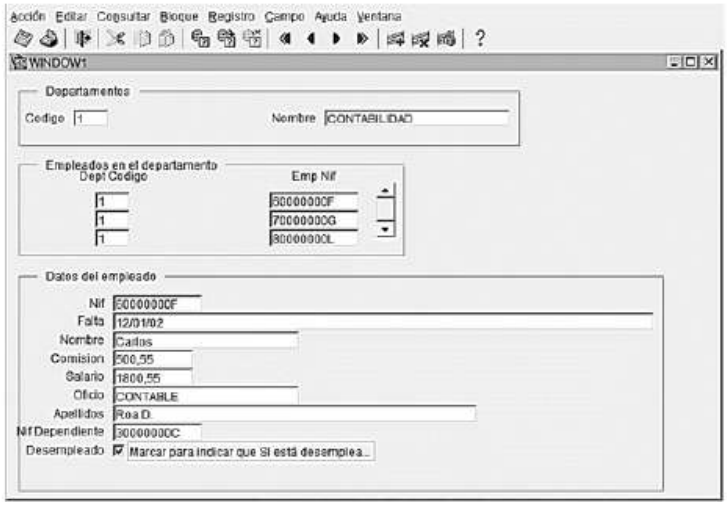


Una vez modificadas las propiedades anteriores del campo *DESEMPLEADO*, cerramos la paleta de propiedades y al volver al editor de diseño podemos comprobar cómo el texto que hemos asociado en la propiedad *Etiqueta* no se visualiza en la pantalla. Esto es debido a que el campo no tiene el tamaño apropiado para poderlo mostrar, por lo que hay que ampliar el marco de visualización de la etiqueta. Para llevarlo a cabo hacemos un clic con el botón izquierdo del ratón sobre la casilla Desempleado (la que aparece con el símbolo √) y estiramos hacia la derecha (desde cualquiera de las esquinas punteadas, hasta que consigamos ver por completo el texto de la etiqueta. Tendremos que llegar a conseguir un efecto similar al que se muestra en la imagen adjunta.

Guardando, compilando y ejecutando el formulario

Concluidas las modificaciones anteriores almacenamos el formulario **(CRTL + S)**, lo compilamos **(SHIFT + CTRL + K)** y generamos **(CTRL + T)**, para en último lugar ejecutarlo **(CTRL + R)** y comprobar cómo han repercutido en el formulario las modificaciones que hemos efectuado.

Se han incluido en este apartado los métodos abreviados (mediante la pulsación de combinaciones de teclas) para llevar a cabo las acciones más comunes dentro de un formulario: guardar, compilar, generar y ejecutar.



Una vez ejecutado el formulario realizamos una consulta general (sin restricciones), lo que hará que se muestre el primer empleado almacenado (el que tiene NIF 60000000F) en el departamento 1. A continuación le marcamos la casilla *Desempleado* como se muestra en la imagen anterior y almacenamos los cambios en la base de datos (**Acción – Guardar**), lo que provocará que se muestre en la barra de estado del formulario el siguiente mensaje: **FRM-40400: Transacción terminada: 1 registros aplicados y guardados**.

Con este paso damos por concluida esta práctica y cerramos las ventanas de ejecución y diseño del formulario.

CREANDO UN LIST ITEM 19

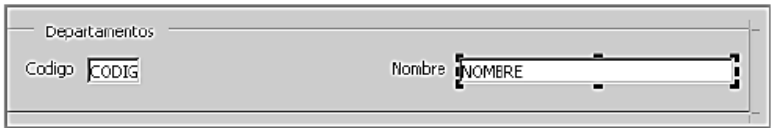
INTRODUCCIÓN

En este capítulo vamos a poner en práctica los conocimientos adquiridos en capítulos anteriores sobre los elementos de lista (List Item).

Apertura de Forms Builder y formulario

Como primer paso abriremos Forms Builder y una vez hecho esto desde la barra de menús **Archivo,** seleccionaremos **Abrir** el fichero **PRACTICA6.FMB** que guardamos en el capítulo anterior.

Eliminando elementos de un bloque



Seguidamente abrimos el **Editor de Diseño** y nos situamos en el marco *Departamentos*, donde marcamos el campo *Nombre* (del departamento) como se muestra en la siguiente imagen.

Para eliminarlo, desde el menú **Editar** seleccionamos la opción **Borrar**. Esto hace que se elimine el campo del canvas (ventana y tapiz de presentación de información en el formulario) y del propio formulario (se elimina también del bloque contenedor del navegador de objetos).

Cambiando las propiedades de un elemento

Sin abandonar el **Editor de Diseño** marcamos el campo *Código* dentro del marco *Departamentos* y a continuación abrimos sus propiedades (**Herramientas – Paleta de Propiedades)**.

A continuación cambiamos la propiedad *Tipo de Elemento* al valor **Elemento de Lista.**

Asignando valores a una lista

Sin abandonar la **Paleta de Propiedades** vamos a asociar valores estáticos a la propiedad *Elementos de Lista*. Los valores que hay que introducir son los siguientes:

Valor a visualizar	Valor de Base de Datos
ACCOUNTING (Contabilidad)	1
RESEARCH (Investigación)	2
SALES (Ventas)	3
OPERATIONS (Logística)	4
OTHERS (Pruebas)	5

Para finalizar la introducción de elementos en la lista, pulsamos el botón **Aceptar** de la imagen mostrada.

Lo que hemos conseguido con esta acción es proporcionar al usuario una manera intuitiva de identificar los distintos departamentos que se han codificado en la base de datos con 1 dígito numérico.

Correspondencia de otros valores de la lista

Sin abandonar la **Paleta de Propiedades** cambiamos la propiedad *Correspondencia de Otros Valores* al valor **5**. De esta forma conseguimos que todos los valores que se puedan leer en la base de datos tengan una correspondencia con los literales de la lista, al asignarle uno de los valores contemplados en la misma.

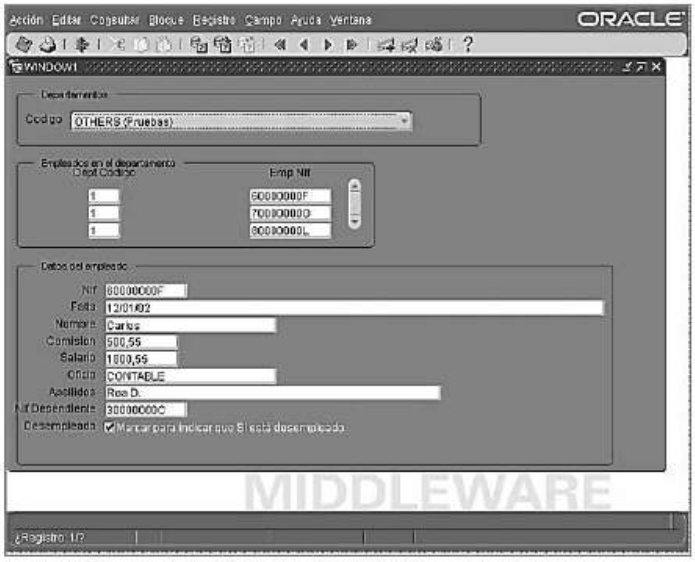
Redimensionando el campo

A continuación cerramos la paleta de propiedades. Al volver al editor de diseño observamos cómo el tamaño del campo *Código* es insuficiente para mostrar la

información de los elementos de la lista, por tanto tenemos que ampliarlo de igual manera que lo hicimos en el capítulo anterior con el campo de tipo Check Box.

Guardando, compilando y ejecutando el formulario

Concluidas las modificaciones anteriores almacenamos el formulario con el nombre **PRACTICA7.FMB**, cambiando también la propiedad *Nombre* del módulo para que sea **PRACTICA7.** A continuación lo compilamos **(SHIFT + CTRL + K)** y generamos **(CTRL + T)**, para en último lugar ejecutarlo **(CTRL + R)** y comprobar cómo han repercutido en el formulario las modificaciones que hemos efectuado.



Una vez ejecutado el formulario realizamos una consulta general (sin restricciones) lo que hará que se muestre el primer empleado almacenado (el que tiene NIF 60000000F) en el departamento 1. Si nos fijamos en el marco *Departamentos* podemos comprobar que se muestra el departamento **OTHERS (Pruebas)**.

Cambiando el estilo de la lista (Text List)

Cerramos la ventana de ejecución del formulario y volvemos al navegador de objetos de Forms Builder. Desplegamos los elementos del bloque **DEPARTAMENTO** y abrimos las propiedades del campo **CODIGO.**

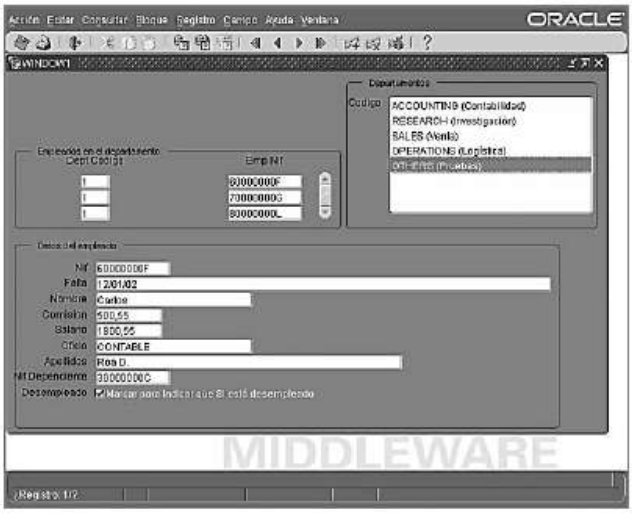
A continuación cambiamos el valor de la propiedad *Estilo de Lista* a **Lista de Texto.**

Ensanchando la lista y el marco

A continuación cerramos la paleta de propiedades y abrimos el **Editor de diseño (F2)**. Al abrirlo nos encontramos con que el campo de tipo lista no ha modificado su

tamaño, pero el estilo de lista seleccionado (lista de texto), requiere de mayor tamaño del campo, por lo que tenemos que ensanchar el marco contenedor *Departamentos* y el propio campo.

Guardando, compilando y ejecutando el formulario

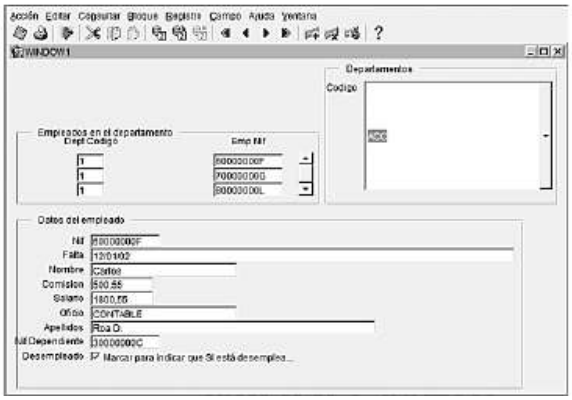


Concluidas las modificaciones anteriores almacenamos el formulario **(CTRL + S),** lo compilamos **(SHIFT + CTRL + K)** y generamos **(CTRL + T)**, para en último lugar ejecutarlo **(CTRL + R)** y comprobar cómo han repercutido en el formulario las modificaciones que hemos efectuado. El resultado de la ejecución después de introducir una consulta sin restricciones será similar al mostrado en la imagen siguiente.

Cambiando el estilo de la lista (Combo Box)

Cerramos la ventana de ejecución del formulario y volvemos al editor de diseño de Forms Builder. Seleccionamos el campo **CODIGO** del marco *Departamentos* y abrimos la paleta de propiedades (**F4)**. A continuación cambiamos el valor de la propiedad *Estilo de Lista* a **Cuadro Combinado**. Seguidamente cambiamos la propiedad *Valor Inicial* a **5**.

Guardando, compilando y ejecutando el formulario



Concluidas las modificaciones anteriores almacenamos el formulario (**CTRL + S),** lo compilamos **(SHIFT + CTRL + K)** y generamos **(CTRL + T)**, para en último lugar ejecutarlo **(CTRL + R)** y comprobar cómo han repercutido en el formulario las modificaciones que hemos efectuado.

CREANDO UN RADIO BUTTON 20

INTRODUCCIÓN

En este capítulo vamos a poner en práctica los conocimientos adquiridos en capítulos anteriores sobre los botones de radio (Radio Button) y los grupos de botones.

Apertura de Forms Builder y formulario

Como primer paso abriremos Forms Builder y una vez hecho esto, desde la barra de menús **Archivo** seleccionaremos **Abrir** el fichero **PRACTICA7.FMB** que guardamos en el capítulo anterior.

Creando un nuevo bloque asistido

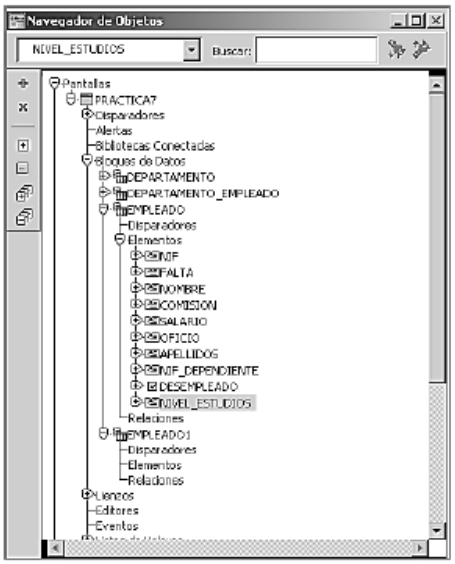
Seguidamente desplegamos los *Bloques de Datos* del navegador de objetos y nos situamos en el bloque **EMPLEADO.** A continuación añadimos un nuevo bloque y seleccionamos la opción **Usar el Asistente de Bloques de Datos.**

Pulsamos el botón **Aceptar** para comenzar la creación del nuevo bloque, y pulsamos **Siguiente** en la pantalla de bienvenida al asistente de bloques de datos. En cuanto al formato de presentación del bloque seleccionamos la opción **Tabla o Vista** y pulsamos el botón **Siguiente.**

A continuación pulsamos el botón **Examinar,** y de la lista de tablas del usuario PEPERF (una vez introducidas las credenciales de conexión a la base de datos), seleccionamos la tabla **EMPLEADO**. De los campos disponibles únicamente elegimos el campo NIVEL_ESTUDIOS.

Pulsamos en esta pantalla y en las sucesivas el botón **Siguiente** hasta llegar a la pantalla en la que se nos solicita crear el bloque solo o crear el bloque e invocar al asistente de diseño, en nuestro caso seleccionamos la opción **Crear sólo el bloque de datos** y pulsamos el botón **Terminar.**

Mover elementos entre bloques



Después del paso anterior se crea un nuevo bloque de datos denominado **EMPLEADO1** que tiene como único elemento el campo **NIVEL_ESTUDIOS**. Marcamos este campo y lo movemos al bloque EMPLEADO justo por debajo del campo *DESEMPLEADO* del mismo, tal y como se muestra en esta imagen.

Eliminando un bloque

A continuación nos situamos en el bloque **EMPLEADO1** y lo borramos (**Editar – Borrar**).

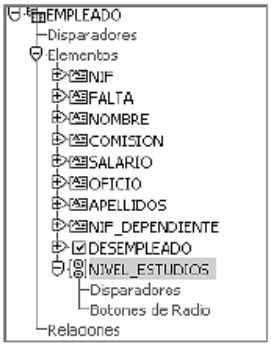
Guardando el formulario

Concluidas las modificaciones anteriores almacenamos el formulario con el nombre **PRACTICA8.FMB**, cambiando también la propiedad *Nombre* del módulo para que sea **PRACTICA8.**

Cambiando el tipo de elemento

Seguidamente nos situamos dentro del navegador de objetos en el campo **NIVEL_ESTUDIOS** del bloque EMPLEADO y abrimos sus propiedades (**F4**). Dentro de ellas cambiamos la propiedad *Tipo de Elemento* al valor **Grupo de Botones de Radio.**

Creando Botones de Radio



A continuación cerramos la paleta de propiedades y volvemos al navegador de objetos para desplegar el bloque *EMPLEADO* y el elemento *NIVEL_ESTUDIOS* como se muestra en esta imagen.

Seguidamente nos situamos en el grupo **Botones de Radio** del campo *NIVEL_ESTUDIOS* y añadimos un nuevo botón (**Editar – Crear)**.

A continuación nos situamos en el nuevo botón de radio y abrimos sus propiedades (**F4**), para cambiar los valores que se indican a continuación:

- Propiedad *Nombre:* **SIN_ESTUDIOS**
- Propiedad *Etiqueta:* **Sin estudios**
- Propiedad *Valor de Botón de Radio:* **00**

Seguidamente creamos 3 botones de radio más, siguiendo los mismos pasos indicados anteriormente y con las siguientes características:

- Segundo Botón:
  - Propiedad *Nombre:* **ENSEÑANZA_PRIMARIA**
  - Propiedad *Etiqueta:* **Enseñanza primaria**
  - Propiedad *Valor de Botón de Radio:* **1**
- Tercer Botón:
  - Propiedad *Nombre:* **ENSEÑANZA_SECUNDARIA**
  - Propiedad *Etiqueta:* **Enseñanza secundaria**
  - Propiedad *Valor de Botón de Radio:* **2**
- Cuarto Botón:
  - Propiedad *Nombre:* **ENSEÑANZA_UNIVERSITARIA**
  - Propiedad *Etiqueta:* **Enseñanza universitaria**
  - Propiedad *Valor de Botón de Radio:* **3**

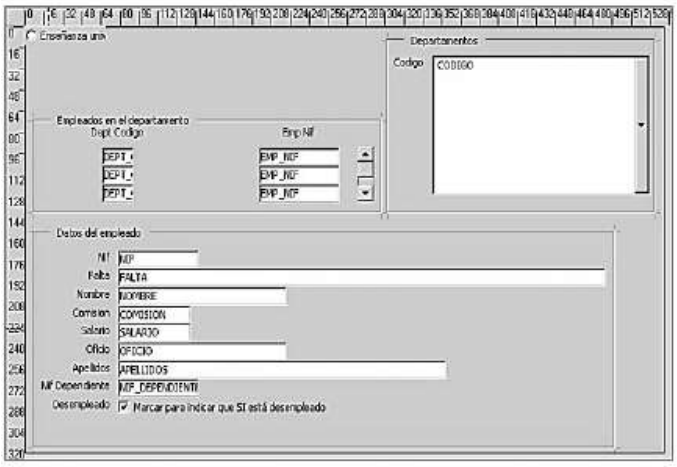
Al terminar este proceso nos encontraremos con los siguientes elementos dependientes del grupo de botones de radio *NIVEL_ESTUDIOS*

Correspondencia de otros valores del Grupo de Radio

Seguidamente abrimos las propiedades del elemento *NIVEL_ESTUDIOS* del bloque *EMPLEADO* y cambiamos las siguientes propiedades a los valores que se indican:

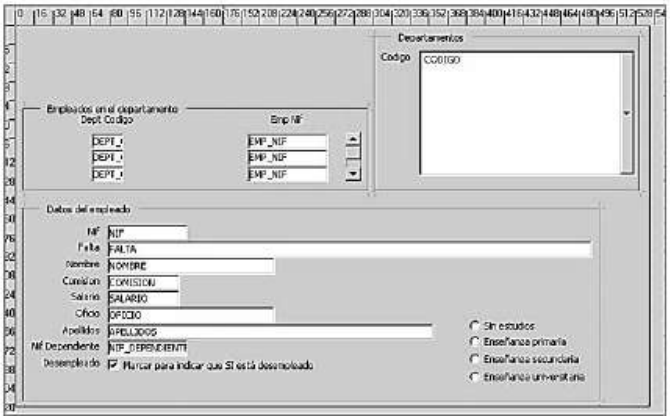
- Propiedad *Correspondencia de otros valores:* **0**
- Propiedad *Lienzo:* **CANVAS4** (o aquel que se muestre en la lista).

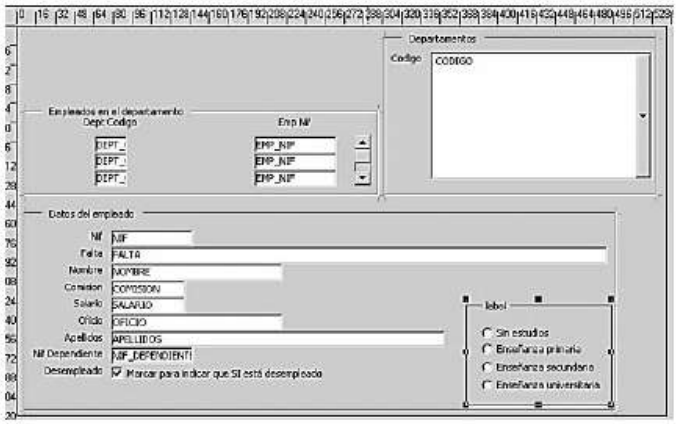
Cambiando propiedades de presentación del grupo



A continuación cerramos la paleta de propiedades y abrimos el editor de diseño (**F2**). Como podemos ver en la imagen que se muestra aquí, los elementos aparecen solapados en la parte superior izquierda del lienzo.

Lo que tenemos que hacer es trasladar cada una de las opciones del grupo de botones de radio a otra posición de la pantalla como se muestra en esta pantalla.





Seguidamente vamos a crear un marco que cubra el conjunto de opciones del grupo de radio para que la presentación de los mismos sea más clara. Para ello en la paleta de herramientas de diseño pulsaremos sobre el elemento *Frame* y pintaremos un rectángulo sobre los botones como se muestra en esta imagen.

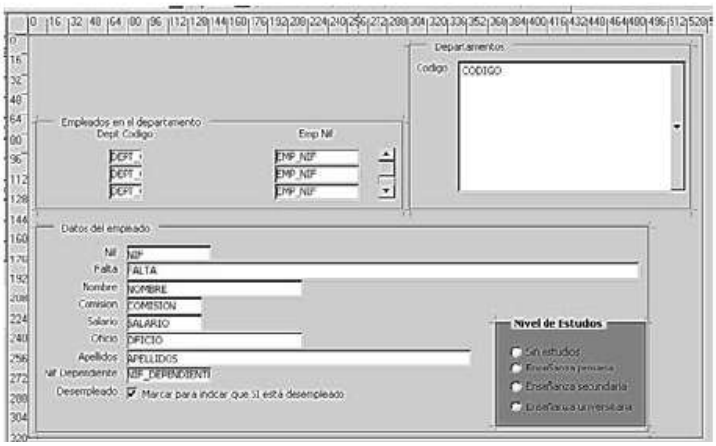
A continuación con el marco seleccionado abrimos la paleta de propiedades del mismo (**F4**) para cambiar las siguientes:

- Propiedad *Título de Marco:* **Nivel de Estudios**
- Propiedad *Grosor de Fuente de Título de Marco:* **Negrita**
- Propiedad *Color de Fondo:* **r100g75b0**
- Propiedad *Patrón de Relleno:* **transparent**

Seguidamente salimos de la paleta de propiedades del marco y seleccionamos en el editor de diseño cada uno de los 4 botones de opciones y abrimos la paleta de propiedades (**F4**) para cambiar la siguiente propiedad a los 4 elementos a la vez:

- Propiedad *Color de Fondo:* **r100g75b0**

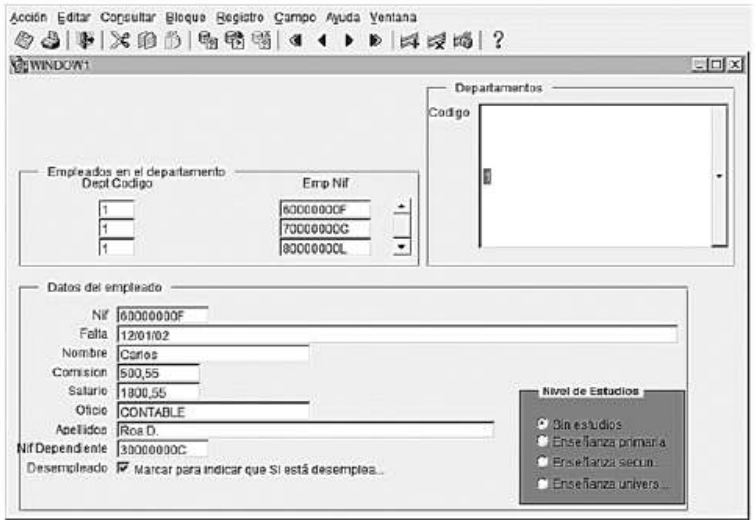
Una vez efectuados todos estos cambios debemos obtener una pantalla como la que se muestra en la siguiente página:



Guardando, compilando y ejecutando el formulario

Concluidas las modificaciones anteriores almacenamos el formulario (**CTRL + S),** lo compilamos **(SHIFT + CTRL + K)** y generamos **(CTRL + T)**, para en último lugar ejecutarlo **(CTRL + R)** y comprobar cómo han repercutido en el formulario las modificaciones que hemos efectuado.

El resultado de la ejecución después de introducir una consulta sin restricciones será similar al mostrado en la siguiente imagen.



CREANDO UN BLOQUE DE CONTROL 21

INTRODUCCIÓN

Hasta el momento todos los bloques que hemos creado en las distintas prácticas han sido bloques asociados a una base de datos, pero en Forms Builder se pueden crear también bloques que no están asociados a ningún objeto de base de datos. Estos bloques se conocen como Bloques de Control.

Por tanto, un bloque de control no está asociado a ningún objeto de la base de datos y sus elementos no están relacionados con ninguna columna de una tabla de la misma. Esto significa que Forms Builder no realiza una consulta automática de este tipo de bloques cuando ejecutemos los comandos *ENTER QUERY* o *EXECUTE QUERY*. Además, las actualizaciones en los valores de los elementos de los bloques de control no provocarán operaciones de modificación de datos (inserción, actualización o borrado) sobre la base de datos.

Apertura de Forms Builder y formulario

Como primer paso abriremos Forms Builder y una vez hecho esto desde la barra de menús **Archivo** seleccionaremos **Abrir** el fichero **PRACTICA8.FMB** que guardamos en el capítulo anterior.

Creando un bloque de control

Seguidamente desplegamos los *Bloques de Datos* del navegador de objetos y nos situamos en el bloque **EMPLEADO.** A continuación añadimos un nuevo bloque y seleccionamos la opción **Crear un nuevo bloque de datos manualmente.**

Pulsamos el botón **Aceptar** y seguidamente abrimos la paleta de propiedades del nuevo bloque (**F4**). Dentro de la paleta de propiedades cambiamos la siguiente:

- Propiedad *Nombre:* **B_CONTROL**

Guardando el formulario

Una vez creado el bloque guardamos el formulario desde el menú **Archivo,** seleccionamos la opción **Guardar Como** y asignamos como nombre de formulario el siguiente: **PRACTICA9.FMB.**

Para mantener la coherencia abrimos las propiedades del formulario y cambiamos la siguiente:

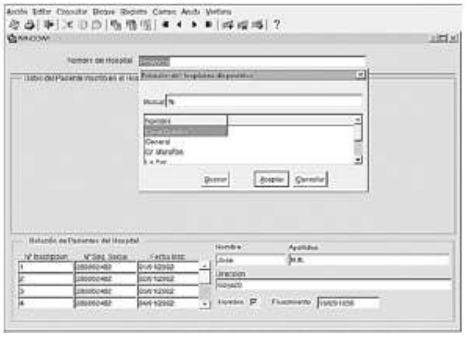
- Propiedad *Nombre*: **PRACTICA9**

SUPUESTO PRÁCTICO 2

MÓDULO ENFERMOS-HOSPITAL

En este supuesto práctico se deberán realizar los siguientes cambios en el formulario **ENFERMOS_HOSPITAL**:

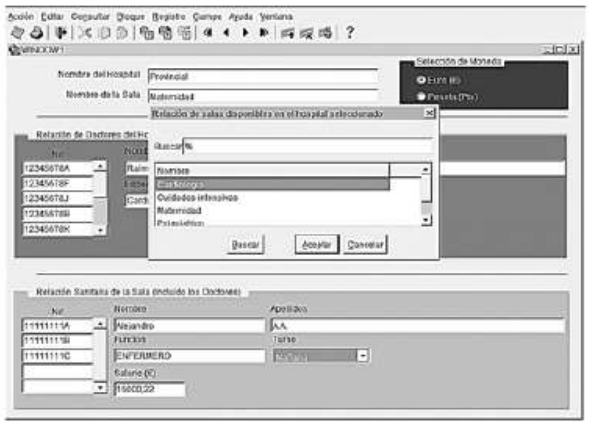
- Crear una lista de valores asociada al campo **Nombre del Hospital** del bloque HOSPITAL, que permita visualizar todos los hospitales que existan en la tabla asociada al bloque, ordenados por el nombre del hospital.
- Cambiar el campo **Sexo** del bloque ENFERMOS para que pase de un tipo de *elemento de texto* a una *Casilla de control (check box)*. Cambiar también el literal asociado al campo para se llame **Hombre.** Los valores correspondientes a los controles de este campo deberán ser:
    - Cuando el campo tenga contenido **M**, la casilla aparecerá marcada.
    - Cuando el campo tenga contenido **F**, la casilla aparecerá desmarcada.
    - El valor inicial de la casilla será desmarcado.
    - No se admitirán valores distintos a estos para el campo.



MÓDULO PLANTILLA-SANITARIA

En este supuesto práctico se deberán realizar los siguientes cambios en el formulario **PLANTILLA_SANITARIA:**

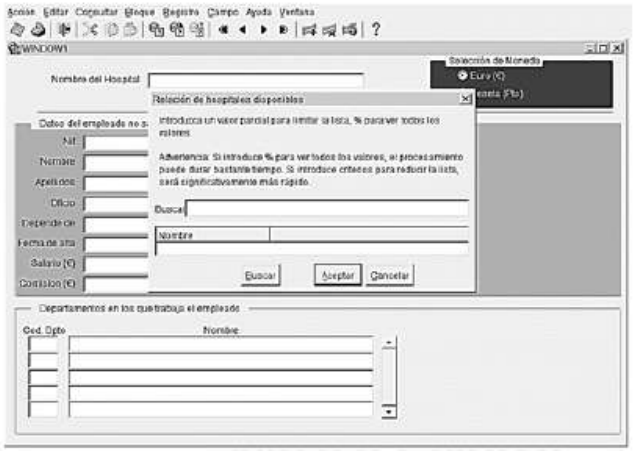
- Crear una lista de valores asociada al campo **Nombre del Hospital** del bloque HOSPITAL, que permita visualizar todos los hospitales que existan en la tabla asociada al bloque.
- Crear una lista de valores asociada al campo **Nombre de la Sala** del bloque SALA, que permita visualizar todas las salas que existan para el hospital seleccionado, ordenadas por el nombre.
- Convertir el campo **Turno** del bloque PLANTILLA de un *elemento de texto* a una *lista desplegable* donde se deben mostrar y manejar los siguientes valores:
  - M: será el turno de Mañana.
  - T: será el turno de Tarde.
  - N: será el turno de Noche.
  - Cualquier otro valor que pueda tener el campo se considerará como turno de Mañana.
- Crear un nuevo bloque denominado **MONEDA** que no esté asignado a tabla y que contenga un Radio Group con 2 elementos:
  - Peseta (con un valor P).
  - Euros (con un valor E). Este será el valor inicial y el valor en el caso de la propiedad "Correspondencia de Otros Valores".
- El bloque anterior deberá ser el primero sobre el que se posicione el cursor al abrir el formulario.
- Crear un marco o frame que contenga el Radio Group creado en el bloque anterior con sus 2 valores (opciones) visualizados.



MÓDULO PLANTILLA-NO-SANITARIA

En este supuesto práctico se deberán realizar los siguientes cambios en el formulario **PLANTILLA_NO_SANITARIA**:

- Crear una lista de valores asociada al campo **Nombre del Hospital** del bloque HOSPITAL que permita visualizar todos los hospitales de la tabla asociada, ordenados por el nombre. Se deberá permitir que el usuario filtre los datos a mostrar en la lista, antes de visualizarlos.
- Crear un nuevo bloque denominado **MONEDA** que no esté asignado a tabla y que contenga un Radio Group con 2 elementos:
  - Peseta (con un valor P).
  - Euros (con un valor E). Este será el valor inicial y el valor en el caso de la propiedad "Correspondencia de Otros Valores".
- El bloque anterior deberá ser el primero sobre el que se posicione el cursor al abrir el formulario.
- Crear un marco o frame que contenga el Radio Group creado en el bloque anterior con sus 2 valores (opciones) visualizados.



ELEMENTOS QUE NO ACEPTAN ENTRADAS 22

INTRODUCCIÓN

Dentro de Forms Builder también podemos crear elementos que no admiten introducción de valores por parte del usuario. Este tipo de elementos son un medio eficaz para visualizar la información o para ejecutar acciones. En este capítulo se describe cómo crear y utilizar este tipo de elementos.

Los elementos que no aceptan la entrada de datos se dividen en tres grupos:

- Los elementos que permiten mostrar datos.
- Los elementos que no permiten mostrar datos.
- Los elementos de área de Bean.

Elementos que permiten mostrar datos

Dentro de los elementos que no aceptan valores de entrada pero que sí permiten visualizarlos nos encontramos los siguientes tipos:

- Elementos mostrados.
- Elementos de imagen.
- Elementos calculados.
- Elementos de árbol jerárquico.

Elementos que no permiten mostrar datos

Dentro de los elementos que no aceptan valores de entrada y tampoco permiten visualizarlos nos encontramos el siguiente tipo:

- Botones.

Elementos de área de Bean

Este tipo de elementos tampoco aceptan valores de entrada, pero están asociados a la funcionalidad del código JavaBean implementado, por tanto dependiendo de cómo se programe el mismo, pueden llegar a mostrar o no información.

ELEMENTOS MOSTRADOS

Un elemento mostrado es similar a un elemento de texto excepto en que no se puede editar su contenido, ni se puede navegar a él en tiempo de ejecución.

Atendiendo al contenido y metodología de recuperación del mismo, un elemento mostrado puede ser de dos tipos:

- Muestra información de una columna de una tabla de la base de datos.
- Muestra valores de datos derivados o programados mediante PL/SQL.

Para crear un elemento mostrado se pueden utilizar los siguientes mecanismos:

- El editor de diseño.
- El icono **+** del navegador de objetos (automáticamente crea un elemento de texto que luego se puede transformar en un elemento mostrado cambiando sus propiedades).
- Convirtiendo un elemento existente en un elemento mostrado cambiando la propiedad *Tipo de Elemento*.

Una vez creado el elemento mostrado podrá definir de qué manera se muestra su contenido a través de las propiedades del mismo:

- Si define la propiedad *Elemento de Base de Datos* a **No**, estará indicando que el contenido del campo se calculará a través de un valor derivado o mediante programación.
- Si define la propiedad *Elemento de Base de Datos* a **Sí**, estará indicando que el contenido del campo se obtendrá de una columna de la base de datos y consecuentemente deberá indicar el nombre de la misma en la propiedad *Nombre de Columna*

ELEMENTOS DE IMAGEN

Se pueden utilizar las imágenes como objetos gráficos en un módulo de pantalla. Una imagen gráfica se muestra automáticamente y no se puede manipular en tiempo de ejecución. Se puede importar desde la propia base de datos o desde el sistema de archivos.

¿Qué es un elemento de imagen?

Un elemento de imagen es un control de interfaz especial que puede almacenar y mostrar imágenes de mapa de bits o vectoriales. Al igual que los elementos de texto almacenan y muestran valores de fecha, numéricos o VARCHAR, los elementos de imagen almacenan y muestran imágenes.

Al igual que el resto de elementos que pueden almacenar valores, los elementos de imagen pueden estar asociados a bloques de datos basados en tabla o a bloques de control.

Visualizar un elemento de imagen

Para recuperar una imagen en un elemento de imagen se pueden utilizar los siguientes métodos:

- Recuperar la información de una columna de la base de datos de tipo **LONG RAW** o **BLOB**.
- Utilizar un disparador y una función incorporada en el lenguaje de programación de Forms que permite rellenar un elemento de imagen.

Almacenamiento de imágenes

Se pueden almacenar las imágenes en la base de datos o bien en el sistema de archivos del ordenador utilizado por la aplicación.

Atendiendo al tamaño de las imágenes tenemos una limitación en cuanto al almacenamiento en base de datos:

- Para un campo de base de datos de tipo LONG RAW el tamaño máximo permitido es de 2 GB.
- Para un campo de base de datos de tipo BLOB el tamaño máximo permitido es de 4 GB.

En el almacenamiento en el sistema de archivos del ordenador utilizado por la aplicación, la única limitación que nos encontramos es el propio espacio libre en el equipo.

Formatos de archivo de imagen permitidos

Forms Builder soporta los siguientes formatos de imagen:

Extensión del archivo	Descripción	Operaciones Permitidas
BMP	Imagen bitmap de MS Windows y OS/2	Lectura/Escritura
CALS	CALS type raster	Lectura/Escritura
GI F	CompuServe	Lectura/Escritura
J FI F	JPEG File Interchange Format	Lectura/Escritura
TI FF	Tag Image File Format	Lectura/Escritura
J PEG	Joint Photographic Experts Group	Lectura/Escritura
PI CT	Macintosh Quickdraw Picture	Lectura/Escritura
RAS	Sun Raster	Lectura/Escritura
TPI C	Truevision Raster Graphics Array Picture	Lectura/Escritura

Debe limitar el número de imágenes que se muestren en su formulario o que se utilicen como imágenes de fondo, dado que cada imagen se descarga del servidor de aplicaciones cada vez porque no se pueden almacenar en caché.

Por ejemplo, para mostrar el logotipo de la compañía en la aplicación, podría incluir esta imagen en la página HTML que se descarga cuando se inicie la aplicación, en lugar de recuperar la imagen de la base de datos o del sistema de archivos por cada formulario. Oracle Forms no soporta archivos TIFF multipáginas.

Propiedades de los elementos de imagen

Los elementos de imagen tienen las siguientes propiedades específicas que permiten modificar el aspecto y comportamiento de las mismas:

- Propiedad **Formato de Imagen:** formato en el que se almacenará la imagen en la base de datos: BMP, CALS, GIF, JFIF, TIFF, JPEG, PICT, RAS, TPIC.
- Propiedad **Profundidad de Imagen:** valor de profundidad de la imagen que se lee o se escribe en un archivo del sistema de archivos (Original, Monocromo, Gris, LUT, RGB).
- Propiedad **Calidad Mostrada:** resolución utilizada para mostrar el elemento de imagen; controla el equilibrio entre calidad y rendimiento (Alta, Media, Baja).
- Propiedad **Estilo de Tamaño:** determina qué cantidad de imagen se muestra cuando el tamaño de imagen no coincide con el tamaño del elemento:
  - *Recortar*: corta los bordes de la imagen para que se ajuste al área de visualización del elemento.
  - *Ajustar*: ajusta la escala de la imagen para que se ajuste al área de visualización del elemento.
- Propiedades **Mostrar Barra de Desplazamiento Horizontal / Mostrar Barra de Desplazamiento Vertical**: muestra barras de herramientas que permiten mover la imagen que no se ajusta al área de visualización del elemento.

Los elementos de imagen no tienen la propiedad *Tipo de Dato*. Si se define la propiedad *Elemento de Base Datos* **= Sí,** Forms Builder entiende que el tipo de datos es LONG RAW.

Lectura y escritura de imágenes desde archivos

Los elementos de imagen pueden cargarse con la imagen contenida en un fichero utilizando una Bulit-in de Forms denominada **READ_IMAGE_FILE.**

Análogamente para almacenar una imagen desde un formulario a un fichero del sistema de archivos utilizaremos otra Built-in de Forms denominada **WRITE_IMAGE_FILE**.

A continuación se muestra la sintaxis de cada una de ellas, así como los parámetros que requieren y ejemplos de programación.

READ_IMAGE_FILE

Esta Built-in se utiliza para la lectura de imágenes dentro del servidor de aplicaciones. No se puede utilizar para la lectura de archivos de imagen desde el cliente del navegador que visualiza el formulario.

La sintaxis de esta Built-in admite las dos alternativas que se indican a continuación:

```
PROCEDURE READ_IMAGE_FILE
(file_name VARCHAR2,
file_type VARCHAR2,
item_id ITEM);


PROCEDURE READ_IMAGE_FILE
(file_name VARCHAR2,
file_type VARCHAR2,
item_name VARCHAR2);
```

El significado de cada uno de los parámetros es el siguiente:

- *file_name*; el nombre del fichero. Este nombre puede incluir la ruta completa donde se encuentre el archivo dentro del equipo. Si se omite la ruta, Oracle Forms busca la imagen en el mismo directorio donde se encuentra el archivo ejecutable del formulario (.FMX).
- *file_type;* el tipo de imagen que se va a leer: BMP, CALS, GIF, JPG, etc. El indicativo del tipo de imagen es opcional. Si se omite, Oracle Forms intenta deducirlo a partir del fichero fuente de la imagen.
- *item_id;* el identificador único que asigna Oracle Forms al elemento de imagen cuando se crea. Puede utilizar la Built-in FIND_ITEM para que devuelva el identificador correspondiente del elemento de imagen. El tipo devuelto por esta función es ITEM.
- *Ítem_name;* el nombre que se le dio a la imagen cuando se creó. El tipo es VARCHAR2.

A continuación se muestra un ejemplo de código programado para Forms que lee una imagen desde un archivo y la carga en un elemento de imagen del formulario:

```
DECLARE
tiff_image_dir  VARCHAR2(80)  := '/usr/staff/photos/';
photo_filename  VARCHAR2(80);
BEGIN
:System.Message_Level  := '25';
photo_filename := tiff_image_dir||
                    LOWER(:emp.userid)||'.tif';
READ_IMAGE_FILE(photo_filename, 'TIFF', 'emp.emp_photo');
IF NOT FORM_SUCCESS THEN
      MESSAGE('No existe foto para el  empleado.');
END IF;
:SYSTEM.MESSAGE_LEVEL := '0';
END;
```

WRITE_IMAGE_FILE

Esta Built-se utiliza para guardar la imagen de un elemento del formulario en un fichero del sistema de archivos del servidor de aplicaciones. No se puede utilizar para la escritura de imágenes en el cliente del navegador que visualiza el formulario

La sintaxis de esta Built-in admite las dos alternativas que se indican a continuación:

```
PROCEDURE WRITE_IMAGE_FILE
(file_name VARCHAR2,
file_type VARCHAR2,
item_id ITEM
compression_quality NUMBER,
image_depth NUMBER);


PROCEDURE READ_IMAGE_FILE
(file_name VARCHAR2,
file_type VARCHAR2,
item_name VARCHAR2
compression_quality NUMBER,
image_depth NUMBER);
```

El significado de cada uno de los parámetros es el siguiente:

- *file_name*; el nombre del fichero. Este nombre puede incluir la ruta completa donde se guardará el archivo dentro del equipo. Si se omite la

ruta, Oracle Forms almacena la imagen en el mismo directorio donde se encuentra el archivo ejecutable del formulario (.FMX).

- *file_type;* el tipo de imagen que se va a guardar: BMP, CALS, GIF, JPG, etc.
- *item_id;* el identificador único que asigna Oracle Forms al elemento de imagen cuando se crea. Puede utilizar la Built-in FIND_ITEM para que devuelva el identificador correspondiente del elemento de imagen. El tipo devuelto por esta función es ITEM.
- *Ítem_name;* el nombre que se le dio a la imagen cuando se creó. El tipo es VARCHAR2.
- *compression_quality;* el porcentaje de compresión que Oracle Forms aplicará a la imagen cuando se almacene en el fichero. Es un parámetro opcional de tipo NUMBER. Los valores admitidos son: NO_COMPRESSION, MINIMIZE_COMPRESSION, LOW_COMPRESSION, MEDIUM_COMPRESSION, HIGH_COMPRESSION o MAXIMIZE_COMPRESSION.
- *Image_depth;* la profundidad de la imagen que se aplicará cuando se almacene. Es un valor opcional de tipo NUMBER. Los valores admitidos son: ORIGINAL_DEPTH, MONOCHROME, GRAYSACLE, LUT (Lookup Table), RGB (Red, Green, Blue).

Esta Built-in presenta una serie de restricciones de uso:

- El indicativo del tipo de fichero debe ser compatible con el tipo actual de la imagen.
- Como con cualquier otro archivo, si se escribe la imagen sobre un archivo existente, se sobrescribirá el contenido del mismo con la imagen.
- Aunque se pueden leer archivos PCD y PCX desde el sistema de archivos o desde la base de datos, no se pueden escribir archivos en el sistema en formato PCD o PCX usando esta Built-in.
- Cuando se escribe una imagen de tipo JPEG desde Oracle Forms a un archivo, siempre habrá pérdida de calidad de resolución.

A continuación se muestra un ejemplo de código programado para Forms que graba una imagen a un archivo desde un elemento de imagen del formulario:

```
BEGIN
   WRITE_IMAGE_FILE('output.tif','TIFF',
                    'emp.photo_image_data',
                    maximize_compression,
                    original_depth);
END;
```

Disparadores para el manejo de elemento de imagen

Para el tratamiento mediante programación de los eventos asociados a un elemento de imagen se utilizan los siguientes disparadores:

- **WRITE-IMAGE-PRESSED**: se dispara cuando detecta que se ha realizado un clic o doble clic (con el ratón) sobre la imagen.
- **WHEN-IMAGE-ACTIVATED:** se dispara cuando el elemento de imagen recibe el foco del cursor tras realizar el clic sobre la misma.

BOTONES

Un botón es un objeto de la interfaz del formulario en el que se hace clic para iniciar una acción. Un botón se muestra como un rectángulo con una etiqueta descriptiva en su interior.

Los botones no pueden almacenar ni mostrar valores.

Puede mejorar el módulo de pantalla agregando botones que proporcionen acceso rápido y directo a las operaciones más necesarias.

Los botones pueden ser de dos tipos:

- **Botones de texto;** se muestran con una etiqueta de texto sobre el botón.
- **Botones icónicos;** se muestran con un gráfico de bitmap en el botón y, a menudo, se utilizan en la barra de herramientas.

Acciones típicas asociadas a un botón

Algunas acciones típicas a las que se puede llamar mediante un botón son las siguientes:

- Mover el foco de entrada.
- Mostrar una lista de valores.
- Llamar a un editor.
- Llamar a otra ventana.
- Confirmar datos.
- Emitir una consulta.
- Realizar cálculos.

Propiedades de un botón

Los botones tienen las siguientes propiedades específicas que permiten modificar el aspecto y comportamiento de los mismos:

- Propiedad **Etiqueta:** etiqueta de texto que aparecerá en el botón en tiempo de ejecución.
- Propiedad **Icónico:** determina si el botón se muestra como un icono en lugar de como una etiqueta.
- Propiedad **Nombre de Archivo de Icono:** nombre del archivo que contiene el icono (solo el nombre del archivo sin la extensión ni la ruta).
- Propiedad **Botón Por Defecto:** determina si este es el botón por defecto del bloque, que puede seleccionar implícitamente pulsando *Seleccionar* sin necesidad de navegar o utilizar el ratón.
- Propiedad **Navegación del Mouse:** determina si Forms navega hasta el elemento cuando se hace clic en el mismo con el ratón.
- Propiedad **Ayuda de Burbuja:** texto de ayuda que aparecerá como una burbuja debajo del botón cuando el ratón se mueve sobre el mismo.
- Propiedad **Grupo de Atributos Visuales de Ayuda de Burbuja:** nombre del atributo visual que se aplica a la ayuda de burbuja en tiempo de ejecución.

ELEMENTOS CALCULADOS

Un elemento calculado permite obtener un resultado a partir de una función de cálculo o una fórmula. Por ejemplo, puede utilizar un elemento calculado para mostrar el total acumulado del importe del sueldo y complementos de un empleado.

Un elemento calculado es de solo lectura. Los usuarios finales no pueden insertar ni modificar el contenido de este tipo de elementos. Por tanto, debe utilizar generalmente elementos mostrados como elementos calculados.

Cualquier elemento que pueda almacenar un valor se puede utilizar como elemento calculado definiendo sus valores de propiedad correspondientes.

Modos de cálculo

Los cálculos se pueden expresar como una fórmula o como un resumen de todos los elementos del bloque. Forms Builder soporta los siguientes modos de cálculo:

- **Fórmula:** el valor del elemento calculado es el resultado de un cálculo horizontal que implica a una o más variable ligadas, como por ejemplo, elementos de pantalla, variables globales y parámetros.
- **Resumen:** el valor del elemento calculado es un cálculo vertical que implica los valores de un único elemento en todas las filas de un único bloque.

Propiedades de un elemento calculado

A diferencia de otros elementos tratados hasta el momento en este capítulo, no hay ningún valor de la propiedad *Tipo de elemento* denominada **elemento calculado.** Las propiedades específicas de cálculo de un elemento lo convierten en elemento calculado. Tanto los elementos de texto como los elementos mostrados soportan elementos calculados.

Las siguientes propiedades son específicas de los elementos calculados:

- Propiedad **Modo de Cálculo:** indica el método de cálculo de los valores del elemento calculado: NINGUNO, FÓRMULA o RESUMEN.
- Propiedad **Fórmula:** expresión PL/SQL que determina el valor calculado de un elemento mediante una fórmula; puede calcular un valor o llamar a un subprograma que la calcule.
- Propiedad **Función de Resumen:** el tipo de función de resumen que se va a realizar en el elemento calculado. Solo se aplica cuando la propiedad *modo de cálculo* tiene el valor *RESUMEN*. Los valores que admite son los siguientes:
    - MEDIA; obtiene el valor medio (media aritmética) del elemento resumido de todos los registros del bloque.
    - RECUENTO; contabiliza todas las instancias no nulas del elemento resumido de todos los registros del bloque.
    - MAX; obtiene el valor máximo del elemento resumido de todos los registros del bloque.
    - MIN; obtiene el valor mínimo del elemento resumido de todos los registros del bloque.
    - DESVENT; obtiene la desviación estándar de los valores del elemento resumido de todos los registros del bloque.
    - SUMA; obtiene la suma de todos los valores del elemento resumido de todos los registros del bloque.

  - VARIANZA; obtiene la varianza (cuadrado de la desviación estándar) de los valores del elementos resumido de todos los registros del bloque.
- Propiedad **Bloque Resumido:** el nombre del bloque sobre el que se aplicará la función de resumen. Si no se especifica se asume que es el bloque actual.
- Propiedad **Elemento Resumido:** el nombre del elemento cuyo valor se resume para asignar un valor al elemento calculado. Es obligatorio si se indica en la propiedad *Modo de Cálculo* el valor *RESUMEN.*

Reglas para fórmulas de elementos calculados

Cuando escriba fórmulas para elementos calculados debe tener en cuenta las siguientes reglas:

- La fórmula (y cualquier subprograma escrito por el usuario que la llame) no debe llamar a ninguna función incorporada restringida.
- La fórmula (y cualquier subprograma escrito por el usuario que la llame) no puede ejecutar ninguna sentencia DML.
- No hay que terminar la expresión PL/SQL con punto y coma.
- Si la expresión PL/SQL implica una asignación, no introduzca la sentencia PL/SQL completa. Forms Builder asigna internamente el código de asignación real.

EJEMPLO

Suponga que tiene un elemento de fórmula denominado *Sueldo_Bruto* en el bloque *EMPLEADO* de una pantalla. Si define la propiedad Fórmula en:

```
NVL(:EMPLEADO.salario,0) * NVL(:EMPLEADO.comision,0)
```

Forms Builder convertirá internamente esta expresión en una sentencia completa de la siguiente forma:

```
:EMPLEADO.Sueldo_Bruto := (NVL(:EMPLEADO.salario,0) *
NVL(:EMPLEADO.comision,0));
```

Reglas para elementos de resumen

Cuando se crean elementos calculados basados en un resumen, se deben seguir las reglas indicadas a continuación:

- El elemento de resumen debe residir en el mismo bloque que el elemento resumido o en un bloque de control cuya propiedad **Registro Único** esté definida a **Sí.**
- El elemento resumido debe residir en un bloque de control o en un bloque de datos cuya propiedad **Consultar todos los registros** o la propiedad **Precalcular Resúmenes** esté definida a **Sí**. Esto asegura que los registros recuperados en el bloque y el valor resumido sean consistentes. En caso contrario, puede que otro usuario actualice un registro que no se ha recuperado aún.
- Defina la propiedad **Tipo de Dato** para un elemento de resumen en **Number,** a menos que la función de resumen sea MAX o MIN, en cuyo caso el tipo de datos debe duplicar el de su elemento resumido asociado. Por ejemplo, un elemento calculado que muestra la fecha más reciente (máxima) en la columna *FECHA_ALTA* debe tener el tipo de datos a **Date**.
- Si los valores de elemento resumido se basan en una fórmula, el elemento resumido debe residir en un bloque cuya propiedad **Consultar todos los registros** esté definida a **Sí**.

ELEMENTOS DE ÁRBOL JERÁRQUICO

Un árbol jerárquico es un elemento que muestra datos en forma de navegador estándar. El aspecto es muy similar al que presenta el explorador de Windows cuando muestra el contenido de una carpeta/directorio.

Los valores de un árbol jerárquico se pueden rellenar con los elementos de un grupo de registros o un texto de consulta.

Propiedades de un árbol jerárquico

Las siguientes propiedades son específicas de los árboles jerárquicos:

- Propiedad **Tipo de Elemento:** debe definirse a *Árbol Jerárquico*.
- Propiedad **Permitir Derivaciones Vacías:** determina si pueden existir nodos de ramas sin secundarios (si se define a valor *No*, los nodos de

ramas sin secundarios se convertirán en nodos de hoja, en cambio si define a valor *Si*, una rama vacía se mostrará como un nodo reducido).

- Propiedad **Selección Múltiple:** determina si se pueden seleccionar varios nodos a la vez.
- Propiedad **Mostrar Líneas**: determina si un árbol jerárquico debe mostrar los símbolos **+** o **–** delante de cada nodo de rama.
- Propiedad **Grupo de Registros:** nombre del grupo de registros desde el que el árbol jerárquico deriva sus valores a mostrar.
- Propiedad **Data Query:** especifica el origen de datos basado en consulta.

Un árbol jerárquico debe ser el único elemento del bloque de datos.

Disparadores para el manejo de árboles jerárquicos

Para el tratamiento mediante programación de los eventos asociados a un elemento de tipo árbol jerárquico se utilizan los siguientes disparadores:

- **WHEN-TREE-NODE-ACTIVATED**: se dispara cuando el usuario hace clic dos veces en un nodo (elemento del árbol) o pulsa ENTER cuando se selecciona un nodo.
- **WHEN-TREE-NODE-EXPANDED:** se dispara cuando el usuario amplía o reduce un nodo.
- **WHEN-TREE-NODE-SELECTED:** se dispara cuando el usuario selecciona o anula la selección de un nodo.

Relleno de árboles jerárquicos

Un árbol jerárquico se puede rellenar con los valores estáticos o dinámicos contenidos en un grupo de registros o un texto de consulta. En tiempo de ejecución se puede agregar, eliminar, modificar o evaluar elementos en un árbol jerárquico mediante programación.

EL PAQUETE FTREE

El paquete FREE contiene funciones incorporadas y constantes para interactuar con los elementos del árbol jerárquico de una pantalla. Para utilizar las funciones incorporadas y las constantes, sus nombres deben ir precedidos del nombre del paquete.

PROCEDIMIENTO FTREE.SET_TREE_PROPERTY

Este procedimiento del paquete FTREE se utiliza para cambiar determinadas propiedades en el elemento del árbol jerárquico indicado. También se puede utilizar para rellenar el elemento del árbol jerárquico indicado desde un grupo de registros.

La sintaxis de esta Built-in es la siguiente:

FTREE.SET_TREE_PROPERTY(ítem_name, ftree.property, value);

Parámetro	Descripción
Item_name	Especifica el nombre del objeto creado durante el diseño. El tipo de datos del nombre es VARCHAR2. También es válida para este argumento una variable que contenga el identificador de elemento Item_Id.
Property	Especifica una de las siguientes acciones: RECORD_GROUP: sustituye el juego de datos del árbol jerárquico por un grupo de registros y hace que se muestre. QUERY_TEXT: sustituye el juego de datos del árbol jerárquico por una consulta SQL y hace que se muestre. ALLOW_EMPTY_BRANCHES: los valores posibles son PROPERTY_TRUE y PROPERTY_FALSE.
Value	Especifica el valor adecuado para la propiedad que se está definiendo.

Para agregar datos a una vista del árbol se deberán seguir alguna de las siguientes acciones:

- Rellenar el árbol con valores contenidos en un grupo de registros o una consulta mediante la función incorporada POPULATE_TREE.
- Agregue datos a un árbol bajo un nodo específico mediante la función incorporada ADD_TREE_DATA.
- Modifique el elemento de un árbol en tiempo de ejecución mediante subprogramas incorporados.
- Agregue o suprima nodos y los elementos de datos bajo los nodos.

EJEMPLO

Este código se puede utilizar en un disparador WHEN-BUTTON-PRESSED para rellenar inicialmente el árbol jerárquico con datos. El ejemplo ubica primero el árbol

jerárquico y a continuación se crea un grupo de registros. Para finalizar se rellena el árbol con dicho grupo.

```
DECLARE
      Htree              ITEM;
      v_ignore           NUMBER;
      rg_emps            RECORDGROUP;
BEGIN
      Htree := FIND_ITEM('tree_block.htree3');
      rg_emps := CREATE_GROUP_FROM_QUERY('rg_emps',
      'select 1, level, last_name, NULL, to_char(employee_id) '
      ||' from employees '||
      'connect by prior employee_id = manager_id '||
      'start with job_id = ''AD_PRES''');
      v_ignore := POPULATE_GROUP(rg_emps);
      FTREE.SET_TREE_PROPERTY(htree, FTREE.RECORD_GROUP,
                              rg_emps);
END;
```

ELEMENTOS DE ÁREA DE BEAN (JAVABEAN)

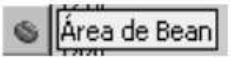
Un JavaBean es un componente desarrollado en Java que se puede conectar a cualquier applet o aplicación de Java. El JavaBean (elemento de área de Bean) le permite ampliar la funcionalidad de Forms agregándolo a su pantalla.

Con el JavaBean se puede interactuar a través de la máquina cliente, realizando funciones tales como:

- Obtener información relativa a la máquina cliente.
- Cargar archivos del cliente en la máquina servidora.
- Modificar la interfaz de usuario en el cliente.
- Comprobar la ortografía de un elemento de texto.
- Mostrar una barra de progreso, un reloj, un calendario o un selector de color con el que el usuario pueda interactuar y seleccionar valores.

Algunas de estas funcionalidades, como el calendario, son posibles mediante la funcionalidad nativa de Forms. Sin embargo, el uso de un JavaBean permite la interacción del cliente sin generar tráfico de red.

Creación de elementos Javabean

Un Javabean se crea mediante la herramienta **Área de Bean** en el editor de diseño. Haga clic en 

y cree un espacio contenedor del elemento en el lienzo.

Al principio el área tendrá el aspecto de un rectángulo vacío.

Propiedades de un Javabean

La propiedad más importante del Javabean es **Clase de Implantación**. Esta propiedad se utiliza para especificar el nombre totalmente cualificado de la clase Java que debe instanciar el elemento Javabean.

Si Javabean tiene un componente visible, Forms Builder muestra el mismo dentro del área definida en el editor de diseño para recibir el Javabean.

Puede definir el tamaño y la posición del Javabean en el editor de diseño o cambiando las propiedades **Posición X**, **Posición Y**, **Altura** y **Ancho**.

Algunos Javabeans no tienen ningún componente visible. Para evitar la distracción que supone un rectángulo vacío grande en la pantalla, se pueden cambiar las propiedades reservadas para el área del objeto, de forma que se muestre como un pequeño punto en una de las esquinas del lienzo. Además puede definir la propiedad **Visible** al valor *Sí* o *No* según decida mostrar o no el Javabean en tiempo de ejecución.

Interacción con el Javabean en tiempo de ejecución

En tiempo de ejecución, el usuario puede interactuar con el Javabean para ejecutar la funcionalidad que se haya definido mediante programación en el mismo.

Independientemente de si el Javabean es visible o no en tiempo de ejecución, debe haber alguna comunicación entre el formulario y las clases Java que componen el Javabean. Para conseguirlo primero debe hacer que la pantalla tenga en cuenta el Javabean, ya sea definiendo su propiedad **Clase de Implantación** durante el diseño del mismo o bien registrando el Javabean y sus eventos en tiempo de ejecución. Una vez que el formulario sabe de la existencia del Javabean, estas se comunican con él de las siguientes formas:

- Llamando a los métodos del Javabean.
- Obteniendo y definiendo propiedades del Javabean.

El Javabean se comunica con la pantalla de las siguientes formas:

- Enviando un evento como, por ejemplo, el hecho de que el usuario ha seleccionado una fecha o un color.
- Enviando una lista que contiene información necesaria para la pantalla como por ejemplo, la fecha o el color seleccionados.
- Devolviendo un valor desde un método llamado.

EL PAQUETE FBEAN

El paquete FBEAN contiene funciones incorporadas de Forms que permiten codificar interacciones con elementos Javabean en PL/SQL, lo que elimina la necesidad de conocer Java para comunicarse con el Javabean.

Muchas de las funciones incorporadas toman algunos de los siguientes argumentos:

- **Nombre del Elemento o Identificador del Elemento (obtenido con la función FIND_ITEM):** el primer argumento para la mayoría de las funciones incorporadas en un Javabean es este.
- **Instancia del Elemento:** una referencia a la instancia del elemento que debe contener el Javabean. Es aplicable cuando el Javabean forma parte de un bloque de varias filas y se muestra más de una instancia del Javabean. Se puede utilizar el comando `ALL_ROWS` (o `FBEAN.ALL_ROWS`) para que el valor de la instancia del elemento se aplique a todas las instancias de este Javabean en el bloque.
- **Valor**: puede aceptar los tipos de datos `BOOLEAN`, `VARCHAR2` o `NUMBER`.

Entre las funciones y procedimientos que nos podemos encontrar en el paquete FBEAN están las siguientes más relevantes:

- **GET_PROPERTY(ITEM, INSTANCE, PROPERTY_NAME):** esta función recupera el valor de la propiedad especificada. Este valor se devuelve con tipo `VARCHAR2`.
- **SET_PROPERTY(ITEM, INSTANCE, PROPERTY_NAME, VALUE):** este procedimiento define una propiedad específica del Javabean, a través del valor que se indique.

- **INVOKE(ITEM, INSTANCE, METHOD_NAME[,ARGUMENTS]):** este procedimiento llama a un método en el Javabean y transfiere opcionalmente argumentos al método.
- **REGISTER_BEAN(ITEM, INSTANCE, BEAN_CLASS):** este procedimiento registra el Javabean definido en la pantalla, en tiempo de ejecución, por lo que todos los métodos y los atributos expuestos están disponibles para el Javabean de la pantalla. El último argumento es el nombre completo de la clase, por ejemplo, *'oracle.forms.demos.beans.ColorPicker'*.
- **ENABLE_EVENT(ITEM, INSTANCE, EVENT_LISTENER_NAME, SUBSCRIBE):** Este procedimiento habilita un evento del Javabean. El último argumento es de tipo BOOLEAN que indica si se suscribe (TRUE) o no (FALSE) al evento.

Recuerde que las llamadas a estas funciones y procedimientos deben ir precedidas del nombre del paquete FBEAN, como por ejemplo, FBEAN.GET_PROPERTY…

CREANDO UN ÁRBOL 23

INTRODUCCIÓN

Durante esta práctica vamos a practicar la creación de un árbol jerárquico con ramas y subramas, a partir de consultas efectuadas contra la base de datos.

Apertura de Forms Builder y formulario

Como primer paso abriremos Forms Builder y una vez hecho esto desde la barra de menús **Archivo** seleccionaremos **Nuevo – Pantalla**, para crear un nuevo formulario.

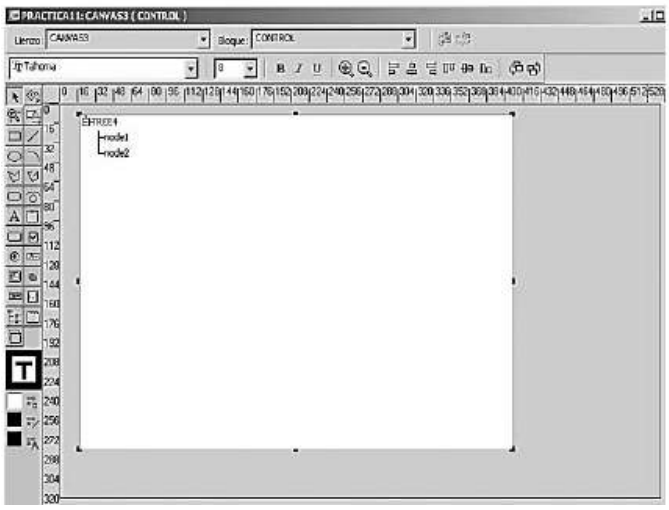
Como nombre del mismo utilizaremos **PRACTICA11**, modificando la propiedad *Nombre* del formulario.

Creando un bloque de datos manualmente

A continuación creamos un bloque de datos manualmente que denominaremos **CONTROL** y que no dependerá de ninguna tabla de la base de datos. Para ello dentro del grupo **Bloque de Datos** pulsamos sobre el botón añadir (**+**) y cuando se nos muestre el asistente de bloques de datos, seleccionamos la opción **Crear un nuevo bloque de datos manualmente** y pulsamos **ACEPTAR.**

Seguidamente dentro de las propiedades del nuevo bloque creado cambiamos el nombre a **CONTROL.**

Creando un elemento árbol jerárquico



Pulsamos F2 para abrir el editor de diseño y a continuación en la paleta de herramientas, seleccionamos el elemento **Árbol Jerárquico** . Nos posicionamos en un punto de la parte superior izquierda del lienzo y abrimos el objeto hasta crear un tamaño del mismo como el que se muestra en esta imagen.

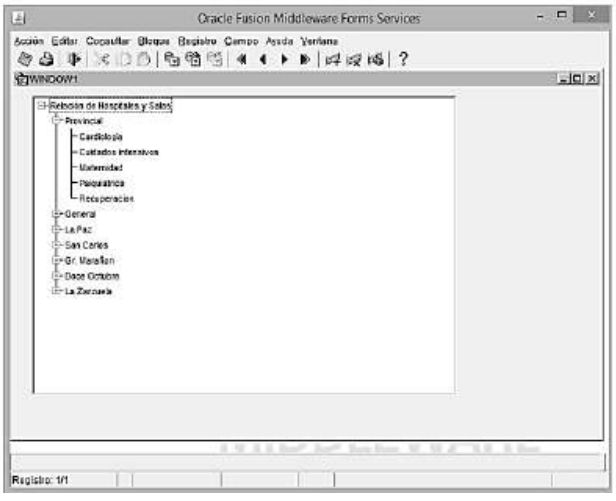
A continuación cerramos el editor de diseño y volvemos al navegador de objetos. Seleccionamos el nuevo elemento (TREEn) creado en el bloque CONTROL, abrimos la paleta de propiedades (F4) y cambiamos la propiedad *Nombre:* **ARBOL_HOSPITAL.** Después cerramos la paleta de propiedades. Seguidamente nos posicionamos en el grupo *Disparadores* a nivel del módulo PRACTICA11 y pulsamos el botón añadir (**+**) para crear un nuevo disparador del tipo **WHEN-NEW-FORM_INSTANCE**.

Dentro del editor PL/SQL que se abre asociado al disparador vamos a introducir el siguiente código:

```
DECLARE
        v_arbol ITEM;
        v_nodo_raiz     FTREE.NODE;
        v_nodo_hosp     FTREE.NODE;
        v_nodo_sala     FTREE.NODE;
    CURSOR C_HOSPITAL IS
           SELECT CODIGO, NOMBRE FROM HOSPITAL
           ORDER BY CODIGO;
   CURSOR C_SALAS (V_HOSPITAL IN HOSPITAL.CODIGO%TYPE) IS
           SELECT CODIGO, NOMBRE FROM SALA
           WHERE HOSP_CODIGO = V_HOSPITAL
           ORDER BY NOMBRE;
BEGIN
v_arbol := FIND_ITEM('CONTROL.ARBOL_HOSPITAL');
v_nodo_raiz := Ftree.Add_Tree_Node(  v_arbol,
                                     Ftree.ROOT_NODE,
                                     Ftree.PARENT_OFFSET,
                                     Ftree.LAST_CHILD,
                                     Ftree.EXPANDED_NODE,
                                     'Relación de Hospitales y Salas',
```

```
                                        NULL,
                                        NULL);

FOR I IN C_HOSPITAL LOOP
        v_nodo_hosp := Ftree.Add_Tree_Node(
           v_arbol,
           v_nodo_raiz,
           Ftree.PARENT_OFFSET,
           Ftree.LAST_CHILD,
           Ftree.EXPANDED_NODE,
           I.NOMBRE,
           NULL,
           'LEVEL '||
        TRIM(TO_CHAR(I.CODIGO)));
FOR J IN C_SALAS(I.CODIGO) LOOP
        v_nodo_sala := Ftree.Add_Tree_Node(
           v_arbol,
           v_nodo_hosp,
           Ftree.PARENT_OFFSET,
           Ftree.LAST_CHILD,
           Ftree.LEAF_NODE,
           J.NOMBRE,
           NULL,
           'LEVEL '||
        TRIM(TO_CHAR(I.CODIGO))||
        TRIM(TO_CHAR(J.CODIGO)));

   END LOOP;
END LOOP;
END;
```

Guardando, compilando y ejecutando el formulario



Concluidas las modificaciones anteriores almacenamos el formulario (**CTRL + S),** lo compilamos **(SHIFT + CTRL + K)** y generamos **(CTRL + T)**, para en último lugar ejecutarlo **(CTRL + R).**

El resultado será similar al mostrado en esta imagen.

CREANDO BOTONES E IMÁGENES 24

INTRODUCCIÓN

Durante esta práctica aprenderemos a usar botones e imágenes, así como los eventos asociados a estos elementos.

MODIFICACIÓN EN LA BASE DE DATOS:

Para poder seguir la práctica que se va a proponer en este capítulo debe de efectuar una alteración en la tabla EMPLEADO de la base de datos, según se especifica a continuación:

- Ejecute SQL*PLUS o cualquier otra herramienta de conexión a la base de datos e intérprete de comandos SQL.
- Conéctese con las credenciales PEPERP/PEPITO (usuario y contraseña respectivamente).
- Ejecute la siguiente sentencia SQL:

```
ALTER TABLE EMPLEADO
ADD (IMAGEN_EMP LONG RAW);
```

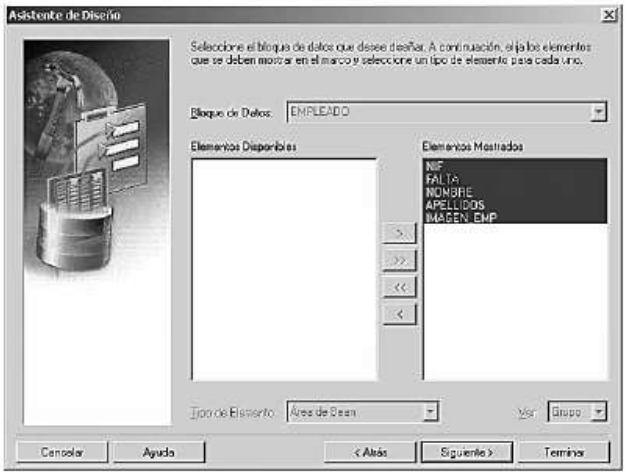
Apertura de Forms Builder y formulario

Como primer paso abriremos Forms Builder y una vez hecho esto desde la barra de menús **Archivo** seleccionaremos **Nuevo – Pantalla**, para crear un nuevo formulario.

Como nombre del mismo utilizaremos **PRACTICA12**, modificando la propiedad *Nombre* del formulario.

Creando un bloque de datos basado en tabla

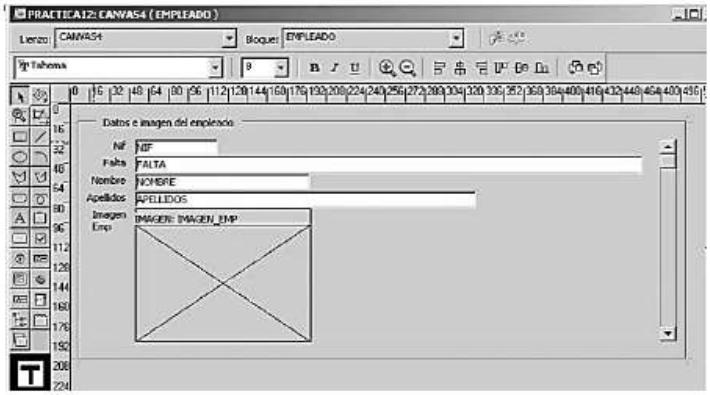
A continuación creamos un bloque de datos basado en la tabla EMPLEADO. Para ello nos situamos dentro del grupo **Bloque de Datos** y pulsamos sobre el botón añadir (**+**) y cuando se nos muestre el asistente de bloques de datos, seleccionamos la opción **Usar el Asistente de Bloque de Datos**. Seguidamente pulsamos **ACEPTAR** sucesivamente hasta llegar a la ventana de selección de la tabla.



Si no nos hemos conectado previamente a la base de datos, lo tendremos que hacer en este punto con las credenciales utilizadas durante el curso: **PEPERF / PEPITO.** Seguidamente seleccionaremos la tabla **EMPLEADO** y dentro de la misma, vamos a utilizar en este formulario las columnas: **NIF, NOMBRE, APELLIDOS, FALTA** e **IMAGEN_EMP**. A continuación pulsamos el botón **Siguiente** hasta llegar a la pantalla en la que se nos solicita si queremos crear el bloque e invocar al Asistente de Diseño, lo cual haremos dejando la opción marcada y pulsando el botón **Terminar,** para seguir pulsando el botón **Siguiente** hasta llegar a esta pantalla.

En ella volveremos a seleccionar todos los elementos disponibles para pintarlos en el lienzo y pulsamos el botón **Siguiente** repetidamente hasta llegar a la pantalla donde se nos solicita el estilo del diseño, en la que dejaremos marcada la opción por defecto de **Pantalla** y pulsaremos el botón **Siguiente** para que se nos presente esta pantalla.

En ella introduciremos las propiedades que se muestran y pulsaremos el botón **Terminar** para completar el diseño del formulario del lienzo. Es muy posible que al completar esta operación y mostrarse el editor de diseño, aparezca este mensaje. En ese caso pulsaremos **Aceptar** y reubicaremos los elementos dentro del formulario para dejarlos con la apariencia que se muestra en la pantalla que aparece a continuación.



Cambiando las propiedades de un elemento de imagen

Sin abandonar el *Editor de Diseño* seleccionamos el elemento **IMAGEN_EMP** y abrimos las propiedades del mismo pulsando **F4.**

Dentro de las propiedades del elemento de imagen vamos a cambiar las siguientes con los valores que se especifican a continuación:

- Propiedad **Formato de Imagen:** debe definirse a *JFIF*.
- Propiedad **Estilo de Tamaño:** debe definirse a *Ajustar*

Una vez completado el cambio de las propiedades, cerramos la paleta de propiedades y también el editor de diseño para situarnos en el navegador de objetos.

Creando un bloque de datos manualmente

A continuación creamos un bloque de datos manualmente que denominaremos CONTROL y que no dependerá de ninguna tabla de la base de datos. Para ello dentro del grupo **Bloque de Datos** nos situamos en el bloque EMPLEADO y pulsamos sobre el botón añadir **(+)** y cuando se nos muestre el asistente de bloques de datos,

seleccionamos la opción **Crear un nuevo bloque de datos manualmente** y pulsamos **ACEPTAR.**

Seguidamente dentro de las propiedades del nuevo bloque creado cambiamos el nombre a **CONTROL.**

Creando un botón dentro de un bloque

Sin abandonar el navegador de objetos y manteniéndonos situados dentro del nuevo bloque **CONTROL**, nos posicionamos en el grupo **Elementos** y pulsamos el botón añadir (**+**) para añadir un nuevo elemento al bloque.

Una vez creado abrimos la paleta de propiedades del nuevo elemento (pulsando **F4**) y cambiamos las propiedades que se indican a continuación con los valores que se especifican:

- Propiedad **Nombre:** debe definirse a *CARGAR_IMAGEN*.
- Propiedad **Tipo de Elemento:** debe definirse a *BOTÓN*.
- Propiedad **Teclado Navegable:** debe definirse a *NO*.
- Propiedad **Navegación del Mouse:** debe definirse a *NO*.
- Propiedad **Etiqueta:** debe definirse a "*Cargar Imagen a B.D.*".
- Propiedad **Lienzo:** debe seleccionarse el único canvas que aparece al desplegar la lista.
- Propiedad **Posición X:** debe definirse a *210*.
- Propiedad **Posición Y:** debe definirse a *123*.
- Propiedad **Ancho:** debe definirse a *99*.
- Propiedad **Alto:** debe definirse a *16*.

Replicando un elemento de un bloque

Cerramos la paleta de propiedades del botón y volviendo al navegador de objetos nos situamos dentro del bloque **CONTROL** sobre el elemento **CARGAR_IMAGEN** que acabamos de crear. A continuación dentro del menú **Editar** seleccionamos **Copiar** y después dentro del mismo menú seleccionamos la opción **Pegar**, lo que produce que se cree un nuevo elemento bajo el anterior con sus mismas propiedades pero con distinto nombre, como se muestra en esta imagen.

Seguidamente vamos a cambiar las propiedades de este nuevo elemento abriendo la paleta de propiedades (pulsando **F4**):

- Propiedad **Nombre:** debe definirse a *GUARDAR_IMAGEN_FICH*.
- Propiedad **Etiqueta:** debe dejarse en blanco eliminando el contenido que haya.
- Propiedad **Icónico:** debe definirse a *SÍ*.
- Propiedad **Nombre de Archivo de Icono:** debe definirse a *MANO_RATON*.
- Propiedad **Posición X:** debe definirse a *367*.
- Propiedad **Posición Y:** debe definirse a *114*.
- Propiedad **Ancho:** debe definirse a *51*.
- Propiedad **Alto:** debe definirse a *53*.
- Propiedad **Petición de Datos:** debe definirse a *"Guardar Imagen a fichero"*.
- Propiedad **Borde de Anexo de Petición de Datos** debe definirse a *ABAJO*.
- Propiedad **Alineamiento de Petición de Datos:** debe definirse a *CENTRO*.

Creando un elemento de tipo casilla de control

Cerramos la paleta de propiedades y volvemos al navegador de objetos para situarnos nuevamente en el bloque **CONTROL** y dentro del mismo en el grupo *Elementos*. Seguidamente pulsamos el botón (**+**) para añadir un nuevo elemento dentro del bloque, que será el primero dentro de la lista de elementos del mismo como se muestra en esta imagen.

Seguidamente vamos a cambiar las propiedades de este nuevo elemento abriendo la paleta de propiedades (pulsando **F4**):

- Propiedad **Nombre:** debe definirse a *IMAGEN_DISPONIBLE*.
- Propiedad **Tipo de Elemento:** debe definirse como *CASILLA DE CONTROL*
- Propiedad **Valor al Activar:** debe definirse a *S*.
- Propiedad **Valor cuando NO Está Activado**: debe definirse a *N*.
- Propiedad **Correspondencia de Casillas de Control de Otros Valores:** debe definirse a *DESACTIVADO.*
- Propiedad **Etiqueta**: debe definirse a *"¿Imagen cargada en B.D.?"*.
- Propiedad **Tipo de Dato**: debe definirse a *CHAR*.
- Propiedad **Longitud Máxima**: debe definirse a *1*.
- Propiedad **Lienzo**: debe seleccionarse el único CANVAS que aparece cuando se despliega la lista.

- Propiedad **Posición X**: debe definirse a *341*.
- Propiedad **Posición Y**: debe definirse a *71*.
- Propiedad **Ancho**: debe definirse a *132*.
- Propiedad **Altura**: debe definirse a *14*.

Creando disparadores asociados a un botón

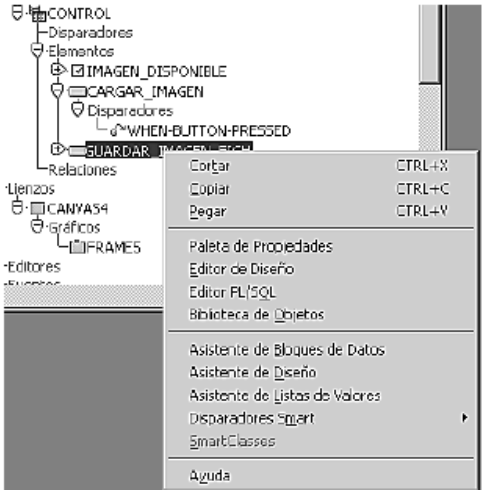
Cerramos la paleta de propiedades y volvemos al navegador de objetos para situarnos nuevamente en el bloque **CONTROL** y dentro del mismo en el elemento **CARGAR_IMAGEN**. Seguidamente desplegamos el contenido de este elemento y dentro del grupo *Disparadores* pulsamos el botón (**+**) para añadir un nuevo disparador del tipo **WHEN-BUTTON-PRESSED,** que nos abre una nueva ventana para introducir el código PL/SQL asociado al mismo. Dentro de la ventana de código, introduciremos el siguiente código PL/SQL:

```
BEGIN
-- El comando read_image_file consta de 3 partes:
/* En la primera parte, se indica la ruta completa donde se encuentra
el fichero indicando también el nombre del mismo
        Z:\Cursos\Cursos Oracle 11g\FORMS\imagenes\
   En la segunda parte, el tipo de fichero de acuerdo a las
especificaciones de Forms. Se puede consultar la ayuda para saberlo
       JPG
   En la tercera parte, se indica el nombre del bloque y del elemento
de tipo imagen que absorberá la imagen.
       empleado.imagen_emp
*/

CASE :EMPLEADO.NIF
WHEN '10000000A' THEN
   READ_IMAGE_FILE('Z:\Cursos\Cursos Oracle
11g\FORMS\imagenes\Empleado2.jpg', 'JPG', 'empleado.imagen_emp');
WHEN '10000000N' THEN
   READ_IMAGE_FILE('Z:\Cursos\Cursos Oracle
11g\FORMS\imagenes\Empleado1.jpg', 'JPG', 'empleado.imagen_emp');
WHEN '11000000P' THEN
   READ_IMAGE_FILE('Z:\Cursos\Cursos Oracle
11g\FORMS\imagenes\Empleado3.jpg', 'JPG', 'empleado.imagen_emp');
WHEN '12000000Q' THEN
   READ_IMAGE_FILE('Z:\Cursos\Cursos Oracle
11g\FORMS\imagenes\Empleado7.jpg', 'JPG', 'empleado.imagen_emp');
WHEN '12345678B' THEN
   READ_IMAGE_FILE('Z:\Cursos\Cursos Oracle
11g\FORMS\imagenes\Empleado8.jpg', 'JPG', 'empleado.imagen_emp');
WHEN '20000000B' THEN
```



```
    READ_IMAGE_FILE('Z:\Cursos\Cursos Oracle
11g\FORMS\imagenes\Empleado6.jpg', 'JPG', 'empleado.imagen_emp');
WHEN '30000000C' THEN
    READ_IMAGE_FILE('Z:\Cursos\Cursos Oracle
11g\FORMS\imagenes\Empleado9.jpg', 'JPG', 'empleado.imagen_emp');
ELSE
       NULL;
END CASE;
END;
```

Tenga en cuenta a la hora de replicar este código que las rutas que aparecen (Z:\Cursos\Cursos Oracle 11g\FORMS\imagenes), se refieren a las que se han utilizado durante la elaboración de este manual, pero que deberán ser ajustadas a las que usted tenga definidas para realizar las prácticas que se sugieren en el mismo. La carpeta **imágenes** con su contenido la podrá descargar desde la Web de la editorial y copiarla a su equipo.



Una vez implementado este código, cerramos el editor PL/SQL y volvemos al navegador de objetos para situarnos nuevamente en el bloque **CONTROL** y dentro del mismo en el elemento **GUARDAR_IMAGEN**. Seguidamente teniendo seleccionado este elemento pulsamos el botón derecho del ratón y se nos muestra el este menú desplegable.

Dentro de este menú seleccionamos la opción **Disparadores Smart** y a continuación el disparador **WHEN-BUTTON-PRESSED.** La diferencia entre utilizar el método anterior y este para implementar un disparador asociado a un elemento es que esta última opción únicamente nos muestra en primer lugar los disparadores más habituales que se pueden asociar a dicho elemento, mientras que la primera opción utilizada muestra todos los triggers compatibles con el elemento seleccionado.

A continuación dentro de la ventana del editor PL/SQL que se ha abierto introducimos el siguiente código:

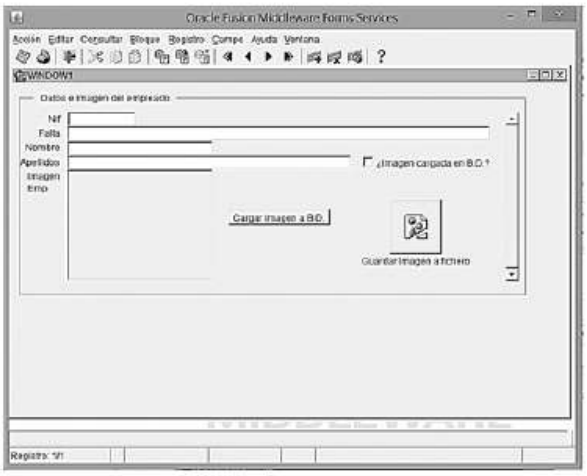
```
BEGIN
-- El comando WRITE_IMAGE_FILE de Forms se compone de los siguientes
argumentos:
/* Primer argumento: Ruta y nombre del fichero en el que se quiere
guardar la imagen
Segundo argumento: Formato de almacenamiento de la imagen según Forms.
```

```
Tercer argumento: Origen de la imagen (de qué bloque y elemento
copiamos la imagen)
Cuarto argumento (opcional): Tipo de compresión de la imagen al
almacenarla
Quinto argumento (opcional): Paleta de colores a utilizar en el
almacenamiento
*/

    BEGIN
   El comando WRITE_IMAGE_FILE de Forms se compone de los siguientes
      argumentos:
    /* Primer argumento: Ruta y nombre del fichero en el que se quiere
guardar la imagen
    Segundo argumento: Formato de almacenamiento de la imagen según
Forms.
    Tercer argumetno: Origen de la imagen (de qué bloque y elemento
copiamos la imagen)
    Cuarto argumento (opcional): Tipo de compresión de la imagen al
almacenarla
    Quinto argumento (opcional): Paleta de colores a utilizar en el
almacenamiento
*/

        IF :EMPLEADO.NIF IS NOT NULL THEN
               WRITE_IMAGE_FILE('Z:\EMPLEADO'||:EMPLEADO.NIF||'.JPG',
        'JFIF',
   'empleado.imagen_emp',
   no_compression,
   RGB);
END IF;
END;
```

Si no dispone de una unidad Z en su ordenador, cámbiela por una unidad que exista en un equipo y en la que tenga permisos de escritura.

Guardando, compilando y ejecutando el formulario

Concluidas las modificaciones anteriores cerramos el editor PL/SQL y almacenamos el formulario (**CTRL + S),** lo compilamos **(SHIFT + CTRL + K)** y generamos **(CTRL + T)**, para en último lugar ejecutarlo **(CTRL + R).**



El resultado de la ejecución será similar al mostrado en esta imagen.

A continuación vamos a ejecutar una serie de pasos durante la ejecución de este formulario:

- Dentro del menú **Consultar** seleccionamos **Ejecutar.**
- Para cada uno de los registros de los empleados mostrados pulsamos el botón **Cargar Imagen a B.D.** y comprobaremos cómo se van cargando las imágenes correspondientes a cada uno de ellos como se muestra en esta imagen.

Cuando hayamos recorrido todos los registros (los 16 existentes), dentro del menú **Acción** pulsamos sobre **Guardar.**

Seguidamente dentro del menú **Acción** seleccionamos la opción **Borrar Todo** y a continuación volvemos a la opción **Consultar** y seleccionamos **Ejecutar,** para ir navegando entre los distintos registros y comprobar que las imágenes han quedado asociadas a los distintos empleados.

Sobre cualquiera de los registros que tenga asociada una imagen pulsamos en el botón **Guardar Imagen a fichero** y a continuación nos vamos a la unidad Z:\ (o aquella que se haya definida en el código) para comprobar si existe un fichero del tipo *EMPLEADO<nif>.JPG* para a continuación abrirlo con un visor de fotografías.

Con estos pasos damos por concluida la práctica.

ALERTAS Y MENSAJES 25

INTRODUCCIÓN

En este capítulo se explica cómo interceptar los mensajes que devuelve el sistema al ejecutar nuestra aplicación y, si lo desea, a sustituirlos por otros que sean más adecuados para la gestión de la aplicación. También se aprenderá a manejar errores mediantes subprogramas incorporados y a crear alertas personalizadas para comunicarse con los usuarios.

TIPOS DE MENSAJES Y ALERTAS

Forms muestra mensajes en tiempo de ejecución para informar al usuario de los eventos que se producen en la sesión abierta contra la aplicación. Como programador, podrá suprimir o modificar alguno de estos mensajes, dependiendo de la naturaleza de los mismos y de su aplicación.

Los tipos de mensajes y alertas que se pueden producir dentro de Forms son los siguientes:

- Mensaje informativo.
- Mensaje de error.
- Mensaje de actividad en proceso.
- Alerta del sistema.
- Mensaje de aplicación.
- Alerta de la aplicación.

Mensaje informativo

Es un mensaje que informa al usuario del estado actual del procesamiento del programa, o proporciona información sensible para el contexto de uso de la aplicación. La visualización por defecto de este tipo de mensajes es en la *línea de mensajes de Forms.*

Se puede evitar la visualización de estos mensajes mediante programación, utilizando el disparador ON-MESSAGE.

Mensaje de error

Es un mensaje que informa al usuario de un error que evita la continuación de la acción actual dentro de la aplicación. La visualización por defecto de este tipo de mensajes es en la *línea de mensajes de Forms*.

Se puede evitar la visualización de estos mensajes mediante programación utilizando el disparador ON-ERROR.

Mensaje de actividad en proceso

Este tipo de mensajes informa al usuario de la actividad que se está registrando actualmente en el formulario (por ejemplo el mensaje: *en proceso*). Igual que los anteriores se muestra por defecto en la *línea de mensajes de Forms*.

Se puede evitar la visualización de estos mensajes definiendo la variable del sistema SUPPRESS_WORKING a valor True:

```
: SYSTEM. SUPPRESS_WORKI NG : = ' TRUE' ;
```

Alerta del sistema

Las alertas ofrecen información al usuario que requiere una confirmación o una respuesta a una pregunta, antes de que la aplicación pueda continuar operando.

Se muestran como una ventana modal, y cuando hay más de un mensaje en espera, entonces se muestran en la *línea de mensajes*, siendo el último mensaje el que se muestra como una alerta.

Mensaje de aplicación

Son mensajes que crea el programador para la aplicación mediante la función incorporada MESSAGE.

La visualización por defecto de estos mensajes es en la *línea de mensajes de Forms*.

Alerta de la aplicación

Son alertas que diseña el programador como parte de la interactividad de la aplicación y se muestran al usuario para obtener una respuesta mediante la función incorporada SHOW_ALERT.

DETECCIÓN DE ERRORES EN TIEMPO DE EJECUCIÓN

Cuando un código programado falla en la aplicación, no causa directamente una excepción en la unidad de programa o el disparador que realiza la llamada. Esto significa que el código siguiente continúa después de un fallo a menos que se realicen acciones para detectar el fallo y actuar en consecuencia.

Por ejemplo un caso que ilustra este mecanismo de funcionamiento es el siguiente:

- Tomemos una aplicación con un lienzo de tipo Toolbar llamado *CLIENTES* en el que hay un botón en el bloque *CONTROL* denominado *BOTON_STOCK.*
- Cuando se hace clic en el botón anterior, se desencadena el código programado en el disparador *WHEN-BUTTON-PRESSED* del mismo, que navega hasta el bloque *INVENTARIO* y allí realiza una consulta:

```
GO_BLOCK('INVENTARIO');
EXECUTE_QUERY;
```

- Si el procedimiento incorporado GO_BLOCK falla porque el bloque *INVENTARIO* no existe, o porque no acepta introducción de valores, entonces el procedimiento EXECUTE_QUERY se va a ejecutar igualmente, intentando consultar en el bloque incorrecto.

Para evitar estas situaciones Forms Builder ofrece funciones incorporadas que nos indican el resultado de la última acción de la pantalla:

- Form_Success.
- Form_Failure.
- Form_Fatal.

Si además queremos ahondar en el nivel de detalle del error que se ha producido, también podemos utilizar las siguientes funciones incorporadas que tratan más información sobre el error:

- Error_code.
- Error_text.
- Error_type.

Form_Success

Esta función devuelve dos posibles valores:

- TRUE: cuando el formulario ha ejecutado una acción correctamente.
- FALSE: cuando se ha producido un error o un error fatal.

Esta es la función que se suele utilizar para el control de errores en los formularios, dado que cuando devuelve FALSE tenemos la seguridad de que no se ha producido ningún tipo de error en la aplicación (sea o no de tipo fatal).

EJEMPLO

A continuación se muestra un código de ejemplo para el control de errores con la función FORM_SUCCESS, en el que se comprueba después de navegar al bloque INVENTARIO si la aplicación ha devuelto o no un error. En caso de que no haya errores se ejecutará la consulta dentro del bloque INVENTARIO y en otro caso, se mostrará un mensaje de error en la *línea de mensajes de Form*.

```
GO_BLOCK('INVENTARIO');
IF FORM_SUCCESS THEN
      EXECUTE_QUERY;
ELSE
      MESSAGE('Ha ocurrido un error al navegar al bloque
      inventario');
END IF;
```


Form_Failure

Esta función devuelve dos posibles valores:

- **TRUE**: cuando se ha producido un error no fatal.
- **FALSE**: cuando no se ha producido ningún error o se ha producido un error fatal.

Form_Fatal

Esta función devuelve dos posibles valores:

- **TRUE**: cuando se ha producido un error fatal.
- **FALSE**: cuando no se ha producido ningún error o se ha producido un error no fatal.

Error_Code

Esta función devuelve el número de error que se ha producido en la aplicación. El valor que devuelve es de tipo NUMBER.

Error_Text

Esta función devuelve el texto descriptivo del error. El valor que devuelve es de tipo CHAR.

Error_Type

Esta función devuelve el tipo de error que se ha producido. El valor que devuelve es de tipo CHAR y puede ser uno de los siguientes:

- **FRM.** Cuando el error devuelto es producto de una operación anómala en Forms Builder.
- **ORA.** Cuando el error es producto de una operación anómala contra la base de datos.

NIVEL DE GRAVEDAD DE LOS MENSAJES

Forms Builder clasifica cada mensaje con un nivel de gravedad que indica si la información es crítica o trivial. Cuanto mayor sea el número recibido en el mensaje de error, más crítico será el mensaje. Hay seis niveles definidos para agrupar los mensajes:

Nivel de Gravedad	Descripción
0	Todos los mensajes.
5	Reafirma una condición obvia
10	El usuario ha cometido un error de procedimiento
15	El usuario intenta una acción para la que no está diseñada la pantalla.
20	No se puede continuar la acción deseada debido a un problema de un disparador o alguna otra condición pendiente.
25	Una condición que puede provocar que la pantalla se ejecute incorrectamente
> 25	Mensajes que no se pueden suprimir.

En un disparador se puede especificar que la aplicación solo emita los mensajes por encima de un nivel de gravedad específico. Para ello se asigna un valor a la variable del sistema MESSAGE_LEVEL.

El valor por defecto que tiene la variable MESSAGE_LEVEL cuando se arranca una aplicación es 0. Esto significa que se muestran todos los mensajes de todos los niveles de gravedad.

Ejemplo de supresión de mensajes por control de nivel

En este ejemplo se muestra un código que se podría utilizar para suprimir ciertos mensajes por el nivel de gravedad de los mismos.

Nos situamos en el supuesto de un disparador WHEN-BUTTON-PRESSED que mueve un registro hacia arriba utilizando la función incorporada UP. Si el cursor ya está en el primer registro, la función falla y normalmente se muestra el mensaje FRM-40100, que es un mensaje de nivel de gravedad 5. Sin embargo, nosotros vamos a suprimir este mensaje y vamos a mostrar uno propio, restableciéndose con posterioridad el nivel de mensajes a valor 0. Todo ello se realiza con el siguiente código:

```
: SYSTEM. MESSAGE_LEVEL  : =  ' 5' ;
UP;
```

```
IF NOT FORM_SUCCESS THEN
MESSAGE('Ya está en el primer registro');
END IF;
:SYSTEM.MESSAGE_LEVEL := '0';
```

MENSAJES DE ACTIVIDAD EN PROCESO

Los mensajes de actividad en proceso, durante la ejecución de un formulario, se muestran cuando Forms está ocupado procesando una acción.

Por ejemplo, cuando realiza una consulta se recibe el mensaje: *Working…* (en castellano *Procesando…*).

Se puede suprimir este tipo de mensajes definiendo la variable del sistema SUPPRESS_WORKING a valor TRUE:

```
:SYSTEM.SUPPRESS_WORKING := 'TRUE';
```

Un método muy habitual es definir estas variables del sistema en cuanto se inicia la aplicación (en su primer formulario) y dentro del disparador WHEN-NEW-FORM-INSTANCE.

LA EXCEPCIÓN FORM_TRIGGER_FAILURE

Los disparadores fallan solo cuando se produce una de las siguientes condiciones:

- Cuando no se ha tratado una excepción, porque no se ha definido dentro del disparador una sección EXCEPTION.
- Cuando se solicita que el disparador falle emitiendo la excepción incorporada FORM_TRIGGER_FAILURE.

Se puede definir y lanzar esta excepción en las siguientes partes del código:

- En la parte ejecutable de un disparador para saltar las acciones restantes del mismo y hacer que el disparador se pare y emita un fallo.
- Dentro de una manejador de excepciones (sección EXCEPTION) para que el disparador falle después de que se hayan ejecutado sus propias acciones de manejo de excepciones.

A continuación se muestra un ejemplo de código de uso de FORMS_TRIGGER_FAILURE dentro de una sección EXCEPTION:

```
BEGIN

    SELECT NOMBRE||' '||APELLIDO1
    INTO :CLIENTES.NOMBRE
    FROM CLIENTES
    WHERE DNI = :CLIENTES.DNI;

EXCEPTION

  WHEN NO_DATA_FOUND THEN
  MESSAGE('Cliente con DNI no identificado');
  RAISE FORM_TRIGGER_FAILURE;

END;
```

DISPARADORES PARA INTERCEPTAR MENSAJES DEL SISTEMA

Dentro de Forms Builder disponemos de dos disparadores que permiten interceptar los mensajes que emite el sistema antes de que se lleguen a mostrar en pantalla. Estos disparadores son:

- ON-ERROR
- ON-MESSAGE

Estos disparadores sustituyen la visualización del mensaje, por tanto el usuario no los verá a menos que el programador haya implementado algún código de visualización dentro de estos propios disparadores.

Estos disparadores se pueden definir a cualquier nivel dentro del formulario. Por ejemplo, un disparador ON-ERROR a nivel de elemento solo interceptará mensajes de error que se produzcan cuando el control está en dicho elemento. No obstante, si define uno o ambos disparadores a nivel de formulario, se interceptarán todos los menajes que provocan el arranque independientemente del objeto de la pantalla actual que cause el error o el mensaje.

On-Error

Este disparador se ejecuta cuando se genera un mensaje de error del sistema. Por tanto, se utiliza para:

- Detectar errores de Forms y del servidor de base de datos Oracle. Este disparador puede llevar a cabo acciones correctivas basadas en el código del error producido.
- Sustituir el mensaje de error por defecto por un mensaje personalizado para esta aplicación.

EJEMPLO

En este ejemplo se captura el mensaje de error del sistema 40202 dentro del disparador ON-ERROR y se envía un nuevo texto del mensaje al usuario.

```
IF ERROR_CODE = 40202 THEN
      MESSAGE('Debe rellenar este campo obligatoriam.');
ELSE
      MESSAGE(ERROR_TYPE||' - '||TO_CHAR(ERROR_CODE)||':'
              ||:ERROR_TEXT);
END IF;

RAISE FORM_TRIGGER_FAILURE;
```

On-Message

Este disparador se ejecuta cuando se genera un mensaje informativo del sistema. Por tanto, se utiliza para capturar estos mensajes y sustituirlos por otros personalizados según sea o no conveniente informar al usuario de los mismos. El manejo es similar al del disparador ON-ERROR pero para capturar el detalle de los mensajes se utilizan las siguientes funciones incorporadas:

Función	Descripción
MESSAGE_CODE	Para obtener el número del mensaje. Es de tipo NUMBER.
MESSAGE_TEXT	Para obtener el texto descriptivo del mensaje. Es de tipo CHAR.
MESSAGE_TYPE	Para obtener el tipo del mensaje. Es de tipo CHAR y puede tener los siguientes valores: • **FRM**. Mensaje que procede de Forms Builder. • **ORA**. Mensaje que procede de la base de datos Oracle. • **NULL**. Cuando no se ha emitido aún ningún mensaje en esta sección.

EJEMPLO

En este ejemplo se capturan los mensajes informativos del sistema 40350 y 40301 dentro del disparador ON-MESSAGE y se envía un nuevo texto del mensaje al usuario.

```
IF MESSAGE_CODE IN (40350,40301) THEN
      MESSAGE('No hay clientes para los criterios de búsqueda
intruducidos.');
ELSE
      MESSAGE(MESSAGE_TYPE||' - '||TO_CHAR(MESSAGE_CODE)
               ||':'||:MESSAGE_TEXT);
END IF;
```

ALERTAS

Las alertas son un método alternativo para comunicarse con el usuario, dado que se muestran en una ventana modal. Las alertas proporcionan una forma efectiva de llamar la atención y forzar que el usuario responda al mensaje antes de que el procesamiento de la aplicación pueda continuar.

Normalmente se utilizan las alertas cuando se necesita llevar a cabo alguna de las siguientes operaciones:

- Mostrar un mensaje que el usuario no puede ignorar y debe confirmar.
- Hacer una pregunta al usuario donde hay hasta tres respuestas adecuadas.

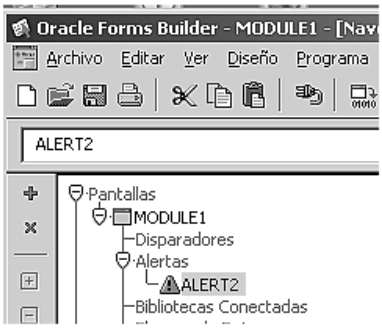
La visualización y las respuestas que se den a las alertas se pueden gestionar mediante subprogramas incorporados en Forms Builder.

Las etapas que hay que cubrir para la gestión de una alerta dentro de Forms son:

- Crear la alerta durante el diseño del formulario y definir sus propiedades en la *Paleta de Propiedades*.
- Activar la alerta en tiempo de ejecución mediante funciones incorporadas y programar las acciones a llevar a cabo en función de la respuesta del usuario.

Cómo crear una alerta

Al igual que el resto de objetos vistos hasta ahora, una alerta se crea en el *Navegador de Objetos* de Forms. En concreto dentro del grupo de objetos **Alertas.**



Una vez dentro del grupo **Alertas** pulsaremos el botón de *Crear* (+) para añadir una nueva alerta a nuestro formulario. A continuación se muestra una imagen del resultado de efectuar esta operación.

Una vez creada la alerta tendremos que ajustar las propiedades del nuevo elemento a través de la *Paleta de Propiedades* de Forms.

Propiedades de una alerta

Las propiedades funcionales de una alerta son las siguientes:

- Título.
- Mensaje.
- Estilo de Alerta.
- Etiqueta del Botón 1, 2 o 3.
- Botón de Alerta por Defecto.

TÍTULO

Esta propiedad permite asignar un título a la ventana modal que se mostrará para la alerta. Dicho texto se mostrará en el marco que recubre a la alerta.

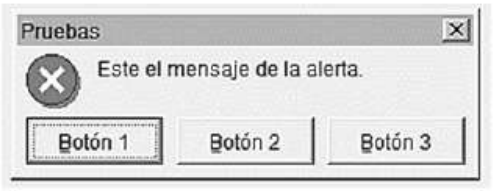
MENSAJE

Esta propiedad permite asignar el texto del mensaje que queremos que se muestre en la alerta. Puede contener varias líneas aunque como máximo se admiten 200 caracteres.

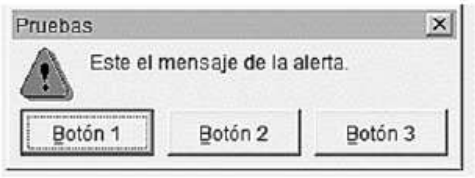
ESTILO DE ALERTA

Esta propiedad permite identificar el tipo de alerta que se quiere mostrar. Los tipos son:

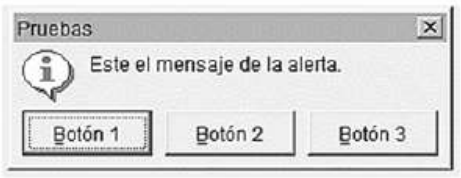
- Parar.
- Atención.
- Nota.



Las alertas de tipo **Parar** tienen esta forma.



Las alertas de tipo **Atención** tienen esta forma.



Las alertas de tipo **Nota** tienen esta forma.

Como se puede observar, la única diferencia que existe entre los tres tipos de alertas es el símbolo que se muestra en cada una de ellas. Normalmente con estos símbolos se pretende captar la atención del usuario en función de la gravedad de la alerta que se esté mostrando, siendo la alerta de tipo **Parar** la que reflejaría mayor gravedad y por tanto el usuario debería mostrar mayor atención.

ETIQUETA DEL BOTÓN 1, 2 O 3

Es el título que queremos que se muestre para cada uno de los botones que van a aparecer en la alerta.

Cuando una etiqueta de botón se deja vacía, significa que ese botón no aparecerá en la alerta.

En cualquiera de los tres tipos de alertas el número de botones que se pueden mostrar es como mínimo 1 botón y como máximo 3.

BOTÓN DE ALERTA POR DEFECTO

Toda alerta debe tener asignada obligatoriamente un botón por defecto que aparecerá resaltado (con una línea de puntos alrededor) cuando se muestre la alerta.

Set_Alert_Property

Este procedimiento incorporado de Forms permite cambiar las propiedades del mensaje y título de la alerta, utilizando programación dentro del formulario.

Esto supone que no es necesario diseñar una alerta por cada mensaje que se quiera mostrar en el formulario, sino que a veces es más útil crear una única alerta por cada tipo de alerta que se quiera mostrar en el programa y a partir de ahí, utilizando el procedimiento SET_ALERT_PROPERTY dentro de nuestro formulario, podemos cambiar las propiedades del mensaje y título de la misma, según nuestras necesidades.

SINTAXIS:

```
SET_ALERT_PROPERTY('alert_name', property, 'message');
```

Parámetro	Descripción
alert_name	Es el nombre de alerta que se ha indicado en el *Navegador de Objetos* de Forms.
property	Especifica el nombre de la propiedad que se quiere modificar. Siendo posible indicar los cambios para las siguientes propiedades: • *ALERT_MESSAGE_TEXT*: para cambiar el texto del mensaje. • *TITLE*: para cambiar el título de la alerta
message	El texto que se asocia a la propiedad que se cambia.

Set_Alert_Button_Property

Este procedimiento incorporado de Forms permite cambiar la propiedad etiqueta de botón de una alerta en tiempo de ejecución.

SINTAXIS:

```
SET_ALERT_BUTTON_PROPERTY('alert_name', button, property,
                          'value');
```

Parámetro	Descripción
alert_name	Es el nombre de alerta que se ha indicado en el *Navegador de Objetos* de Forms.
button	Especifica el número correspondiente al botón de la alerta que se pretende modificar. Los valores admitidos son: • *ALERT_BUTTON1* • *ALERT_BUTTON2* • *ALERT_BUTTON3*
property	Se incluirá siempre la palabra *LABEL* que especifica que se modifica la propiedad etiqueta del botón.
value	Se indicará el texto que queremos mostrar en el botón. Cuando queremos que no se muestre uno de los botones, en este parámetro introduciremos NULL.

Show_Alert

Esta función incorporada en Forms permite mostrar una alerta en tiempo de ejecución y devolver la respuesta del usuario (el botón pulsado por el usuario).

SINTAXIS:

```
variable := SHOW_ALERT('alert_name');
```

Parámetro	Descripción
alert_name	Es el nombre de alerta que se ha indicado en el *Navegador de Objetos* de Forms.

Al ser SHOW_ALERT una función, siempre devuelve un valor. Por tanto será necesario recoger el valor de esta función por una variable que se haya definido en la aplicación: puede ser una variable a nivel de la sección declarativa de un subprograma, un elemento de un bloque, una variable del sistema o un parámetro. Esta variable en cualquier caso deberá ser de tipo NUMBER.

Pese a devolver un valor de tipo NUMBER, cuando queramos comparar dicho valor con el botón que ha pulsado el usuario en la alerta, deberemos referenciar dicho valor utilizando las siguientes constantes:

Si el número es igual a	El usuario ha seleccionado…
ALERT_BUTTON1	El botón 1
ALERT_BUTTON2	El botón 2
ALERT_BUTTON3	El botón 3

EJEMPLO

A continuación se muestra un breve código de programa que visualiza una alerta y devuelve el resultado en una variable, que posteriormente se compara para determinar si el usuario ha pulsado o no el botón 1. En caso afirmativo realiza un borrado del registro visualizado dentro del bloque.

```
DECLARE
   RESULTADO              NUMBER;
BEGIN
   RESULTADO := SHOW_ALERT('alerta1');
   IF RESULTADO = ALERT_BUTTON1 THEN
      DELETE_RECORD;
   END IF;
END;
```

CREANDO UNA ALERTA 26

INTRODUCCIÓN

Durante esta práctica aprenderemos a crear una alerta y a gestionarla en tiempo de ejecución.

Apertura de Forms Builder y formulario

Como primer paso abriremos Forms Builder y una vez hecho esto desde la barra de menús **Archivo** seleccionaremos **Abrir,** para seleccionar el formulario **PRACTICA12** que creamos en los capítulos anteriores.

Guardando el formulario con otro nombre

Seguidamente cambiamos la propiedad *Nombre* del formulario para ponerle **PRACTICA13** y a continuación desde el menú **Archivo** seleccionamos **Guardar Como** e indicamos **PRACTICA13.FMB**.

Creando una alerta

Comenzamos la creación de la alerta situándonos en el grupo **Alertas** dentro del *Navegador de Objetos* de Forms y pulsamos el botón añadir (**+**) para crear una nueva alerta.

Ajustando las propiedades de una alerta

A continuación nos situamos en la nueva alerta creada y pulsamos **F4** para abrir la *Paleta de Propiedades* del objeto, donde asignaremos los siguientes valores para las propiedades que se indican:

- Propiedad **Nombre:** debe definirse a *FALTA_CODEMP*.
- Propiedad **Título:** debe definirse como *'Control de errores de la aplicación'.*
- Propiedad **Mensaje:** debe definirse a *'Para poder cargar una imagen antes debe de consultar los empleados para que se muestre el código que tiene asignado cada uno'*.
- Propiedad **Estilo de Alerta**: debe definirse a *PARAR*.
- Propiedad **Etiqueta de Botón 1:** debe definirse a *ACEPTAR.*
- Propiedad **Etiqueta de Botón 2:** debe dejarse sin contenido (vacío).
- Propiedad **Etiqueta de Botón 3:** debe dejarse sin contenido (vacío).
- Propiedad **Color de Fondo:** debe definirse a *r100g88b88.*
- Propiedad **Nombre de Fuente:** debe definirse a *VERDANA.*
- Propiedad **Tamaño de fuente:** debe definirse a *10.*

Si no nos hemos conectado previamente a la base de datos, lo tenderemos que hacer en este punto con las credenciales utilizadas durante el curso: **PEPERF / PEPITO.**

Añadiendo más alertas

Seguidamente cerramos la paleta de propiedades y volvemos al *Navegador de objetos* para crear una nueva alerta de la misma forma que se ha indicado antes (pulsando el botón de añadir).

Una vez creada la nueva alerta le asociamos las siguientes propiedades:

- Propiedad **Nombre:** debe definirse a *AVISOS_VARIOS*.
- Propiedad **Estilo de Alerta**: debe definirse a *ATENCIÓN*.
- Propiedad **Etiqueta de Botón 1:** debe definirse a *ACEPTAR.*
- Propiedad **Etiqueta de Botón 2:** debe definirse a CANCELAR.

Añadiendo programación para gestión de las alertas

Cerramos la paleta de propiedades de la última alerta creada y nos situamos dentro del código PL/SQL del disparador WHEN-BUTTON-PRESSED del elemento **GUARDAR_IMAGEN_FICH** del bloque **CONTROL.**

Una vez dentro del código del disparador, añadimos el siguiente código:

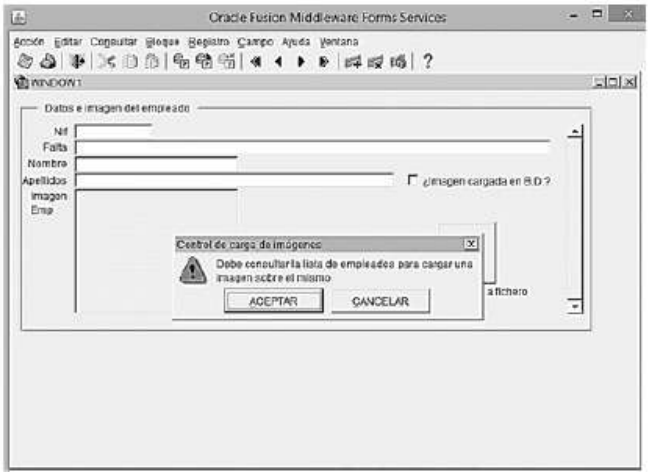
```
DECLARE
       RESULTADO                      NUMBER;
       V_Existe_el PDF                BOOLEAN;
       V_Long_File                    NUMBER;
       V_Block_File                   NUMBER;
BEGIN
       IF ….
              WRITE_IMAGE ….
              UTL_FILE. FGETATTR('K:\','EMPLEADO'||:EMPLEADO.NIF||
   '.JPG',V_Existe_el PDF,V_Long_File,  V_Block_File);

              IF NOT V_Existe_el PDF THEN
                     SET_ALERT_PROPERTY('AVISOS_VARIOS',
                            TITLE,
                            'Control del código de empleado');
                     SET_ALERT_PROPERTY('AVISOS_VARIOS',
                            ALERT_MESSAGE_TEXT,
                            'Al empleado codificado con '||
                            'el NIF '||
                            TRIM(TO_CHAR(:EMPLEADO.NIF))||
                            'no se le ha asignado una '||
                            'fotografía. No obstante '||
                            '¿Quiere continuar?');
                     RESULTADO := SHOW_ALERT('AVISOS_VARIOS');
                     IF RESULTADO = ALERT_BUTTON1 THEN
                            READ_IMAGE_FILE('K:\Cursos Oracle\Cursos
   Oracle 11g\FORMS\imagenes\Empleado_sinimagen.jpg',
                                   'JPG', 'empleado.imagen_emp');
                     END IF;
      END IF;
   ELSE
       RESULTADO := SHOW_ALERT('FALTA_CODEMP');
   END IF;
END;
```

Ahora cerramos el editor PL/SQL y volvemos al *Navegador de Objetos* para abrir el código PL/SQL del disparador WHEN-BUTTON-PRESSED del elemento **CARGAR_IMAGEN** del bloque **CONTROL.**

Una vez dentro del código del disparador, añadimos el siguiente código:

```
DECLARE
      RESULTADO    NUMBER;
      ….
      ….
WHEN '30000000C' THEN
      READ_IMAGE_FILE('Z:\Cursos\Cursos Oracle
      11g\FORMS\imagenes\Empleado9.jpg', 'JPG',
      'empleado.imagen_emp');
ELSE
      IF :EMPLEADO.NIF IS NULL THEN
            SET_ALERT_PROPERTY('AVISOS_VARIOS',TITLE,
                        'Control de carga de imágenes');
            SET_ALERT_PROPERTY('AVISOS_VARIOS',
                                    ALERT_MESSAGE_TEXT,
                        'Debe consultar la lista de '||
                        'empleados para cargar una '||
                        'imagen sobre el mismo');
            RESULTADO := SHOW_ALERT('AVISOS_VARIOS');
      END IF;

END CASE;
```

Guardando, compilando y ejecutando el formulario

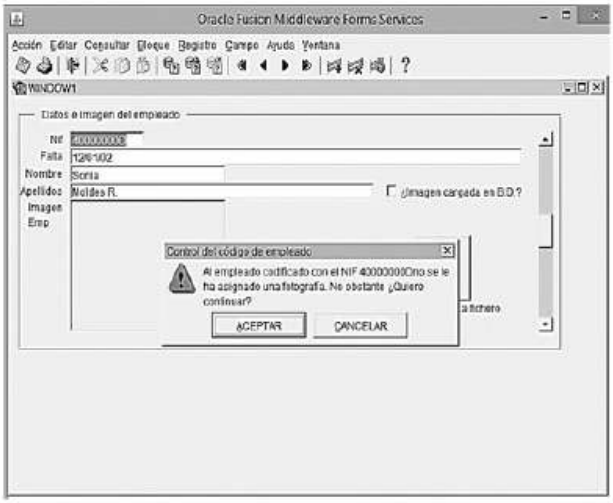


Concluidas las modificaciones anteriores cerramos el editor PL/SQL y almacenamos el formulario (**CTRL + S),** lo compilamos **(SHIFT + CTRL + K)** y generamos **(CTRL + T)**, para en último lugar ejecutarlo **(CTRL + R).**

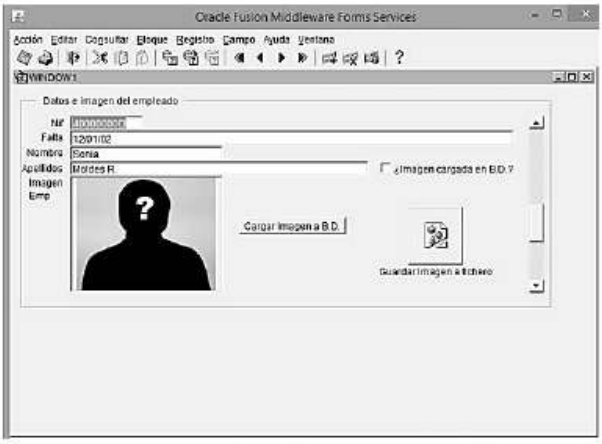
Una vez ejecutada la aplicación, no debemos realizar ninguna operación de consulta.

En cuanto aparezca el formulario pulsaremos sobre el botón **Cargar_Imagen a B.D.**, lo cual provocará que se

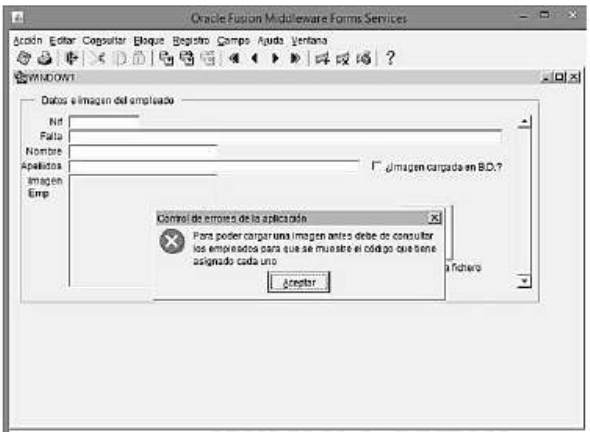
desencadene el código que hemos añadido en la sección anterior y se mostrará la alerta *AVISOS_VARIOS* como se muestra en la imagen de la página anterior.



Pulsamos **Aceptar** y realizamos una nueva prueba de funcionamiento. En este caso vamos a pulsar sobre el menú **Consultar** y luego **Ejecutar**. Con la barra de desplazamiento iremos navegando hasta encontrarnos con el empleado con NIF *40000000D* y a continuación pulsaremos sobre el botón **Guardar Imagen a fichero** lo que provocará que se muestre una nueva alerta como la que se muestra en esta imagen.



Pulsamos **Aceptar** y automáticamente (como habíamos programado) se carga una imagen para este empleado, obteniendo el resultado que se muestra en esta pantalla.



A continuación realizamos la última prueba de funcionamiento. Para ello desde el menú **Registro** seleccionamos la opción **Insertar** y seguidamente pulsamos el botón **Guardar Imagen a fichero** lo que provocará que se muestre esta alerta.

SUPUESTO PRÁCTICO 3

MODIFICACIONES PREVIAS EN LA BASE DE DATOS

Antes de comenzar a desarrollar este supuesto práctico, es necesario realizar las siguientes modificaciones en la base de datos hospital que estamos utilizando durante los supuestos prácticos de este manual:

MODIFICACIÓN EN LA BASE DE DATOS:

Para poder seguir las prácticas propuestas en este curso debe de efectuar una alteración en la tabla EMPLEADO de la base de datos, según se especifica a continuación:

- Ejecute SQL*PLUS o cualquier otra herramienta de conexión a la base de datos e intérprete de comandos SQL.
- Conéctese con las credenciales PEPERF/PEPITO (usuario y contraseña respectivamente).
- Ejecute la siguiente sentencia SQL:

```
ALTER TABLE ENFERMO
ADD (FOTOCASA LONG RAW);
ALTER TABLE PLANTILLA
ADD (FOTODNI LONG RAW);
```

MÓDULO ENFERMOS-HOSPITAL

En este supuesto práctico se deberán realizar los siguientes cambios en el formulario **ENFERMOS_HOSPITAL**:

- Añadir al bloque **ENFERMO** el nuevo campo **FOTOCASA** que se ha añadido a la tabla con el mismo nombre de la base de datos, y asignarle las propiedades adecuadas para que soporte imágenes de tipo JGP, ajustando el ancho y el largo del cuadro de la imagen.
- Visualizar el nuevo campo **FOTOCASA** dentro del marco **Datos del Paciente Inscrito en el Hospital**.
- Mostrar como campos de solo lectura, dentro del marco **Datos del Paciente Inscrito en el Hospital,** los siguientes valores del enfermo: **Nombre, Apellidos, Dirección.**
- Cada vez que se seleccione un enfermo del marco **Relación de Pacientes del Hospital**, aparecerá automáticamente la información indicada en el marco **Datos del Paciente Inscrito**. Para conseguir esto hay que duplicar los elementos que se necesiten en el bloque ENFERMO, ponerlos como elementos de solo lectura sin estar asociados a base de datos y poner la propiedad del elemento de visualización que se denomina **Sincronizar con Elemento.**
- Crear una alerta denominada **NOCASA** del tipo ***Nota*** con un único botón y las siguientes propiedades:
  - Texto del título: **Mensajes de control del desarrollador**.
  - Texto del mensaje: **El código de inscripción está vacío o no se corresponde con ninguno de los válidos para la carga de la imagen**.
  - Texto del botón 1: **Aceptar.**
- Crear un bloque manual que NO esté asociado a base de datos, denominado **CONTROLES**. En dicho bloque se deberán crear 2 botones con las operaciones que se indican a continuación:
  - El primer botón se etiquetará como **"Visualizar lista de valores"** y no aceptará navegación por teclado ni por ratón. Al pulsarlo (WHEN-BUTTON-PRESSED) deberá ejecutar el código siguiente:

    ```
    DO_KEY( ' LI ST_VALUES' ) ;
    ```

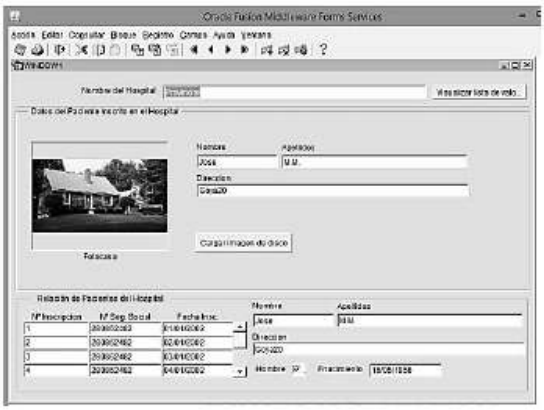
  - El segundo botón que tampoco aceptará navegación por teclado ni por ratón, se etiquetará como **"Cargar imagen de disco"** y al pulsarlo deberá ejecutar el siguiente código, asignando a la variable RUTA el valor correspondiente al directorio de Windows donde usted ubique las imágenes que se proporcionan para llevar

a cabo este supuesto práctico. Tenga en cuenta que la ruta debe ser completa, y no terminará en /. Por ejemplo una ruta válida posible sería 'C:\IMAGENES\CURSO_FORMS':

```
DECLARE
        RESULTADO       NUMBER;
        RUTA            VARCHAR2(1000);
BEGIN
        RUTA := ruta carga imágenes;
        IF (:ENFERMO.NUMSEGSOCIAL < 280000000) OR
           (:ENFERMO.NUMSEGSOCIAL > 289999999) THEN
               RESULTADO := SHOW_ALERT('NOCASA');
        ELSIF       :ENFERMO.NUMSEGSOCIAL       BETWEEN
280000001 AND 280862480 THEN
                READ_IMAGE_FILE(RUTA||'\casa1.jpg',
                                'JFIF',
                                'enfermo.fotocasa');
        ELSIF :ENFERMO.NUMSEGSOCIAL = 280862481 THEN
                READ_IMAGE_FILE(RUTA||'\casa2.jpg',
                                'JFIF',
                                'enfermo.fotocasa');
        ELSIF :ENFERMO.NUMSEGSOCIAL = 280862482 THEN
                READ_IMAGE_FILE(RUTA||'\casa3.jpg',
                                'JFIF',
                                'enfermo.fotocasa');
        ELSIF :ENFERMO.NUMSEGSOCIAL = 280862483 THEN
                READ_IMAGE_FILE(RUTA||'\casa4.jpg',
                                'JFIF',
                                'enfermo.fotocasa');
        ELSIF :ENFERMO.NUMSEGSOCIAL = 280862484 THEN
                READ_IMAGE_FILE(RUTA||'\casa5.jpg',
                                'JFIF',
                                'enfermo.fotocasa');
        ELSE
                READ_IMAGE_FILE(RUTA||'\casa10.jpg',
                                'JFIF',
                                'enfermo.fotocasa');
        END IF;
END;
```

- Para poder almacenar imágenes en la base de datos se realizará el siguiente proceso en tiempo de ejecución:
    - Se deberá utilizar el botón **Cargar imagen de disco** para poder visualizar una imagen en el marco.
    - Una vez visualizada se almacenará con el botón de **Grabar** que se encuentra en la barra predefinida de Forms Runtime.

- Se ejecutará este proceso para al menos 2 enfermos distintos de cada hospital.
- Se recorrerán todos los enfermos para observar las diferencias entre los que tienen y no tienen fotografía de su casa.



A continuación se muestra una imagen para ilustrar cómo debe de quedar el formulario después de implementar los cambios indicados en este supuesto.

MÓDULO PLANTILLA-SANITARIA

En este supuesto práctico se deberán realizar los siguientes cambios en el formulario **PLANTILLA_SANITARIA:**

- Añadir al bloque **PLANTILLA** el nuevo campo que se ha incorporado a la base de datos: **FOTODNI,** para que pueda ver y cargar fotografías de tipo JFIF, ajustadas al marco de la imagen.
- Crear un botón con la etiqueta **Cargar imagen de disco** dentro del bloque **MONEDA** que realice la siguiente operativa, teniendo en cuenta que la ruta debe ser completa, y no terminará en /. Por ejemplo una ruta válida posible sería 'C:\IMAGENES\CURSO_FORMS':

```
DECLARE
    RESULTADO     NUMBER;
    RUTA          VARCHAR2(1000);

BEGIN
    RUTA := ruta carga imágenes;
    if substr(:plantilla.nif,1,1) = '1' then
          read_image_file(RUTA||'\Empleado1.jpg','JFIF',
                          'PLANTILLA.FOTODNI');
    elsif substr(:plantilla.nif,1,1) = '2' then
          read_image_file(RUTA||'\Empleado2.jpg','JFIF',
                          'PLANTILLA.FOTODNI');
```

```
    elsif substr(:plantilla.nif,1,1) = '3' then
          read_image_file(RUTA||'\Empleado3.jpg','JFIF',
                          'PLANTILLA.FOTODNI');
    elsif substr(:plantilla.nif,1,1) = '4' then
          read_image_file(RUTA||'\Empleado4.jpg','JFIF',
                          'PLANTILLA.FOTODNI');
    elsif substr(:plantilla.nif,1,1) = '5' then
          read_image_file(RUTA||'\Empleado5.jpg','JFIF',
                          'PLANTILLA.FOTODNI');
    else resultado:=show_alert('NOFOTO');
    end if;

END;
```

- Crear una alerta denominada **NOFOTO** del tipo ***Atención*** con un único botón y las siguientes propiedades:
  - Texto del título: **Mensajes del desarrollador del programa**.
  - Texto del mensaje: **No se dispone de fotografía para el NIF que actualmente está consultando o bien el mismo está vacío.**
  - Texto del botón 1: **Aceptar.**
- Crear un botón denominar **LISTAR** dentro del bloque **MONEDA** que permite visualizar una lista de valores de cualquier campo.
- Probar la aplicación realizando operaciones de grabación de imágenes en la base de datos.

A continuación se muestra una imagen que ilustra cómo debe de quedar el formulario después de implementar los cambios indicados en este supuesto.

VENTANAS Y LIENZOS 27

INTRODUCCIÓN

Oracle Forms se beneficia del entorno de interfaz gráfico del que dispone el usuario en su equipo y permite mostrar un módulo de pantalla en diversos lienzos y en varias ventanas. En este capítulo podrá familiarizarse con el objeto *ventana* y los tipos de lienzos que se pueden manejar.

¿Qué es una ventana?

Una ventana es un contenedor para todos los objetos visuales que componen una aplicación de Forms. Es similar a un marco de imagen vacío.

El gestor de ventanas proporciona los controles que dan funcionalidad a la misma, tales como desplazar, mover y cambiar el tamaño, y por tanto también se puede minimizar o maximizar.

Una única pantalla de una aplicación puede incluir varias ventanas.

¿Qué es un lienzo?

Un lienzo es una superficie dentro de un contenedor de ventana en el que se colocan objetos visuales de la interfaz gráfica (campos, botones, etc.). Es similar al lienzo en el que se pinta un cuadro.

Para ver un lienzo y su contenido en tiempo de ejecución, debe mostrarse dentro de una ventana y por tanto únicamente se podrá visualizar en la ventana que tenga asociada.

Cuando un elemento a visualizar en un lienzo no tiene asociado ninguno, se dice que es un *elemento de lienzo nulo* y no se mostrará en tiempo de ejecución.

¿Qué es un Viewport?

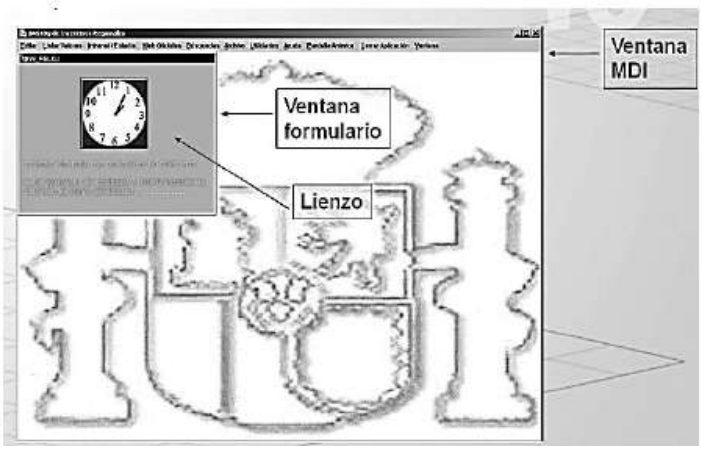
Un Viewport es un atributo de un lienzo. Es de hecho la parte visible del lienzo o una vista del mismo.

¿Qué es la ventana MDI?

La ventana MDI (Runform) es el "frame" (marco) que contiene la ejecución de la aplicación, es decir, es la contenedora de todas las ventanas y lienzos que se diseñan en un formulario.

No se pueden manejar las propiedades en tiempo de diseño de la ventana MDI dado que no es un objeto propio del formulario. Únicamente se pueden alterar las propiedades mediante código programado.

El nombre de la ventana MDI para hacer referencia dentro del código programado es FORMS_MDI_WINDOW.



En esta imagen se puede observar la diferencia de representación de la ventana MDI, una ventana de un formulario y un lienzo de contenido que se mostraría en una aplicación en tiempo de ejecución.

LA VENTANA POR DEFECTO

Cuando se crea un nuevo módulo de pantalla, Forms Builder crea implícitamente una nueva ventana. Por tanto cada módulo de pantalla nuevo tiene una ventana predefinida que se denomina *WINDOW1*. Se puede suprimir o cambiar este nombre dentro de los atributos de dicho objeto ventana.

Usos y ventajas de una ventana nueva

Se pueden crear ventanas adicionales en las que mostrar su aplicación en pantalla. Una nueva o segunda ventana ofrece la posibilidad de realizar las siguientes acciones:

- Mostrar dos o más lienzos de contenidos a la vez.
- Crear módulos de contenidos de pantalla diferenciados (dividir la pantalla en módulos).
- Cambiar entre lienzos sin sustituir el inicial.
- Aprovechar funcionalidades del gestor de ventanas de Windows, como por ejemplo, minimizar o maximizar.

Las nuevas ventanas que se crean no tienen por qué ser del mismo tipo que las anteriores, se pueden crear ventanas de distinto tipo.

TIPOS DE VENTANAS

Se pueden diseñar dos tipos de ventanas:

- Ventanas modales
- Ventanas NO modales.

A su vez dentro de estos tipos, se puede crear dos estilos diferentes de ventanas en cuanto a la visualización que se va a ofrecer de las mismas:

- Ventanas de estilo *Documento* (son las ventanas de contenido).
- Ventanas de estilo *Diálogo* (son las ventanas con aspecto similar a una alerta).

Ventana Modal

Es una ventana restringida a la que el usuario debe responder antes de mover el foco de entrada a otra ventana. Las ventanas modales deben cumplir lo siguiente:

- Deben cerrarse para poder devolver el control a una ventana NO modal.
- Se activan cuando se muestran.
- Requieren un medio de salida o desactivación.

Ventana No Modal

Es una ventana no restringida de la que el usuario puede salir libremente. Las ventanas no modales:

- Se pueden mostrar muchas a la vez.
- No están necesariamente activas cuando se muestran.
- Son el tipo de ventana por defecto.

PROPIEDADES DE UNA VENTANA

Las propiedades básicas que puede definir para un objeto de tipo VENTANA son las siguientes:

- Título.
- Lienzo Primario.
- Lienzo de Barra de Herramientas Horizontal / Vertical.
- Estilo de Ventana.
- Modal.
- Ocultar al Salir.
- Cierre Permitido.
- Movimiento Permitido.
- Cambio de Tamaño Permitido.
- Maximización / Minimización Permitida.
- Título Minimizado.
- Nombre de Archivo de Icono.
- Posición X / Y.
- Mostar Barra de Desplazamiento Horizontal / Vertical.
- Ancho y Altura.

TÍTULO

Esta propiedad permite asignar un título a la ventana para que se muestre en el marco que recubre a la ventana.

LIENZO PRIMARIO

Esta propiedad asigna el nombre del lienzo principal que aparecerá en la ventana cuando se muestre la misma. Este nombre de lienzo deberá existir previamente en la lista de objetos de tipo lienzo.

LIENZO DE BARRA DE HERRAMIENTAS HORIZONTAL / VERTICAL

Esta propiedad asigna el nombre del lienzo de tipo barra de herramientas que se va a mostrar horizontalmente (sobre la parte superior) o verticalmente (a la izquierda de la ventana). Estos nombres de lienzo deberán existir previamente en la lista de objetos de tipo lienzo.

ESTILO DE VENTANA

Esta propiedad únicamente permite asignar dos tipos de valores:

- *Documento:* es el tipo por defecto para la mayoría de las ventanas. Las ventanas definidas de esta forma pueden ser minimizadas, maximizadas o reajustando su tamaño, sin que la visualización de la ventana salga del frame de la aplicación.
- *Diálogo:* es el tipo que se utiliza cuando se pretende definir una ventana con el mismo aspecto que una alerta o cuadro de diálogo. Las ventanas definidas de esta forma no se pueden minimizar, maximizar o reajustar su tamaño, y si se definen con la propiedad *Movimiento Permitido,* por lo que podrán moverse incluso fuera del frame de la aplicación.

MODAL

Esta propiedad especifica si la ventana es modal, lo que requiere que el usuario salga de la ventana antes de poder continuar con la interacción del usuario.

OCULTAR AL SALIR

Esta propiedad se asocia a las ventanas modales y permite cerrar automáticamente la ventana cuando el usuario navega a un campo que se encuentra fuera de la ventana a cerrar.

CIERRE PERMITIDO

Esta propiedad activa el mecanismo de cierre de la ventana (presenta el icono de Windows para cerrar una ventana). Además, activa el disparador WHEN-WINDOW-CLOSED.

MOVIMIENTO PERMITIDO

Esta propiedad determina si el usuario puede mover la ventana.

CAMBIO DE TAMAÑO PERMITIDO

Esta propiedad determina si el usuario puede reconfigurar el tamaño de la ventana en tiempo de ejecución.

MAXIMIZACIÓN Y MINIMIZACIÓN PERMITIDA

Estas propiedades activan los mecanismos de maximización y minimización de la ventana (presentan los iconos de Windows que realizan estas operaciones).

TÍTULO MINIMIZADO

Cuando se activa la propiedad *Minimización Permitida* y queremos que se muestre un título de la ventana cuando está minimizada, habrá que introducirlo en esta propiedad.

NOMBRE DE ARCHIVO DE ICONO

Cuando se activa la propiedad *Minimización Permitida* y queremos que se muestre un icono de la ventana cuando está minimizada, habrá que introducirlo el nombre de archivo de icono en esta propiedad. El nombre del archivo NO debe llevar la extensión.

POSICIÓN X, Y

Esta propiedad permite asociar las coordenadas X e Y donde se mostrará la ventana dentro del frame de la aplicación.

MOSTRAR BARRA DE DESPLAZAMIENTO HORIZONTAL / VERTICAL

Estas propiedades nos permiten mostrar las barras para desplazarse de forma horizontal o vertical, cuando la información contenida en la ventana, supera el tamaño del frame de la aplicación.

ANCHO Y ALTURA

Estas propiedades nos permiten asociar el tamaño (ancho y alto) de la ventana.

TIPOS DE LIENZO

Forms Builder proporciona distintos tipos de lienzos, siendo este (el de contenido) el lienzo base que ocupa todo el panel de contenido de la ventana que se muestra. Los tipos de lienzo permitidos son:

- Lienzo de Contenido.
- Lienzo Apilado.
- Lienzo de Barra de Herramienta Vertical.
- Lienzo de Barra de Herramienta Horizontal.
- Lienzo Separador.

El lienzo de Contenido

Forms Builder proporciona distintos tipos de lienzos, siendo este (el de contenido) el lienzo base que ocupa todo el panel de contenido de la ventana que se muestra. Es el tipo de lienzo por defecto y el que se utiliza en la mayoría de los casos.

Relación entre Ventanas y Lienzos de Contenido

Cuando se diseña una aplicación se debe crear al menos un lienzo de contenido para cada ventana de la misma y cuando se ejecuta la pantalla, solo se puede mostrar un lienzo de contenido en la misma ventana, a la vez, incluso aunque se pueda asignar más de un lienzo de contenido a la misma ventana durante el diseño.

En tiempo de ejecución, un lienzo de contenido siempre rellena completamente su ventana asociada. En cuanto el usuario cambia el tamaño de la ventana, Forms ajusta automáticamente el tamaño del lienzo a la misma. Si la ventana es demasiado pequeña para mostrar todos los elementos del lienzo, Forms desplaza automáticamente el lienzo para que sea visible el elemento actual.

El lienzo Apilado

El tipo de lienzo apilado se utiliza para mostrar información en un espacio de pantalla diferente y superpuesto a otro lienzo de contenido o apilado. El lienzo apilado comparte la misma ventana que aquel al que se superpone y normalmente es más pequeño que la ventana en la que se muestra para ofrecer el aspecto de superposición.

El uso y las ventajas que tienen los lienzos apilados se resumen en los siguientes puntos:

- Vistas desplazadas de la información.
- Creación de un efecto de superposición en una única ventana.
- Visualización de cabeceras de información constante como, por ejemplo, el nombre de la compañía.
- Creación de un efecto de aparición en cascada de una única ventana.
- Visualización de información adicional.
- Visualización de información de forma condicional.
- Visualización de ayuda sensible al contexto.
- Ocultación de información.

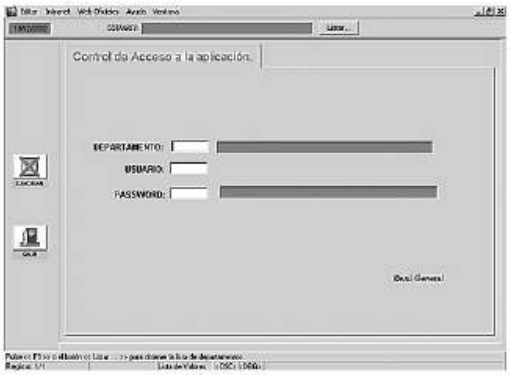
Los lienzos de barra de herramientas

Este tipo de lienzos únicamente se pueden crear para mostrarse en la parte superior del lienzo de contenido principal como barra de herramientas horizontal, o en la parte izquierda como barra de herramientas vertical.

Normalmente este tipo de lienzos se diseñan para colocar una botonadura de uso común para todos los lienzos del formulario, aunque también es posible introducir cualquier otro tipo de elementos.

El uso y las ventajas que tienen los lienzos de barra de herramientas se resumen en los siguientes puntos:

- Proporcionar un aspecto estándar en los lienzos mostrados en la misma ventana.
- Reducir el tiempo de mantenimiento del módulo de pantalla.
- Aumentar la facilidad de uso de la aplicación.
- Crear aplicaciones similares a otras utilizadas en el mismo centro de trabajo.
- Ofrecer una alternativa a las aplicaciones controladas por menús o teclas de función.



A continuación se muestra un ejemplo de un formulario con una barra de herramientas horizontal (con elementos mostrados como la fecha y el usuario) y otra vertical (con botones).

Los lienzos Separador

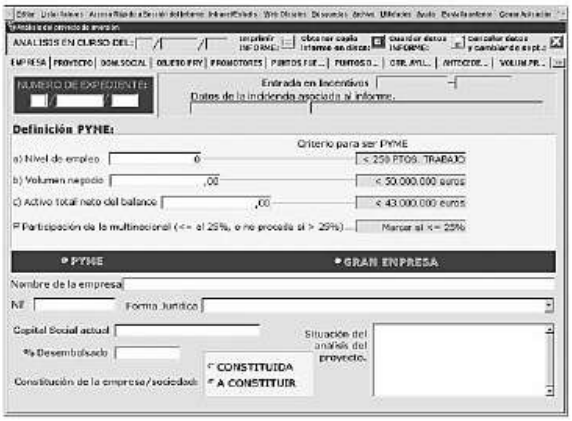
Este tipo de lienzos muestran un conjunto de lienzos de contenido con una pestaña identificativa para cada uno, lo cual permite organizar y separar la información a mostrar.

Siempre se muestran encima de un lienzo de contenido.

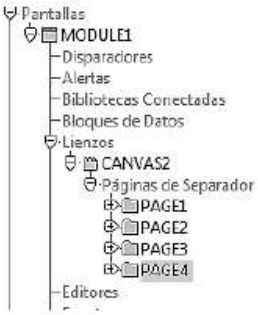
Todo lienzo separador dispone de 1 o más de una *página separadora* (solapa de aspecto visual). Cada página separadora es un subobjeto del lienzo que tiene su propio título. Toda página separadora ocupa la misma cantidad de espacio que se haya definido en el lienzo Separador.

El uso y las ventajas que tienen los lienzos de barra de herramientas se resumen en los siguientes puntos:

- Crear un efecto de superposición en una única ventana.
- Mostrar grandes cantidades de información en un único lienzo.
- Acceder fácilmente a la información necesaria haciendo clic en la solapa.



Aquí se muestra un ejemplo de un formulario con un lienzo separador con varias páginas (solapas). Y a continuación se muestra el aspecto que tendría en Forms Builder un lienzo separador con 4 páginas (solapas).



PROPIEDADES DE UN LIENZO

Las propiedades básicas que puede definir para un objeto de tipo LIENZO son las siguientes, aparte del *Tipo de Lienzo* que ya hemos visto anteriormente:

- Visible.
- Ventana.
- Posición X, Y de Viewport en Lienzo.
- Ancho y Altura.
- Bisel.
- Etiqueta (*solo para las páginas separadoras de un lienzo separador).*

VISIBLE

Esta propiedad define si se visualiza o no un lienzo dentro de un formulario. Aunque exista el lienzo en el formulario, si esta propiedad no está a valor *SÍ*, el mismo no se visualizará en el formulario.

VENTANA

Esta propiedad define en cuál de las ventanas definidas en el formulario se va a mostrar el lienzo. Por tanto, la ventana debe existir previamente.

POSICIÓN X, Y DE VIEWPORT EN LIENZO

Estas propiedades definen las posiciones X e Y en las que se va a mostrar el lienzo dentro de la ventana en relación con la posición 0,0 de la misma.

ANCHO Y ALTURA

Estas propiedades definen el tamaño del lienzo (ancho y alto).

BISEL

Esta propiedad define la característica de relieve que se le va a dar al marco que rodea al lienzo. Las opciones disponibles son:

- Resaltado.
- Hundido.
- Ninguno.
- Sin Relieve.
- Con Relieve.
- Normal.

ETIQUETA

Esta propiedad define el título a mostrar en la página separadora correspondiente. Por tanto esta propiedad únicamente se mostrará en cada página separadora de un lienzo separador.

SET_WINDOW_PROPERTY (CAMBIAR PROPIEDADES DE UNA VENTANA EN TIEMPO DE EJECUCIÓN)

Para modificar las propiedades de una ventana en tiempo de ejecución disponemos de la built-it **SET_WINDOW_PROPERTY**.

Sintaxis:

```
SET_WINDOW_PROPERTY( nombre_ventana, propiedad, valor )
```

Las propiedades que se pueden modificar en tiempo de ejecución con la instrucción SET_WINDOW_PROPERTY son las siguientes:

- Background_Color.
- Direction.
- Fill_Pattern.
- Font_Name.
- Font_Size.
- Font_Spacing.
- Font_Style.
- Font _Weight.
- Foreground_Color.
- Height.
- Hide_On_Exit.
- Icon_Name.
- Position.
- Title.
- Visible.
- Window_Size.
- Window_State.
- Width.
- X_Pos.
- Y_Pos.

BACKGROUND_COLOR

Esta propiedad permite asociar el color de fondo para la ventana.

DIRECTION

Esta propiedad solo se utiliza con aplicaciones que soportan lenguaje bidireccional (NLS) y permite cambiar la dirección de los objetos bidireccionales de acuerdo a los siguientes valores:

- DIRECTION_DEFAULT.
- RIGHT_TO_LEFT.
- LEFT_TO_RIGHT.

FILL_PATTERN

Esta propiedad permite asociar un patrón de relleno a la ventana.

FONT_NAME

Esta propiedad permite asociar el nombre de la fuente que se utilizará en la ventana. Este nombre de fuente debe existir en el sistema.

FONT_SIZE

Esta propiedad permite asociar el tamaño de la fuente (en puntos) que se utilizará en la ventana. Este nombre de fuente debe existir en el sistema.

FONT_SPACING

Esta propiedad permite asociar el espacio de la fuente que se utilizará en la ventana. Los valores admitidos son:

- FONT_NORMAL.
- FONT_ULTRADENSE.
- FONT_EXTRADENSE.
- FONT_DENSE.
- FONT_SEMIDENSE.
- FONT_SEMIEXPAND.
- FONT_EXPAND.
- FONT_EXTRAEXPAND.
- FONT_ULTRAEXPAND.

FONT_STYLE

Esta propiedad permite asociar el estilo de la fuente que se utilizará en la ventana. Los valores admitidos son:

- FONT_PLAIN.
- FONT_ITALIC.
- FONT_OBLIQUE.
- FONT_UNDERLINE.
- FONT_OUTLINE.
- FONT_SHADOW.
- FONT_INVERTED.
- FONT_OVERSTRIKE
- FONT_BLINK.

FONT_WEIGHT

Esta propiedad permite asociar la densidad de la fuente que se utilizará en la ventana. Los valores admitidos son:

- FONT_MEDIUM.
- FONT_ULTRALIGHT.
- FONT_EXTRALIGHT.
- FONT_LIGHT.
- FONT_DEMILIGHT.
- FONT_DEMIBOLD.
- FONT_BOLD.
- FONT_EXTRABOLD.
- FONT_ULTRABOLD.

FOREGROUND_COLOR

Esta propiedad permite asociar el color de fuente del primer plano de una ventana.

HEIGHT

Esta propiedad permite asociar la altura de una ventana.

HIDE_ON_EXIT

Esta propiedad permite especificar si la ventana se oculta automáticamente al navegar a un objeto fuera de la misma.

ICON_NAME

Esta propiedad permite asociar el nombre del icono que se va a utilizar cuando se minimice la pantalla.

POSITION

Esta propiedad permite asociar las posiciones X e Y, donde se va a abrir la ventana dentro del formulario.

TITLE

Esta propiedad permite asociar el texto que aparece como título en la ventana.

VISIBLE

Esta propiedad permite indicar si la ventana es o no visible.

WINDOW_SIZE

Esta propiedad permite asociar el tamaño de la ventana, indicado el ancho y alto de la misma.

WINDOW_STATE

Esta propiedad permite asociar el estado con el que se abrirá la ventana. Los valores permitidos son:

- NORMAL
- MAXIMIZE
- MINIMIZE

WIDTH

Esta propiedad permite asociar el ancho de una ventana.

X_POS

Esta propiedad permite asociar la posición X en la que se mostrará la ventana dentro del formulario.

Y_POS

Esta propiedad permite asociar la posición Y en la que se mostrará la ventana dentro del formulario.

Ejemplo

En este ejemplo se muestra cómo se puede cambiar el título de la ventana MDI de la aplicación y maximizar a la entrada de la misma. Para ello se utilizaría el disparador de Forms Builder WHEN-NEW-FORM-INSTANCE y el siguiente código:

```
SET_WINDOW_PROPERTY( FORMS_MDI_WINDOW, TITLE, 'Bienvenido' );
SET_WINDOW_PROPERTY( FORMS_MDI_WINDOW, WINDOW_STATE, MAXIMIZE);
```

SET_CANVAS_PROPERTY (CAMBIAR PROPIEDADES DE UN LIENZO EN TIEMPO DE EJECUCIÓN)

Para modificar las propiedades de un lienzo en tiempo de ejecución disponemos de la built-in **SET_CANVAS_PROPERTY**.

Sintaxis:

```
SET_CANVAS_PROPERTY( nombre_lienzo, propiedad, valor)
```

Las propiedades que se pueden modificar en tiempo de ejecución con la instrucción SET_WINDOW_PROPERTY son las siguientes:

- Background_Color.
- Canvas_Size.
- Fill_Pattern.
- Font_Name.
- Font_Size.
- Font_Spacing.
- Font_Style.
- Font _Weight.
- Foreground_Color.
- Height.
- Topmost_Tab_Page.
- Visual_Attribute.
- Width.

BACKGROUND_COLOR

Esta propiedad permite asociar el color de fondo para el lienzo.

CANVAS_SIZE

Esta propiedad permite asociar el tamaño del lienzo, indicado el ancho y alto del mismo.

FILL_PATTERN

Esta propiedad permite asociar un patrón de relleno del lienzo.

FONT_NAME

Esta propiedad permite asociar el nombre de la fuente que se utilizará en el lienzo. Este nombre de fuente debe existir en el sistema.

FONT_SIZE

Esta propiedad permite asociar el tamaño de la fuente (en puntos) que se utilizará en el lienzo. Este nombre de fuente debe existir en el sistema.

FONT_SPACING

Esta propiedad permite asociar el espacio de la fuente que se utilizará en el lienzo. Los valores admitidos son los mismos que se han indicado en la propiedad FONT_SPACING de SET_WINDOW_PROPERTY.

FONT_STYLE

Esta propiedad permite asociar el estilo de la fuente que se utilizará en el lienzo. Los valores admitidos son los mismos que se han indicado en la propiedad FONT_STYLE de SET_WINDOW_PROPERTY.

FONT_WEIGHT

Esta propiedad permite asociar la densidad de la fuente que se utilizará en el lienzo. Los valores admitidos son los mismos que se han indicado en la propiedad FONT_WEIGHT de SET_WINDOW_PROPERTY.

FOREGROUND_COLOR

Esta propiedad permite asociar el color de fuente del primer plano de un lienzo.

HEIGHT

Esta propiedad permite asociar la altura de un lienzo.

TOPMOST_TAB_PAGE

Esta propiedad permite especificar el texto del título que aparecerá en la solapa de una página separadora.

VISUAL_ATTRIBUTE

Esta propiedad permite asociar el nombre de los atributos visuales que se asocian al lienzo. Este nombre debe de existir previamente en el formulario (dentro del objeto *Visual Attributes*).

WIDTH

Esta propiedad permite asociar el ancho de un lienzo.

SHOW_WINDOW / HIDE_WINDOW

Estas dos built-in permiten mostrar y ocultar una ventana.

La built-in **SHOW_WINDOW** se utiliza para mostrar una ventana en tiempo de ejecución. Tiene la siguiente sintáxis:

```
SHOW_WINDOW( nombre_ventana [, x, y])
```

Nos permite mostrar la ventana en la posición que se haya definido para la misma en el diseño, o bien en una posición nueva (asociando los valores X e Y en la propia built-in.

La built-in **HIDE_WINDOW** se utiliza para ocultar una ventana abierta en tiempo de ejecución. Tiene la siguiente sintáxis:

```
HIDE_WINDOW( nombre_ventana)
```

CREANDO MÚLTIPLES VENTANAS 28

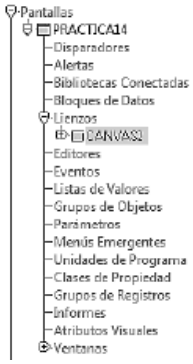
INTRODUCCIÓN

Durante esta práctica aprenderemos a usar y manejar múltiples ventanas mediante el diseño de Forms Builder, y el código programado con las built-in vistas en el capítulo anterior.

Apertura de un nuevo formulario y almacenamiento

Como primer paso abriremos Forms Builder con un nuevo formulario. Seguidamente cambiamos la propiedad *Nombre* del formulario para ponerle **PRACTICA14** y a continuación desde el menú **Archivo** seleccionamos **Guardar Como** e indicamos **PRACTICA14.FMB**.

Creando un nuevo lienzo



Comenzamos el diseño del formulario creando un nuevo lienzo, para ello nos situamos en el grupo **Lienzos** y pulsamos el botón añadir para crear uno nuevo.

Cambiando las propiedades de un lienzo

Una vez creado el nuevo lienzo abrimos la paleta de propiedades (F4) y le asociamos las siguientes propiedades:

- Propiedad **Nombre:** debe definirse a *LIENZO1*.
- Propiedad **Color de Fondo**: debe definirse a *r100g88b88*.

Creando un bloque de datos manual

Seguidamente cerramos la paleta de propiedades y volvemos al navegador de objetos de Forms Builder para situarnos en el grupo **Bloques de Datos**, donde pulsaremos el botón añadir para crear un nuevo bloque que será de tipo manual y que llamaremos con el nombre *CONTROL1*.

Añadiendo un botón al bloque manual

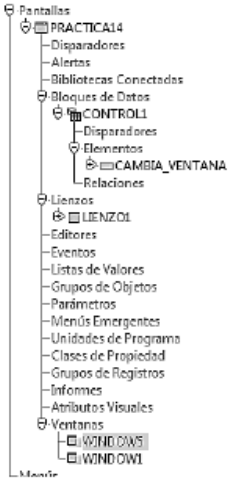
A continuación nos situamos en el grupo **Elementos** del bloque *CONTROL1* y pulsamos añadir para crear un nuevo elemento.

Cambiando las propiedades de un elemento del bloque

Una vez creado el nuevo elemento en el bloque abrimos la paleta de propiedades (F4) y le asociamos las siguientes propiedades:

- Propiedad **Nombre:** debe definirse a *CAMBIA_VENTANA*.
- Propiedad **Tipo de elemento**: debe definirse a *Botón*.
- Propiedad **Etiqueta:** debe definirse a "*Pulse para ir a la próxima ventana*".
- Propiedad **Lienzo**: debe definirse a *LIENZO1.*
- Propiedad **Posición X**: debe definirse a *137*.
- Propiedad **Posición Y**: debe definirse a *121*.
- Propiedad **Ancho**: debe definirse a *234.*
- Propiedad **Alto**: debe definirse a *28.*

Creando una nueva ventana



Cerramos la paleta de propiedades y volvemos al navegador de objetos para situarnos en el grupo de elementos **Ventanas**, donde pulsaremos el botón de añadir para crear una nueva ventana (aparte de la que ya existe).

Cambiando las propiedades de una ventana

Una vez creada la nueva ventana, abrimos la paleta de propiedades (F4) y le asociamos las siguientes propiedades:

- Propiedad **Nombre:** debe definirse a *VENTANA_MODAL*.
- Propiedad **Título:** debe definirse a *"¿Y ahora como salgo de aquí?"*.
- Propiedad **Estilo de Ventana:** debe definirse a *Diálogo*.
- Propiedad **Modal:** debe definirse a *SÍ.*
- Propiedad **Ocultar al Salir:** debe definirse a *SÍ*.
- Propiedad **Cierre Permitido:** debe definirse a *NO*.
- Propiedad **Movimiento Permitido:** debe definirse a *NO*.
- Propiedad **Cambio de Tamaño Permitido:** debe definirse a *NO*.
- Propiedad **Maximización Permitida:** debe definirse a *NO*.
- Propiedad **Minimización Permitida:** debe definirse a *NO*.
- Propiedad **Posición X:** debe definirse a *100*.
- Propiedad **Posición Y:** debe definirse a *100*.
- Propiedad **Ancho:** debe definirse a *200*.
- Propiedad **Alto:** debe definirse a *200*.

Creando más lienzos

Cerramos la paleta de propiedades y volvemos al navegador de objetos donde nos situamos sobre el lienzo *LIENZO1* y pulsamos el botón añadir para crear un nuevo lienzo, al que le modificaremos las siguientes propiedades:

- Propiedad **Nombre:** debe definirse a *LIENZO2*.
- Propiedad **Ancho:** debe definirse a *199*.
- Propiedad **Alto:** debe definirse a *199*.
- Propiedad **Ventana:** debe definirse a *VENTANA_MODAL*.

Seguidamente volvemos a cerrar la paleta de propiedades y al volver al navegador de objetos, volvemos a pulsar el botón añadir para crear otro lienzo más al que asociaremos estas propiedades:

- Propiedad **Nombre:** debe definirse a *BARRA_V*.
- Propiedad **Tipo de Lienzo:** debe definirse a *Barra de Herramientas Vertical*.
- Propiedad **Ventana:** debe definirse a *WINDOW1*.
- Propiedad **Alto:** debe definirse a *70*.
- Propiedad **Bisel:** debe definirse a *Resaltado*.

Creando más botones

Cerramos la paleta de propiedades y volvemos al navegador de objetos donde nos situamos sobre el elemento *CAMBIA_VENTANA* del bloque *CONTROL1*. Allí pulsaremos añadir para crear un nuevo elemento que será de tipo botón con las siguientes propiedades:

- Propiedad **Nombre:** debe definirse a *SALIR*.
- Propiedad **Tipo de elemento**: debe definirse a *Botón*.
- Propiedad **Icónico:** debe definirse a *SÍ*.
- Propiedad **Nombre de Archivo de Icono**: debe definirse a *salir.*
- Propiedad **Lienzo**: debe definirse a *BARRA_V*.
- Propiedad **Posición X**: debe definirse a *6*.
- Propiedad **Posición Y**: debe definirse a *6.*
- Propiedad **Ancho**: debe definirse a *60.*
- Propiedad **Alto**: debe definirse a *60.*
- Propiedad **Color de Fondo**: debe definirse a *white.*

Creando un disparador WHEN-NEW-FORM-INSTANCE

Cerramos la paleta de propiedades y volvemos al navegador de objetos donde nos situamos sobre el grupo de elementos **Disparadores** que cuelga bajo el nombre del formulario y pulsamos el botón añadir para crear un nuevo disparador del tipo WHEN-NEW-FORM-INSTANTE, al que asociaremos el siguiente código PL/SQL:

```
SET_WINDOW_PROPERTY(FORMS_MDI_WINDOW, WINDOW_STATE, MAXIMIZE);
SET_WINDOW_PROPERTY(FORMS_MDI_WINDOW, TITLE, 'Curso Forms 11g');
SET_WINDOW_PROPERTY('WINDOW1', WINDOW_STATE, MAXIMIZE);
SET_WINDOW_PROPERTY('WINDOW1', TITLE, 'Práctica 14 del Curso');
```

A continuación compilamos el código y cerramos la ventana de programación, lo que nos devolverá al navegador de objetos y tendremos una imagen de los elementos que hay en el mismo.

Creando un disparador a nivel de botón

A continuación nos situamos en el elemento *CAMBIA_VENTANA* del bloque *CONTROL1* y desplegamos su contenido, situándonos en el grupo **Disparadores**, donde pulsaremos el botón añadir para crear un disparador WHEN-BUTTON-PRESSED con el código PL/SQL que se indica a continuación:

```
GO_ITEM('CONTROL1.VOLVER');
SHOW_VIEW('LIENZO2');
```

Creando más botones

Compilamos el código del disparador y cerramos la ventana para volver al navegador de objetos, donde nos situaremos en el botón *SALIR* y pulsaremos añadir para crear un nuevo elemento con las siguientes propiedades:

- Propiedad **Nombre:** debe definirse a *VOLVER*.
- Propiedad **Tipo de elemento**: debe definirse a *Botón*.
- Propiedad **Etiqueta:** debe definirse a "*Volver a la pantalla anterior*"
- Propiedad **Lienzo:** debe definirse a *LIENZO2.*
- Propiedad **Posición X:** debe definirse a *22*.
- Propiedad **Posición Y:** debe definirse a *73*.
- Propiedad **Ancho** debe definirse a *128*.
- Propiedad **Alto:** debe definirse a *19*.
- Propiedad **Color de Fondo:** debe definirse a *Gray.*

Seguidamente nos situamos en el nuevo botón *VOLVER* y desplegamos su contenido, para crear un nuevo disparador asociado al mismo del tipo WHEN-BUTTON-PRESSED, que llevará el siguiente código PL/SQL:

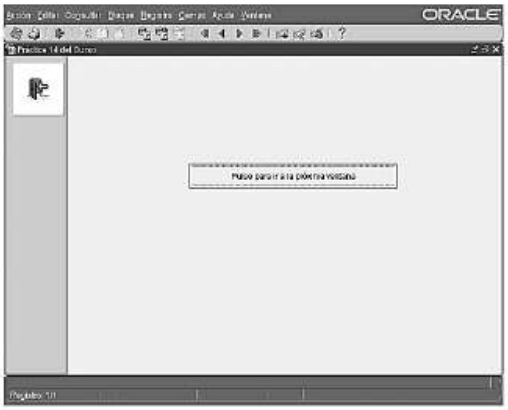
```
GO_ITEM('CONTROL1.CAMBIA_VENTANA');
HIDE_VIEW('LIENZO2');
HIDE_WINDOW('VENTANA_MODAL');
```

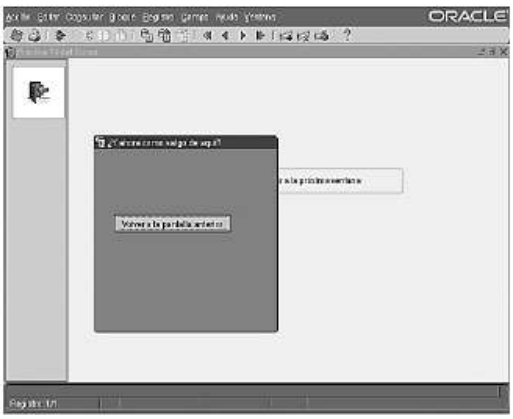
Creando más disparadores a nivel de botón

A continuación compilamos el código de los disparados y cerramos la ventana para volver al navegador de objetos, donde nos situaremos en el botón *SALIR* y desplegamos su contenido, para crear un nuevo disparador asociado al mismo del tipo WHEN-BUTTON-PRESSED, que llevará el siguiente código PL/SQL:

```
EXIT_FORM;
```

Guardando, compilando y ejecutando el formulario





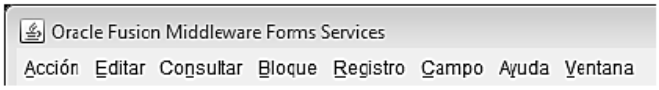
Concluida la compilación del código anterior cerramos el editor PL/SQL del disparador y una vez en el navegador de objetos, almacenamos el formulario (**CTRL + S),** lo compilamos **(SHIFT + CTRL + K)** y generamos **(CTRL + T)**, para en último lugar ejecutarlo **(CTRL + R).** Una vez ejecutada la aplicación se nos mostrará una primera pantalla con este aspecto. Pulsaremos sobre el botón **Pulse para ir a la próxima ventana** que nos mostrará superpuesta otra ventana. Si intentamos pulsar fuera de la ventana superpuesta (por ejemplo al botón del icono salir o al botón que está debajo de la ventana), veremos que el formulario no hace nada, al ser la ventana superpuesta una ventana de diálogo. La única forma que tenemos para abandonar la misma el pulsar el botón **Volver a la pantalla anterior.** Una vez hecho esto, si queremos cerrar la aplicación podemos pulsar el botón del icono salir.

LAS BUILT-IN DO_KEY 29

INTRODUCCIÓN

El código ya programado por Oracle en Forms que se puede invocar en cualquier formulario se denomina BUILT-IN.

En este capítulo se van a tratar las BUILT-IN del tipo DO_KEY. Este código programado se utiliza para ejecutar las mismas acciones que se derivan de pulsar los diferentes elementos del menú contextual de la barra de herramientas preestablecida de un formulario.



Equivalencia menú Acción con Built-in DO_KEY

El menú **Acción** de un formulario tiene este aspecto.

La equivalencia entre sus elementos y las Built-in DO_KEY es la que se especifica en el siguiente cuadro:

Elemento del Menú	Built-in DO_KEY
Guardar	DO_KEY(COMMI T_FORM)
Borrar_Todo	DO_KEY(CLEAR_FORM)
Imprimir	DO_KEY(PRI NT)
Salir	DO_KEY(EXI T_FORM)

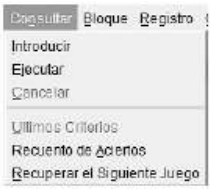
Equivalencia menú Editar con Built-in DO_KEY

El menú **Editar** de un formulario tiene este aspecto.

La equivalencia entre sus elementos y las Built-in DO_KEY es la que se especifica en el siguiente cuadro:

Elemento del Menú	Built-in DO_KEY
Cortar	Sin equivalencia
Copiar	Sin equivalencia
Pegar	Sin equivalencia
Editar	DO_KEY(EDIT_TEXTITEM)
Mostar Lista	DO_KEY(LIST_VALUES)

Equivalencia menú Consultar con Built-in DO_KEY

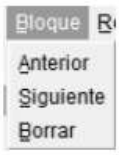


El menú **Consultar** de un formulario tiene este aspecto.

La equivalencia entre sus elementos y las Built-in DO_KEY es la que se especifica en el siguiente cuadro:

Elemento del Menú	Built-in DO_KEY
Introducir	DO_KEY(ENTER_QUERY)
Ejecutar	DO_KEY(EXECUTE_QUERY)
Cancelar	DO_KEY(EXIT_FORM)
Últimos Criterios	Sin equivalencia
Recuento de Aciertos	DO_KEY(COUNT_QUERY)
Recuperar el Siguiente Juego	DO_KEY(NEXT_SET)

Equivalencia menú Bloque con Built-in DO_KEY

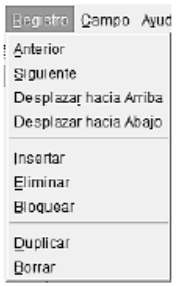


El menú **Bloque** de un formulario tiene este aspecto.

La equivalencia entre sus elementos y las Built-in DO_KEY es la que se especifica en el siguiente cuadro:

Elemento del Menú	Built-in DO_KEY
Anterior	DO_KEY(PREVIOUS_BLOCK)
Siguiente	DO_KEY(NEXT_BLOCK)
Borrar	DO_KEY(CLEAR_BLOCK)

Equivalencia menú Registro con Built-in DO_KEY



El menú **Bloque** de un formulario tiene este aspecto.

La equivalencia entre sus elementos y las Built-in DO_KEY es la que se especifica en el siguiente cuadro:

Elemento del Menú	Built-in DO_KEY
Anterior	DO_KEY(PREVI OUS_RECORD)
Siguiente	DO_KEY(NEXT_RECORD)
Desplazar hacia Arriba	DO_KEY(UP)
Desplazar hacia Abajo	DO_KEY(DOWN)
Insertar	DO_KEY(CREATE_RECORD)
Eliminar	DO_KEY(DELETE_RECORD)
Bloquear	DO_KEY(LOCK_RECORD)
Duplicar	DO_KEY(DUPLI CATE_RECORD)
Borrar	DO_KEY(CLEAR_RECORD)

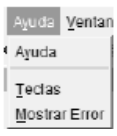
Equivalencia menú Campo con Built-in DO_KEY

El menú **Campo** de un formulario tiene este aspecto.

La equivalencia entre sus elementos y las Built-in DO_KEY es la que se especifica en el siguiente cuadro:

Elemento del Menú	Built-in DO_KEY
Anterior	DO_KEY(PREVI OUS_I TEM)
Siguiente	DO_KEY(NEXT_I TEM)
Borrar	Si n equi val enci a
Duplicar	DO_KEY(DUPLI CATE_I TEM)

Equivalencia menú Ayuda con Built-in DO_KEY



El menú **Ayuda** de un formulario tiene este aspecto.

La equivalencia entre sus elementos y las Built-in DO_KEY es la que se especifica en el siguiente cuadro:

Elemento del Menú	Built-in DO_KEY
Ayuda	DO_KEY(HELP)
Teclas	MENU_SHOW_KEYS
Mostrar Error	DI SPLAY_ERROR;

SUPUESTO PRÁCTICO 4

MÓDULO BIENVENIDO

Abrimos el módulo **BIENVENIDO.FMB** que se proporciona junto con este manual y realizamos las siguientes modificaciones en el mismo:

- Eliminar del módulo el botón **CONTINUAR.**
- Eliminar la barra de estado.
- Eliminar el menú por defecto y la barra de menú icónica por defecto.
- Sustituir la ruta j ava/ i magenes que aparece en el disparador WHEN-NEW-FORM-INSTANTE, por aquella en la que se haya instalado la carpeta i magenes que se proporciona con el curso.
- Sustituir el título de la ventana MDI de Forms por "Developer 11G. Curso Práctico de Formación".

En la página siguiente se muestra una imagen que ilustra cómo debe de quedar el formulario después de implementar los cambios indicados en este supuesto.

PARA TODOS LOS MÓDULOS

Será necesario realizar las siguientes modificaciones en el resto de módulos que se han desarrollado hasta ahora, es decir: ENFERMOS_HOSPITAL, HOSPITAL_SALA, PLANTILLA_NO_SANITARIA, PLANTILLA_SANITARIA:

- Introducir un trigger de tipo **WHEN-NEW-FORM-INSTANCE** en el que se incluyan las siguientes sentencias sustituyéndose `nombre_ventana_modulo` por el nombre de la ventana que está en cada módulo.

```
SET_WINDOW_PROPERTY(FORMS_MDI_WINDOW, WINDOW_STATE, MAXIMIZE);
SET_WINDOW_PROPERTY(FORMS_MDI_WINDOW, TITLE,'Curso Forms 11g');
SET_WINDOW_PROPERTY(nombre_ventana_modulo, WINDOW_STATE, MAXIMIZE);
```

- Indicar un título para cada ventana que exista en cada módulo desde sus propias propiedades.
- Impedir que la ventana del módulo se puede mover, minimizar o cerrar.
- Incluir una barra de herramienta vertical denominada **L_VERTICAL** que incluya la siguiente botonadura que se asociará a un nuevo bloque denominado **CONTROL_VERTICAL:**

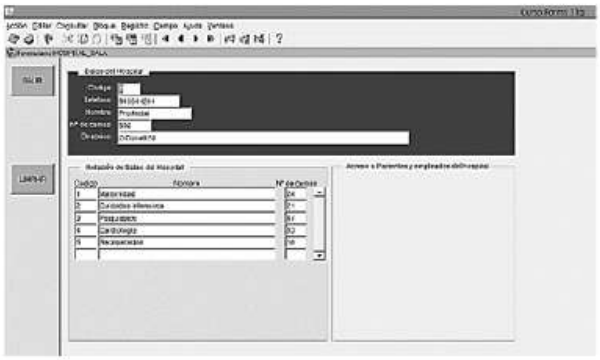
Al pulsar este botón saldremos del formulario.

Al pulsar este botón limpiaremos todos los elementos del formulario.

Además, la barra de herramientas vertical debe llevar un bisel **Resaltado** y un color de fondo **r88g100b75.**

- Crear la funcionalidad `DO_KEY` apropiada para los dos botones.
- Eliminar de los bloques ENFERMO, HOSPITAL, EMPLEADO_HOSPITAL, EMPLEADO_HOSPITAL, DOCTOR_HOSPITAL y HOSPITAL_ENFERMO de cualquier formulario, la posibilidad de insertar nueva información en cualquiera de sus campos y tampoco borrar o actualizar la que ya exista.

A continuación se ilustra cómo debe de quedar el resto de formularios después de implementar los cambios indicados en este supuesto.



LOS MENÚS DE USUARIO 30

INTRODUCCIÓN

Hasta el momento hemos visto que un formulario puede disponer de los menús preestablecidos. Entre ellos el menú contextual por defecto es el siguiente:

Pero también es posible crear menús personalizados para las distintas necesidades del usuario.

Los menús de Forms también se diseñan dentro del Forms Builder, pero se almacenan con una extensión diferente a los formularios:

- .MMB Identifica el código fuente de un menú de usuario.
- .MMX Identifica el código compilado de un menú de usuario.

Un menú de usuario se divide en los siguientes elementos:

- **Menú principal**: es aquel que aparece en la barra horizontal.
- **Menú individual**: es aquel que se abre desde un menú principal (se despliega de manera vertical).
- **Submenú**: es el que se abre desde un menú individual (se despliega verticalmente a la derecha del menú individual).

Para facilitar la creación de los menús de usuario, Forms Builder tiene una herramienta de diseño de menús denominada **Editor de Menús.**

Propiedades de un menú

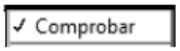
Las propiedades básicas que definen un elemento de un menú de usuario son las siguientes:

- Tipo de opción de menú.
- Tipo de comando.
- Activado.
- Etiqueta.
- Elemento mágico.
- Grupo de botones de Radio de opción de menú.
- Código de Opción de Menú.
- Acelerador de Teclado.
- Visible en Menú.
- Icono en Menú.
- Nombre de Archivo de Icono.

TIPO DE OPCIÓN DE MENÚ

Esta propiedad permite indicar el tipo de elemento que se visualizará dentro del ítem de menú. Las opciones permitidas son:

- **Normal:** es la opción por defecto de un ítem de menú. Se representa con el nombre del ítem de menú. 

- **Comprobar**: se representa con un elemento gráfico de tipo check a la izquierda del ítem. 

- **Radio**: se representa con un elemento gráfico de tipo radio button a la izquierda del ítem. 

- **Separador**: este tipo de elementos no se representan con una etiqueta sino con una línea. Sirven para separar un ítem de menú de otro. 

- **Mágico**: este tipo de elementos asumen la funcionalidad estándar de otros elementos preestablecidos de Windows como cortar, copiar, etc.

TIPO DE COMANDO

Nos permite identificar cómo se debe comportar el ítem de menú, permitiéndose las siguientes opciones:

- **Menú**: indica que el ítem de menú es un menú individual del que podrán derivarse otros ítems de menú.
- **PL/SQL**: indica que el ítem de menú lleva un código programado para realizar la operación deseada.
- **Nulo**: indica que el ítem de menú no lleva ni código programa, ni es un menú individual. Se utiliza por ejemplo en los tipos de opción de menú *separador* y *mágico*.

ETIQUETA

Nos permite representar con un texto el ítem de menú.

ELEMENTO MÁGICO

Para indicar un elemento mágico asociado al ítem de menú. Los elementos mágicos existentes son:

- Cortar.
- Copiar.
- Pegar.
- Borrar.
- Deshacer.
- Ayuda.
- Acerca de.
- Salir.
- Ventana.

GRUPO DE BOTONES DEL RADIO DE OPCIÓN DE MENÚ

Especifica el nombre del grupo de botones de radio al que pertenece el ítem de menú, de forma que solo se permita seleccionar uno de ellos.

Todos los elementos de ítem de menú pertenecientes al mismo grupo deben aparecer en el mismo menú individual.

Solo se permite un grupo de botones de radio por menú individual.

CÓDIGO DE OPCIÓN DE MENÚ

Solo se incluye cuando el ítem de menú tiene como tipo de comando la propiedad PL/SQL. En ese caso, en esta propiedad se indicará la programación que queremos que se ejecute para llevar a cabo la acción que determine el botón.

ACELERADOR DE TECLADO

Especifica una tecla de función lógica que se asocia al ítem de menú para ejecutarlo más rápido al pulsarla. Los nombres lógicos de estos aceleradores son ACCELERATOR1 a ACCELERATOR5.

Las teclas de función que se definen para cada ACCELERATOR se crean en un fichero de recursos de ejecución (FRMWEB.RES) y tienen el contenido siguiente:

```
#  FMRWEB.RES is the key definition file for webforms.  The syntax is:
#
#    JFN : JMN : URKS : FFN : URFD   (whitespace ignored)
#
#      JFN = Java function number
#      JMN = Java modifiers number
#     URKS = User-readable key sequence (double-quoted)
#      FFN = Forms function number
#     URFD = User-readable function description (double-quoted)
#

#  JAVA FUNCTION NUMBER
#         33 = PageUp
#         34 = PageDown
#         35 = End
#         36 = Home
#         37 = LeftArrow
#         38 = UpArrow
#         39 = RightArrow
#         40 = DownArrow
#    65 - 90 = Ctrl+A thru Ctrl+Z (These will always have the control
#              modifier explicitly included, as well as any other
#              modifiers that might be used.)
#  112 - 123 = F1 thru F12
#          9 = Tab (Ctrl+I, without the control modifier)
#         10 = Return (Ctrl+J, without the control modifier)
#
#  JAVA MODIFIERS NUMBER
#  Equal to the sum of the values for the modifier keys:
#    0 = None
#    1 = Shift
#    2 = Control
#    4 = Meta
#    8 = Alt
#
```

```
#  FORMS FUNCTION NUMBER
#  The Forms function numbers match the function numbers found in a
#  typical  Forms key binding file.
#
#  USER-READABLE STRINGS
#  The double-quoted strings appear when users click [Show Keys], and
#  are used for this purpose only. These strings can be translated as
#  needed. Note that the strings do not affect what actually happens
#  when end users press a particular key sequence.
#
9    : 0 : "Tab"            : 1  : "Next Field"
9    : 1 : "Shift+Tab"      : 2  : "Previous Field"
116  : 0 : "F5"             : 3  : "Clear Field"
38   : 0 : "Up"             : 6  : "Up"
40   : 0 : "Down"           : 7  : "Down"
33   : 0 : "PageUp"         : 12 : "Scroll Up"
34   : 0 : "PageDown"       : 13 : "Scroll Down"
69   : 2 : "Ctrl+E"         : 22 : "Edit"
10   : 0 : "Return"         : 27 : "Return"
76   : 2 : "Ctrl+L"         : 29 : "List of Values"
115  : 0 : "F4"             : 32 : "Exit"
75   : 2 : "Ctrl+K"         : 35 : "Show Keys"
83   : 2 : "Ctrl+S"         : 36 : "Commit"
118  : 1 : "Shift+F7"       : 61 : "Next Primary Key"
117  : 0 : "F6"             : 62 : "Clear Record"
38   : 2 : "Ctrl+Up"        : 63 : "Delete Record"
117  : 1 : "Shift+F6"       : 64 : "Duplicate Record"
40   : 2 : "Ctrl+Down"      : 65 : "Insert Record"
119  : 1 : "Shift+F8"       : 66 : "Next Set of Records"
1005 : 0 : "Down"           : 67 : "Next Record"
1004 : 0 : "Up"             : 68 : "Previous Record"
118  : 0 : "F7"             : 69 : "Clear Block"
66   : 2 : "Ctrl+B"         : 70 : "Block Menu"
34   : 1 : "Shift+PageDown" : 71 : "Next Block"
33   : 1 : "Shift+PageUp"   : 72 : "Previous Block"
116  : 1 : "Shift+F5"       : 73 : "Duplicate Field"
119  : 0 : "F8"             : 74 : "Clear Form"
122  : 0 : "F11"            : 76 : "Enter Query"
122  : 2 : "Ctrl+F11"       : 77 : "Execute Query"
69   : 3 : "Shift+Ctrl+E"   : 78 : "Display Error"
80   : 2 : "Ctrl+P"         : 79 : "Print"
123  : 0 : "F12"            : 80 : "Count Query"
85   : 2 : "Ctrl+U"         : 81 : "Update Record"

121  : 3 : "Shift+Ctrl+F10" : 82 : "Function 0"
112  : 3 : "Shift+Ctrl+F1"  : 83 : "Function 1"
113  : 3 : "Shift+Ctrl+F2"  : 84 : "Function 2"
114  : 3 : "Shift+Ctrl+F3"  : 85 : "Function 3"
115  : 3 : "Shift+Ctrl+F4"  : 86 : "Function 4"
116  : 3 : "Shift+Ctrl+F5"  : 87 : "Function 5"
117  : 3 : "Shift+Ctrl+F6"  : 88 : "Function 6"
118  : 3 : "Shift+Ctrl+F7"  : 89 : "Function 7"
119  : 3 : "Shift+Ctrl+F8"  : 90 : "Function 8"
```

```
120  : 3 : "Shift+Ctrl+F9"  : 91 : "Function 9"
113  : 0 : "F2"             : 95 : "List Tab Pages"
72   : 2 : "Ctrl+H"         : 30 : "Help"
```

Por ejemplo si queremos crear un ítem de menú, que tenga en la propiedad acelerador como ACCELERATOR1, y que se componga con la combinación de teclas "Ctrl+F2", tendremos que añadir a la sección USER-READABLE STRINGS del fichero **frmweb.res** la siguiente línea:

```
113 : 2 : "Ctrl+F2" : 11022 : "ACCELERATOR1"
```

Esta línea se compone de los siguientes elementos:

- El código 113 identifica al JFN (Java function number).
- El código 2 identifica al JMN (Java modifiers number).
- Entre comillas se indica la combinación de teclas "Ctrl+F2".
- El código 11022 identifica al FFN (Forms function number).
- Por último entre comillas se indica el nombre lógico del acelerador para Forms.

Los códigos FFN (Forms Function Number) que identifican a cada acelerador son:

- 11022 Accelerator Key 1
- 11023 Accelerator Key 2
- 11024 Accelerator Key 3
- 11025 Accelerator Key 4
- 11026 Accelerator Key 5
- 11027 Accelerator Key 6
- 11028 Accelerator Key 7
- 11029 Accelerator Key 8
- 11030 Accelerator Key 9
- 11031 Accelerator Key 10

VISIBLE EN MENÚ

Para especificar si el ítem de menú se muestra o no en el formulario (dentro del conjunto de elementos del menú).

ICONO EN MENÚ

Para especificar si se debería visualizar o no un icono al lado del ítem de menú.

NOMBRE DE ARCHIVO DE ICONO

Para especificar el nombre del archivo donde se encuentra el icono que se quiere visualizar al lado del ítem de menú.

ACTIVADO

Para mostrar el ítem de menú como habilitado/en negro (permite pulsarse sobre dicho ítem) o deshabilitado/en gris (no permite pulsar sobre dicho ítem).

GENERAR EL FICHERO EJECUTABLE MMX

Para poder utilizar un menú de usuario dentro de un formulario hay que generar el fichero ejecutable .MMX, para ello se compilará el menú desde el menú **Programa – Compilar PL/SQL - Todo** y luego generaremos el ejecutable desde **Programa – Compilar Módulo**.

UTILIZAR UN MENÚ DE USUARIO EN UN FORMULARIO

Una vez generado el fichero ejecutable del menú (.mmx), es posible utilizar este menú dentro de un formulario, para ello entraremos en la propiedad **Módulo de Menús** del formulario e incluiremos el nombre (sin extensión) y la ruta donde se encuentre el menú como en el ejemplo siguiente:

General	
Nombre	PRUEBA
Información de Subclase	
Comentarios	
Título del Libro de Ayuda	
Funcional	
Título	MODULE1
Ventana de Consola	WINDOW1
Módulo de Menús	D:\Proyecto_Prueba\MODULE2
Menú Inicial	
Diferir Forzado Necesario	No
Seguridad de Menús	
Rol de Menú	

EJECUCIÓN DE VARIOS FORMULARIOS 31

INTRODUCCIÓN

En este capítulo trataremos las metodologías para enlazar dos o más formularios, dado que una aplicación generalmente consta de varios de estos formularios que han de encadenarse para satisfacer las necesidades del usuario.

Funcionamiento habitual de una aplicación real

Cuando se diseña una aplicación real, se define un formulario de comienzo que es el que abre una sesión entre el usuario y el servidor de aplicaciones. Este formulario de inicio se define en el Forms Runtime.

El resto de módulos de pantalla que componen la aplicación se van abriendo en la misma sesión mediante disparadores incorporados.

En el diseño de una aplicación podemos crear ventanas independientes en las que el usuario pueda trabajar con varios formularios de forma simultánea en la misma sesión (utilizando OPEN_FORM). En estos casos el usuario podrá navegar entre los bloques visibles de los distintos formularios abiertos del mismo modo que si estuvieran en un único formulario (pantalla visible).

Igualmente se pueden crear transiciones entre los distintos formularios de forma que el usuario únicamente pueda trabajar con un formulario a la vez y cuando lo cierre vuelva al formulario anterior o incluso salga de la aplicación.

En la navegación entre formularios es posible compartir e intercambiar información de la siguiente forma:

- Mediante variables globales que tienen vigencia en la sesión.
- Mediante listas de parámetros que transfieren valores entre los formularios correspondientes.
- Mediante grupos de registros globales.
- Mediante variables PL/SQL en bibliotecas compartidas.

Para poder compartir código entre formularios, tenemos que recurrir a elementos de base de datos (como paquetes, procedimientos o funciones) o a bibliotecas de código PL/SQL conectadas a cada formulario.

Tipos de Built-in para invocar a formularios

Dentro de Forms Builder se dispone de 3 Built-in destinadas a la apertura de un nuevo formulario en la misma sesión:

- OPEN_FORM.
- CALL_FORM.
- NEW_FORM.

Cada una de estas Built-in presenta funcionalidades diferentes para la apertura del formulario que se irán desarrollando en este capítulo.

OPEN_FORM

Esta Built-in permite abrir otro formulario sin cerrar el anterior (desde el que se realiza la llamada). Permite trabajar de forma simultánea entre los dos formularios.

Esta Built-in no puede ser invocada en el modo Enter Query de un formulario.

SINTAXIS:

```
OPEN_FORM('nombre_formulario',modo_activado,
          modo_sesión, modo_datos,
          listaparámetros);
```

NOMBRE_FORMULARIO

En este parámetro y entre comillas, se indicará el nombre del formulario que se quiere abrir. No hay que indicar en el nombre la extensión del formulario (es decir, no se pone .FMX).

MODO_ACTIVADO

Presenta dos posibles valores:

- **ACTIVATE**: es la opción por defecto, y permite que el foco del formulario se coloque en el nuevo formulario que se abre.
- **NO_ACTIVATE**: abre el nuevo formulario pero el foco se mantiene en el formulario llamante.

MODO_SESIÓN

Presenta dos posibles valores:

- **NO_SESSION**: es la opción por defecto, e indica que el formulario que se abre comparte la misma sesión que el formulario llamante.
- **SESSION**: abre el nuevo formulario pero el foco se mantiene en el formulario llamante.

MODO_DATOS

Presenta dos posibles valores:

- **NO_SHARE_LIBRARY_DATA**: es la opción por defecto, e indica que no se compartirán datos entre los formularios abiertos que tengan idénticas librerías atachadas/conectadas (en tiempo de diseño) .
- **SHARE_LIBRARY_DATA**: se compartirán datos entre los formularios abiertos que tengan idénticas librerías atachadas/conectadas (en tiempo de diseño).

LISTAPARÁMETROS

Especifica el identificador único que se ha asignado a la lista de parámetros en el momento de la creación.

CALL_FORM

Esta Built-in permite abrir otro formulario manteniendo activo el anterior (desde el que se realiza la llamada), pero no permite trabajar de forma simultánea entre los dos formularios. Hasta que no se cierra el último formulario llamado mediante CALL_FORM no se puede trabajar con el anterior (desde el que se había realizado la llamada).

Esta Built-in sí puede ser invocada en el modo Enter Query de un formulario.

SINTAXIS:

```
CALL_FORM('nombre_formulario', visualización, menú,
          modo_consulta, modo_datos, listaparámetros);
```

NOMBRE_FORMULARIO

En este parámetro, y entre comillas, se indicará el nombre del formulario que se quiere abrir. No hay que indicar en el nombre la extensión del formulario (es decir, no se pone .FMX).

VISUALIZACIÓN

Este parámetro tiene dos valores posibles:

- **HIDE:** es el valor por defecto e indica que el formulario desde el que se produce la llamada se ocultará para mostrar el formulario llamado.
- **NO_HIDE:** el formulario desde el que se produce la llamada permanecerá visible cuando se muestra el formulario llamado.

MENÚ

Este parámetro tiene dos valores posibles:

- **NO_REPLACE:** es el valor por defecto e indica que el formulario llamado mantendrá su menú asociado cuando se abra.
- **DO_REPLACE:** el formulario llamado visualizará el mismo menú que el formulario que lo llama.

MODO_CONSULTA

Este parámetro tiene dos valores posibles:

- **NO_QUERY_ONLY:** es el valor por defecto e indica que el formulario llamado se ejecutará en modo normal, permitiéndose operaciones de inserción, actualización y borrado.
- **QUERY_ONLY:** el formulario llamado se ejecutará en modo consulta, lo que impedirá que se ejecute cualquier otra sentencia de manipulación de datos.

MODO_DATOS

Presenta dos posibles valores:

- **NO_SHARE_LIBRARY_DATA**: es la opción por defecto, e indica que no se compartirán datos entre los formularios abiertos que tengan idénticas librerías atachadas/conectadas (en tiempo de diseño).
- **SHARE_LIBRARY_DATA**: se compartirán datos entre los formularios abiertos que tengan idénticas librerías atachadas/conectadas (en tiempo de diseño).

LISTAPARÁMETROS

Especifica el identificador único que se ha asignado a la lista de parámetros en el momento de la creación.

NEW_FORM

Esta Built-in permite abrir otro formulario cerrando el anterior (desde el que se realiza la llamada).

Esta Built-in no puede ser invocada en el modo Enter Query de un formulario.

SINTAXIS:

```
CALL_FORM('nombre_formulario', modo_rollback,
          modo_consulta, modo_datos, listaparámetros);
```

NOMBRE_FORMULARIO

En este parámetro, y entre comillas, se indicará el nombre del formulario que se quiere abrir. No hay que indicar en el nombre la extensión del formulario (es decir, no se pone .FMX)

MODO_ROLLBACK

Presenta tres posibles valores:

- **TO_SAVEPOINT**: es la opción por defecto, e indica que cuando se salga del formulario llamado y haya cambios no confirmados con COMMIT, estos se deshacen (ROLLBACK) hasta el último punto de salva (SAVEPOINT) del formulario.
- **NO_ROLLBACK**: indica que se saldrá del formulario llamado sin que se realice ningún proceso de ROLLBACK (deshacer).
- **FULL_ROLLBACK**: indica que cuando se salga del formulario llamado y haya cambios no confirmados con COMMIT, todos ellos se deshacen (ROLLBACK).

MODO_CONSULTA

Este parámetro tiene dos valores posibles:

- **NO_QUERY_ONLY:** es el valor por defecto e indica que el formulario llamado se ejecutará en modo normal, permitiéndose operaciones de inserción, actualización y borrado.
- **QUERY_ONLY:** el formulario llamado se ejecutará en modo consulta, lo que impedirá que se ejecute cualquier otra sentencia de manipulación de datos.

MODO_DATOS

Presenta dos posibles valores:

- **NO_SHARE_LIBRARY_DATA**: es la opción por defecto, e indica que no se compartirán datos entre los formularios abiertos que tengan idénticas librerías atachadas/conectadas (en tiempo de diseño).
- **SHARE_LIBRARY_DATA**: se compartirán datos entre los formularios abiertos que tengan idénticas librerías atachadas/conectadas (en tiempo de diseño).

LISTAPARÁMETROS

Especifica el identificador único que se ha asignado a la lista de parámetros en el momento de la creación.

USO COMPARTIDO DE DATOS ENTRE MÓDULOS

Como se ha indicado en la introducción, los datos se pueden intercambiar entre formularios de las siguientes formas:

- Mediante variables globales que abarcan a la sesión.
- Mediante listas de parámetros que transfieren valores entre los formularios correspondientes.
- Mediante grupos de registros globales.
- Mediante variables PL/SQL en bibliotecas compartidas.

Compartición mediante variables globales

Las variables globales se definen en tiempo de diseño dentro de cualquier elemento del formulario que admita programación PL/SQL.

Las variables globales no se declaran, sino que se les asigna un valor y en ese instante quedan declaradas. Toda variable global es de tipo carácter (VARCHAR2) y admite como máximo 255 caracteres.

Una variable global se identifica con el prefijo **: GLOBAL.**

Dentro de una misma sesión, el nombre de las variables globales es único, es decir, si definimos una variable global en el primer formulario, con un nombre, no se podrá definir otra variable con el mismo nombre en otro formulario de la misma sesión, porque lo que se producirá será un cambio de contenido de la variable.

Por ejemplo, tomemos este ejemplo:

Formulario 1 (trigger WHEN-NEW-FORM-INSTANCE)

```
:GLOBAL.EDAD := 34; -- Se crea la variable global EDAD a valor 34.
```

Formulario 2 (trigger WHEN-NEW-FORM-INSTANCE) de la misma sesión

```
:GLOBAL.EDAD := 36;   -- No se crea la variable global EDAD porque
                      -- ya estaba creada en el formulario 1, por
                      -- tanto lo que se hace es actualizar el valor
                      -- a 36.
```

Si se intenta leer en tiempo de ejecución una variable global no definida (no se le ha asignado aún ningún valor, aunque sea NULL), provocará el siguiente error en tiempo de ejecución: FRM-40815: Variable GLOBAL.<nombre> does not exist .

Compartición mediante lista de parámetros

Una pantalla puede contener cualquier número de parámetros necesarios para su ejecución.

A diferencia de las variables globales los parámetros de una pantalla pueden ser de 3 tipos (CHAR, DATE, NUMBER).

Dentro de Forms Builder, los parámetros se crean en la sección **Parámetros** del navegador de objetos.

Cada parámetro definido en esta sección consta de las siguientes propiedades:

- **Nombre:** nombre del parámetro.
- **Tipo de Dato:** tipo de datos (CHAR, DATE, NUMBER).
- **Valor inicial:** valor que se asocia al parámetro cuando se define.

Una vez definido un parámetro, es posible utilizarlo en cualquier parte del formulario referenciándolo con el prefijo **:PARAMETER.**

Por ejemplo, para asignar un valor al parámetro definido en la imagen de la página anterior, haríamos lo siguiente:

```
:PARAMETER.PARAMETER1 := 10;
```

Una lista de parámetros es un conjunto de nombres de parámetros y de sus valores correspondientes que queremos pasarle a otro formulario para su ejecución. El formulario que recibe la llamada con la lista de parámetros debe contener con el mismo nombre cada uno de los parámetros que se indican en la lista que se le ha pasado.

Los pasos a seguir para crear una lista de parámetros son:

- Crear la lista de parámetros. Normalmente primero se comprueba que no exista una con el mismo nombre (si existe se borra) y luego se crea utilizando un código como el siguiente:

```
DECLARE
      Cod_lista    PARAMLIST;
      Nom_lista    VARCHAR2(10) := 'ltempo1';
BEGIN
      Cod_lista := GET_PARAMETER_LIST(Nom_lista);
      -- Se comprueba si existe la lista
      IF ID_NULL(Cod_lista) THEN
            -- Si no existe se crea
            Cod_lista := CREATE_PARAMETER_LIST(Nom_lista);
      ELSE
            -- Si existe se destruye y luego se crea
            DESTROY_PARAMETER_LIST(Cod_lista);
            Cod_lista := CREATE_PARAMETER_LIST(Nom_lista);
      END IF;
```

- Se agregan los parámetros necesarios para pasarle al formulario que se pretende abrir y ejecutar, utilizando un código como el siguiente:

```
-- Añadimos el parámetro PARAMETER1 a la lista y le
-- definimos de tipo texto (TEXT_PARAMETER) con un valor
-- TO_CHAR(:bloque1.item1. Es decir el valor del campo ITEM1
-- del bloque BLOQUE1
      ADD_PARAMETER(Cod_lista,'parameter1',TEXT_PARAMETER,
                          TO_CHAR(:bloque1.item1));
-- Se realiza la llamada al formulario pasándole la lista de
-- parámetros con un código como el siguiente:
-- Abrimos el formulario FORMULARIO2 pasándose la lista
-- de parámetros.
      OPEN_FORM('formulario2',ACTIVATE,NO_SESSION,Cod_lista);
```

Resumiendo, disponemos de las siguientes funciones para el manejo de listas de parámetros:

- **GET_PARAMETER_LIST**: obtiene el código identificativo de una lista pasándole el nombre.
- **CREATE_PARAMETER_LIST**: crea una lista.
- **DESTROY_PARAMETER_LIST**: elimina una lista.
- **ADD_PARAMETER**: añade un parámetro. En cuanto al tipo de los parámetros solo se admite TEXT_PARAMETER (para pasar un valor

concreto de un parámetro como texto), DATA_PARAMETER (para pasar el nombre de un grupo de valores de tipo RECORD_GROUP).

- **DELETE_PARAMETER**: elimina un parámetro.

Compartición mediante variables en bibliotecas PL/SQL compartidas

Esta es la forma más simple y eficiente de compartir datos entre formularios, dado que dentro de un paquete de una biblioteca compartida PL/SQL podemos definir variables de cualquier tipo de dato, incluso definidas por el propio usuario.

Una biblioteca se crea con al menos una especificación de paquete que contiene una o más declaración de variables. A continuación se conecta la biblioteca al formulario o formularios en los que se va a utilizar.

Para permitir que todos los formularios abiertos compartan los datos de la biblioteca, cuando se abra cada formulario habrá de hacerse con la opción SHARE_LIBRARY_DATA.

CIERRE Y VALIDACIÓN DE FORMULARIOS

Para poder cerrar un formulario abierto mediante código programado, se utilizan dos Built-in: EXIT_FORM y COMMIT_FORM.

Exit_Form

Permite cerrar un formulario confirmando los COMMIT y acciones específicas de ROLLBACK.

- EXIT_FORM cierra el formulario y si hay cambios que no han sido confirmados mediante el COMMIT, el propio comando EXIT_FORM proporciona un mecanismo para confirmarlos antes de abandonar el formulario.
- Si el formulario que se va a cerrar se encuentra en modo ENTER QUERY, el comando EXIT_FORM únicamente sale de este modo sin abandonar el formulario.

SINTAXIS:

```
EXIT_FORM(modo_commit, modo_rollback);
```

MODO_COMMIT

Este parámetro es opcional y permite las siguientes cuatro opciones:

- **ASK_COMMIT:** permite solicitar por pantalla confirmación antes de completar el proceso COMMIT de las operaciones sin confirmar.
- **DO_COMMIT:** confirma (COMMIT) los cambios pendientes antes de salir del formulario sin solicitar nada por pantalla.
- **NO_COMMIT:** valida los cambios y abandona el formulario sin realizar COMMIT de los cambios pendientes de confirmación.
- **NO_VALIDATE:** abandona el formulario sin validar cambios y sin realizar COMMIT de los que estén pendientes de confirmación.

MODO_ROLLBACK

Este parámetro es opcional y permite las siguientes tres opciones:

- **TO_SAVEPOINT:** realiza un proceso de rollback (deshacer) de los cambios pendientes de confirmación hasta el último punto de salva (SAVEPOINT) del formulario.
- **FULL_ROLLBACK:** realiza un rollback (deshacer) completo de todos los cambios pendientes de confirmación.
- **NO_ROLLBACK:** se abandona el formulario sin realizar rollback de los cambios pendientes de confirmación hasta el último punto de salva (SAVEPOINT).

Commit_Form

Permite cerrar un formulario validando los cambios del formulario y confirmándolos mediante un COMMIT antes de abandonarlo.

SINTAXIS:

```
COMMIT_FORM;
```

CREAR LIBRERÍAS EN FORMS 32

INTRODUCCIÓN

Las librerías de código PL/SQL de Forms es una recopilación de unidades de programa PL/SQL, incluidos procedimientos, funciones y paquetes. Una única biblioteca puede contener muchas unidades de programa que pueden compartir las aplicaciones.

Una biblioteca:

- Se crea como un módulo independiente y se almacena en un archivo o en la base de datos.
- Proporciona un método adecuado de almacenamiento del código del cliente y su uso compartido entre aplicaciones.
- Supone que una única copia de unidades de programa puede ser utilizada por muchos módulos de pantalla, de menú o de informe.
- Soporta la carga dinámica de unidades de programa.

Una librería se crea dentro de Oracle Forms Builder y se almacena con la extensión **.PLL** en el sistema de archivos. Esta extensión indica que el fichero contiene el código fuente de la librería.

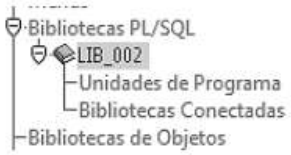
Cuando queremos que el código contenido en una librería se pueda ejecutar en nuestros formularios, tenemos que compilarla para obtener un fichero con extensión **.PLX**.

Cuando vayamos a utilizar una librería en uno de nuestros formularios, tendremos que adjuntar (atachar) la misma dentro de Oracle Forms Builder.

Una biblioteca no puede utilizar variables ligadas con los formularios, menús o informes dado que están fuera del ámbito de la misma.

CREAR UNA BIBLIOTECA

En primer lugar abrimos Forms Builder y dentro del navegador de objetos nos situamos en el elemento **Bibliotecas PL/SQL.**



A continuación pulsamos el botón  para añadir una nueva biblioteca y el sistema asigna un nombre automáticamente para la misma.

Si queremos que tenga un nombre distinto, en las propiedades (**F4**) de la biblioteca se lo podemos cambiar, o también se puede realizar esta operación al "guardar como" la librería dentro del menú **Archivo**.

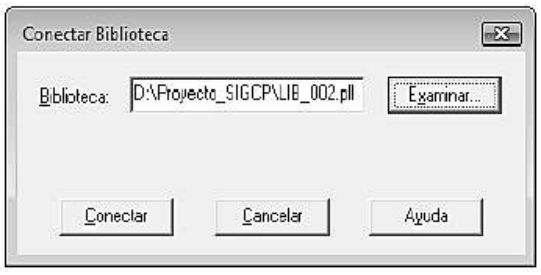
COMPILAR UNA BIBLIOTECA

Una vez que hemos almacenado el código fuente de la biblioteca en un fichero **.PLL**, tenemos que compilarlo para depurar errores. Esto se realiza desde el menú **Programa – Compilar PL/SQL - Todo.**

Si no hay errores de compilación el siguiente paso consiste en crear el fichero ejecutable (.**PLX**) para poder ejecutar el código de la librería en nuestros formularios. Esto se realiza desde el menú **Programa – Compilar Módulo.**

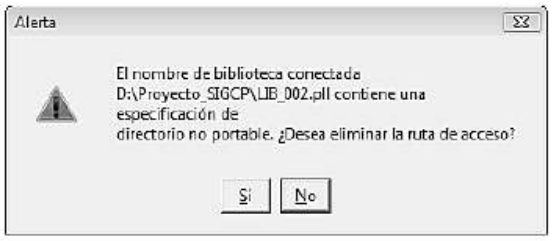
CONECTAR UNA BIBLIOTECA A UN MÓDULO

Antes de que se puede hacer referencia a una unidad de programa de una biblioteca de código PL/SQL, desde un formulario, menú o informe, es necesario conectarla al módulo. Para realizar esta acción en primer lugar abrimos el módulo en Forms Builder y a continuación dentro del navegador de objetos nos situamos en el elemento **Bibliotecas Conectadas.**



A continuación pulsamos el botón  para conectar la biblioteca, y se nos muestra una pantalla donde nos solicita la ruta completa donde se encuentra la biblioteca con extensión (.PLL). Es

importante reseñar que para conectar la biblioteca al módulo solo es necesario que exista el código fuente de la misma (.PLL), pero si queremos que luego el módulo ejecute la programación de la biblioteca entonces será necesario que también exista el fichero ejecutable (.PLX).



Una vez indicada la ruta completa y el nombre de la librería, pulsamos el botón **Conectar**. Al hacerlo se presenta una alerta del sistema como la que se muestra a continuación:

Las dos posibles opciones son **SÍ** o **NO**. La diferencia entre elegir una opción y otra es la siguiente:

- **SÍ:** supone que se asocia la librería al módulo pero no la ruta donde se encuentra la misma. Por tanto, cuando se ejecute el módulo, se buscará la librería ejecutable (.PLX) en la ruta por defecto que se haya indicado en la instalación para las librerías, o en su defecto en la misma ruta donde se encuentra el módulo.
- **NO:** supone que se asocia la librería y la ruta de la misma al módulo por lo que se buscará la librería ejecutable (.PLX) en la misma ruta indicada en la vinculación.

En cualquiera de las dos opciones al finalizar la vinculación se presenta la biblioteca colgando del elemento **Bibliotecas Conectadas** y se puede visualizar el código que tiene.

DESCONECTAR UNA BIBLIOTECA

Para desconectar una biblioteca de un módulo, simplemente hay que suprimir la biblioteca del grupo **Bibliotecas Conectadas** del módulo.

REFERENCIAR A UNIDADES DE PROGRAMA DE BIBLIOTECAS CONECTADAS

Se hace referencia a una unidad de programa de una biblioteca conectada a un módulo, de la misma forma que a cualquier otra unidad de programa, es decir, por el nombre de la misma sin anteponerle nada.

Por ejemplo en los ejemplos mostrados hasta el momento de la librería **LIB_002.PLL**, el único paquete se invocaría por su nombre: **PRUEBA.**

Orden de ejecución de unidades de programa

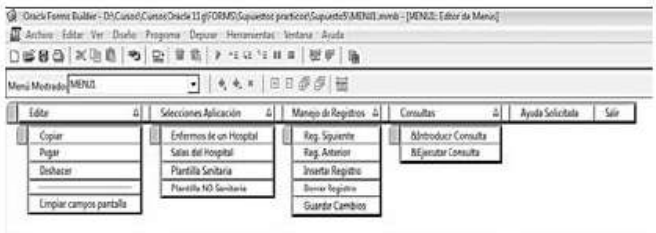
Cuando en un mismo módulo coexisten unidades de programa del propio módulo y de una o varias librerías conectadas (con el mismo nombre o diferente) el método de búsqueda que hace Oracle Forms, a la hora de ejecutarlas es el siguiente:

1. Se busca la unidad de programa dentro del módulo.
2. Se busca la unidad de programa dentro de la primera librería que se encuentra conectada al módulo.
3. Se busca la unidad de programa en el resto de librerías conectadas al módulo en orden descendente según aparecen en el propio módulo.

SUPUESTO PRÁCTICO 5

MENÚ DE LA APLICACIÓN

Será necesario crear un menú personalizado para nuestra aplicación del hospital que se almacene con el nombre MENU1.MMB y que se componga de los elementos que se muestran en la siguiente imagen:



El menú se compondrá de 6 elementos principales:

- Editar.
- Selecciones Aplicación.
- Manejo de Registros.
- Consultas.
- Ayuda Solicitada.
- Salir

Asimismo, 4 de estos elementos dispondrán de un submenú de la siguiente forma:

- El menú **Edita**r tendrá el siguiente submenú:
  - Copiar
  - Pegar
  - Deshacer
  - ----------
  - Limpiar campos pantalla
- El menú **Selecciones Aplicación** tendrá el siguiente submenú:
  - Enfermos de un Hospital.
  - Salas del Hospital.
  - Plantilla Sanitaria.
  - Plantilla NO Sanitaria.
- El menú **Manejo de Registros** tendrá el siguiente submenú:
  - Reg. Siguiente.
  - Reg. Anterior.
  - Insertar Registro.
  - Borrar Registro.
  - Guardar Cambios.
- El menú **Consultas** tendrá el siguiente submenú:
  - &Introducir Consulta.
  - &Ejecutar Consulta.

Los elementos del submenú **Selecciones Aplicación** formarán parte del mismo grupo de Radio, de forma que solo se podrá marcar uno de ellos.

Una vez creado el menú se le deberá dotar de la funcionalidad que se indica a continuación:

- El submenú **Enfermos de un Hospital** deberá acceder al formulario ENFERMOS_HOSPITAL.FMX sin cerrar el formulario en el que se encuentre el usuario.
- El submenú **Salas del Hospital** deberá acceder al formulario HOSPITAL_SALA.FMX sin cerrar el formulario en el que se encuentre el usuario.
- El submenú **Plantilla Sanitaria** deberá acceder al formulario PLANTILLA_SANITARIA.FMX sin cerrar el formulario en el que se encuentre el usuario.
- El submenú **Plantilla NO Sanitaria** deberá acceder al formulario PLANTILLA_NO_SANITARIA sin cerrar el formulario en el que se encuentre el usuario.
- El submenú **Reg. Siguiente** deberá navegar al registro siguiente.

- El submenú **Reg. Anterior** deberá navegar al registro anterior.
- El submenú **Insertar Registro** deberá crear un nuevo registro.
- El submenú **Delete Record** deberá borrar un registro.
- El submenú **Guardar Cambios** deberá realizar un commit del formulario.
- El submenú **&Introducir Consulta** deberá ejecutar el modo ENTER_QUERY del formulario.
- El submenú **&Ejecutar Consulta** deberá ejecutar el modo EXECUTE_QUERY del formulario.
- El menú **Ayuda Solicitada** deberá mostrar la ayuda de Forms.
- El menú **Salir** deberá salir del formulario.

PARA TODOS LOS MÓDULOS

Será necesario adjuntar el menú creado anteriormente a todos los módulos de la aplicación es decir: BIENVENIDO.FMX, ENFERMOS_HOSPITAL, HOSPITAL_SALA, PLANTILLA_NO_SANITARIA, PLANTILLA_SANITARIA.

DISPARADORES 33

INTRODUCCIÓN

Los disparadores son uno de los mecanismos más importantes que se pueden utilizar para modificar o agregar funcionalidad a una pantalla. En este tema aprenderá las reglas y las propiedades esenciales de los disparadores para que pueda utilizarlos en su aplicación.

Un disparador es una unidad de programa que se ejecuta (arranca) cuando se produce un evento. Los disparadores de Forms Builder se escriben en PL/SQL. Cada disparador que se define está asociado a un evento específico. Forms Builder define un amplio rango de eventos para los que se puede arrancar un disparador. Entre estos eventos se incluyen:

- Eventos relacionados con consultas.
- Entrada y validación de datos.
- Navegación lógica o física.
- Interacción del operador con elementos de la pantalla.
- Eventos internos en la pantalla.
- Errores y mensajes.

Los eventos provocan la activación, o arranque, de determinados tipos de disparadores.

CATEGORÍA DE DISPARADORES

Los disparadores se pueden clasificar según sus funciones:

Categoría	Arranca	Ejemplos
Procesamiento de bloque	En respuesta a eventos relacionados con la gestión de registros en un bloque.	WHEN- CREATE- RECORD WHEN- CLEAR- BLOCK WHEN- DATABASE- RECORD WHEN- REMOVE- RECORD
Evento de interfaz	En respuesta a eventos que se producen en la interfaz de pantalla	KEY- [ALL] WHEN- BUTTON- PRESSED WHEN- CHECKBOX- CHANGED WHEN- LI ST- CHANGED WHEN- RADI O- CHANGED WHEN- I MAGE- ACTI VATED WHEN- I MAGE- PRESSED WHEN- TI MER- EXPI RED WHEN- WI NDOW- ACTI VATED WHEN- WI NDOW- DEACTI VATED WHEN- WI NDOW- CLOSED WHEN- WI NDOW- RESI ZED
Maestro-detalle	Para forzar la coordinación entre registros en un bloque detalle y el registro maestro en un bloque maestro.	ON- CHECK- DELETE- MASTER ON- CLEAR- DETAI LS ON- POPULATE- DETAI LS
Manejo de mensajes	En respuesta a eventos de mensajes por defecto	ON- ERROR ON- MESSAGE
Navegación	En respuesta a eventos de navegación	PRE- FORM PRE- BLOCK PRE- RECORD PRE- TEXT- I TEM POST- FORM POST- BLOCK POST- RECORD POST- TEXT- I TEM WHEN- NEW- FORM- I NSTANCE WHEN- NEW- BLOCK- I NSTANCE WHEN- NEW- RECORD- I NSTANCE WHEN- NEW- I TEM- I NSTANCE
Tiempo de consulta	Inmediatamente antes y después de que el operador o la aplicación ejecuten una consulta en un bloque	PRE- QUERY POST- QUERY
Validación	Cuando Forms valida datos en un elemento o registro	WHEN- VALI DATE- I TEM WHEN- VALI DATE- RECORD

Los disparadores también se pueden clasificar según sus nombres:

Categoría	Arranca
WHEN- EVENT	Punto en el que se debe ***aumentar*** el procesamiento por defecto de Forms con operaciones o tareas adicionales.
ON- EVENT	Punto en el que se puede ***sustituir*** el procesamiento por defecto de Forms.
PRE- EVENT	Punto inmediatamente anterior a la incidencia de un evento WHEN-EVENT u ON-EVENT; se utiliza para preparar objetos o datos para el próximo evento.
POST- EVENT	Punto inmediatamente siguiente a la incidencia de un evento WHEN-EVENT u ON-EVENT; se utiliza para validar o realizar algunas tareas de auditoría basadas en el evento anterior.
Disparadores KEY	Se arrancan cuando el operador pulsa una tecla o una secuencia de teclas específica.

RELACIÓN COMPLETA DE DISPARADORES

A continuación se relaciona la lista completa de disparadores que se presentan en la versión Oracle Forms Builder 12c:

Nombre	Nombre	Nombre
KEY- CLRBLK	KEY- CLRFRM	KEY- CLRREC
KEY- COMMI T	KEY- CQUERY	KEY- CREREC
KEY- DELREC	KEY- DOWN	KEY- DUP- I TEM
KEY- DUPREC	KEY- EDI T	KEY- ENTER
KEY- ENTQRY	KEY- EXEQRY	KEY- EXI T
KEY- F0	KEY- F1	KEY- F2
KEY- F3	KEY- F4	KEY- F5
KEY- F6	KEY- F7	KEY- F8
KEY- F9	KEY- HELP	KEY- LI STVAL
KEY- MENU	KEY- NEXT- I TEM	KEY- NXTBLK
KEY- NXTKEY	KEY- NXTREC	KEY- NXTSET
KEY- OTHERS	KEY- PREV- I TEM	KEY- PRI NT
KEY- PRVBLK	KEY- PRVREC	KEY- SCRDOWN
KEY- SCRUP	KEY- UP	KEY- UPDREC
ON- CHECK- DELETE- MASTER	ON- CHECK- UNI QUE	ON- CLEAR- DETAI LS
ON- CLOSE	ON- COLUMN- SECURI TY	ON- COMMI T
ON- COUNT	ON- DELETE	ON- ERROR
ON- FETCH	ON- I NSERT	ON- LOCK
ON- LOGON	ON- LOGOUT	ON- MESSAGE
ON- POPULATE- DETAI LS	ON- ROLLBACK	ON- SAVEPOI NT
ON- SELECT	ON- SEQUENCE- NUMBER	ON- UPDATE
POST- BLOCK	POST- CHANGE	POST- DATABASE- COMMI T
POST- DELETE	POST- FORM	POST- FORMS- COMMI T
POST- I NSERT	POST- LOGON	POST- LOGOUT

POST- QUERY	POST- RECORD	POST- SELECT
POST- TEXT- I TEM	POST- UPDATE	PRE- BLOCK
PRE- COMMI T	PRE- DELETE	PRE- FORM
PRE- I NSERT	PRE- LOGON	PRE- LOGOUT
PRE- POPUP- MENU	PRE- QUERY	PRE- RECORD
PRE- SELECT	PRE- TEXT- I TEM	PRE- UPDATE
WHEN- BUTTON- PRESSED	WHEN- CHECKBOX- CHANGED	WHEN- CLEAR- BLOCK
WHEN- CREATE- RECORD	WHEN- CUSTOM- I TEM- EVENT	WHEN- CUSTOM- J AVASCRI PT- EVENT
WHEN- DATABASE- RECORD	WHEN- EVENT- RAI SED	WHEN- FORM- NAVI GATE
WHEN- I MAE- ACTI VATED	WHEN- I MAGE- PRESSED	WHEN- LI ST- ACTI VATED
WHEN- LI ST- CHANGED	WHEN- MOUSE- CLI CK	WHEN- MOUSE- DOUBLECLI CK
WHEN- MOUSE- DOWN	WHEN- MOUSE- ENTER	WHEN- MOUSE- LEAVE
WHEN- MOUSE- MOVE	WHEN- MOUSE- UP	WHEN- NEW- BLOCK- I NSTANCE
WHEN- NEW- FORM- I NSTANCE	WHEN- NEW- I TEM- I NSTANCE	WHEN- NEW- RECORD- I NSTANCE
WHEN- RADI O- CHANGED	WHEN- REMOVE- RECORD	WHEN- TAB- PAGE- CHANGED
WHEN- TI MER- EXPI RED	WHEN- TREE- NODE- ACTI VATED	WHEN- TREE- NODE- EXPANDED
WHEN- TREE- NODE- SELECTED	WHEN- VALI DATE- I TEM	WHEN- VALI DATE- RECORD
WHEN- WI NDOW- ACTI VATED	WHEN- WI NDOW- CLOSED	WHEN- WI NDOW- DEACTI VATED
WHEN- WI NDOW- RESI ZED		

COMPONENTES DE UN DISPARADOR

Cuando se diseña un disparador de Forms Builder se deben tener en cuenta tres componentes principales:

- **Tipo de disparador:** define el evento específico que provocará el arranque del disparador.
- **Código del disparador:** el cuerpo de PL/SQL que define las acciones del disparador.
- **Ámbito del disparador:** El nivel en un módulo de pantalla en el que se define el disparador; determina el alcance de los eventos que detectará el disparador.

Tipo de disparador

El tipo de disparador determina el tipo de evento que arranca. Hay más de 100 disparadores incorporados, cada uno de ellos identificado por un nombre específico. En la mayoría de los casos, los disparadores son arrancados por eventos en un módulo de pantalla. Los módulos de menú pueden iniciar un evento en una pantalla, pero el módulo de pantalla es el propietario del disparador que se arranca.

El nombre de un disparador identifica su tipo. Todos los tipos de disparadores incorporados están asociados a un evento y sus nombres siempre contienen un guion (-). Por ejemplo:

- WHEN-VALIDATE-ITEM se arranca cuando Forms valida un elemento.
- PRE-QUERY se arranca antes de que Forms emita una consulta para un bloque.

Forms Builder soporta los disparadores creados por el usuario así como disparadores incorporados estándar. Los disparadores creados por el usuario son los que define el diseñador del programa. Estos disparadores solo se arrancan si los llama otro disparador o una unidad de programa que utilice funciones de código incorporadas.

La primera parte del nombre de un disparador (antes del primer guion) sigue una regla estándar; esto le ayuda a comprender la naturaleza general del tipo de disparador y planificar los tipos que se van a utilizar.

Prefijo del disparador	Descripción
KEY-	Se arranca en lugar de la acción estándar de una tecla de función
ON-	Se arranca en lugar del procesamiento estándar (se utiliza para sustituir o ignorar un proceso)
PRE-	Se arranca inmediatamente antes de la acción denominada en el tipo de disparador (por ejemplo, antes de que se ejecute una consulta)
POST-	Se arranca inmediatamente después de la acción denominada en el tipo de disparador (por ejemplo, después de que se ejecute una consulta)
WHEN-	Se arranca además del procesamiento estándar (se utiliza para aumentar la funcionalidad)

Código del disparador

El código del disparador define las acciones que este llevará a cabo cuando se arranque. Debe escribir este código como un bloque PL/SQL anónimo mediante el

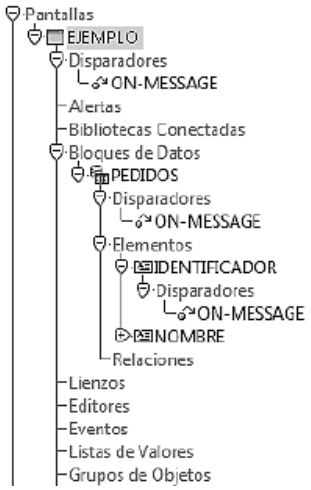
editor de PL/SQL integrado en Forms Builder, es decir, tiene que introducir la estructura BEGI N…END en el texto del disparador solo si inicia el bloque con una sentencia DECLARE o si tiene que codificar subbloques.

Las sentencias que se pueden escribir en un disparador son las mismas que se podrían escribir en cualquier otro procedimiento, paquete o función del formulario, con la salvedad de las sentencias de control de transacciones: COMMIT, ROLLBACK o SAVEPOINT que no deben incluirse directamente como sentencias dentro de un disparador, ya que pueden provocar el lanzamiento de otros disparadores del propio Forms. Por ejemplo si dentro de un disparador WHEN-BUTTON-PRESSED (que se ejecuta al pulsar un botón), después de un conjunto de operaciones, indicamos la instrucción COMMIT, provocará que se ejecute también el trigger COMMIT_FORM.

Ámbito del disparador

El ámbito de un disparador viene determinado por su posición en la jerarquía de objetos de pantalla, es decir, el tipo de objeto bajo el que se crea el disparador. Hay tres posibles niveles:

- **Nivel de pantalla:** el disparador pertenece a la pantalla y se puede arrancar gracias a eventos de toda la pantalla.
- **Nivel de bloque:** el disparador pertenece a un bloque y solo se puede arrancar cuando este es el bloque actual.
- **Nivel de elemento:** el disparador pertenece a un elemento individual y solo se puede arrancar cuando este elemento es el actual.



Algunos disparadores no se pueden definir por debajo de un determinado nivel. Por ejemplo, los disparadores POST-QUERY no se pueden definir a nivel de un elemento dado que se arranca debido a una consulta global o restringida dentro de un bloque. Por defecto solo se arranca el disparador que es más específico a la ubicación actual del cursor. Para comprender mejor esta afirmación, situémonos en el ejemplo mostrado en la imagen.

Como podemos ver hay definidos 3 disparadores del tipo ON-MESSAGE (se arranca cuando se muestra un mensaje) en ámbitos diferentes. Entonces la pregunta sería: ¿Cuándo se ejecutará cada uno si se genera un mensaje? Pues bien, el disparador ON-MESSAGE a nivel del campo IDENTIFICADOR solo se ejecutará cuando se genere un mensaje estando el

puntero dentro de dicho campo. El disparador ON-MESSAGE del bloque PEDIDOS se ejecutará cuando se produzca un mensaje estando el puntero dentro del campo NOMBRE, que es un elemento perteneciente al propio bloque PEDIDOS. Por último el disparador ON-MESSAGE del formulario EJEMPLO se ejecutará cuando se genere un mensaje estando el puntero fuera del bloque PEDIDOS.

Orden de apertura de un formulario

Cuando se programan diversos disparadores en un formulario, el orden que se sigue para la apertura es el siguiente:

- Objeto Módulo
  - PRE-FORM
  - WHEN NEW-FORM-INSTANCE
  - Objeto Bloque
    - PRE-BLOCK
    - WHEN-NEW-BLOCK-INSTANCE
    - Objeto Item
      - PRE-TEXT-ITEM
      - WHEN-NEW-ITEM-INSTANCE

JERARQUÍA DE EJECUCIÓN

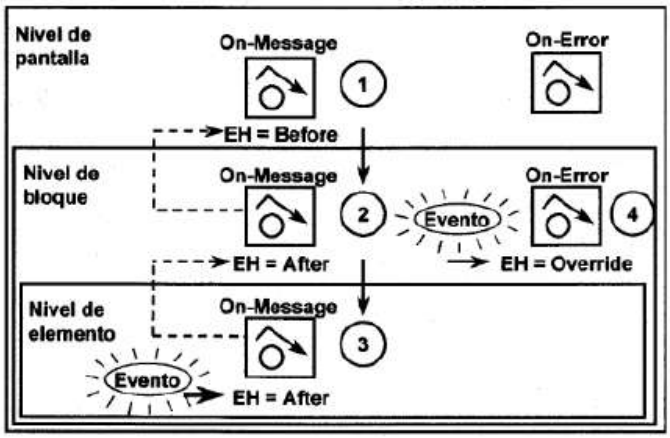
Como ya se ha mencionado, cuando hay más de un disparador del mismo tipo, Forms arranca normalmente el disparador más específico a la ubicación del cursor. Para modificar la secuencia de arranque de un disparador, defina la propiedad del disparador **Jerarquía de Ejecución.**

Para acceder a las propiedades de cualquier disparador únicamente nos tenemos que situar en el navegador de objetos, marcar el disparador y pulsar **F4**, lo que nos abrirá la paleta de propiedades del disparador.

La propiedad **Jerarquía de Ejecución** permite indicar cómo se debe ejecutar el código del disparador actual si hay un disparador con el mismo nombre definido a un nivel superior en la jerarquía de objetos. La definición de esta propiedad a nivel de formulario no tiene efecto, dado que no hay ningún disparador de nivel superior.

Los valores admitidos para esta propiedad son:

- **Sustituir (Override):** solo se arranca el disparador más específico de la ubicación del cursor. Esta es la opción por defecto.
- **Después (After): e**l disparador se arranca *después* de que se arranque el mismo disparador, si lo hay en el siguiente nivel superior.
- **Antes (Before):** el disparador se arranca *antes* de que se arranque el mismo disparador, si lo hay en el siguiente nivel superior.



En este ejemplo se muestra cómo la definición de la propiedad **Jerarquía de Ejecución** afecta al orden de arranque de un mismo disparador en distintos ámbitos.

AÑADIR UN DISPARADOR A UN FORMULARIO

Para añadir un disparador a nuestro formulario hay que seguir estos pasos:

- Seleccionar el ámbito de creación.
- Añadir el disparador.
- Definir el código del mismo.
- Compilar el código.

Seleccionar el ámbito del disparador

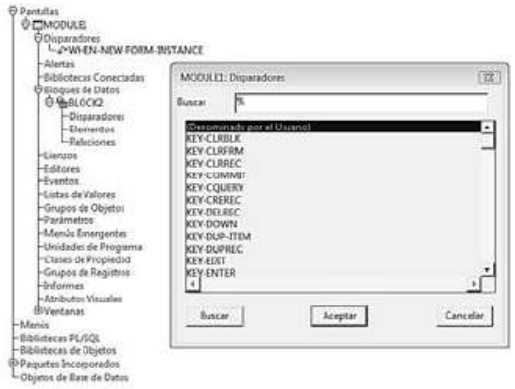
Nos situamos dentro del navegador de objetos del formulario en la sección **Disparadores** del ámbito en el que vamos a crear el disparador: formulario, bloque o elemento.

Añadir el disparador

Para añadir el disparador tenemos dos métodos: a través de los disparadores Smart o seleccionándolo de la lista completa.

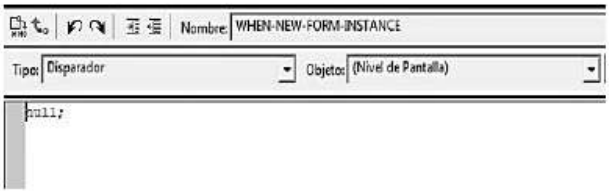
1. Elegir un disparador de los conocidos como Smart (recomendaciones más habituales para el ámbito seleccionado). Para ello pulsaremos con el botón derecho dentro de la sección disparadores del ámbito concreto y el diseñador nos mostrará una pequeña lista para seleccionar un disparador. A continuación se puede ver un ejemplo:

2. Si queremos seleccionar un disparador concreto dentro de la lista completa, pulsaremos el botón añadir en la sección disparadores del ámbito concreto y se nos mostrará la lista íntegra de disparadores, como se puede ver en el ejemplo.



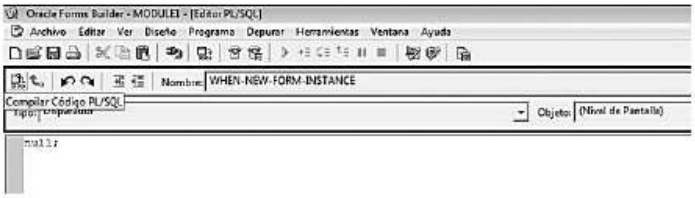
Definir el código del disparador

Una vez que hemos seleccionado el disparador para añadirlo al formulario, inmediatamente se muestra el **Editor PL/SQL** con objeto de programar el código que se va ejecutar cuando se desencadene el disparador.



Esto se produce porque no se permite crear disparadores sin código. Por tanto, si por motivos de diseño previo de un formulario queremos agregar los disparadores que se van a utilizar en un formulario para definir su código con posterioridad (esqueleto de ejecución de un formulario), al menos deberemos crear todos los disparadores con la instrucción NULL (que equivale a que el trigger no ejecuta nada, pero contiene un código válido).

Compilar el disparador



Una vez introducido el código programado que se va a ejecutar dentro del disparador, tenemos que compilarlo. Para ello pulsaremos sobre el botón que se muestra en la siguiente imagen.

PROPIEDADES DE UN DISPARADOR

Se pueden definir las siguientes propiedades del disparador:

- **Nombre: d**ebe ser uno de los disparadores existentes dentro de Forms Builder.
- **Texto del Disparador:** el código PL/SQL definido que se ejecutará cuando se lance el disparador.
- **Ejecutar en Modo Introducir Consulta:** indica si el disparador se arrancará cuando se produzca el evento correspondiente también en modo Consulta (**SI**) o solo cuando se esté en modo normal (**NO**).
- **Jerarquía de Ejecución:** definida con anterioridad.
- **Mostrar en 'Ayuda de Teclado':** solo se puede utilizar con disparadores KEY-, y especifica si se desea que el nombre o la descripción aparezca en la ventana de "Mostrar Teclas".
- **Texto de 'Ayuda de Teclado':** solo se puede utilizar con disparadores KEY-, y permite introducir la descripción que queremos que se muestre en vez de la que aparece por defecto.

ESCRITURA DEL CÓDIGO DE UN DISPARADOR

El texto del código de un disparador de Forms Builder es un bloque PL/SQL que consta de tres secciones:

- Una sección de declaración para variables, constantes y excepciones (opcional).
- Una sección de sentencias ejecutable (obligatoria).
- Una sección de control de errores o manejo de excepciones (opcional).

Si el código del disparador no requiere variables definidas, no es necesario recubrir el bloque de código ejecutable con las palabras BEGIN. . END, dado que estas son agregadas implícitamente.

Ejemplo 1: Disparador sin variables

Si el disparador no requiere sentencias declarativas de variables, las palabras BEGIN y END son opcionales. A continuación se muestra un código ejemplo para un disparador:

```
IF :PEDIDOS.precio_unitario IS NULL THEN
      :PEDIDOS.precio_unitario := :PRODUCTOS.lista_precio;
END IF;
CALCULO_TOTAL;       -- Llamada a procedimiento definido por el usuario
                     -- en el formulario
```

Ejemplo 2: Disparador con variables

Si el disparador tiene sentencias declarativas de variables, las palabras BEGIN y END son obligatorias. A continuación se muestra un ejemplo para un disparador:

```
DECLARE
      V_descuento  NUMBER;
BEGIN
      V_descuento := CALCULA_DESCUENTO(:PRODUCTOS.Id);
END;
```

Ejemplo 3: Disparador con control de errores

Si el disparador tiene sentencias para el control de errores (excepciones), las palabras BEGIN y END son opcionales, si no tiene declaración de variables. A continuación se muestra un ejemplo para un disparador:

```
INSERT INTO LOG_TABLA (LOG_VALOR, LOG_USUARIO)
VALUES (:DPTOS.departamento_id, :GLOBAL.usuario);
EXCEPTION
       WHEN OTHERS THEN
       MESSAGE('Error: '||SQLERRM);
END;
```

USO DE VARIABLES EN DISPARADORES

En disparadores y subprogramas, Forms Builder acepta generalmente dos tipos de variables para almacenar valores:

- **Variables PL/SQL:** se deben declarar en una sección DECLARE y permanecen disponibles hasta el final del bloque de declaración. No llevan dos puntos como prefijo. Si se declaran en un paquete PL/SQL, hay una variable accesible en todos los disparadores que acceden a este paquete.
- **Variables de Forms Builder:** tipos de variables mantenidos por Forms Builder. PL/SQL las ve como variables externas y requieren un prefijo de dos puntos (:) para distinguirlas de los objetos PL/SQL (excepto cuando su nombre se transfiere a un subprograma como una cadena de caracteres). Las variables de Forms Builder no se declaran formalmente en una sección DECLARE y pueden existir fuera del ámbito de un bloque PL/SQL.

WHEN-BUTTON-PRESSED

Este disparador se arranca cuando el usuario hace clic en un botón. Puede definir el disparador en un elemento individual o en niveles superiores si es necesario. WHEN-BUTTON-PRESSED acepta tanto funciones incorporadas restringidas como no restringidas. Puede utilizar botones para proporcionar un amplio rango de funciones a los usuarios. Entre ellas se incluyen:

- Navegación.
- Visualización de lista de valores.
- Llamada a cálculos y otras funciones.

WHEN-WINDOW-CLOSED

Este disparador se arranca cuando se cierra una ventana mediante un comando de cierre específico del gestor de ventanas. Dicho disparador se define a nivel de pantalla. El disparador WHEN-WINDOW-CLOSED acepta funciones incorporadas restringidas y no restringidas. Este disparador se utiliza para cerrar una ventana mediante programación cuando el operador emite el comando CLOSE del gestor de ventanas. Forms Builder no cierra la ventana cuando el operador emite un comando de cierre específico del gestor de ventanas; solo arranca el disparador WHEN-WINDOW-CLOSED. Es responsabilidad del desarrollador escribir la funcionalidad necesaria en este disparador. Puede cerrar una ventana con los subprogramas incorporados HIDE_WINDOW, SET_WINDOW_PROPERTY y EXIT_FORM. No se puede ocultar la ventana que contiene el elemento actual.

WHEN-CHECKBOX-CHANGED

Este disparador se arranca cuando se produce una modificación en una casilla de control. Cuando el usuario activa o desactiva una casilla de control, se define el valor asociado al estado y puede que desee realizar acciones de disparador basadas en este cambio. Tenga en cuenta que la función **CHEKBOX_CHECKED** le permite probar el estado de una casilla de control sin necesidad de conocer los valores asociados al elemento.

WHEN-LIST-CHANGED

Puede utilizar este disparador para interrumpir la selección del usuario en una lista de valores. Para las listas de texto, puede interrumpir los dobles clic con **WHEN-LIST_ACTIVATED.**

DISPARADORES ASOCIADOS A LAS CONSULTAS

Los disparadores de transacciones para el procesamiento de consultas son los siguientes:

Categoría	Arranca
ON- CLOSE	Arranca cuando Forms cierra una consulta (aumenta, en lugar de sustituir, el procesamiento por defecto).
ON- COUNT	Arranca cuando Forms normalmente realizaría un procesamiento de Consulta de Recuento por defecto para determinar el número de filas que coinciden con las condiciones de la consulta.

ON- FETCH	Arranca cuando Forms realiza una recuperación para un juego de filas. El disparador se sigue arrancando hasta que: • No se crea ningún registro consultado durante una ejecución única del disparador. • El usuario o la función incorporada ABORT_QUERY ejecutada desde otro disparador cierran la consulta. • El disparador emite FORM_TRIGGER_FAILURE.
PRE- SELECT	Arranca después de que Forms haya creado la sentencia SELECT del bloque basado en las condiciones de la consulta, pero antes de que ejecute esta sentencia.
ON- SELECT	Arranca cuando Forms normalmente ejecutaría la sentencia SELECT del bloque (el disparador sustituye las fases de cursor abierto, análisis y ejecución de una consulta).
POST-SELECT	Arranca después de que Forms haya creado y ejecutado la sentencia SELECT de bloque, pero antes de que recupere los registros.

Los disparadores de consulta **PRE-QUERY** y **POST-QUERY** se arrancan debido al propio proceso de consulta y normalmente están definidos en el bloque en que tiene lugar la consulta.

Cuando se inicia una consulta en un bloque de datos, ya sea por el operador o por un subprograma incorporado, tienen lugar los siguientes eventos principales:

1. En el modo *Introducir Consulta*, Forms arranca el disparador **PRE-QUERY** si está definido.
2. Si el disparador **PRE-QUERY** es correcto, Forms crea la sentencia SELECT de consulta, basada en cualquier criterio existente en el bloque (bien introducido por el operador o por el disparador PRE-QUERY).
3. Se ejecuta la consulta.
4. Forms recupera los valores de columna de una fila en los elementos de tabla base de un nuevo registro del bloque.
5. El registro se marca como válido.
6. Forms arranca el disparador **POST-QUERY**. Si falla, este registro se vacía del bloque.
7. Si el registro ha cambiado (debido a un disparador), Forms realiza la validación de elementos y de registros.
8. Se repiten los pasos 4 a 7 para los demás registros restantes de esta consulta.

SUBPROGRAMAS 34

INTRODUCCIÓN

Dentro de Forms Builder se pueden crear los siguientes tipos de subprogramas:

- Paquetes.
- Procedimientos.
- Funciones.
- Disparadores.

Además de ellos existen una serie de subprogramas predefinidos, ya incorporados en Forms Builder, que se pueden utilizar. En este capítulo aprenderemos cuáles son y cómo utilizarlos.

VARIABLES DE FORMS BUILDER

Dentro de Forms Builder se pueden utilizar los siguientes tipos de variables**.**

Tipo variable	Objetivo	Sintáxis
Elementos	Presentación e interacción del usuario	:<nom_block>.<nom_item>
Variable global	Variable que permanece activa durante la sesión	:GLOBAL.<nom_variable>
Variable del sistema	Control y estado de pantalla	:SYSTEM.<nom_variable>
Parámetros	Transferencia de valores desde y hacia el módulo	:PARAMETER.<nombre>

Además de estas variables que se pueden invocar desde cualquier subprograma, también se pueden definir variables locales dentro de cada subprograma, declarándolas dentro de la sección DECLARE del mismo.

Ejemplo 1: Uso de variables de elementos

Las referencias a elementos deben llevar como prefijo el nombre del bloque de Forms Builder propietario, lo que evita la ambigüedad cuando existen elementos con el mismo nombre en bloques distintos:

```
:BLOQUE1.ID_PRODUCTO := :BLOQUE_PEDIDO.ID_PRODUCTO;
```

Ejemplo 2: Uso de variables globales

Las referencias a variables globales deben llevar la palabra :GLOBAL como prefijo. Se pueden crear como resultado de una asignación:

```
:GLOBAL.CLIENTE_ID := BLOQUE1.ID;
```

Ejemplo 3: Uso de variables del sistema

Las referencias a variables del sistema deben llevar como prefijo la palabra :SYSTEM, y el contenido debe estar en mayúsculas ('NORMAL', no 'normal'):

```
DECLARE
      V_salir_del_bloque      BOOLEAN;
BEGIN
      IF :SYSTEM.MODE = 'NORMAL' THEN
            V_salir_del_bloque := TRUE;
      END IF;
END;
```

Ejemplo 4: Uso de parámetros

Los parámetros definidos durante el diseño tienen el prefijo :PARAMETER.

```
IF :PARAMETER.punto_comiento = 2 THEN
      GO_BLOCK('BLOQUE2');
END IF;
```

Inicialización de variables globales por defecto

Puede utilizar la función incorporada DEFAULT_VALUE para asignar un valor a una variable global. Forms Builder crea la variable global si no existe. Si el valor de la variable indicada no es nulo, DEFAULT_VALUE no hace nada.

El siguiente ejemplo crea una variable global denominada PAIS y la inicializa con el valor 'TURQUÍA':

```
DEFAULT_VALUE( ' TURQUÍ A' , ' GLOBAL. PAI S' ) ;
```

Eliminación de variables globales

Puede utilizar la función incorporada ERASE para eliminar una variable global. Las variables globales siempre asignan 255 bytes de almacenamiento. Para asegurarse de que el rendimiento no se vea más afectado de lo necesario, borre siempre cualquier variable global cuando ya no se necesite.

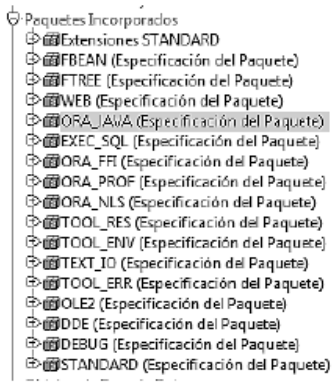
```
ERASE( ' GLOBAL. PAI S' ) ;
```

SUBPROGRAMAS INCORPORADOS

Forms Builder proporciona un conjunto de subprogramas predefinidos que se pueden utilizar en la programación de los formularios para aumentar la funcionalidad de los mismos. Estos subprogramas pertenecen a uno de los siguientes tipos de paquetes:

- **Extensiones de paquetes estándar:** estas funciones están integradas en el conjunto de comandos PL/SQL estándar de Forms Builder. Se pueden invocar directamente, sin ningún prefijo de paquete. Se pueden utilizar más de cien funciones incorporadas estándar y a esta categoría pertenecen todas las Built-in que se han mencionado a lo largo de este manual. Por ejemplo: EXECUTE_QUERY;
- **Otros paquetes de Forms Builder:** los subprogramas de otros paquetes incorporados ofrecen funcionalidad relacionada con una función particular soportada y requieren el nombre del paquete como prefijo cuando se llaman. Por ejemplo: ORA_J AVA. CLEAR_EXCEPTI ON.

Lista de paquetes incorporados



Para acceder a la lista de paquetes incorporados dentro del navegador de objetos de Forms Builder, tenemos que desplegar el grupo **Paquetes Incorporados** y se mostrará esta lista.

En el primer elemento (**Extensiones STANDARD**) se encuentran todas las Built-in que permite utilizar Forms Builder sin predicado de paquete, y justo debajo de este elemento se relacionan el resto de paquetes incorporados que si requieren un descriptor de paquete.

En el último elemento (**STANDARD**) se incluyen todas las funciones del lenguaje SQL y PL/SQL y que tampoco requieren ningún descriptor de paquete para ser utilizadas en la programación dentro de Forms Builder.

A continuación se describe la utilidad del resto de paquetes incorporados que sí requieren de un nombre de descriptor de paquete para su invocación:

Paquete	Descripción
DDE	Proporciona soporte de DDE (Intercambio dinámico de datos). Por ejemplo para compartir datos entre un formulario de Oracle y un documento de Microsoft.
DEBUG	Proporciona funciones incorporadas para depurar unidades de programa PL/SQL.
EXEC_SQL	Proporciona funciones incorporadas para ejecutar SQL dinámico en procedimientos PL/SQL.
FBEAN	Proporciona funciones incorporadas para interactuar con Java Beans del cliente.
FTREE	Proporciona funciones incorporadas para manipular elementos de árboles jerárquicos.
OLE2	Proporciona una API PL/SQL para la creación, manipulación y acceso a los atributos de objetos de automatización OLE2.
ORA_FFI	Proporciona funciones incorporadas para llamar a funciones © ajenas desde PL/SQL
ORA_J AVA	Permite llamar a procedimientos Java desde PL/SQL.
ORA_NLS	Permite extraer información de alto nivel acerca del entorno de idioma actual.
ORA_PROF	Proporciona funciones incorporadas para ajustar unidades de programa PL/SQL.
TEXT_I O	Proporciona funciones incorporadas para escribir información en archivos y leer información de ellos.

TOOL_ENV	Permite interactuar con variables de entorno Oracle.
TOOL_ERR	Permite acceder y manipular la pila de error creada por otros paquetes incorporados como, por ejemplo, Debug
TOOL_RES	Proporciona funciones incorporadas para manipular archivos de recursos.
WEB	Proporciona funciones incorporadas para el entorno Web.

Algunos de estos paquetes como, por ejemplo OLE2, ORA_FFI y TEXT_IO, funcionan en el servidor de aplicaciones, no en el cliente. Por ejemplo TOOL_ENV le permite obtener y definir variables de entorno en el servidor de aplicaciones. Para interactuar con el cliente, necesitaría ofrecer una funcionalidad similar en un JavaBean.

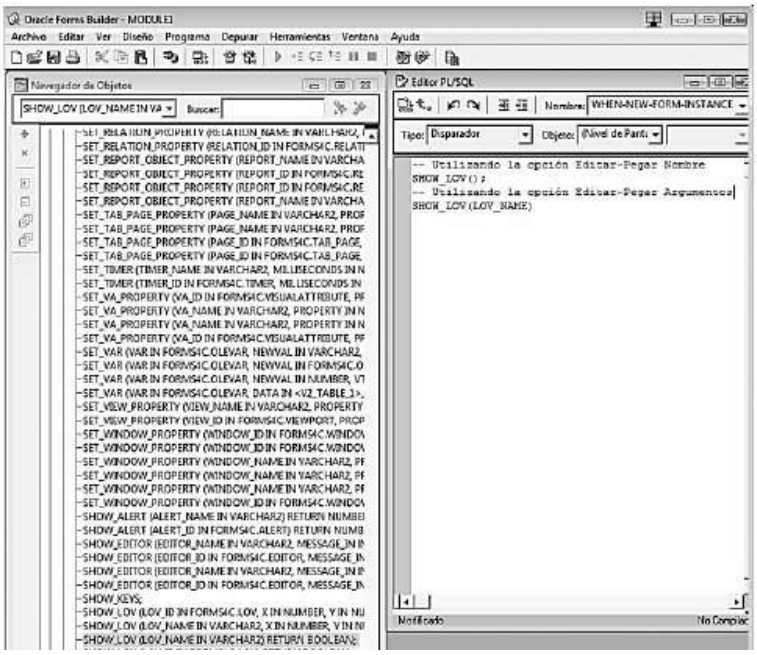
LÍMITES DE USO DE LOS SUBPROGRAMAS INCORPORADOS

Las funciones y los paquetes incorporados se pueden invocar en cualquier disparador o subprograma diseñado por el usuario, en el que se utilice PL/SQL. Sin embargo, algunas funciones incorporadas ofrecen funcionalidades que no se permiten en determinados tipos de disparadores. Las funciones incorporadas se dividen, por tanto, en dos grupos:

- **Funciones incorporadas no restringidas:** estas funciones no están restringidas porque no afectan a la navegación lógica o física y se pueden invocar desde cualquier disparador o desde cualquier subprograma.
- **Funciones incorporadas restringidas:** estas funciones afectan a la navegación en la pantalla, ya sea navegación de pantalla externa o navegación interna. Se pueden invocar estas funciones solo desde disparadores cuando no se produce ninguna navegación interna. La ayuda de Forms Builder especifica si una función incorporada tiene alguna restricción.

La llamada a una función incorporada restringida desde un disparador de navegación se compila correctamente, pero provoca un error en tiempo de ejecución. Por ejemplo, puede compilar un disparador PRE-TEXT-ITEM que llama a la función incorporada restringida GO_ITEM, pero cuando el disparador se arranca en tiempo de ejecución, genera el error FRM-40737.

USAR DEFINICIONES DE FUNCIONES INCORPORADAS



Cuando se escribe el código PL/SQL de un disparador o unidad de programa, Forms Builder le permite buscar las definiciones de las funciones incorporadas y, opcionalmente, copiar su nombre y esqueleto de argumentos necesarios en el código. A continuación se muestra un ejemplo de ello:

Para conseguir el resultado del ejemplo hay que seguir estos pasos:

1. En un nuevo formulario nos situamos en el grupo **Disparadores** a nivel de formulario y creamos el trigger WHEN-NEW-FORM-INSTANCE, lo que provocará que se abra el editor PL/SQL.
2. Mantenemos abiertas las ventanas del editor PL/SQL del disparador y del navegador de objetos.
3. A continuación desplegamos el grupo **Paquetes Incorporados** dentro del Navegador de Objetos, y a su vez, dentro de este, el grupo **Extensiones STANDARD.**
4. Seguidamente seleccionamos la Built-in `SHOW_LOV(LOV_NAME IN VARCHAR2) RETURN BOOLEN` y pulsamos en el menú principal la opción **Editar – Pegar Nombre**, lo que provocará que en el editor PL/SQL del disparador se escriba la instrucción `SHOW_LOV()`.
5. A continuación volvemos al navegador de objetos y señalamos otra vez la misma Built-in, pero ahora dentro del menú **Editar** seleccionamos la opción **Pegar Argumentos,** lo que provocará que en el editor PL/SQL del disparador se escriba la instrucción `SHOW_LOV(LOV_NAME)`.

FUNCIONES INCORPORADAS DE USO HABITUAL

La tabla siguiente describe algunas funciones incorporadas que se pueden utilizar en disparadores para agregar funcionalidad a los elementos.

Subprograma incorporado	Descripción
Procedimiento EDIT_TEXTITEM	Llama al editor de elementos de Form Runtime para el elemento de texto actual.
Procedimiento ENTER_QUERY	Borra el bloque actual y crea un registro de ejemplo. Los operadores pueden especificar luego las condiciones de consulta antes de ejecutar la consulta con un comando de botón o menú. Si hay cambios que confirmar, Forms Builder solicita al operador que los confirme antes de continuar el procesamiento de ENTER_QUERY.
Procedimiento EXECUTE_QUERY	Borra el bloque actual, abre una consulta y recupera una serie de registros seleccionados. Si hay cambios que confirmar, Forms Builder solicita al operador que los confirme antes de continuar el procesamiento EXECUTE_QUERY.
Procedimiento EXIT_FORM	En el modo normal, sale de la pantalla actual; en el modo Introducir Consulta, cancela la consulta.
Función GET_ITEM_PROPERTY	Devuelve valores de propiedad especificados para el elemento indicado.
Procedimiento GO_BLOCK	Navega al bloque especificado.
Procedimiento GO_ITEM	Navega al elemento especificado.
Procedimiento HIDE_VIEW	Oculta el lienzo indicado.
Procedimiento LIST_VALUES	Llama a la lista de valores adjunta al elemento actual.
Procedimiento MESSAGE	Muestra texto especificado en la línea de mensajes.
Procedimiento SET_ITEM_PROPERTY	Cambia los valores de la propiedad especificada para un elemento.
Función SHOW_ALERT	Muestra la alerta dada y devuelve un valor numérico cuando el operador selecciona uno de los tres botones de alerta.
Procedimiento SHOW_EDITOR	Muestra el editor especificado en las coordenadas dadas y transfiere una cadena al editor, o bien recupera una cadena existente del editor.
Función SHOW_LOV	Llama a una lista de valores especificada y devuelve un valor booleano que indica si el usuario ha seleccionado un valor de la lista.
Procedimiento SHOW_VIEW	Muestra el lienzo indicado en las coordenadas especificadas por los valores de propiedad del lienzo: X Position e Y Position. Si ya se muestra la vista, SHOW_VIEW la coloca delante de cualquier otra vista de la misma pantalla.

EL PROCESO DE DEPURACIÓN 35

INTRODUCCIÓN

Cuando se empieza a agregar código a las pantallas, escribiendo disparadores o unidades de programa, probablemente habrá ocasiones en las que obtendrá resultados incorrectos o inesperados.

En una aplicación grande, puede resultar difícil determinar el origen del problema, y para facilitar la búsqueda de estos problemas, existe el depurador PL/SQL que incorpora Forms Builder. En este capítulo se explica su funcionamiento.

EL PROCESO DE DEPURACIÓN

Dentro de Forms Builder se puede controlar y depurar el código PL/SQL de varias formas:

- **Compilación:** al ejecutar este método, Forms Builder verifica la sintaxis del código PL/SQL, informando de los posibles errores en la sintaxis y/o referenciación a objetos. Esto permite corregir estos problemas en el editor PL/SQL antes de ejecutar el formulario.
- **Ejecución de una pantalla con el parámetro de tiempo de ejecución:** en el modo de depuración, se puede solicitar que se muestren mensajes para indicar cuándo se arrancan los distintos disparadores incluidos en una pantalla. Esto ayuda a ver si determinados disparadores se arrancan, su origen y su nivel, y la hora a la que se arrancan. Si se produce un error, se puede mirar el último disparador que se arrancó para limitar el ámbito del origen del error. Para conseguir esta funcionalidad hay que marcar la

opción **Mensajes de Depuración** dentro de la solapa **Ejecución** de las preferencias del Forms Builder. Para acceder a estas preferencias hay que pulsar en el menú **Editar** y a continuación en **Preferencias.**

- **Llamada al depurador PL/SQL:** cuando se quiere ejecutar el depurador de Forms Builder para que el formulario se ejecute paso a paso hay que pulsar en el menú **Depurar** y a continuación en **Depurar Módulo** o bien pulsar en el siguiente icono  de la barra de herramientas:

Con el depurador se puede controlar la ejecución de código de un disparador y otras unidades de programa. Se puede desplazar en la ejecución por el código, línea a línea y conforme lo hace, puede controlar los subprogramas y las variables a las que se llama. También puede modificar variables mientras se ejecuta la pantalla, lo que le permite probar cómo afectarán a la aplicación diversos cambios en los valores de elementos de pantalla y los valores de variables.

LA CONSOLA DE DEPURACIÓN

La consola de depuración es un espacio de trabajo en Forms Builder que permite ver diversos aspectos de la pantalla en ejecución y su entorno. Para ver la consola de depuración hay que seleccionar dentro del menú **Depurar** la opción **Consola de Depuración**. También aparecerá automáticamente cuando se encuentre un punto de ruptura en un formulario que se esté ejecutando en el modo depuración.

Con la consulta de depuración se pueden mostrar los siguientes tipos de paneles:

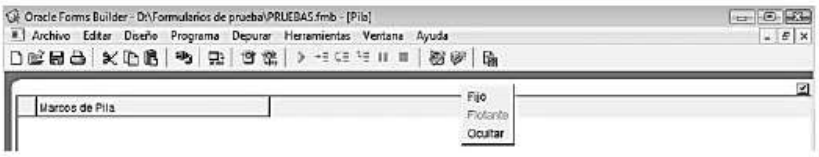
Icono	Nombre
	Pila
	Variables
	Observar
	Valores de Pantalla
	Paquetes PL/SQL
	Variables del Sistema/Globales
	Puntos de Ruptura

Como cualquier otra pantalla incluida en Forms Builder, es posible cambiar el tamaño de la consola de depuración y de los paneles mostrados en ella.

Para mostrar qué paneles se quieren mostrar o cerrar, hay que hacer clic en cada uno de los botones de conmutación de la barra de herramientas de la consulta de depuración (que se han indicado en el cuadro anterior), o bien en el menú **Depurar.** A continuación hay que seleccionar la opción **Ventanas de Depuración,** e ir marcando cada una de las ventanas que queremos mostrar u ocultar. Mientras muestra y oculta los paneles en la consola, estos se amplían y contraen automáticamente para rellenarla.

Para anular la fijación de cualquiera de los paneles y así poderlo mostrar fuera de la consola de depuración, hay que hacer clic en la fecha hacia arriba  que hay en la parte derecha de cada marco de ventana de depuración:

Y para volver a fijar el panel dentro de la consulta de depuración, hay que pulsar en la flecha hacia abajo: 



Si hace clic con el botón derecho del ratón sobre el área junto a la flecha de fijación/anulación de fijación, verá un menú emergente que le permite ocultar, fijar o anular la fijación del panel como se muestra en esta imagen.

Panel Pila

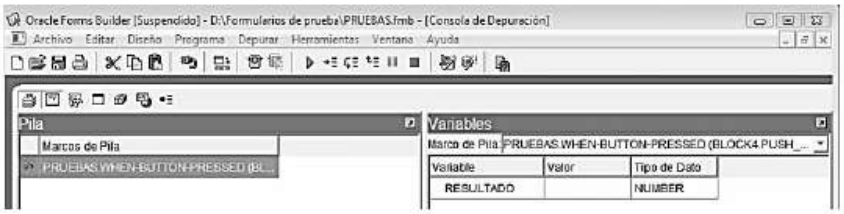
La pila de llamada representa la cadena de subprogramas que comienza desde el punto de entrada inicial hasta el subprograma actualmente en ejecución.

En el panel de pila se muestran los subprogramas ejecutados en orden inverso. El subprograma inicial se coloca en la parte final de la pila, mientras que el subprograma en el que se encuentra actualmente el proceso de depuración se coloca al principio de la pila.

Panel Variables

El panel de variables muestra las variables del marco de pila actual, junto a sus valores. Hay una lista emergente desde la que se puede seleccionar el marco de pila cuyas variables se desean ver. También puede cambiar las variables mostradas en el panel de variables haciendo clic en un marco de pila distinto en el panel de pila, si está abierto también. Esto no cambia el orden de ejecución de las sentencias del programa, sino solo la información mostrada en la consola de depuración.

Una función del depurador es que cuando se suspende la pantalla se pueden cambiar los valores de variables haciendo clic en la columna de valor e introduciendo un valor nuevo. Cuando el programa continúa, utiliza el nuevo valor que se haya introducido, de forma que puede probar el efecto que tiene cambiar un valor en el resultado final.



Algunas variables, como por ejemplo los parámetros, no se pueden modificar. Cuando se hace clic en una variable de solo lectura, se resalta toda la celda. Al hacer clic en una variable modificable, se resalta solo el valor, no toda la celda. En esta imagen se muestra un ejemplo de este panel.

Panel Observar

Una aplicación en ejecución puede tener docenas de variables que se pueden mostrar en diversos paneles en la consola de depuración, pero puede que haya muy pocas que se tengan que controlar. El panel Observar proporciona un lugar central desde el que se puede realizar el seguimiento de cualquier variable válida que se especifique. Las variables de paquetes almacenados no son válidas.

Una vez que la variable se muestra en la lista de comprobaciones, cuando se vuelve a suspender la ejecución en una ubicación diferente, los valores de variables de la lista se actualizan como sea necesario si están disponibles en el contexto de ejecución actual. Si una variable no está definida en el subprograma actualmente en ejecución, en la celda se muestra #### en lugar de un valor.

En esta imagen se muestra un ejemplo de este panel. Para agregar una variable en el **panel Observar**, realice los siguientes pasos:

1. Abra la ventana en la que se muestra la variable.
2. Indique en la primera columna el nombre de la variable.

Para suprimir un elemento del **panel Observar**:

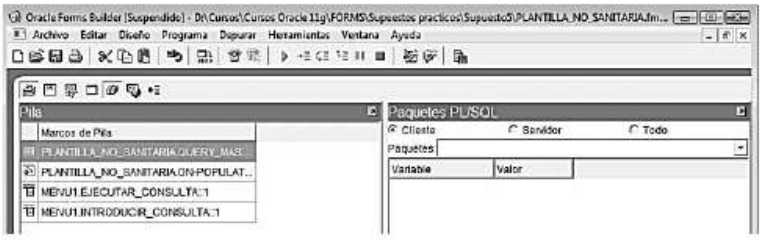
- Selecciónelo en el panel Observar, haga clic con el botón derecho y seleccione Eliminar en el menú emergente.
- Para borrar toda variable del panel Observar, seleccione Eliminar Todo.

Panel Valores de Pantalla

Puede utilizar el panel de valores de pantalla para mostrar los valores de todos los elementos y parámetros de módulos que están actualmente en ejecución. Para cambiar entre una vista de elementos y una de parámetros, haga clic en los separadores correspondientes en el panel.

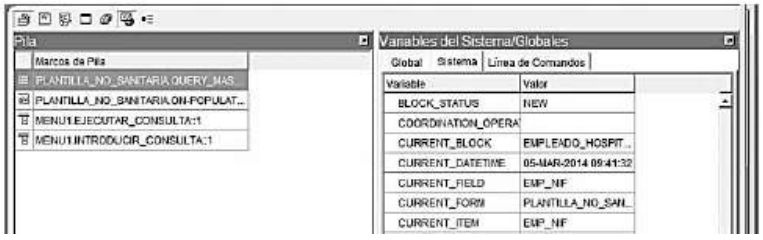
Puede cambiar los valores de los elementos modificables para probar los efectos de tales cambios. Si el valor es de solo lectura, como por ejemplo los valores de elementos mostrados, se resalta toda la celda al intentar editarla. Si el valor es modificable, solo se resalta el valor de la celda cuando se selecciona.

Panel Paquetes PL/SQL



Utilice el panel PL/SQL para examinar los paquetes PL/SQL que se han instanciado. Aparecen tanto las variables globales de la especificación del paquete como las del cuerpo del paquete. Solo puede ver los paquetes mientras el proceso de ejecución de pantalla esté ejecutando actualmente PL/SQL. Puede seleccionar también un botón de opción para determinar qué paquetes se visualizan: Cliente, Servidor o Todos.

Panel Variables del Sistema/Globales



Utilice este panel para mostrar las variables del sistema, globales y de línea de comandos actuales, así como sus valores. Para cambiar entre estos tipos de variables, haga clic en los separadores correspondientes en el panel.

Las variables de línea de comandos y la mayoría de las variables del sistema son de solo lectura. Las únicas variables del sistema modificables son DATE_THRESHOLD, EFFECTIVE_DATE, MESSAGE_LEVEL y SUPPRESS_WORKING. Puede modificar las variables globales, y la aplicación en ejecución utilizará posteriormente los nuevos valores.

Panel Puntos de Ruptura

Este panel consta de dos separadores:

- El separador **Puntos de Ruptura** muestra cualquier punto de ruptura definitivo en el código durante la sesión actual de Forms Builder por orden de creación de los puntos de ruptura.

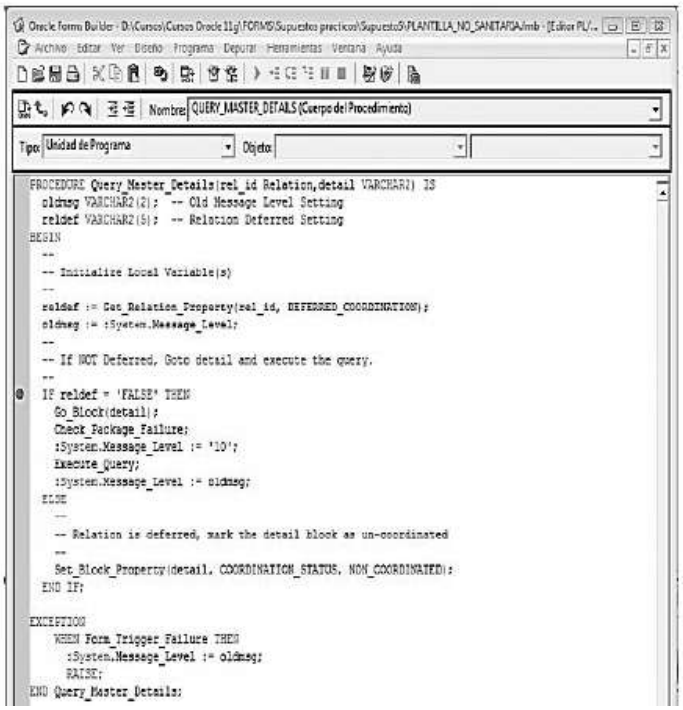
  La visualización incluye:
  - Nombre del disparador o unidad de programa.
  - Número de línea en que está definido el punto de ruptura.
  - Una casilla de control para activar o desactivar temporalmente el punto de ruptura.

- Permite la navegación al código de origen donde se ha definido el punto de ruptura:
  - Al hacer clic dos veces en el nombre del punto de ruptura.
  - Al resaltarlo (haciendo clic una vez) y seleccionar a continuación **Ver Código Fuente** en el menú emergente (hacer clic con el botón derecho del ratón). Desde este menú emergente también puede eliminar el punto de ruptura o todos los puntos de ruptura.

- El separador **Break On Exceptions** muestra una lista de excepciones del sistema utilizadas frecuentemente que puede monitorizar durante la depuración. La visualización incluye:
  - Nombre de excepción.
  - Número de error ORA-asociado.
  - Casilla de control donde se define o se anula la definición del punto de ruptura.

DEFINICIÓN DE PUNTOS DE RUPTURA

Los puntos de ruptura se definen en el código PL/SQL del formulario. Cuando se ejecuta dicho formulario en modo depuración y se llega hasta el punto señalado, la ejecución del formulario se suspende y se muestra la ventana de depuración, lo que permite controlar o cambiar el entorno de dicho punto. Se pueden incorporar tantos puntos de ruptura como desee el programador.



Adjunto se muestra un ejemplo de un formulario al que se le ha añadido un punto de ruptura (círculo que aparece a la izquierda del código PL/SQL).

Cuando se define un punto de ruptura, este permanece activo hasta que se sale de la sesión de Forms Builder. No obstante, puede desactivar y activar los puntos de ruptura según sea necesario, desactivando y volviendo a activar la casilla de control en el panel Puntos de Ruptura de la consola de depuración.

Puede definir los puntos de ruptura solo en líneas de código ejecutable como, por ejemplo, asignaciones o llamadas a subprogramas. Un punto de ruptura se puede definir de tres formas:

- Haciendo clic dos veces a la izquierda de una línea de código en el editor PL/SQL.
- Haciendo clic con el botón derecho en una línea de código y seleccionando **Insertar/Eliminar Punto de Ruptura**.
- Seleccionando **Depurar** y dentro de este menú la opción **Insertar/Eliminar Punto de Ruptura**.

Al ejecutar la misma acción otra vez, se anula la definición del punto de ruptura. También puede eliminar uno o todos los puntos de ruptura desde el menú emergente en el panel Punto de Ruptura, tal como se ha descrito anteriormente.

Si dentro de Forms Builder hemos abierto una conexión a la base de datos, se pueden llegar a definir puntos de ruptura en funciones, procedimientos y paquetes de base de datos almacenados. Para ello debe realizar los siguientes pasos:

1. Amplíe el elemento **Objetos de Base de Datos** y a su vez amplíe el usuario de base de datos al que quiera acceder para ver los objetos almacenados.
2. Seleccione el objeto que quiera abrir.
3. Ábralo en el Editor PL/SQL y marque un punto de ruptura.

Los puntos de ruptura no se pueden definir en disparadores de base de datos ni bibliotecas PL/SQL almacenadas.

EJECUTAR UN FORMULARIO EN MODO DEPURACIÓN

Para poder ejecutar un formulario en el modo depuración se pueden utilizar dos mecanismos:

1. Con un formulario abierto en Forms Builder, seleccionamos la opción **Depurar** y dentro de ella **Depurar Módulo.**
2. La segunda opción es pulsar el botón 

Una vez ejecutado el formulario en modo depuración, la ejecución del mismo se para cuando encuentra un punto de ruptura. En ese instante se pueden utilizar los comandos de depuración para reanudar la ejecución o desplazarse por el código, de varias formas, para ver cómo afecta cada línea a la aplicación.

Desplazamiento por el código

Cuando el depurador encuentra un punto de ruptura se para la ejecución y se habilitan una serie de funciones para desplazarnos por el programa, o bien a través de la botonadura que se muestra en la imagen siguiente, o bien a través del menú **Depurar.**

A continuación se muestra una tabla con el significado de cada uno de estos botones, así como su equivalencia en el menú **Depurar**:

Botón	Menú	Descripción
	Depurar- Ir	Reanuda la ejecución hasta que el programa termina normalmente o lo interrumpe el siguiente punto de ruptura.
	Depurar – Ir a primera línea ejecutable	Ejecuta la sentencia siguiente dentro del código que se abre en depuración cuando se encuentra el punto de ruptura.
	Depurar – Ir a siguiente línea de código	Ejecuta la sentencia siguiente sin desplazase a un subprograma anidado, saltando el bucle de control (si lo hubiese).
	Depurar – Salir de Método	Finaliza el subprograma anidado y se desplaza a la sentencia ejecutable siguiente en el programa que realiza la llamada.
	Depurar – Pausa	Realiza una pausa en la ejecución del código PL/SQL en ejecución, para permitir que el programa examine el entorno de ejecución. Por ejemplo para comprobar los valores de las variables.
	Depurar - Parar	Termina la depuración y la ejecución del programa por completo; la consola de depuración y cualquier panel de depuración abierto se cierran y termina la ejecución de la aplicación.

EL PROCESO DE VALIDACIÓN 36

INTRODUCCIÓN

Forms realiza un proceso de validación a varios niveles para asegurar que los registros y los valores individuales siguen las reglas adecuadas. Si la validación falla, el control se vuelve a transferir al nivel adecuado, de forma que el operador puede realizar correcciones. La validación se produce a:

- **Nivel de elemento:** Forms registra un estado para cada elemento, con objeto de determinar si es válido actualmente. Si un elemento se ha cambiado y no está marcado aún como válido, Forms realiza primero comprobaciones de validación estándar para asegurarse de que el valor cumple las propiedades del elemento. Estas comprobaciones se llevan a cabo antes de arrancar cualquier disparador WHEN-VALIDATE-ITEM que se haya definido. Entre las comprobaciones estándar se incluyen las siguientes propiedades de un elemento:
    - Máscara de formato.
    - Necesaria (si es así, entonces ¿es nulo el elemento?)
    - Tipo de datos.
    - Rango (Valor Menor-Mayor Permitido)
    - Validar desde la Lista.
- **Nivel de registro:** después de salir de un registro, Forms comprueba si el registro es válido. Si no, se comprueba el estado de cada elemento del registro y se arranca un disparador WHEN-VALIDATE-RECORD, si está presente. Cuando el registro transfiere estas comprobaciones, se define un válido.

- **Nivel de bloque y de pantalla:** a nivel de bloque o de pantalla se validan todos los registros que haya por debajo de dicho nivel. Por ejemplo, si se confirman (guardan) los cambios en la pantalla, se validan todos los registros de la pantalla, a menos que haya suprimido esta acción.

CUÁNDO SE PRODUCE LA VALIDACIÓN

Forms lleva a cabo la validación de la unidad de validación en las siguientes condiciones:

- Se ha pulsado la tecla ENTER o se ha ejecutado el procedimiento de función incorporada ENTER. El objetivo de la función incorporada ENTER es forzar la validación inmediatamente.
- El operador o un disparador navegan fuera de la unidad de validación. Esto incluye la confirmación de los cambios. La unidad de validación por defecto es el elemento, pero el disparador también la puede definir en el registro, el bloque o la pantalla. La unidad de validación se analiza en la sección siguiente.

USO DE PROPIEDADES DE OBJETOS PARA CONTROLAR LA VALIDACIÓN

Puede controlar cuándo y cómo se produce la validación en una pantalla, incluso sin disparadores. Para ello, deberá definir propiedades para la pantalla y para los elementos individuales en su interior.

La unidad de validación

La unidad de validación define la cantidad máxima de datos que un operador puede introducir en la pantalla antes de que Forms inicie la validación. **Validation Unit** es una propiedad del módulo de pantalla y se puede definir en la paleta de propiedades en uno de los siguientes valores:

- Default.
- Item.
- Record.
- Block.
- Form.

El valor por defecto es el nivel de elemento. Normalmente se elige dicho valor.

En la práctica, una unidad de validación a nivel de elemento significa que Forms valida los cambios cuando un operador navega fuera de un elemento cambiado. De esta forma, se pueden realizar inmediatamente las comprobaciones de validación estándar y el arranque del disparador WHEN-VALIDATE-ITEM de dicho elemento. En consecuencia, los operadores conocen los fallos de validación en cuanto intentan salir del elemento.

En las unidades de validación superiores (nivel de registro, bloque o pantalla), las comprobaciones anteriores se posponen hasta que la navegación sale de dicha unidad. Todos los elementos y registros pendientes se validan juntos, incluido el arranque de disparadores WHEN-VALIDATE-ITEM y WHEN-VALIDATE-RECORD.

Debe definir una unidad de validación por encima del nivel de elemento en una de las siguientes circunstancias:

- La validación implica referencias de base de datos y desea posponer el tráfico hasta que el operador haya completado un registro (nivel de registro).
- Para mejorar el rendimiento reduciendo los recorridos de ida y vuelta al servidor de aplicaciones.

USO DE LISTAS DE VALORES PARA VALIDACIÓN

Cuando se utiliza una lista de valores asociada a un elemento del formulario, opcionalmente se puede utilizar el contenido de la lista de valores para validar los datos introducidos en dicho elemento. Para hacer esto hay que definir a **SÍ** la propiedad **Validar desde la Lista** del elemento que tiene asociada la lista de valores.

En el momento de la validación Forms utiliza automáticamente el valor del elemento como una cadena de búsqueda no sensible a mayúsculas/minúsculas en el contenido de la lista de valores. A continuación, dependiendo de las circunstancias, se producen los siguientes eventos:

- Si el valor del elemento de texto coincide con uno de los valores de la primera columna de la lista de valores, la validación es correcta y la lista de valores no se mostrará, continuando el procesamiento normalmente.
- Si el valor del elemento hace que se encuentre un único registro en la lista de valores, pero es un valor parcial del valor de la lista de valores,

entonces se devuelve sobre el elemento, el valor completo de la columna de la lista de valores (siempre que dicho elemento esté definido como elemento de retorno en la lista de valores). De esta forma, el elemento supera la fase de validación con éxito.

- Si el valor del elemento hace que se encuentren varios registros en la lista de valores, Forms muestra la lista con los valores encontrados que se asemejen al indicado en el elemento, y deja que sea el usuario el que seleccione el correcto.
- Si no se encuentra ninguna coincidencia, se muestra la lista completa de valores para que el usuario seleccione uno.

Cuando diseñe una lista de valores con fines de validación, debe tener en cuenta que la primera columna debe ser la columna de la validación y tener un ancho de visualización mayor de 0. Igualmente deberá definir dicha columna como elemento de retorno y asociarla al elemento del formulario en el que se vaya a validar.

Por razones de rendimiento se desaconseja utilizar lista de valores con la propiedad **Validar desde la Lista = SI**, cuando dichas listas tengan un gran volumen de valores.

CONTROL DE VALIDACIÓN MEDIANTE DISPARADORES

Hay disparadores que se arrancan debido a la validación: WHEN-VALIDATE-ITEM y WHEN-VALIDATE-RECORD, son dos de estos casos. En dichos disparadores el desarrollador puede programar acciones que se deben realizar en la validación.

When-Validate-Item

Este disparador se aplica a nivel de elemento y se arranca después de la validación estándar del elemento y el foco de entrada se devuelva al mismo si el disparador falla.

Imaginemos el campo FECHA_PEDIDO del bloque PEDIDOS, de tipo fecha, en el que definimos el siguiente código en el disparador WHEN-VALIDATE-ITEM:

```
IF :PEDIDOS.FECHA_PEDIDO > SYSDATE THEN
   MESSAGE('La fecha del pedido es mayor a la fecha de hoy');
   RAISE FORM_TRIGGER_FAILURE;
END IF;
```

Cuando el cursor de nuestra aplicación salga del elemento FECHA_PEDIDO del bloque PEDIDOS, se realizará la validación estándar: comprueba que el campo es de tipo fecha y que la fecha es correcta (dentro del calendario). Seguidamente se arrancará el disparador WHEN-VALIDATE-ITEM y comprobará que el campo FECHA_PEDIDO tiene una fecha inferior o igual a la fecha del sistema (SYSDATE). Si no es así, se mostrará un mensaje en la barra de estado y se abortará el salto del cursor al siguiente campo (RAISE FORM_TRIGGER_FAILURE), quedándose el mismo en el elemento FECHA_PEDIDO hasta que se introduzca una nueva fecha y se valide.

When-Validate-Record

Este disparador se aplica a nivel de registro y se arranca después de una validación estándar a nivel de registro, cuando el cursor ha salido de un registro nuevo o saltado a otro registro existente. Dado que Forms ya ha comprobado que los elementos necesarios para el registro son válidos, puede utilizar este disparador para realizar comprobaciones adicionales que pueden implicar a más de un elemento del registro, en el orden en que se introdujeron.

Este disparador se debe definir a nivel de bloque o superior.

SEGUIMIENTO DEL ESTADO DE VALIDACIÓN

Cuando el cursor de Forms sale de un objeto del formulario, normalmente se validan los cambios realizados en el contenido del mismo. Para determinar si se debe llevar a cabo la validación, Forms realiza el seguimiento del estado de validación de elementos y registros.

Estado de validación de un Item

Los estados de validación de un elemento de tipo ITEM son los siguientes:

Estado	Definición
NEW	Cuando se crea un registro, Forms marca como nuevo cada elemento (ITEM) del registro. Esto es así incluso si el elemento se rellena mediante las propiedades del elemento **Copiar Valor de Elemento** o **Valor Inicial**, o mediante el disparador WHEN-CREATE-RECORD.
CHANGED	Forms marca un elemento como cambiado en las siguientes condiciones: • Cuando el usuario o un disparador cambia el contenido del elemento. • Cuando se cambia el contenido de cualquier elemento (ITEM) en un registro nuevo, todos los elementos de dicho registro se marcan como cambiados.

VALI D	Forms marca un elemento como válido en las siguientes condiciones: • Todos los elementos (ITEM) de un registro que se recuperan de la base de datos se marcan como válidos. • Si la validación del elemento ha sido correcta. • Después de un envío o una confirmación correctos. • Cada elemento de un registro duplicado hereda el estado de su origen.

Estado de validación de un Registro

Los estados de validación de un elemento de tipo RECORD (registro) son los siguientes:

Estado	Definición
NEW	Cuando se crea un registro, Forms marca dicho registro como nuevo. Esto es así incluso si los elementos del registro se rellenan mediante las propiedades **Copiar Valor de Elemento** o **Valor Inicial**, o mediante el disparador WHEN-CREATE-RECORD.
CHANGED	Cuando un elemento de un registro se marca como cambiado, Forms marca el registro entero como cambiado.
VALI D	Forms marca un registro como válido en las siguientes condiciones: • Despúes de que se hayan validado correctamente todos los elementos del registro. • Todos los registros que se recuperan de la base de datos se mancan como válidos. • Después de un envío o una confirmación correctos. • Un registro duplicado hereda el estado de su origen.

CONTROL DE LA VALIDACIÓN CON FUNCIONES INCORPORADAS

Se pueden utilizar los siguientes subprogramas incorporados en disparadores para que afecten a la validación:

- CLEAR_BLOCK
- CLEAR_FORM
- EXIT_FORM
- ITEM_IS_VALID
- ENTER
- SET_FORM_PROPERTY
- VALIDATE

Clear_Block, Clear_Form y Exit_Form

El primer parámetro de estas funciones incorporadas (COMMIT_MODE) controla qué se hará con los cambios no aplicados cuando se borra un bloque (CLEAR_BLOCK), se borra la pantalla (CLEAR_FORM) o se sale de la misma (EXIT_FORM), respectivamente. Cuando el parámetro se define en NO_VALIDATE, los cambios no se validan ni confirman (por defecto, se le solicita dicha acción al usuario).

Item_is_valid

Puede utilizar funciones incorporadas como GET_ITEM_PROPERTY y SET_ITEM_PROPERTY con el parámetro ITEM_IS_VALID para obtener o definir el estado de validación de un elemento (ítem), pero no puede obtener ni definir directamente el estado de validación de un registro entero.

Enter

La función incorporada ENTER realiza la misma acción que la tecla Enter del teclado del ordenador. Es decir, fuerza la validación de los datos en la unidad de validación actual donde se encuentre el cursor.

Set_Form_Property

Se puede utilizar esta función incorporada para desactivar la validación de Forms. Por ejemplo, suponga que está probando un formulario y necesita ignorar la validación normal del mismo; puede entonces definir la propiedad VALIDATION en esta función con False, de la siguiente forma:

```
SET_FORM_PROPERTY('nombre_formulario', VALIDATION,
                  PROPERTY_FALSE);
```

También puede utilizar esta función incorporada para cambiar la unidad de validación mediante programación:

```
SET_FORM_PROPERTY('nombre_formulario', VALIDATION_UNIT,
                  scope);
```

Validate

VALIDATE(scope) fuerza a Forms a ejecutar inmediatamente el procesamiento de validación para el ámbito indicado. Dicho ámbito puede ser DEFAULT_SCOPE, BLOCK_SCOPE, RECORD_SCOPE o ITEM_SCOPE.

NAVEGACIÓN 37

INTRODUCCIÓN

Forms ofrece diversas formas de controlar el movimiento del cursor. Este capítulo se centra en los distintos métodos para forzar la navegación, tanto de manera visible como no visible.

Qué es la unidad de navegación

La unidad de navegación es un objeto interno e invisible que determina el estado de navegación de una pantalla. Forms utiliza la unidad de navegación para realizar un seguimiento del objeto donde se encuentra actualmente el foco de un proceso de navegación. La unidad de navegación puede ser uno de los objetos de la siguiente jerarquía:

- Fuera de la pantalla.
- Pantalla (form).
- Bloque (block).
- Registro (record).
- Elemento (ítem).

Cuando Forms navega, cambia la unidad de navegación desplazándose a través de esta jerarquía de objetos hasta que se alcanza el objetivo.

Entrada y salida de objetos

Durante la navegación, Forms sale y entra en los distintos objetos del formulario. La entrada en un objeto supone el cambio de la unidad de navegación desde el objeto por encima en la jerarquía. La salida de un objeto significa el cambio de la unidad de navegación hasta el objeto situado por debajo.

El cursor y cómo se relaciona con la unidad de navegación

El cursor es un objeto externo y visible que indica el foco de entrada actual. Forms no moverá el cursor hasta que la unidad de navegación se haya convertido correctamente en el elemento de destino.

Qué ocurre si falla la navegación

Si la navegación falla, Forms revierte la ruta de navegación e intenta mover la unidad de navegación de vuelta a su ubicación inicial. Observe que el cursor sigue en su posición inicial. Si Forms no puede mover la unidad de navegación de vuelta a su ubicación inicial, sale de la pantalla.

LA NAVEGACIÓN INTERNA

La navegación se produce cuando el usuario o un disparador hacen que el foco de entrada se desplace de un objeto a otro. Ha visto que la navegación implica cambiar la ubicación física del cursor en la pantalla. Además de esta navegación visible, también tiene lugar la navegación lógica. A esta navegación se le conoce como navegación interna.

Pongamos como ejemplo un formulario que consta de un único bloque, un registro con varias filas y varios elementos (ítem) dentro del registro. La navegación interna se produciría de la siguiente forma si no hay errores:

- Entrada en la pantalla.
- Entrada en el bloque.
- Entrada en el registro.
- Entrada en el primer elemento del registro.

A medida que se vaya validando la información y se produzcan inserciones, actualizaciones o supresiones en los elementos espejo de la base de datos, se producirá también el movimiento inverso, sin que el foco de entrada se vea que se haya movido, pero internamente si se producen los siguientes eventos de navegación interna:

- Salir de último elemento del registro validado.
- Salir del registro validado.
- Salir del bloque actual después de la validación.

PROPIEDADES QUE AFECTAN A LA NAVEGACIÓN

Las propiedades que afectan a la navegación se agrupan en los siguientes objetos de un formulario.

Objeto	Propiedades que afectan a la navegación
BLOQUE	ESTILO DE NAVEGACIÓN BLOQUE DE DATOS DE LA NAVEGACIÓN ANTERIOR SIGUIENTE BLOQUE DE DATOS DE NAVEGACIÓN
ELEMENTO	ACTIVADO TECLADO NAVEGABLE ELEMENTO DE NAVEGACIÓN ANTERIOR SIGUIENTE ELEMENTO DE NAVEGACIÓN NAVEGACIÓN DEL MOUSE
FORMULARIO	LÍMITE DE NAVEGACIÓN DEL MOUSE PRIMER BLOQUE DE DATOS DE NAVEGACIÓN

Propiedad Navegación del Mouse

Esta propiedad es válida para los siguientes elementos de un formulario:

- Botones.
- Casillas de control.
- Elementos de lista.
- Grupos de botones de radio.
- Elementos de árbol jerárquico.
- Elementos de área de bean.

Esta propiedad tiene por defecto el valor SÍ.

El significado de los valores que puede asumir es el siguiente:

Valor	Significado
SI	Forms navega al nuevo elemento, lo que provoca que se arranquen los disparadores de navegación y de validación.
NO	Forms no navega al nuevo elemento, ni valida el elemento actual cuando el usuario activa con el ratón el nuevo elemento.

DISPARADORES DE NAVEGACIÓN

Los disparadores de navegación se pueden subdividir en dos grupos generales:

- Disparadores de navegación PRE- y POST-.
- Disparadores WHEN-NEW-<*objeto*>-INSTANCE.

Cuándo se arrancan los disparadores PRE- y POST-

Los disparadores de navegación PRE- y POST- se arrancan durante la navegación, justo antes de entrar (PRE-) o salir (POST-) del objeto especificado como parte del nombre del disparador.

Por ejemplo en el caso de un disparador PRE-TEXT-ITEM, se arranca justo antes de entrar en un elemento de texto.

Hay que tener en cuenta que los disparadores PRE- y POST- no se arrancan si pertenecen a una unidad que está más abajo en la jerarquía que la unidad de validación actual. Es decir, por ejemplo si la unidad de validación es un registro (RECORD) en los que existen disparadores PRE-TEXT-ITEM o POST-TEXT-ITEM en alguno de los elementos del registro, entonces estos nunca se arrancarán. En cambio sí se arrancarían los PRE-RECORD (antes de entrar en el registro) y POST-RECORD (después de salir del registro).

Los tipos de disparadores que podemos encontrar con estas condiciones son los siguientes:

Disparador	Se utiliza para

PRE- FORM	• Validar
	  o Usuario.
	  o Hora del día.
	• Inicializar bloques de control
	• Llamar a otra pantalla para mostrar mensajes.

POST- FORM	• Realizar tareas de mantenimiento como por ejemplo, borrado de variables globales. • Mostrar mensajes al usuario antes de salir.

PRE-BLOCK	• Autorizar acceso al bloque. • Definir variables globales.
POST-BLOCK	• Validar el último registro que tuvo el foco de entrada. • Probar una condición y evitar que el usuario salga del bloque.
PRE-RECORD	• Definir variables globales.
POST-RECORD	• Borrar variables globales. • Definir un atributo visual para un elemento cuando el usuario se desplaza por un juego de registros. • Realizar validaciones cruzadas de un campo.
PRE-TEXT-ITEM	• Derivar un valor por defecto complejo. • Registrar el valor anterior de un elemento de texto.
POST-TEXT-ITEM	• Calcular o cambiar valores de elementos.

Cuándo se arrancan los disparadores WHEN-NEW-objeto-INSTANCE

Este tipo de disparadores se arrancan después de los disparadores PRE- y antes de los disparadores POST-, es decir, justo cuando el foco de entrada llega al elemento correspondiente.

Los tipos de disparadores que podemos encontrar con estas condiciones son los siguientes:

Disparador	Momento del arranque
WHEN-NEW-FORM-INSTANCE	Cuando se ejecuta una pantalla, después de una navegación correcta a la misma.
WHEN-NEW-BLOCK-INSTANCE	Después de una navegación correcta a un bloque.
WHEN-NEW-RECORD-INSTANCE	Después de una navegación correcta al registro.
WHEN-NEW-ITEM-INSTANCE	Después de una navegación correcta a una nueva instancia del elemento.

A continuación se muestra un ejemplo del disparador *WHEN-NEW-BLOCK-INSTANCE* mediante el cual se define condicionalmente la propiedad *DELETE_ALLOWED* (borrado permitido) a *FALSE.*

```
IF GET_APPLICATION_PROPERTY(username) = 'SCOTT' THEN
   SET_BLOCK_PROPERTY('PEDIDOS', DELETE_ALLOWED, PROPERTY_FALSE);
END IF;
```

En el siguiente ejemplo con el disparador *WHEN-NEW-FORM-INSTANCE* se consigue ejecutar la consulta del primer bloque definido en el formulario en cuanto se abra el mismo.

```
EXECUTE_QUERY;
```

Uso de funciones incorporadas de Navegación

Se puede incluir la navegación mediante programación (desde disparadores) llamando a los subprogramas incorporados, como por ejemplo *GO_ITEM* y *PREVIOUS_BLOCK*.

Función	Descripción del funcionamiento
GO_FORM	Navega a una pantalla abierta en una aplicación de varias pantallas.
GO_BLOCK GO_ITEM GO_RECORD	Navegan al bloque, elemento o registro indicado.
NEXT_BLOCK NEXT_ITEM NEXT_KEY	Navegan al siguiente bloque, elemento o elemento de clave primaria que permita introducir valores.
NEXT_RECORD PREVIOUS_RECORD	Navegan al primer elemento del registro siguiente o anterior que admita introducir valores.
NEXT_SET	Recupera otro juego de registros de la base de datos y navega al primer registro que se recupera.
UP DOWN	Navegan a la instancia del elemento actual en el registro anterior (UP) o siguiente (DOWN).
PREVIOUS_BLOCK PREVIOUS_ITEM	Navegan al bloque o elemento anterior que permite introducir valores.
SCROLL_UP SCROLL_DOWN	Desplazan el bloque de forma que se muestren los registros por encima del más alto visible o por debajo del más bajo visible.

Este tipo de funciones no se pueden utilizar en los disparadores que se arrancan durante el proceso de navegación, es decir, en los disparadores PRE- y POST-, debido a que no se puede modificar la navegación en el trascurso de llevar a cabo la misma. Durante los disparadores PRE- y POST- la navegación está en curso, y hasta que no finalizan no termina también la navegación.

En el siguiente ejemplo dentro de un disparador PRE-TEXT-ITEM sería incorrecta, dado que estamos intentando cambiar la navegación del disparador PRE- a otro ítem sin que aún haya acabado la propia navegación del disparador:

```
IF CHECKBOX_CHECKED('PEDIDOS.online') THEN
   GO_ITEM('PEDIDOS.estado');
END IF;
```

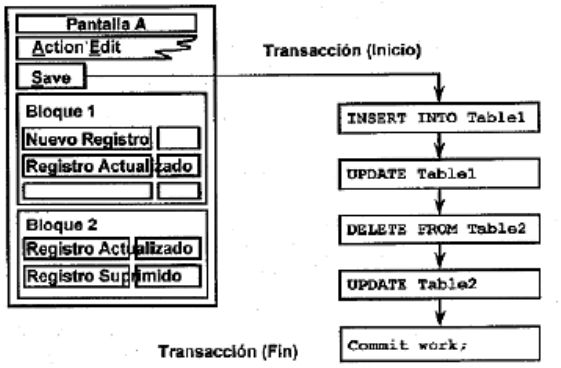
En cambio sí podemos utilizar las funciones de navegación dentro de disparadores de tipo WHEN-NEW-*<objeto>*-INSTANCE, como en el siguiente ejemplo sobre un disparador WHEN-NEW-ITEM-INSTANCE:

```
IF CHECKBOX_CHECKED('PEDIDOS.online') THEN
   GO_ITEM('PEDIDOS.estado');
END IF;
```

PROCESAMIENTO DE TRANSACCIONES 38

INTRODUCCIÓN

Cuando se aplican los cambios en los datos de un formulario a las tablas de la base de datos, Forms Builder permite arrancar disparadores para modificar el comportamiento por defecto de estas operaciones o para agregar otros comportamientos. En la siguiente imagen podemos ver el mecanismo habitual que tiene lugar cuando se confirman los cambios en un formulario que repercute sobre la base de datos.



Cuando un usuario pide a Forms que guarde los cambios realizados en una pantalla, tiene lugar un proceso que implica eventos en la transición de la base de datos correspondiente. Este proceso incluye los siguientes puntos:

- Procesamiento de transacciones de Forms por defecto: aplica los cambios del usuario a las tablas de la base de datos.

- Arranque de disparadores de transacciones: necesarios para realizar acciones adicionales o modificar otras acciones que deben realizarse durante el proceso de almacenamiento en base de datos definido por el diseñador.

Cuando todas estas acciones se completan correctamente, Forms confirma la transacción y hace los cambios permanentes en base de datos.

El proceso de transacción se produce como resultado de una de las siguientes acciones llevadas a cabo por el usuario en el formulario:

- Se pulsa el botón **Guardar** que haya diseñado el programador.
- Se selecciona la opción **Guardar** dentro del menú **Acción** (menú por defecto de Forms Builder).
- Se pulsa el botón **Guardar** que hay dentro de la barra de herramientas por defecto de Forms Builder.
- Se ejecuta la sentencia **COMMIT_FORM** en alguno de los disparadores o subprogramas incorporados en el formulario.

En cualquier caso, el proceso de transacción implica dos fases: envío y confirmación.

- **Envío:** el envío escribe los cambios del usuario en las tablas de la base de datos mediante sentencias *INSERT*, *UPDATE* o *DELETE*. Los cambios se aplican en el orden de secuencia de bloques, según aparecen en el navegador de objetos durante el diseño. Para cada bloque, primero se llevan a cabo los borrados de filas, luego las inserciones y por último las actualizaciones. Los disparadores de transacciones se arrancan durante este ciclo si los define el diseñador dentro del formulario.
- **Confirmación:** lleva a cabo la confirmación de los elementos de la base de datos, lo que hace que los cambios aplicados sean permanentes y se liberan los bloqueos.

Otros eventos relacionados con las transacciones incluyen rollbacks, puntos de grabación y bloqueos.

Las sentencias del lenguaje Sql: *COMMIT, ROLLBACK* y *SAVEPOINT* no se pueden utilizar en el código de disparadores o subprogramas de un formulario. Si se incluyen, Forms Builder las tratará como las siguientes funciones incorporadas: *COMMIT_FORM, ROLLBACK* y CLEAR_FORM.

Si queremos utilizar el COMMIT del lenguaje SQL en el código de un formulario tenemos que utilizar **standard.commit**, que realizará la misma operación que el COMMIT de SQL.

En cuanto al comando ROLLBACK no existe equivalente en el paquete **standard** de Forms, pero sí que podemos utilizar un rollback hasta un punto de salva mediante **standard.rollback_sv(***'punto de ruptura'***)**.

De la misma forma podemos definir un punto de salva o SAVEPOINT en Forms mediante la instrucción **standard.savepoint(***'punto de ruptura'***)**.

Rollbacks

Forms lleva a cabo un rollback (deshacer) de los cambios aplicados, hasta un punto de grabación, si se produce un error en su procesamiento por defecto o cuando falla un disparador de transacción.

Por defecto se informa al usuario del error a través de un mensaje. Un fallo en una inserción o una actualización en la base de datos hace que se vuelva a mostrar el registro afectado en el formulario, o bien que el foco del cursor se quede en el elemento que ha fallado. El usuario puede entonces intentar corregir el error antes de volver a intentar grabarlo.

Puntos de grabación

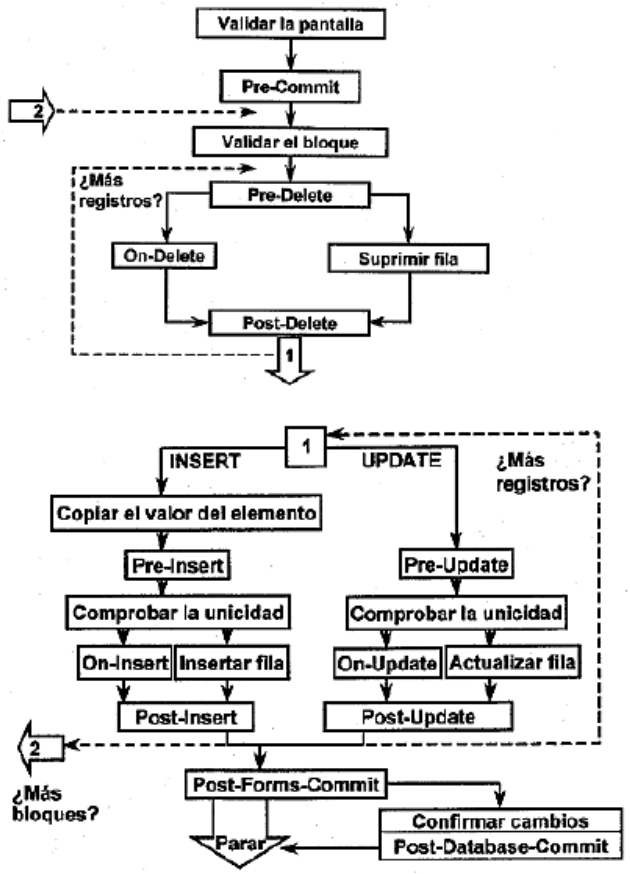
Forms genera automáticamente puntos de grabación en una transacción y realiza un rollback hasta el último punto de grabación si se producen determinados eventos. Generalmente estos puntos de grabación son para uso interno de Forms, pero determinadas funciones incorporadas como, por ejemplo, el procedimiento *EXIT_FORM* pueden solicitar que se realice un rollback hasta el último punto de grabación mediante la opción *TO_SAVEPOINT*.

Bloqueo

Cuando se actualizan o se eliminan filas de una tabla de la base de datos, a partir de un formulario, se aplican automáticamente bloqueos de dichas filas en la base de datos. Los bloqueos también se aplican durante la fase de envío de una transacción y para cualquier sentencia DML que se utilice explícitamente en el código (disparadores o subprogramas) del formulario.

SECUENCIA DE CONFIRMACIÓN DE EVENTOS

A continuación se muestra de forma gráfica el proceso que se lleva a cabo en un formulario para la confirmación de eventos de eliminación, actualización o inserción de filas.



La secuencia de confirmación de eventos es la siguiente:

1. Se valida la pantalla.
2. Se procesa el punto de grabación.
3. Se arranca el disparador *PRE-COMMIT*.
4. Se validan los bloques (en orden secuencial tal cual se han diseñado en el formulario).
5. Llevar a cabo la instrucción DML correspondiente:
   - Para todos los registros suprimidos del bloque (en orden inverso de supresión):
     a) Se arranca el disparador *PRE-DELETE*.
     b) Se suprime la fila de la tabla de base de datos o si se arranca el disparador *PRE-DELETE*.

- Para todos los registros insertados en bloque (en orden secuencial):
  a) Se copia el valor del elemento.
  b) Se comprueba el disparador *PRE-INSERT*.
  c) Se comprueba la unicidad del registro (no existe una duplicidad en la clave primaria o en la clave única) y las restricciones propias de la base de datos (tamaños y tipos de columna, claves ajenas, constraints, etc.).
  d) Se inserta la fila en la tabla de la base de datos o se arranca el disparador *ON-INSERT*.
- Para todos los registros actualizados del bloque (en orden secuencial):
  a) Se arranca el disparador *PRE-UPDATE*.
  b) Se comprueba la unicidad del registro (no existe una duplicidad en la clave primaria o en la clave única) y las restricciones propias de la base de datos (tamaños y tipos de columna, claves ajenas, constraints, etc.).
  c) Se actualiza la fila de la tabla de base de datos o se arranca el disparador *ON-UPDATE*.
  d) Se arranca el disparador *POST-UPDATE*.

6. Se arranca el disparador *POST-FORMS-COMMIT*.
7. Se emite una sentencia SQL COMMIT.
8. Se arranca el disparador *POST-DATABASE-COMMIT*.

Si falla un disparador de confirmación, excepto en el caso del disparador *POST-DATABASE-COMMIT*, se hace rollback de la transacción hasta el punto de grabación definido al inicio del procesamiento de confirmación actual. Esto también significa que no se hace rollback de los envíos no confirmados y emitidos antes del punto de grabación.

DISPARADORES DE CONFIRMACIÓN

A continuación se muestra la lista con los disparadores de confirmación que existen en Forms Builder, así como las características de cada uno.

Disparador	Características
PRE-COMMIT	Se arranca una vez durante el procesamiento de confirmación, antes de que se procesen los bloques de la tabla de base de datos. Se arranca si hay cambios en los elementos de la tabla de base de datos con respecto a los valores equivalentes en los campos del formulario, o si se han enviado cambios pero no se han confirmado aún. Se arranca siempre en caso de envíos no confirmados, incluso si no hay cambios que enviar.

PRE- DML POST- DML	Se arranca para cada registro que está marcado para insertar, actualizar o eliminar. Se arranca inmediatamente antes o después de que se inserte, actualice o se elimina una fila de la tabla de base de datos.
ON- DML	Se arranca para cada registro que está marcado para insertar, actualizar o eliminar cuando Forms emite su sentencia INSERT, UPDATE o DELETE, sustituyendo las sentencias DML por defecto. Incluye una llamada a las funciones incorporadas INSERT_RECORD, UPDATE_RECORD o DELETE_RECORD para realizar el procesamiento por defecto para estos disparadores.
POST-FORMS-COMMIT	Se arranca una vez durante el procesamiento de confirmación, después de que se procesen los bloques de la tabla de base de datos, pero antes de que se emita la sentencia SQL COMMIT. Se arranca incluso aunque no haya cambios que enviar o confirmar.
POST-DATABASE-COMMIT	Se arranca una vez durante el procesamiento de confirmación, después de emitir la sentencia SQL COMMIT. Se arranca incluso aunque no haya cambios que enviar o confirmar (esto también es así para la propia sentencia SQL COMMIT).

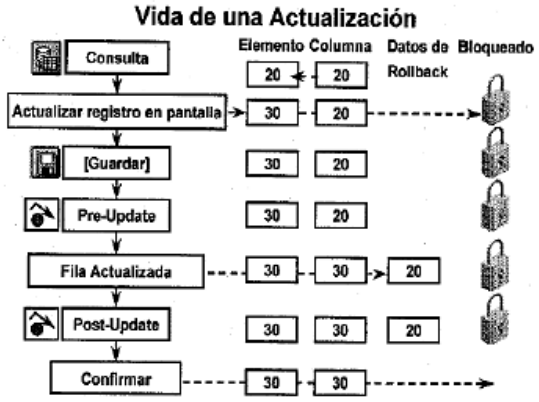
USO COMÚN DE LOS DISPARADORES DE CONFIRMACIÓN

Una vez que conoce cuándo se arranca un disparador de confirmación, debe poder elegir el disparador de confirmación correcto para la funcionalidad que desee. En la tabla siguiente se muestran los usos más habituales que se dan a cada uno.

Disparador	Uso común
PRE- COMMIT	Para comprobar la autorización del usuario. Para configurar requisitos de bloque especiales.
PRE- DELETE	Para escribir en una tabla de auditoría (registro diario). Para implementar eliminación de filas restringidas o en cascada dentro de las tablas de la base de datos.
PRE- INSERT	Para escribir en una tabla de auditoría (registro diario). Para rellenar columnas generadas automáticamente. Para generar números de secuencia. Para comprobar restricciones.
PRE- UPDATE	Para escribir en una tabla de auditoría (registro diario). Para rellenar columnas generadas automáticamente. Para comprobar restricciones. Para implementar actualizaciones restringidas o en cascada sobre tablas de la base de datos.

POST-DELETE POST-INSERT POST-UPDATE	Se utilizan con poca frecuencia, pero en cualquier caso sería para realizar operaciones posteriores a la confirmación de los datos.
ON-DELETE ON-INSERT ON-UPDATE	Se utilizan para sustituir las sentencias DML del bloque por defecto; por ejemplo, para implementar una pseudo supresión o para actualizar una vista de unión.
POST-FORMS-COMMIT	Se utiliza para comprobar las restricciones complejas de varias filas.
POST-DATABASE-COMMIT	Se utiliza para determinar si la confirmación de los datos fue correcta. Determina si hay cambios sin confirmar que ya hayan sido enviados.

Para poder comprender mejor el proceso de confirmación, se visualiza a continuación un ejemplo de actualización.



Validación al suprimir filas en base de datos

Los bloques maestro-detalle de un formulario que están enlazados por una relación con la regla de supresión no aislada, evitan automáticamente que se supriman los registros del bloque maestro del formulario si existen filas hijas en el bloque detalle.

Sin embargo, quizás desee implementar una comprobación similar, de la siguiente manera, cuando se aplica una supresión de filas en la base de datos (siempre y cuando no se haya programado un trigger de base de datos que ya realice lo mismo o una restricción del mismo tipo):

- Una comprobación final para asegurarse de que otro usuario no haya insertado ninguna fila de detalle dependiente de la fila maestro que se va a borrar en la tabla de base de datos, dado que el registro maestro se marcó para la supresión en el formulario.
- Una comprobación final en los datos de pantalla o comprobaciones que implican acciones en la aplicación.

Para conseguir esto habría que programar el disparador *PRE-DELETE*. A continuación se muestra un ejemplo de este tipo de programación en la que se comprueba en el disparador *PRE-DELETE* que no se intente borrar un cliente que aún tenga pedidos en base de datos con su identificador de cliente.

```
DECLARE
       CURSOR C1 IS SELECT * FROM PEDIDOS
       WHERE cliente_id = :CLIENTES.ID;
       V_REG  PEDIDOS%ROWTYPE;
BEGIN
OPEN C1;
FETCH C1 INTO V_REG;
IF C1%FOUND THEN
       CLOSE C1;
       MESSAGE('Hay pedidos para ese cliente. No se puede borrar');
       RAISE FORM_TRIGGER_FAILURE;
ELSE
       CLOSE C1;
END IF;
END;
```

Prueba de resultados DML en disparadores o subprogramas

Cuando programa en un disparador o subprograma una sentencia DML de inserción, actualización o eliminación (INSERT, UPDATE o DELETE), puede que necesite comprobar los resultados de la misma.

A diferencia de la sentencia SELECT, las sentencias DML de modificación de datos antes mencionadas no emiten excepciones cuando se procesan cero o varias filas. PL/SQL proporciona algunos atributos útiles para obtener resultados del cursor implícito utilizado para procesar la última sentencia DML de modificación. A continuación se muestra una tabla con los atributos disponibles y sus posibles valores.

Atributo	Valores
SQL%FOUND	TRUE: indica que la instrucción ha procesado + de 0 filas. FALSE: indica que la instrucción ha procesado 0 filas.
SQL%NOTFOUND	TRUE: indica que la instrucción ha procesado 0 filas. FALSE: indica que la instrucción ha procesado + de 0 filas.
SQL%ROWCOUNT	Devuelve un valor entero que indica el número de filas que se han procesado en la instrucción.

Por ejemplo tomemos el siguiente caso en el que se ejecuta una instrucción de actualización (UPDATE) dentro de un subprograma y queremos programar que cuando no se actualice ninguna fila se devuelva un mensaje al formulario:

```
BEGIN
      UPDATE PEDIDOS
      SET fecha_pedido = SYSDATE
      WHERE pedido_id = :PEDIDOS.id;

      IF SQL%NOTFOUND THEN
            MESSAGE('No se actualiza ningún pedido con ese ID');
            RAISE FORM_TRIGGER_FAILURE;
      END IF;
END;
```

Asignación de números de secuencia a registros

Se pueden asignar valores por defecto durante la creación de elementos de una secuencia de Oracle para proporcionar automáticamente claves únicas. No obstante, si el usuario no completa un registro, el número de secuencia asignado se "desperdicia".

Un método alternativo consiste en asignar claves únicas a registros desde un disparador *PRE-INSERT*, justo antes de su inserción en la tabla de base de datos, en cuyo momento el usuario ha terminado el registro y emitido el comando de Guardar.

La asignación de claves únicas en un disparador *PRE-INSERT* puede conseguir los siguientes efectos:

- Reducir intervalos en los números asignados.
- Reducir el tráfico de datos durante la creación de registros, especialmente si estos se descartan antes de guardarse.

En el siguiente ejemplo de programación del disparador *PRE-INSERT,* en el bloque PEDIDOS se asigna un identificador de pedidos desde la secuencia PEDIDOS_SEQ que escribirá en la columna PEDIDO_ID cuando la fila se inserta en la tabla de la base de datos:

```
BEGIN
   SELECT PEDIDOS_SEQ.nextval
   INTO :BLOQUE.pedido_id
   FROM SYS.dual;
END;
```

Para poder completar correctamente el funcionamiento de este ejemplo, las propiedades **Inserción Permitida** y **Teclado Navegable** del campo :BLOQUE.pedido_id deberían tener valor **No**, de forma que el usuario no pueda introducir manualmente un identificador en este campo.

SENTENCIAS DE CONFIRMACIÓN POR DEFECTO

Si no se ha modificado el procesamiento de confirmación por defecto, Forms emite sentencias DML durante la confirmación para cada registro de la base de datos que se inserte, actualice o suprima construyendo dichas sentencias como se muestra a continuación:

```
INSERT INTO bd_tabla (col1, col2, …)
VALUES (:BLOQUE.ITEM1, BLOQUE.ITEM2, …);
UPDATE bd_tabla
SET
col1 = :BLOQUE.ITEM1
, col2 = :BLOQUE.ITEM2
   …

WHERE ROWID = :ROWID;
DELETE FROM bd_tabla
WHERE ROWID = :ROWID;
```

Además, Forms tiene las siguientes reglas para la formación de las sentencias anteriores:

- Las sentencias construidas pueden arrancar disparadores (triggers) de base de datos asociados a las tablas que se modifican.
- Forms utiliza la construcción ROWID = :ROWID solo cuando la propiedad de bloque **Modo Clave** está definida en **Único** o **Automático** (que es el valor por defecto). En caso contrario, la clave primaria se utiliza para construir la cláusula WHERE.
- Si Forms inserta correctamente una fila en la base de datos, también recupera el ROWID de dicha fila.

- Si la propiedad de bloque **Sólo Actualizar Columnas Cambiadas** está definida en **Sí**, entonces en la sentencia UPDATE solo se incluyen columnas de la tabla de base de datos con valores cambiados.
- Si la propiedad de bloque **Forzar Seguridad de Columna** está definida en **Sí**, entonces todas las columnas de la tabla de base de datos para las que el usuario actual no tiene privilegios de actualización se excluyen de la sentencia UPDATE.

Forms no emite sentencias de bloqueo durante el procesamiento de confirmación por defecto; se emiten en cuanto el usuario actualiza o suprime un registro en la pantalla cuyo bloque está asociado a una tabla de base de datos.

Si define la propiedad del bloque **Modo Bloqueo** al valor **Retrasado**, entonces Forms espera hasta el momento de la confirmación para bloquear la fila correspondiente de la tabla de base de datos.

OBTENCIÓN DEL ESTADO DE CONFIRMACIÓN

Forms proporciona mecanismos para determinar en tiempo de ejecución el estado de confirmación en el que se encuentran los siguientes elementos:

- Un formulario.
- Un registro.
- Un bloque.

Para poder consultar el estado de un formulario preguntaremos por el valor de **:SYSTEM.FORMS_STATUS**.

Valor	Descripción
NEW	Indica que todos los bloques del formulario tienen el estado NEW (nuevo).
QUERY	Indica que al menos un bloque del formulario tiene el estado QUERY (consulta) y todos los demás bloques tienen el estado NEW.
CHANGED	Indica que al menos un bloque del formulario tiene el estado CHANGED (cambiado).

Para poder consultar el estado de un registro preguntaremos por el valor de **:SYSTEM.RECORD_STATUS**.

Valor	Descripción
NEW	Indica que se ha creado el registro, pero que no se ha cambiado aún ninguno de sus elementos (se ha podido rellenar el registro con valores por defecto).
I NSERT	Indica que se han cambiado uno o más elementos de un registro recién creado. El registro se procesará como una inserción durante el siguiente proceso de

	confirmación si su bloque tiene el estado CHANGED (cambiado). Tenga en cuenta que cuando se cambia un elemento de control de un registro NEW, el estado del registro también se convierte en INSERT (insertado).
QUERY	Indica que el registro corresponde a una fila de la base de datos, pero que no se ha cambiado ninguno de sus elementos de la tabla de base de datos.
CHANGED	Indica que se han cambiado uno o más elementos de la tabla de base de datos en un registro. Dicho registro se procesará como una actualización (o supresión) durante el siguiente proceso de confirmación.

Para poder consultar el estado de un bloque preguntaremos por el valor de **:SYSTEM.BLOCK_STATUS**.

Valor	Descripción
NEW	Indica que todos los registros del bloque tienen el estado NEW (nuevo). Tenga en cuenta que un bloque de una tabla de base de datos con el estado NEW, también puede contener registros con el estado INSERT (insertado), provocado al cambiar elementos de control.
QUERY	Indica que todos los registros del bloque tienen el estado QUERY (consultado) si el bloque es un bloque asociado a una tabla de base de datos. Un bloque de control tiene el estado QUERY si contiene al menos un registro con el estado INSERT (insertado).
CHANGED	Indica que el bloque contiene al menos un registro con el estado INSERT (insertado) o CHANGED (cambiado) si el bloque está asociado a una tabla de base de datos. El bloque se procesará durante el siguiente proceso de confirmación. Tenga en cuenta que un bloque de control no puede tener el estado CHANGED.

Las variables del sistema **:SYSTEM.RECORD_STATUS** y **:SYSTEM.BLOCK_STATUS** se utilizan sobre el registro o bloque donde se encuentra actualmente situado el foco del cursor. Si queremos obtener el estado de otros bloques o registros donde no está el cursor, hay que utilizar las siguientes funciones (Built-In) que incorpora Forms Builder.

Función incorporada	Descripción
GET_BLOCK_PROPERTY	Utiliza la propiedad STATUS para obtener el estado del bloque especificado.
GET_RECORD_PROPERTY	Utiliza la propiedad STATUS para obtener el estado del registro especificado del bloque indicado.
SET_RECORD_PROPERTY	Define la propiedad STATUS del registro especificado del bloque indicado en una de las siguientes constantes: • NEW_STATUS • I NSERT_STATUS • QUERY_STATUS • CHANGED_STATUS

Tomemos el siguiente ejemplo en el que comprobamos si el tercer registro del bloque PEDIDOS tiene el estado a CHANGED (cambiado), si es así, le ponemos el estado QUERY a dicho registro:

```
BEGIN
IF GET_RECORD_PROPERTY(3,'PEDIDOS',STATUS) = 'CHANGED'
THEN
    SET_RECORD_PROPERTY(3,'PEDIDOS',STATUS,QUERY_STATUS);
END IF;
END;
```

PROCESAMIENTO DE MATRICES DML

El procesamiento de matrices es una opción de Forms Builder que modifica la forma en que se procesan los registros. El comportamiento por defecto de Forms consiste en procesar un único registro cada vez. Activando el procesamiento de matrices, se pueden procesar grupos de registros a la vez, lo que reduce el tráfico de red y aumenta, por tanto, el rendimiento. Con el procesamiento de matrices, una estructura (una matriz de datos) que contiene varios registros, se envía o se devuelve desde o hacia el servidor para el procesamiento.

Forms Builder soporta tanto el procesamiento de recuperación de matrices como el procesamiento de matrices DML. Para las operaciones DML (insert, update o delete) y de consulta (select), puede determinar el tamaño de matriz para optimizar el rendimiento según las necesidades. Este apartado del capítulo se centra en el estudio del procesamiento de matrices DML.

El procesamiento de matrices está disponible para las operaciones DML y de consulta en bloques que estén asociados a tablas de la base de datos, o sobre vistas, procedimientos y subconsultas; y no están permitidos en bloques basados en disparadores de transacciones.

Implementar matrices DML

Se realiza en dos fases:

1. Dentro de las preferencias de Forms Builder:
    a. Seleccionamos dentro del menú **Editar** la opción **Preferencias.**
    b. Nos situamos en la solapa **Ejecución.**
    c. Marcamos la opción **Procesamiento Matrices.**

2. Dentro de las propiedades del bloque:
    a. Nos situamos dentro del grupo de propiedades **Base de Datos Avanzada.**
    b. Indicamos un valor superior a 0 en la propiedad **Tamaño de Matriz DML.** Este número indicará la cantidad de registros que se procesará a la vez para este bloque. Si este valor es > 1, entonces hay que especificar la clave primaria.

Efecto de las matrices DML en disparadores de transacciones

Cuando la propiedad **Tamaño de Matriz DML** está definida a valor 1, los disparadores PRE-INSERT, PRE-UPDATE y PRE-DELETE se arrancan para cada registro nuevo, cambiado o suprimido; se emite la sentencia DML y se arranca el disparador POST- para dicho registro.

Cuando la propiedad **Tamaño de Matriz DML** está definida a un valor mayor que 1, se arrancan los disparadores PRE- adecuados para todas las filas nuevas, cambiadas y suprimidas; se emiten todas las sentencias DML y se arrancan todos los disparadores POST-.

Un ejemplo de funcionamiento sería el siguiente tomando en consideración que la propiedad **Tamaño de Matriz DML** de un bloque está a valor 20: si cambiamos 100 filas, se obtienen 100 disparadores PRE-, 5 matrices de 20 sentencias DML y 100 disparadores POST-.

GESTIÓN EN TIEMPO DE EJECUCIÓN 39

INTRODUCCIÓN

En este capítulo veremos distintas funciones incorporadas en Forms, así como variables útiles para controlar nuestro programa en tiempo de ejecución.

VARIABLES PARA EL CONTROL DEL FOCO

A continuación se relacionan las variables del sistema que devuelven el elemento del formulario donde se encuentra actualmente el foco de entrada.

Variable	Descripción
CURSOR_BLOCK	Devuelve el nombre del bloque donde se encuentra el foco.
CURSOR_RECORD	Devuelve el número del registro donde se encuentra el foco. El valor devuelto se hace en forma de una cadena.
CURSOR_ITEM	Devuelve el nombre del campo con el formato *bloque.nombre*, donde se encuentra el bloque. Por ejemplo 'pedidos.numero_pedido'.
CURSOR_VALUE	Devuelve el valor del campo donde se encuentra el foco. El valor devuelto siempre lo hará en forma de una cadena.

Tomemos el siguiente ejemplo en el que definimos el código para un trigger WHEN-BUTTON-PRESSED de un botón correspondiente a un bloque de control (no asociado a tabla de base de datos).

```
IF :SYSTEM.CURSOR_BLOCK = 'PEDIDOS' THEN
   GO_BLOCK('DATOS_PEDIDO');
ELSIF :SYSTEM.CURSOR_BLOCK = 'DATOS_PEDIDO' THEN
   GO_BLOCK('INVENTARIO');
ELSIF :SYSTEM.CURSOR_BLOCK = 'INVENTARIO' THEN
   GO_BLOCK('PEDIDOS');
END IF;
```

Con dicho código conseguiremos cambiar la navegación del formulario a uno u otro bloque del mismo, dependiendo de dónde se encuentre el foco en el momento de pulsar el botón donde está definido este código. Hay que tener en cuenta que dicho botón tendrá que tener la propiedad **Navegación del Mouse** a valor **No**, para que cuando se pulse el botón no identifique que el foco está sobre el bloque donde está el botón.

VARIABLES PARA EL CONTROL DEL FOCO DE UN DISPARADOR

A continuación se relacionan las variables del sistema que devuelven el elemento del formulario donde se encontraba el foco cuando se arrancó un disparador de Forms.

Variable	Descripción
TRIGGER_BLOCK	Devuelve el bloque en el que se encontraba el foco de entrada cuando se arrancó inicialmente el disparador.
TRIGGER_RECORD	Devuelve el número del registro que está procesando Forms.
TRIGGER_ITEM	Devuelve el bloque y el nombre del elemento en el que estaba el foco de entrada cuando se arrancó inicialmente el disparador.

Tomemos el siguiente ejemplo en el que definimos el código para un trigger WHEN-BUTTON-PRESSED de un botón correspondiente a un bloque de control (no asociado a tabla de base de datos).

FUNCIONES INCORPORADAS

En este apartado del capítulo iremos conociendo diversas funciones para el manejo de nuestra aplicación en tiempo de ejecución.

Get_Application_Property

Esta función devuelve información acerca de la aplicación Forms actual. La sintáxis de la misma es:

```
GET_APPLICATION_PROPERTY(<propiedad>);
```

Esta aplicación devuelve un valor de tipo VARCHAR2 y los valores que puede asumir el parámetro <propiedad> son los que se muestran a continuación:

Propiedad	Descripción
APPLICATION_INSTANCE	Devuelve el valor del puntero del manejador de la instancia. Solo es válido para plataformas Windows. En el resto de plataformas devuelve NULL.
BUILTIN_DATE_FORMAT	Devuelve el valor actual de la máscara de formato.
CALLING_FORM	Devuelve el nombre del formulario llamado si se hizo mediante la instrucción CALL_FORM. En caso contrario devolverá NULL.
CONFIG	Devuelve el nombre de la sección de configuración de una aplicación que está siendo usada.
CONNECT_STRING	Devuelve la cadena de conexión a la base de datos.
CURRENT_FORM	Devuelve el nombre del fichero .FMX del formulario actual que está siendo ejecutado.
CURRENT_FORM_NAME	Devuelve el nombre del formulario (el valor de la propiedad Nombre que tiene dicho formulario) que se está ejecutando.
CURSOR_STYLE	Devuelve el nombre actual de la propiedad Estilo de Cursor. Los posibles valores a devolver son: • BUSY • CROSSHAIR • DEFAULT • HELP • INSERTION
DATASOURCE	Devuelve el nombre de la base de datos que está actualmente en uso.
DATETIME_LOCAL_TZ	Devuelve la hora local de la zona (región donde nos encontramos)
DATETIME_SERVER_TZ	Devuelve la hora del servidor de la zona (región donde se encuentra el servidor)
DISPLAY_HEIGHT	Devuelve la altura de visualización.
DISPLAY_WIDTH	Devuelve la anchura de visualización.
ERROR_DATE/ DATETIME_FORMAT	Devuelve el valor actual del error de la fecha o la máscara de formato de fecha y hora.
FLAG_USER_VALUE_TOO_LONG	Devuelve el valor actual de esta propiedad.
IS_PROXY_CONNECTION	Devuelve si la aplicación se conecta a la base de datos usando un proxy (TRUE) o no (FALSE)
OPERATING_SYSTEM	Devuelve el nombre del sistema operativo que se está usando. Los valores válidos devueltos son: • MSWINDOWS • MSWINDOWS23 • WIN32COMMON • UNIX • SUNOS • MACINTOSH

	• VMS • HP-UX.
PASSWORD	Devuelve la password del usuario del sistema operativo que se ha conectado a la aplicación.
PLSQL_DATE_FORMAT	Devuelve el valor actual de la máscara de formato de fecha para PL/SQL.
SAVEPOINT_NAME	Devuelve el nombre del último punto de ruptura salvado (SAVEPOINT) que ha alcanzado Forms en la ejecución actual.
SSO_SUBDN	Devuelve la cadena que contiene el nombre que distingue al suscriptor para la conexión Single Sign On, si el usuario se ha autenticado a través de un Servidor de Autenticación.
SSO_USRID	Devuelve la cadena que contiene el ID del usuario para la conexión Single Sign On, si el usuario se ha autenticado a través de un Servidor de Autenticación.
SSO_USRDN	Devuelve la cadena que contiene el nombre que distingue al usuario (dn) en la conexión Single Sign On, si el usuario se ha autenticado a través de un Servidor de Autenticación.
TIMER_NAME	Devuelve el nombre del TIMER (contador de tiempo) que más recientemente ha concluido.
USER_DATE/ DATE_TIME_FORMAT	Devuelve el valor actual de la fecha del sistema del equipo del usuario, o el formato de máscara de fecha y hora.
USER_INTERFACE	Devuelve el nombre de la interfaz del usuario que está actualmente en uso. Los valores válidos de retorno son: • MOTIF • MACINTOSH • MSWINDOWS • MSWINDOWS32 • WIN32COMMON • WEB • PM • BLOCKMODE • X • UNKNOWN
USER_NLS_CHARACTER_SET	Devuelve la porción correspondiente al juego de caracteres que haya en la variable de entorno del usuario USER_NLS_LANG. Si dicha variable no está definida, devolverá el valor de la variable NLS_LANG.
USER_NLS_DATE_FORMAT	Devuelve el formato de máscara de fecha correspondiente al lenguaje de la región (NLS). Es

	equivalente al valor de la variable de entorno NLS_DATE_FORMAT. Si dicha variable no está definida, devolverá el valor que corresponda al formato de fecha del territorio que indique la variable NLS
USER_NLS_LANG	Devuelve el valor completo actual de la variable de entorno USER_NLS_LANG, que indica el lenguaje nacional que se soporta. Si no está definida dicha variable, devolverá el valor de la variable NLS_LANG. USER_NLS_LANG es equivalente a concatenar las variables de entorno siguientes: • USER_NLS_LANGUAGE • USER_NLS_TERRITORY • USER_NLS_CHARACTER_SET
USER_NLS_LANGUAGE	Devuelve la porción correspondiente al lenguaje que haya en la variable de entorno USER_NLS_LANG. Si no está definida dicha variable, devuelve el valor de la variable NLS_LANG.
USER_NLS_TERRI TORY	Devuelve la porción correspondiente al territorio que haya en la variable de entorno USER_NLS_LANG. Si no está definida dicha variable, devuelve el valor de la variable NLS_LANG.
USERNAME	Devuelve el nombre del usuario del sistema operativo que se ha conectado a la aplicación.

En el siguiente ejemplo capturamos en dos variables globales el nombre del usuario del sistema operativo que se ha conectado a la aplicación y el nombre del propio sistema operativo de dicho usuario.

```
: GLOBAL. SO_USER := GET_APPLI CATI ON_PROPERTY( USERNAME) ;
: GLOBAL. SO := GET_APPLI CATI ON_PROPERTY( OPERATI NG_SYSTEM) ;
```

Hay que resaltar que en un sistema de varias capas donde la aplicación se ejecuta a través de un Servidor de Aplicaciones, la función GET_APPLICATION_PROPERTY devuelve la información acerca del servidor de aplicaciones que se encuentra en la capa media. Si se necesita conocer la información acerca de la máquina cliente, habrá que utilizar un JavaBean o bien Oracle WebUtil.

Get_Block_Property

Esta función devuelve información acerca de un bloque especificado. La sintaxis de la misma es:

```
GET_BLOCK_PROPERTY(<nombre del bloque>, propiedad);
```

Esta aplicación devuelve un valor de tipo VARCHAR2 y los valores que puede asumir el parámetro <propiedad> son los que se muestran a continuación:

Propiedad	Descripción
ALL_RECORDS	Especifica si todos los registros equivalentes al criterio de búsqueda deberían ser cargados en el bloque de datos cuando se ejecute la consulta.
BLOCKSCROLLBAR_X_POS	Devuelve la posición X de la barra de desplazamiento del bloque.
BLOCKSCROLLBAR_Y_POS	Devuelve la posición Y de la barra de desplazamiento del bloque.
COLUMN_SECURITY	Devuelve 'TRUE' si la seguridad de la columna está asociada a SÍ. En caso contrario 'FALSE'.
COORDINATION_STATUS	Para un bloque detalle de una relación maestro-detalle, devuelve el estado de coordinación del block detalle respecto al maestro. Los posibles valores a devolver son: • NON_COORDINATED • COORDINATED
CURRENT_RECORD	Devuelve el número del registro actual.
CURRENT_RECORD_ATTRIBUTE	Devuelve el nombre del atributo visual que está asociado al bloque.
CURRENT_ROW_BACKGROUND_COLOR	Devuelve el color de fondo (background).
CURRENT_ROW_FILL_PATTERN	Devuelve el patrón de relleno.
CURRENT_ROW_FONT_NAME	Devuelve el nombre de la fuente.
CURRENT_ROW_FONT_SIZE	Devuelve el tamaño de la fuente.
CURRENT_ROW_FONT_SPACING	Devuelve el espaciado de la fuente.
CURRENT_ROW_FONT_STYLE	Devuelve el estilo de la fuente.
CURRENT_ROW_FONT_WEIGHT	Devuelve el grosor de la fuente.
CURRENT_ROW_FOREGROUND_COLOR	Devuelve el color de frente (foreground).
DEFAULT_WHERE	Devuelve el texto que se haya indicado en la cláusula WHERE del bloque.
DELETE_ALLOWED	Devuelve 'TRUE' si está permitido borrar elementos en el bloque. En caso contrario 'FALSE'.
DML_DATA_TARGET_NAME	Devuelve el nombre de la fuente de datos DML asociada al bloque.
DML_DATA_TARGET_TYPE	Devuelve el valor indicado para la propiedad **Tipo de Destino de Datos DML**. Los posibles a devolver son: • NONE. • TABLE

	• STORED PROCEDURE • TRANSACTIONAL TRIGGER
ENFORCE_PRIMARY_KEY	Devuelve 'TRUE' si la propiedad **Forzar Clave Primaria** tiene el valor SÍ. En caso contrario 'FALSE'.
ENTERABLE	Devuelve 'TRUE' si se puede acceder al bloque porque alguno de sus elementos tenga las propiedades **Activado** y **Teclado Navegable** a SÍ. En caso contrario 'FALSE'.
FIRST_DETAIL_RELATION	Devuelve el nombre de la primera relación con la que el bloque actual actúa como detalle.
FIRST_ITEM	Devuelve el nombre del primer elemento del bloque.
FIRST_MASTER_RELATION	Devuelve el nombre de la primera relación en la que el bloque actual actúa como maestro.
INSERT_ALLOWED	Devuelve 'TRUE' si la propiedad **Inserción Permitida** tiene el valor SÍ. En caso contrario 'FALSE'.
KEY_MODE	Devuelve el valor actual de la configuración del modo de clave del bloque actual. Los valores posibles son: • UNIQUE_KEY • UPDATEABLE_PRIMARY_KEY • NON_UPDATEABLE_PRIMARY_KEY
LAST_ITEM	Devuelve el nombre del último elemento del bloque.
LAST_QUERY	Devuelve la sentencia SQL de la última consulta ejecutada en el bloque.
LOCKING_MODE	Devuelve el valor 'IMMEDIATE' si las filas se bloquean inmediatamente cuando se efectúe un cambio en los elementos de la tabla de la base de datos asociada al bloque. En caso contrario 'DELAYED'.
MAX_QUERY_TIME	Devuelve el valor actual de la propiedad **Tiempo de Consulta Máximo**. Esta propiedad permite al desarrollador programar si el usuario puede abortar la consulta cuando se supere el tiempo indicado en esta propiedad.
MAX_RECORDS_FETCHED	Devuelve el número máximo de registros que se pueden recuperar en una consulta.
NAVIGATION_STYLE	Devuelve el valor que indica la configuración de la propiedad **Estilo de Navegación.** Los valores posibles son: • SAME_RECORD

	• CHANGE_RECORD • CHANGE_BLOCK
NEXTBLOCK	Devuelve el nombre del siguiente bloque. Devolverá NULL si el bloque actual es el último.
NEXT_NAVI GATI ON_BLOCK	Devuelve el nombre del siguiente bloque al que se va a navegar.
ONTI ME_WHERE	Devuelve el valor actual de la cláusula WHERE en ejecución para el bloque especificado.
OPTI MI ZER_HI NT	Devuelve una señal (hint) de tipo VARCHAR2 para indicar que el formulario pasó el optimizador RDBMS cuando construyó la consulta.
ORDER_BY	Devuelve el valor por defecto de la cláusula ORDER BY que afecta al bloque.
PRECOMPUTE_SUMMARI ES	Especifica que el valor de cualquier campo calculado del bloque se calcule antes de que la consulta sea emitida en el bloque.
PREVI OUSBLOCK	Devuelve el nombre del bloque que tiene la secuencia más baja en la ordenación dentro del formulario. Devolverá NULL cuando estemos en el primer bloque del formulario.
PREVI OUS_NAVI GATI ON_BLOCK	Devuelve el nombre (en formato VARCHAR2) del bloque anterior al que se ha navegado.
QUERY_ALLOWED	Devuelve 'TRUE' si la propiedad **Consulta Permitida** tiene el valor SÍ. Con esta propiedad permitimos al usuario consultar registros dentro del bloque.
QUERY_DATA_SOURCE_NAME	Devuelve el nombre del origen de datos asociada a la consulta del bloque. Corresponde con la propiedad **Nombre de Origen de Datos de Consulta.**
QUERY_DATA_SOURCE_TYPE	Devuelve el valor (en formato VARCHAR2) de la propiedad **Consultar Tipo de Origen de Datos.** Los posibles valores que devuelve son: • NONE • TABLE • STORED PROCEDURE • TRANSACTIONAL TRIGGER • SUB-QUERY
QUERY_HI TS	Devuelve un valor que indica el número de registros identificados por la operación COUNT_QUERY. Es decir, QUERY_HINTS especifica el número de registros que se han recuperado.

QUERY_OPTIONS	Devuelve un valor dentro de los siguientes: • VIEW • FOR UPDATE • COUNT_QUERY En caso contrario devolverá NULL.
RECORDS_DISPLAYED	Devuelve el número de registros del bloque que se pueden visualizar.
RECORDS_TO_FETCH	Devuelve el número de registros que se esperan en el proceso de recuperación del cursor asociado a la query.
STATUS	Devuelve el valor 'NEW' si el bloque contiene únicamente nuevos registros. 'CHANGED' si el bloque contiene al menos 1 registro modificado y 'QUERY' si el bloque contiene solo registros válidos que hayan sido recuperados de la base de datos.
TOP_RECORD	Devuelve el número de registro del registro visible con valor más alto del bloque dado.
UPDATE_ALLOWED	Devuelve el valor 'TRUE' si la propiedad actualización permitida tiene valor SÍ en el bloque. En caso contrario devuelve 'FALSE'.
UPDATE_CHANGED_COLUMNS	Especifica que solo las columnas actualizadas por un operador se envían a la base de datos.

Get_Item_Property

Esta función devuelve información acerca de un elemento especificado. La sintaxis de la misma es:

```
GET_ITEM_PROPERTY(<nombre del elemento>, propiedad);
```

Esta aplicación devuelve un valor de tipo VARCHAR2 y los valores que puede asumir el parámetro <propiedad> son los que se muestran a continuación:

Propiedad	Descripción
AUTO_HINT	Devuelve 'TRUE' cuando la propiedad **Mostrar Indicio Automáticamente** tiene el valor SÍ. En caso contrario 'FALSE'.
AUTO_SKIP	Devuelve 'TRUE' cuando la propiedad **Salto Automático** tiene el valor SÍ. En caso contrario 'FALSE'.
BACKGROUND_COLOR	Devuelve el nombre del color de fondo.
BLOCK_NAME	Devuelve el nombre del bloque asociado al elemento consultado.

BORDER_BEVEL	Devuelve el valor de la propiedad **Bisel**. Los posibles valores que devuelve son: • RAISED • LOWERED • PLAIN
CASE_I NSENSI TI VE_QUERY	Devuelve 'TRUE' cuando la propiedad **Consulta No Sensible a Mayúsculas/Minúsculas** tiene el valor SÍ. En caso contrario 'FALSE'.
CASE_RESTRI CTI ON	Devuelve el valor de la propiedad **Restricción Mayúsculas/Minúsculas**. Los posibles valores que devuelve son: • UPPERCASE • LOWERCASE • NONE
COLUMN_NAME	Devuelve el nombre de la columna en la base de datos al que está asociado el elemento.
CONCEAL_DATA	Devuelve 'TRUE' si el texto que introduce un usuario está oculto. En caso contrario 'FALSE'.
CURRENT_RECORD_ATTRI BUTE	Devuelve el nombre del atributo visual del registro donde se encuentra el elemento.
CURRENT_ROW_BACKGROUND_COLOR	Devuelve el color de fondo del registro donde se encuentra el elemento.
CURRENT_ROW_FI LL_PATTERN	Devuelve el nombre del patrón de relleno del registro donde se encuentra el elemento.
CURRENT_ROW_FONT_NAME	Devuelve el nombre de la fuente del registro donde se encuentra el elemento.
CURRENT_ROW_FONT_SI ZE	Devuelve el tamaño de la fuente del registro donde se encuentra el elemento.
CURRENT_ROW_FONT_SPACI NG	Devuelve el espaciado de la fuente del registro donde se encuentra el elemento.
CURRENT_ROW_FONT_STYLE	Devuelve el estilo de la fuente del registro donde se encuentra el elemento.
CURRENT_ROW_FONT_WEI GHT	Devuelve el ancho de la fuente del registro donde se encuentra el elemento.
CURRENT_ROW_FOREGROUND_COLOR	Devuelve el color frontal del registro donde se encuentra el elemento.
DATA_LENGTH_SEMANTI CS	Determina la semántica de las propiedades **Longitud Máxima** y **Longitud de Consulta** cuando se utiliza para forzar el tamaño máximo del valor interno del elemento.
DATABASE_VALUE	Para un elemento asociado a base de datos, devuelve el valor que se recuperó originalmente de la base de datos.

DATATYPE	Devuelve el tipo de elemento. Los posibles valores que devuelve son: • ALPHA • CHAR • DATE • JDATE • EDATE • DATETIME • INT • RINT • MONEY • RMONEY • NUMBER • RNUMBER • TIME • LONG • GRAPHICS • IMAGE.
DIRECTION	Devuelve la dirección del tapiz para objetos bidireccionales. Los posibles valores son: • RIGHT_TO_LEFT • LEFT_TO_RIGHT
DISPLAYED	Devuelve 'TRUE' si se visualiza el elemento. En caso contrario 'FALSE'.
ECHO	Devuelve 'TRUE' sin la propiedad **Ocultar Datos** tiene el valor SÍ. En caso contrario 'FALSE'.
EDITOR_NAME	Devuelve el nombre del editor asociado al elemento.
EDITOR_X_POS	Devuelve la posición X de visualización del editor.
EDITOR_Y_POS	Devuelve la posición Y de visualización del editor.
ENFORCE_KEY	Devuelve el nombre del elemento cuyo valor se copia a este elemento como una clave ajena cuando se crea un nuevo registro como parte de una relación maestro-detalle. Corresponde con la propiedad **Copiar Valor de Elemento**.
ENABLED	Devuelve 'TRUE' si la propiedad **Activado** tiene el valor SÍ. En caso contrario 'FALSE'.
FILL_PATTERN	Devuelve el patrón de relleno del elemento.
FONT_NAME	Devuelve el nombre de la fuente del elemento.
FONT_SIZE	Devuelve el tamaño de la fuente del elemento.
FONT_SPACING	Devuelve el espacio de la fuente del elemento.
FONT_STYLE	Devuelve el estilo de la fuente del elemento.

FONT_WEI GHT	Devuelve el ancho de la fuente del elemento.
FOREGROUND_COLOR	Devuelve el color frontal del elemento.
FORMAT_MASK	Devuelve la máscara de formato del elemento.
HEI GHT	Devuelve la altura del elemento.
HI NT_TEXT	Devuelve el texto de ayuda que se muestra en la línea de mensajes en tiempo de ejecución.
I CON_NAME	Devuelve el nombre del icono asociado al elemento cuando este es un botón y tiene la propiedad **Icónico** a valor SÍ.
I CONI C_BUTTON	Devuelve el valor 'TRUE' si el botón está definido como icónico. En caso contrario 'FALSE'.
I NSERT_ALLOWED	Devuelve 'TRUE' si la propiedad **Inserción Permitida** tiene valor SÍ. En caso contrario 'FALSE'.
I TEM_CANVAS	Devuelve el nombre del lienzo del elemento.
I TEM_I S_VALI D	Devuelve 'TRUE' si el valor del elemento es válido. En caso contrario 'FALSE'.
I TEM_NAME	Devuelve el nombre del elemento.
I TEM_TAB_PAGE	Devuelve el nombre de la solapa a la que está asignada el elemento.
I TEM_TYPE	Devuelve el tipo del elemento. Los posibles valores son: • BUTTON • CHART ITEM • CHECKBOX • DISPLAY ITEM • IMAGE • LIST • RADIO GROUP • TEXT ITEM • USER AREA
J USTI FI CATI ON	Devuelve el tipo de alineamiento que tiene asignado el elemento. Los posibles valores son: • START • END • LEFT • CENTER • RIGHT
KEEP_POSI TI ON	Devuelve 'TRUE' si el cursor está re-entrando en la misma localización que cuando salió del elemento. En caso contrario 'FALSE'.
LABEL	Devuelve el valor correspondiente a la propiedad "Etiqueta" del elemento.

LI ST	Devuelve 'TRUE' si el elemento es un TEXT ITEM asociado a una lista de valores LOV. En caso contrario 'FALSE'.
LOCK_RECORD_ON_CHANGE	Devuelve 'TRUE' si Oracle Forms debería intentar bloquear una fila basada en un cambio potencial de este elemento. En caso contrario 'FALSE'.
LOV_NAME	Devuelve el nombre de la lista de valores asociada al elemento.
LOV_X_POS	Devuelve el valor de la coordenada X donde se visualiza la lista LOV.
LOV_Y_POS	Devuelve el valor de la coordenada Y donde se visualiza la lista LOV.
MAX_LENGTH	Devuelve la longitud máxima configurada por el elemento.
MASTER_MI RROR_I TEM	Devuelve el nombre del elemento espejo maestro o el nombre del propio elemento.
MOUSE_NAVI GATE	Devuelve 'TRUE' si esta propiedad tiene el valor SÍ para el elemento. En caso contrario 'FALSE'.
MULTI _LI NE	Devuelve 'TRUE' si el elemento es multi-lineal. En caso contrario 'FALSE'.
NAVI GABLE	Devuelve 'TRUE' si esta propiedad tiene el valor TRUE a nivel de elemento. En caso contrario 'FALSE'.
NEXTI TEM	Devuelve el nombre del siguiente elemento en la secuencia de navegación por defecto.
NEXT_NAVI GATI ON_I TEM	Devuelve el nombre del siguiente elemento que está definido en la propiedad "Siguiente elemento de Navegación".
PREVI OUSI TEM	Devuelve el nombre del elemento previo.
PREVI OUS_NAVI GATI ON_I TEM	Devuelve el nombre del elemento anterior que está definido en la propiedad "Elemento de Navegación Anterior".
PRI MARY_KEY	Devuelve el valor 'TRUE' si el elemento es una clave primaria. En caso contrario 'FALSE'.
PROMPT_ALI GNMENT_OFFSET	Devuelve la distancia entre el elemento y su etiqueta.
PROMPT_DI SPLAY_STYLE	Devuelve el estilo de visualización de la etiqueta. Los posibles valores son: • FIRST_RECORD • HIDDEN
PROMPT_EDGE	Devuelve el valor que indica cómo se coloca la etiqueta con respecto al elemento. Los posibles valores son: • START

	• END • TOP • BOTTOM
PROMPT_EDGE_ALI GNEMENT	Devuelve el valor que indica cómo se alinea la etiqueta del elemento. Los posibles valores son: • START • END • CENTER
PROMPT_EDGE_OFFSET	Devuelve la distancia entre el elemento y su etiqueta.
PROMPT_FI LL_PATTERN	El patrón que se utiliza para rellenar la región del objeto.
PROMPT_FONT_NAME	El nombre de la familia de la fuente.
PROMPT_FONT_SI ZE	El tamaño de la fuente especificada en puntos.
PROMPT_FONT_SPACI NG	El ancho de la fuente, que es la cantidad de espacio entre caracteres.
PROMPT_FONT_STYLE	El estilo de la fuente
PROMPT_FONT_WEI GHT	El grosor de la fuente
PROMPT_FOREGROUND_COLOR	El color de la región de fondo del objeto.
PROMPT_TEXT	El texto de la etiqueta que se visualiza en el elemento.
PROMPT_TEXT_ALI GNMENT	Devuelve el valor que indica cómo se justifica la etiqueta. Los posibles valores son: • START • LEFT • RIGHT • CENTER • END
PROMPT_VI SUAL_ATTRI BUTE	Devuelve un valor que indica los atributos visuales de la etiqueta.
QUERYABLE	Devuelve 'TRUE' si el elemento puede ser incluido en una consulta. En caso contrario 'FALSE'.
QUERY_LENGTH	Devuelve el número de caracteres que se permite introducir en el elemento en el modo Consulta.
QUERY_ONLY	Devuelve 'TRUE' si la propiedad tiene el valor SÍ. En caso contrario 'FALSE'.
RANGE_HI GH	Devuelve el valor más alto del rango límite permitido.
RANGE_LOW	Devuelve el valor más pequeño del rango límite permitido.
RECORDS_DI SPLAYED	Devuelve el número de registros que se pueden visualizar en el bloque.

REQUI RED	Para los elementos multi-lineales devuelve el valor 'TRUE' si esta propiedad tiene el valor verdadero.
SCROLLBAR	Devuelve 'TRUE' si esta propiedad tiene el valor SÍ. En caso contrario 'FALSE'.
TOOLTI P_BACKGROUND_COLOR	El color de la región de fondo del TOOLTIP.
TOOLTI P_FI LL_PATTERN	El patrón de relleno que se utiliza en el TOOLTIP.
TOOLTI P_FONT_NAME	El nombre de la familia de la fuente de TOOPLTIP.
TOOLTI P_FONT_SI ZE	El tamaño de la fuente especificado en puntos.
TOOLTI P_FONT_SPACI NG	El ancho de la fuente, que es la cantidad de espacio entre caracteres.
TOOLTI P_FONT_STYLE	El estilo de la fuente.
TOOLTI P_FONT_WEI GHT	El grosor de la fuente.
TOOLTI P_FOREGROUND_COLOR	El color frontal de la región en el TOOLTIP.
UPDATE_ALLOWED	Devuelve 'TRUE' si esta propiedad tiene el valor verdadero. En caso contrario 'FALSE'.
UPDATE_COLUMN	Devuelve 'TRUE' si Oracle Forms debería tratar el elemento como actualizable. En caso contrario 'FALSE'.
UPDATE_NULL	Devuelve 'TRUE' si el elemento debería ser actualizado solo si tiene previamente un valor NULL. En caso contrario 'FALSE'.
UPDATE_PERMI SSI ON	Devuelve 'TRUE' si esta propiedad tiene el valor ON, al combinar las propiedades UPDATEABLE y UPDATE_NULL si ambas tienen valor verdadero. En caso de que ambas tengan valor falso, devolverá 'FALSE'.
VALI DATE_FROM_LI ST	Devuelve 'TRUE' si Oracle Forms debería validar el valor del elemento contra el valor de la LOV a la que está asociado.
VI SI BLE	Devuelve 'TRUE' si la propiedad tiene valor SÍ. En caso contrario 'FALSE'.
VI SUAL_ATTRI BUTE	Especifica el nombre de los atributos visuales que se deberían aplicar al objeto en tiempo de ejecución.
WI DTH	Devuelve el ancho del elemento.
WI NDOW_HANDLE	Devuelve la constante única interna que se usa para referirse al objeto. Devolverá el valor '0' si la plataforma no es Microsoft Windows.
WRAP_STYLE	Especifica el estilo de ajuste de texto. Los posibles valores son: • CHARACTER

	• WORD • NONE
X_POS	Devuelve la coordenada X que refleja el lugar en el que se posiciona el elemento en relación con la esquina superior izquierda.
Y_POS	Devuelve la coordenada Y que refleja el lugar en el que se posiciona el elemento en relación con la esquina superior izquierda.

Set_Item_Instance_Property

Esta función modifica la instancia específica de un elemento en un bloque, ajustando sus propiedades. La sintaxis de la misma es:

```
SET_ITEM_INSTANCE_PROPERTY(<nombre del elemento>,
                           <número_de_registro>,
                           propiedad, valor);
```

Los valores que puede asumir el parámetro <propiedad> son los que se muestran a continuación:

Propiedad	Descripción
BORDER_BEVEL	Especifica el borde del bisel del elemento. Los posibles valores admisibles son: • RAISED • LOWERED • PLAIN
INSERT_ALLOWED	Se aplica solo a los registros no recuperados de la base de datos y si se indica PROPERTY_TRUE permite modificaciones a nivel de instancia, elemento o bloque. En caso contrario con PROPERTY_FALSE, se prohíbe modificar la instancia.
NAVIGABLE	Cuando se indica PROPERTY_TRUE se permite navegar a la instancia del elemento usando el teclado de navegación. Cuando se indica PROPERTY_FALSE se prohíbe la navegación mediante teclado.
REQUIRED	Cuando se indica PROPERTY_TRUE se fuerza a que el usuario introduzca un valor distinto de vacío en un elemento de la instancia.
UPDATE_ALLOWED	Solo se aplica a los registros recuperados de la base de datos. Cuando se indica PROPERTY_TRUE .
VISUAL_ATTRIBUTE	Especifica el nombre del atributo visual que se utilizará a nivel de instancia.

Set_Menu_Item_Property

Esta función modifica las propiedades dadas de una opción de menú. La sintaxis de la misma es:

```
SET_MENU_ITEM_PROPERTY(<nombre del menú.nombre del ítem de menú>, propiedad, valor);
```

Los valores que puede asumir el parámetro <propiedad> son los que se muestran a continuación:

Propiedad	Descripción
CHECKED	Para indicar si un ítem de menú de tipo CHECK BOX o RADIO MENU tiene activada o desactivada la propiedad chequeado.
ENABLED	Para habilitar o deshabilitar un ítem de menú.
ICON_IN_HTOOLBAR	Para indicar si un ítem de menú aparece representado como un icono en una barra de menús horizontal.
ICON_IN_MENU	Para especificar si un icono se visualiza en el menú debajo del ítem de menú.
ICON_IN_VTOOLBAR	Para indicar si un ítem de menú aparece representado como un icono en una barra de menús vertical.
ICON_NAME	Especifica el nombre del fichero del icono asociado con el ítem de menú. Solo se utiliza si la propiedad ICON_IN_MENU tiene valor verdadero.
LABEL	Especifica el texto correspondiente a la etiqueta del ítem de menú.
VISIBLE	Especifica si el ítem de menú se visualiza.

Set_Tab_Page_Property

Esta función define las propiedades de una solapa de un lienzo de tipo página de solapas. La sintaxis de la misma es:

```
SET_TAB_PAGE_PROPERTY(<nombre de la solapa>, propiedad,
                      valor);
```

Los valores que puede asumir el parámetro <propiedad> son los que se muestran a continuación:

Propiedad	Descripción
BACKGROUND_COLOR	El color de la región de fondo del objeto.
ENABLED	Especificar 'TRUE' para habilitar la solapa y 'FALSE' para deshabilitarla.

FILL_PATTERN	El patrón se utiliza para rellenar la región del objeto.
FONT_NAME	El nombre de la familia de la fuente que se usará para el texto del objeto.
FONT_SIZE	El tamaño de la fuente.
FONT_SPACING	El ancho de la fuente, que es la cantidad de espacio entre caracteres.
FONT_STYLE	El estilo de la fuente.
FONT_WEIGHT	El grosor de la fuente.
FOREGORUND_COLOR	El color de la región frontal del objeto.
LABEL	El texto que aparece como título en la solapa.
VISIBLE	Especificar 'TRUE' para hacer visible la solapa y 'FALSE' para no visualizarla.
VISUAL_ATTRIBUTE	Especifica el nombre de los atributos visuales.

REFERENCIA A OBJETOS MEDIANTE EL IDENTIFICADOR INTERNO

Forms Builder asigna un identificador a cada objeto que cree. Un identificador de objeto es un valor interno que no se muestra nunca. Para obtenerlo, se utiliza el subprograma FIND_<objeto>.

Existe un subprograma FIND por cada tipo de objeto representado en Forms Builder:

Subprograma	Descripción
FIND_ALERT	Para recuperar el identificador de una alerta.
FIND_BLOCK	Para recuperar el identificador de un bloque.
FIND_CANVAS	Para recuperar el identificador de un lienzo.
FIND_COLUMN	Para recuperar el identificador de una columna.
FIND_EDITOR	Para recuperar el identificador de un editor.
FIND_EVEN_OBJECT	Para recuperar el identificador de un evento.
FIND_FORM	Para recuperar el identificador de un formulario.
FIND_GROUP	Para recuperar el identificador de un grupo.
FIND_ITEM	Para recuperar el identificador de un elemento.
FIND_LOV	Para recuperar el identificador de una lista de valores (LOV).
FIND_MENU_ITEM	Para recuperar el identificador de un ítem de menú.
FIND_RELATION	Para recuperar el identificador de una relación.
FIND_REPORT_OBJECT	Para recuperar el identificador de un informe.
FIND_TAB_PAGE	Para recuperar el identificador de una solapa de un lienzo.
FIND_TIMER	Para recuperar el identificador de un contador (timer).
FIND_TREE_NODE	Para recuperar el identificador de un nodo del árbol.
FIND_VA	Para recuperar el identificador de un atributo visual.

FIND_VIEW	Para recuperar el identificador de una vista.
FIND_WINDOW	Para recuperar el identificador de una ventana.

Los subprogramas FIND_<objeto> requieren, como parámetro, un nombre de objeto totalmente cualificado.

Los valores de retorno de los subprogramas FIND_<objeto> son de un tipo específico. Los tipos de identificadores de objeto están predefinidos en Forms Builder. Hay un tipo distinto para cada objeto.

Las razones para utilizar identificadores de objeto son:

- Mejora del rendimiento. Forms busca el objeto solo una vez cuando se llama inicialmente al subprograma FIND_<objeto> para obtener el identificador. Cuando se hace referencia a un objeto por el nombre en un disparador, Forms debe buscar el identificador de objeto cada vez.
- Escritura de código más genérico.
- Probar si un objeto existe. Para ello se utilizan las funciones ID_NULL o FIND_objeto.

DECLARACIÓN DE VARIABLES PARA IDENTIFICADORES DE OBJETO

Para utilizar un identificador de objeto, primero debe asignarlo a una variable. Debe declarar una variable del mismo tipo que el identificador de objeto.

En el siguiente ejemplo se utiliza la función incorporada FIND_ITEM para asignar el identificador del elemento que tiene actualmente el foco de entrada a la variable ID_VAR.

Una vez que se asigna un identificador de objeto a una variable en un disparador o una unidad de programa PL/SQL, puede utilizar dicha variable para hacer referencia al objeto, en lugar de hacer referencia al objeto por el nombre.

```
DECLARE
        Id_var          ITEM;
BEGIN
        Id_var  := FIND_ITEM(:SYSTEM.CURSOR_ITEM);
        ...
END;
```

En los dos siguientes ejemplos se muestra que se puede transferir un nombre de elemento o un identificador de elemento al subprograma incorporado SET_ITEM_PROPERTY. Las llamadas siguientes son equivalentes lógicamente:

```
SET_ITEM_PROPERTY('PEDIDO.num_id', position, 50, 35);
SET_ITEM_PROPERTY(Id_var, position, 50, 35);
```

Puede utilizar identificadores de objeto o nombre de objeto en la misma lista de argumentos, siempre que cada argumento individual haga referencia a un objeto distinto.

Sin embargo, no puede utilizar un identificador de objeto y un nombre de objeto para formar un nombre de objeto totalmente cualificado (NOMBRE_BLOQUE.NOMBRE_ELEMENTO). La siguiente llamada no es válida:

```
GO_ITEM(Id_bloque.'nombre_elemento');
```

USO DE IDENTIFICADORES DE OBJETO FUERA DEL BLOQUE PL/SQL INICIAL

Se ha visto con anterioridad cómo se hacen referencias a los identificadores de objeto en el disparador o unidad de programa mediante variables PL/SQL. Puede hacer referencia a estas variables PL/SQL solo en el bloque PL/SQL actual; sin embargo, puede aumentar el ámbito de un identificador de objeto.

Para hacer referencia a un identificador de objeto fuera del bloque PL/SQL inicial, tiene que convertir el identificador a un formato numérico utilizando una extensión .ID para la variable PL/SQL declarada y a continuación, asignarla a una variable global.

En el siguiente ejemplo de código de disparador, se asigna el identificador de objeto a una variable PL/SQL local (var_elemento) inicialmente, y a continuación, a una variable global (GLOBAL.elemento):

```
DECLARE
        Var_elemento        ITEM;
BEGIN
        Var_elemento := FIND_ITEM(:SYSTEM.CURSOR_ITEM);
        :GLOBAL.elemento := Var_elemento.ID;
END;
```

USO COMPARTIDO DE OBJETOS 40

INTRODUCCIÓN

Al desarrollar aplicaciones, se debe compartir y reutilizar los objetos y el código cuando sea posible para:

- **Mejorar la productividad:** puede desarrollar aplicaciones de forma mucho más efectiva y eficiente si no intenta "comenzar desde el principio" cada vez que escribe un fragmento de código. Al compartir y reutilizar objetos y código utilizados con frecuencia, puede acortar el tiempo de desarrollo y aumentar la productividad.
- **Disminuir el mantenimiento:** al crear aplicaciones que utilizan o llaman al mismo objeto o fragmento de códigos varias veces, puede reducir el tiempo de mantenimiento.
- **Aumentar la modularidad:** al compartir y reutilizar el código, se aumenta la modularidad de las aplicaciones.
- **Mantener estándares:** puede mantener los estándares reutilizando objetos y código. Si crea un objeto una vez y lo vuelve a copiar una y otra vez, no corre el riesgo de introducir pequeños cambios. De hecho, puede crear un juego de objetos estándar y algunos fragmentos de código estándar y utilizarlos como punto de partida para todos los módulos de pantalla nuevos.

- **Rendimiento de aplicación mejorado:** cuando Forms Services comunica la interfaz de usuario al cliente de Forms, envía metadatos acerca de los elementos de la pantalla. Esos metadatos incluyen valores de propiedades que difieren de los valores por defecto. Una vez que se define un elemento, los metadatos acerca del siguiente elemento solo incluyen aquellas propiedades que difieren del elemento anterior. Esto se conoce como función de envío de diferencias. Fomentar las similitudes entre elementos mediante los métodos de reutilización de objetos presentados en este capítulo mejora la eficiencia de la función de envío de diferencias y de este modo reduce el tráfico de red y mejora el rendimiento.

Una de las formas más sencillas que tiene un desarrollador para aumentar la eficiencia del rendimiento de una aplicación es mediante el uso de estándares consistentes para todos los objetos de una aplicación. Los elementos de tipos distintos deben tener al menos valores comunes para las propiedades comunes o compartidas.

Para maximizar la reutilización, el desarrollador debe aplicar las siguientes instrucciones en el orden indicado:

- **Aceptar las propiedades por defecto siempre que sea posible:** si no se sobrescriben las propiedades para cada objeto, el valor de las propiedades comunes será el mismo independientemente del tipo de objeto, a excepción de la posición y el tamaño.
- **Utilizar SmartClasses para describir un objeto:** si, a causa de los estándares de diseño, el uso de las propiedades por defecto no es una opción viable, la creación de subclases de objetos a partir de un juego de SmartClasses garantiza que se satisfacen los estándares de desarrollo. También fuerza un alto grado de uso compartido de propiedades en objetos gráficos. Los elementos del mismo tipo tendrán entonces (a menos que se sustituyan) las mismas propiedades y, por tanto, podrán compartir propiedades de forma más efectiva. Aprenderá acerca de las SmartClasses más adelante en esta misma lección.
- **Uso de juegos de atributos visuales:** si no se utilizan SmartClasses para aplicar los estándares y las propiedades comunes, entonces utilice juegos de atributos visuales parciales para aplicar un juego común de propiedades en objetos de distinto tipo; por ejemplo, fuente, tamaño de fuente, color de primer plano, color de fondo, etc. Estos juegos de atributos visuales se pueden definir como clases de propiedad.

CLASE DE PROPIEDAD

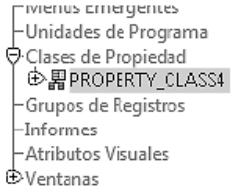
Una clase de propiedad es un objeto "con nombre" que contiene una lista de propiedades y sus valores.

Las clases de propiedad se utilizan para:

- Aumentar la productividad definiendo valores estándar o utilizados frecuentemente para las propiedades comunes y asociándolos con diversos objetos de Forms Builder. Puede utilizar las clases de propiedad para definir propiedades estándar no solo para un objeto particular, sino para varios a la vez. Esto da como resultado un aumento de la productividad, ya que elimina el tiempo dedicado a definir propiedades idénticas para varios objetos.
- Mejorar el rendimiento de red aumentando la eficiencia de la función de envío de diferencias.

Creación de una Clase de Propiedad

Al crear una clase de propiedad, tiene todas las propiedades de cada objeto de Forms Builder disponibles. Debe seleccionar las propiedades y los valores a incluir en la clase de propiedad. Puede crear una clase de propiedad situándose en el grupo **Clases de Propiedad** del navegador de objetos y pulsando el botón  .



En esta imagen se muestra el resultado de la creación de una clase de propiedad en Forms Builder.

Para poder añadir propiedades a la clase hay que abrir con la **Paleta de Propiedades** la clase (pulsando **F4**) y una vez dentro de la paleta, se pueden añadir propiedades a la clase pulsando el botón  .

Si se quiere eliminar una propiedad de las que se hayan incluido en la clase, hay que pulsar el botón X que se encuentra a la derecha del botón de añadir propiedad.

Herencia desde una Clase de Propiedad

Una vez que se crea una clase de propiedad y se agregan propiedades, puede utilizar dicha clase de propiedad. Para aplicar las propiedades de una clase de propiedad a un objeto, utilice la propiedad **Información de Subclase** de cualquier elemento.

Una **propiedad heredada** es aquella que toma su valor de la clase de propiedad que asoció al objeto. Una propiedad heredada se muestra con una flecha a la izquierda del nombre de la propiedad.

Una **propiedad variante** es aquella que tiene un valor modificado incluso aunque se haya heredado de la clase de propiedad asociada al objeto. Puede sustituir el valor de cualquier propiedad heredada para hacer que dicha propiedad sea variante. Las propiedades variantes se muestran con una cruz roja sobre una flecha.

Para heredar valores de propiedad desde una clase de propiedad hay que seguir estos pasos:

- Haga clic en el objeto al que dese aplicar las propiedades desde la clase de propiedad.
- Haga clic en la propiedad Información de Subclase de la paleta de propiedades del objeto.
- Seleccione la clase de propiedad cuyas propiedades desea utilizar. El objeto toma los valores de dicha clase de propiedad y las propiedades heredadas se muestran como un símbolo de flecha, siempre y cuando la clase haya modificado los valores por defecto de la propiedad del elemento. Cuando se aplica sobre un elemento de texto, solo aparece con flecha la propiedad **Justificación** porque es la única a la que se le cambia el valor por defecto (Principio) que tiene el objeto:

CONVERSIÓN DE PROPIEDAD HEREDADA A PROPIEDAD VARIANTE

Para convertir una propiedad heredada en una propiedad variante, simplemente hay que introducir un valor diferente en la propiedad heredada.

CONVERSIÓN DE PROPIEDAD VARIANTE A PROPIEDAD HEREDADA

Para convertir una propiedad variante en una propiedad heredada, haga clic en el icono  (heredar).

Recuerde que para poder hacer esto, previamente el elemento deberá tener incluida una clase de propiedad que se utilizará con el mismo, dentro de la propiedad **Información de Subclase.**

GRUPO DE OBJETOS

Un grupo de objetos es un contenedor lógico para un juego de objetos de Forms Builder.

Estos grupos se definen cuando se desea realizar operaciones como las siguientes:

- Empaquetar objetos relacionados para copiarlos o crear subclases en otro módulo.
- Agrupar numerosos objetos en bloques integrantes de nivel superior que se puedan volver a utilizar en otra aplicación.

Un ejemplo muy típico de la necesidad de crear un grupo de objetos es el caso del calendario. Imagine que diseña en Forms un calendario y quiere que el mismo se reutilice en otros formularios, entonces puede agrupar todos los elementos que componen el calendario en un grupo de objetos y luego basta con copiar dicho grupo a otro formulario para disponer del calendario y sus elementos.

Creación y uso de grupos de objetos

Para crear un grupo de objetos hay que realizar los siguientes pasos en Forms Builder:

- Dentro del navegador de objetos de Forms Builder hay que hacer clic en **Grupo de Objetos**.
- A continuación dentro del menú de Forms pulsamos en **Editar – Crear**, lo que provoca que se muestre una nueva entrada en el grupo de objetos.
- Una vez creado el objeto se puede cambiar el nombre por defecto que se muestra (OBJECT_GROUPx) por el nombre que se desee.
- Para añadir elementos al grupo de objetos hay que arrastrarlos desde el propio formulario hasta el grupo de objetos. Para ello nos situamos en el nombre del formulario (dentro del navegador de objetos) y dentro del menú seleccionamos **Ver – Ampliar Todo**.
- Para añadir los objetos se mantendrá pulsada la tecla CONTROL y se irá haciendo clic en cada objeto que se quiera incluir en el grupo de objetos.
- A continuación se arrastrarán esos objetos al Grupo de Objetos.

Hay que tener en cuenta las siguientes circunstancias a la hora de crear un grupo de objetos:

- La inclusión de un bloque en un grupo de objetos también incluye sus elementos, los disparadores de nivel de elemento, los disparadores de nivel de bloque y las relaciones. No puede utilizar ninguno de estos objetos en un grupo de objetos sin el bloque.
- No es posible incluir otro grupo de objetos.
- La supresión de un objeto de un módulo suprime automáticamente el/los objeto/s del grupo de objetos.
- La supresión de un grupo de objetos de un módulo no suprime los objetos del módulo que contiene.

COPIA Y CREACIÓN DE SUBCLASES DE OBJETOS

Puede copiar o crear subclases de objetos de la siguiente forma:

- Entre módulos (formularios) se pueden arrastrar objetos entre los módulos en el navegador de objetos.

- En un único módulo seleccionado el objeto en el navegador de objetos, presionando la tecla CONTROL y arrastrándolo para crear el objeto nuevo.

Cuando se arrastran objetos, aparece un recuadro de diálogo que pregunta si se desea copiar el objeto o crear subclases.

Una vez arrastrado el objeto a otro formulario y seleccionada la opción **Subclase** en el cuadro de diálogo, se mostrará el objeto en el formulario destino con un indicativo de una flecha roja en la parte inferior izquierda del mismo.

Al copiar un objeto, se crea una versión única e independiente de dicho objeto en el módulo de destino. También se copia cualquier objeto que sea propiedad del objeto copiado. Con la creación de subclases puede realizar una copia exacta y, a continuación, modificar las propiedades de algunos objetos si lo desea. Si cambia la clase principal, los cambios también se aplican a las propiedades del objeto subclasificado que no ha modificado. Sin embargo, cualquier propiedad que sustituya permanece sustituida. Esto proporciona un potente modelo de herencia de objetos.

Cuando se crea una subclase de bloque de datos, se puede:

- Cambiar la estructura del principal, propagando automáticamente los cambios al secundario.
- Agregar o cambiar propiedades al secundario para sustituir la herencia.

Cuando se crea una subclase de bloque de datos, no se puede:

- Suprimir elementos del secundario.
- Cambiar el orden de los elementos en el secundario.
- Agregar elementos al secundario a menos que se agreguen al final.

Capacidad para agregar a un objeto

Puede crear una copia exacta de un objeto y puede agregarla al objeto subclasificado. Por ejemplo, puede agregar elementos adicionales al final de un bloque subclasificado.

Capacidad para modificar propiedades

Con la creación de subclases, puede realizar una copia exacta y, a continuación, modificar las propiedades de algunos objetos. Si cambia la clase principal, los cambios también se aplican a las propiedades del objeto subclasificado que no ha modificado. Sin embargo, cualquier propiedad que sustituya permanece sustituida.

Capacidad para heredar cambios

Cuando se cambian las propiedades de un objeto principal, todos los objetos secundarios heredan esas propiedades si no se han sustituido todavía.

El secundario hereda cambios:

- Inmediatamente, si los objetos principal y secundario están en la misma pantalla.
- Cuando se vuelve a cargar la pantalla que contiene un objeto secundario.

Capacidad para volver a heredar

Si realiza cambios en el objeto secundario para sustituir las propiedades del objeto principal, puede hacer clic en el icono Heredar para volver a heredar la propiedad del objeto principal.

Iconos que identifican las propiedades de un elemento

Hay una serie de iconos que se asocian a las propiedades de un elemento que permiten rápidamente identificar el tipo de propiedad. A continuación se indican cuáles son:

Icono de la propiedad	Significado
 Círculo	La propiedad tiene el valor por defecto.
 Cuadrado	El valor de la propiedad ha cambiado respecto al valor por defecto.
 Flecha	El valor de la propiedad ha sido heredado.
 Flecha con cruz roja	El valor de la propiedad se ha heredado pero ha sido sustituido por otro valor.

BIBLIOTECAS DE OBJETOS

Las bibliotecas de objetos son contenedores de objetos adecuados para su reutilización. Simplifican la reutilización en entornos complejos y soportan estándares corporativos, de proyectos y personales.

Una biblioteca de objetos puede contener objetos simples, clases de propiedades, grupos de objetos y unidades de programa, pero están protegidos frente a cambios en la biblioteca. Los objetos se pueden utilizar como estándares (clases) para otros objetos.

Las bibliotecas de objetos simplifican el uso compartido de componentes reutilizables. La reutilización de componentes le permite:

- Aplicar estándares a objetos simples como, por ejemplo, botones y elementos, para que tengan un aspecto consistente. Esto también mejora el rendimiento de red, fomentando similitudes entre objetos, lo que aumenta la eficiencia de la función de envío de diferencias.
- Reutilizar objetos complejos como, por ejemplo, un navegador.

La diferencia entre utilizar bibliotecas de objetos y grupos de objetos reside en los siguientes puntos:

- Las bibliotecas de objetos son externas a la pantalla, por tanto, se comparten con facilidad entre módulos de pantalla.

- Las bibliotecas de objetos pueden contener elementos individuales; por ejemplo, botones icónicos. La unidad más pequeña aceptada en un grupo de objetos es un bloque.
- Las bibliotecas de objetos aceptan unidades de programación PL/SQL.
- Si cambia un objeto en una biblioteca de objetos, todas las pantallas que contienen el objeto subclasificado reflejarán el cambio.

Las bibliotecas de objetos aparecen en el navegador si están abiertas. Puede crear, abrir y cerrar bibliotecas de objetos como otros módulos. Forms Builder abre automáticamente todas las bibliotecas de objetos que estaban abiertas cuando cerró Forms Builder por última vez.

Las bibliotecas de objetos se almacenan como archivos con extensión **.OLB**.

No se permite la modificación de objetos dentro de la misma biblioteca de objetos. Para realizar cualquier cambio en la misma, hay que arrastrar el objeto desde la biblioteca a un formulario, cambiarlo en dicho formulario y volver a arrastrarlo a la biblioteca de objetos.

Crear una biblioteca de objetos

Para crear una biblioteca de objetos hay que seguir estos pasos:

- Abra Forms Builder y un formulario del que vaya a utilizar objetos para crear la biblioteca.
- A continuación en el menú **Archivo** pulse sobre la opción **Nuevo** y dentro de esta sobre **Biblioteca de Objetos.** Esto producirá la creación de una biblioteca con un nombre genérico dentro del grupo Bibliotecas de Objetos.
- Seguidamente haga doble clic sobre el nombre de la biblioteca de objetos para abrir las solapas contenedoras del mismo.
- Por último, se irán arrastrando, desde el formulario abierto inicialmente a la biblioteca de objetos, todos aquellos elementos que se deseen incorporar a la misma.
- Por último cambiamos el nombre por defecto de la librería, y eliminamos aquellas solapas de la misma que no tienen contenido.
- Para guardar la librería entramos en el menú **Archivo** y pulsamos en **Guardar** lo que provoca que tengamos que elegir la ruta y el nombre del archivo (.OLB) donde vamos a almacenar la librería.

SMARTCLASS

SmartClass es un miembro especial de una biblioteca de objetos. Se puede utilizar para crear fácilmente subclases de objetos existentes en una pantalla mediante la opción **SmartClass** que se muestra dentro de cualquier elemento cuando se pulsa el botón derecho del ratón sobre el mismo.

A continuación se muestra un ejemplo.

Hay que tener en cuenta que la opción SmartClass dentro del menú desplegable solo aparecerá habilitada si existe una biblioteca de objetos que esté abierta en Forms Builder con algún elemento de tipo SmartClass.

Los elementos de una biblioteca de objetos que están marcados como SmartClass aparecen en la misma con un aspa a su izquierda. En la imagen anterior, podemos ver que el elemento (de tipo alerta) NOFOTO es un SmartClass.

Para convertir un elemento de una biblioteca de objetos en un SmartClass hay que seleccionarlo y desde el menú **Editar** seleccionar la opción **SmartClass.** De esta forma el elemento quedará marcado con un aspa a la izquierda, tal y como se muestra en esta imagen:

BIBLIOTECAS DE CÓDIGO PL/SQL

Una biblioteca de código es una recopilación de unidades de programa PL/SQL, incluidos procedimientos, funciones y paquetes.

Una única biblioteca puede contener muchas unidades de programa que pueden compartir las aplicaciones y los módulos de Oracle Forms Developer que necesitan utilizarlos.

Una biblioteca:

- Se crea como un módulo independiente y se almacena en un archivo o en la base de datos.
- Proporciona un método adecuado de almacenamiento del código del cliente y su uso compartido entre aplicaciones.
- Supone que una única copia de unidades de programa puede ser utilizada por muchos módulos de pantalla, de menú o de informe.
- Soporta la carga dinámica de unidades de programa.

Dado que las bibliotecas se compilan independientemente de los módulos de Forms que las utilizan, las variables ligadas en las pantallas, menús e informes están fuera del ámbito de la biblioteca. Esto significa que no tiene que hacer referencia directamente a variables que no son locales en otro módulo, dado que el compilador no las conoce cuando se compilan las unidades de programa de la biblioteca.

Las bibliotecas de código PL/SQL se almacenan en archivos con dos tipos de extensiones:

- **.PLL**: son los archivos que contiene el código fuente de la biblioteca.
- **.PLX**: son los archivos que contiene el código compilado de la biblioteca.

Hay dos maneras de evitar las referencias directas a variables ligadas en las bibliotecas PL/SQL:

- Puede hacer referencia a las variables globales y las variables del sistema en las pantallas de forma indirecta como cadenas entre comillas utilizando determinados subprogramas incorporados.
- Escriba unidades de programa con los parámetros IN e IN OUT que están diseñados para aceptar referencias a variables ligadas. A continuación, puede transferir los nombres de las variables ligadas como parámetros al llamar a las unidades de programa de biblioteca desde las aplicaciones de Forms.

Crear una biblioteca de código PL/SQL

Para crear una biblioteca de objetos hay que seguir estos pasos:

- Abra Forms Builder y sitúese en el elemento **Bibliotecas PL/SQL** del navegador de objetos.
- A continuación pulse en el menú **Archivo** la opción **Nuevo** y seguidamente **Biblioteca PL/SQL.**

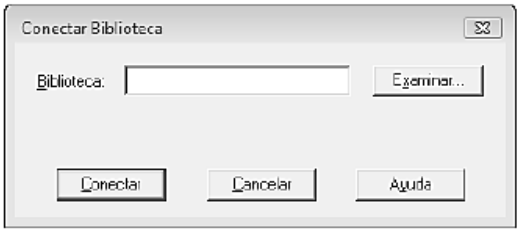
- Esto provocará que se cree un elemento de tipo Biblioteca PL/SQL con un nombre por defecto: LIB_NNN.
- Dentro de la nueva biblioteca de código PL/SQL encontrará dos elementos: Unidades de Programa y Bibliotecas Conectadas.
- En el grupo de elementos unidades de programa, puede crear tantos procedimientos, funciones y paquetes como necesite.
- En el grupo de elementos Bibliotecas Conectadas, podrá conectar tantas bibliotecas de código PL/SQL a esta, como necesite.
- Para almacenar la biblioteca sitúese en el nombre por defecto de la misma y a continuación vaya al menú **Archivo**, desde allí seleccione **Guardar Como**, y cuando se muestre el nombre del archivo con el que quiere almacenar la librería, inserte uno nuevo sin eliminar la extensión .PLL. Este nombre de archivo será también el que se le otorgue a la librería.
- Para generar el archivo .PLX con el código compilado sitúese en el menú **Programa** y a continuación pulse la opción **Compilar Módulo**. Si no hay errores de compilación, en la misma ruta donde almacenó el código fuente aparecerá el fichero con extensión .PLX.

Conectar una biblioteca de código

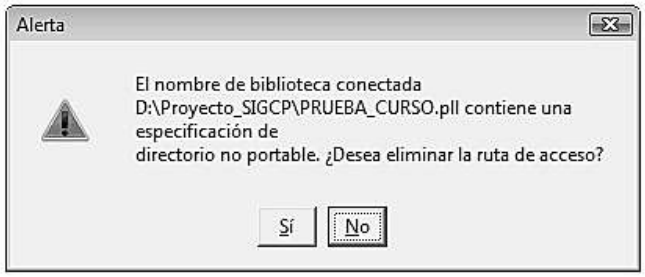
Una biblioteca de código PL/SQL se puede conectar a un formulario para que este utilice el código de la misma, o también se puede conectar a otra biblioteca de código PL/SQL con el mismo objetivo.

Para conectar una biblioteca de código hay que seguir estos pasos.

- Abra Forms Builder y el objeto al que quiera conectar la biblioteca: el formulario u otra biblioteca.
- Sitúese en el elemento **Bibliotecas Conectadas** del navegador de objetos y a continuación dentro del menú **Editar** pulse la opción **Crear.** Al hacerlo se mostrará un cuadro de diálogo como se muestra en la siguiente imagen:



- Pulse el botón *Examinar* y dentro del árbol de carpetas de su sistema busque y seleccione el archivo de biblioteca de código (.PLL) que quiera conectar al formulario. Y a continuación pulse el botón *Conectar*. Esto provocará que se muestre un nuevo cuadro de diálogo como el siguiente:



- En este cuadro de diálogo se presentan dos alternativas cuyo significado son:

  - **NO**: es la opción por defecto y supone que al conectar la biblioteca al formulario también se almacena la ruta donde se encuentra la misma.
  - **SI**: esta opción conecta la biblioteca pero no almacena la ruta donde se encuentra la misma. Es la opción más aconsejable cuando desconocemos si la ruta donde se encuentra la biblioteca va a estar disponible en todos los entornos donde pretendamos utilizar el formulario.

- Una vez seleccionada una de las dos opciones, la biblioteca quedará conectada en el elemento (formulario u otra biblioteca) y se podrá referenciar a los elementos de la misma.

Desconectar una biblioteca de código

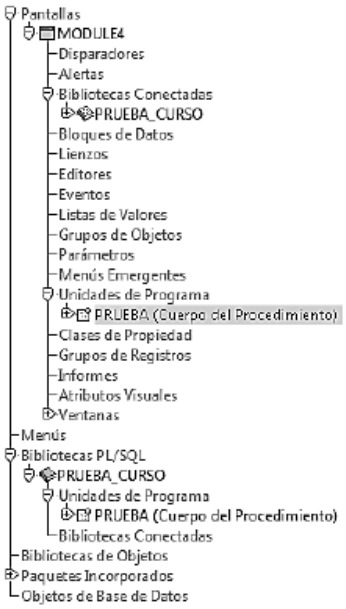
Para quitar o desconectar una biblioteca de código PL/SQL de un elemento (formulario u otra biblioteca) únicamente hay que eliminar dicha biblioteca del grupo **Bibliotecas Conectadas** dentro del navegador de objetos.

Referenciar a elementos de una biblioteca de código

Para hacer referencia a cualquier procedimiento, función o paquete de una biblioteca de código conectada, no hay que anteponer al nombre de estos elementos

ningún prefijo, simplemente hay que referenciarlos por el nombre como si fuesen cualquier otra unidad de programa perteneciente al formulario o biblioteca.

En el caso de que exista igualdad de nombres entre un elemento de código (procedimiento, función o paquete) del formulario y de la biblioteca conectada, la búsqueda de dicho elemento comenzará siempre por el formulario.



Pongamos el siguiente ejemplo donde hay una unidad de programa a nivel del formulario que se llama PRUEBA y otra a nivel de la biblioteca conectada (PRUEBA_CURSO) también con el mismo nombre:

Si ahora dentro del formulario creamos un bloque de código como el siguiente:

```
BEGI N
    PRUEBA;
END;
```

lo que se ejecutará será el procedimiento PRUEBA del formulario, no de la biblioteca conectada.

Cuando hay varias bibliotecas conectadas en un mismo elemento (formulario u otra biblioteca), las búsquedas de elementos se realizan en orden, de arriba abajo, según se visualizan en el navegador de objetos.

MISCELÁNEA FORMS 41

INTRODUCCIÓN

En este capítulo vamos a ver una serie de utilidades que se pueden desarrollar en Oracle Forms.

APERTURA DE UNA PÁGINA WEB DESDE FORMS

Forms suministra una utilidad que permite abrir directamente páginas web en el navegador por defecto que se tenga instalado en el cliente. Para realizar esta operación hay que utilizar la siguiente Built-in:

```
WEB.SHOW_DOCUMENT(url, destino)
```

Los parámetros que utiliza son:

- **Url:** se indicará en forma de string (entrecomillada), la dirección completa de la página web que se quiere consultar.
- **Destino**: será uno de los siguientes:
  - ***'_self'***: para indicar que la página web se abra en el mismo marco que la aplicación.
  - ***'_parent'***: para indicar que la página web se abra en la ventana raíz de la aplicación.
  - ***'_top'***: para indicar que la página web se abra en la ventana que contiene el enlace a la apertura de la aplicación, sustituyendo lo que haya.
  - ***'_blank'***: para indicar que la página web se abra en una nueva ventana.

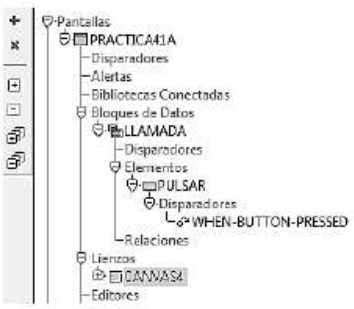
Practicando la función de apertura de página web

En esta práctica vamos a crear un formulario con un único botón que permita invocar la apertura de la página web siguiente: **www.oracle.es**

En primer lugar vamos a abrir Forms Builder con un nuevo formulario y dentro del elemento LIENZOS del navegador añadimos un lienzo.

Seguidamente nos situaremos en el elemento GRUPO DE OBJETOS del navegador de objetos de Forms y crearemos un *Bloque Manual* que denominaremos **LLAMADA**. Dentro de dicho bloque crearemos un único elemento de tipo BOTÓN que denominaremos **PULSAR** y que se visualice en el lienzo anterior. Dentro del botón programaremos un evento de tipo WHEN-BUTTON-PRESSED con el código siguiente:

```
BEGIN
      WEB.SHOW_DOCUMENT('http://www.oracle.es','_blank');
END;
```



Después de hacer esto, si desplegamos en el navegador de objetos el bloque **LLAMADA** tendríamos que tener lo siguiente.

A continuación almacenamos el formulario con el nombre **PRACTICA41A.FMB** y lo compilamos y generamos.

Cuando ejecutamos el formulario y pulsamos el botón **PULSAR** obtendremos el siguiente resultado.

APERTURA DE PROGRAMAS EXTERNOS EN FORMS

Forms también permite la llamada y apertura de programas externos invocando el ejecutable del mismo a través de dos Built-in:

- HOST (programa);
- CLIENT_HOST(programa);

La diferencia entre ambas es que la primera (HOST) abre el programa en el servidor, y la segunda (CLIENT_HOST) lo abre en el equipo del cliente. Para ejecutar comandos del lado del cliente, como CLIENT_HOST, el formulario debe tener asociadas las librerías de objetos de WEBUTIL.

El parámetro **programa** debe incluir, entre comillas, la ruta completa donde se encuentra el ejecutable del programa, así como el nombre del mismo con la extensión.

Cuando se ejecuta un comando HOST o CLIENT_HOST, el programa externo se abre en segundo plano, es decir, se abre pero se queda en la barra de tareas de Windows. Y hay que tener en cuenta que el foco sale de la aplicación para situarse en el programa externo. Hasta que no se cierre dicho programa, no se podrá volver a la aplicación para continuar con el trabajo en la misma.

ORACLE REPORTS. CONCEPTOS BÁSICOS 42

INTRODUCCIÓN

Oracle Reports es una herramienta que permite el diseño y la generación de informes, que contiene una herramienta de diseño de informes denominada Reports Builder.

Para ejecutar los informes generados con Reports Builder hay que tener un servidor de Oracle Reports donde se pueda ejecutar.

Los informes de Reports se pueden publicar en variedad de formatos, incluyendo HTML, XML, PDF, XLS, texto delimitado, PostScript y RTF, y se pueden enviar a diversos destinos incluyendo e-mail, navegador Web, Oracle Portal y almacenar como un archivo más dentro del administrador de archivos.

Además de los formatos indicados anteriormente, también se puede almacenar el listado en el formato nativo de Oracle Reports que es: RDF.

Un informe de Oracle puede contener los siguientes elementos:

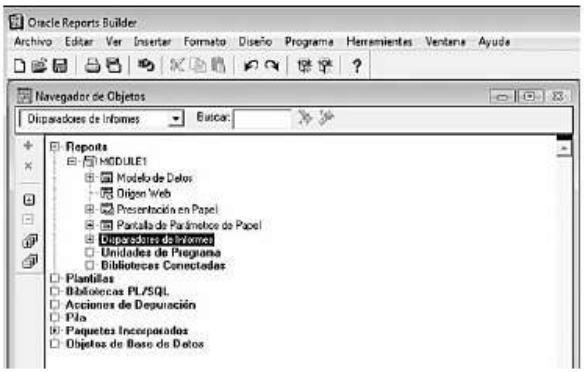
- Objetos del modelo de datos: consultas, grupos, columnas, enlaces y parámetros de usuario.
- Objetos de la imagen del listado: marcos de repetición, marcos, campos, etc.
- Parámetros y campos.
- Objetos de programación PL/SQL: unidades de programa, disparadores.
- Referencias a librerías externas PL/SQL.
- Código mostrado en la vista del navegador Web.

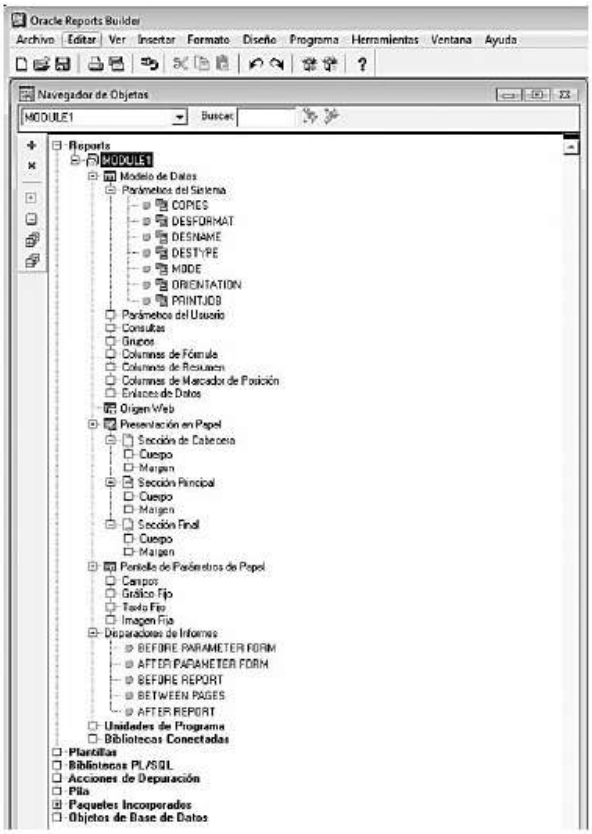
REPORT BUILDER

Es la herramienta de diseño de los informes.

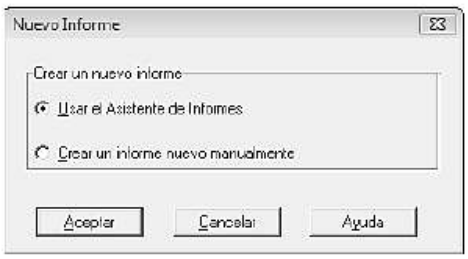
Para la apertura de esta herramienta hay que ejecutar el fichero **rwbuilder.bat** que normalmente se encontrará en la ruta **config\reports\bin** donde se haya instalado la instancia de Oracle.

Al igual que ocurre con Forms Builder, esta herramienta dispone de un navegador de objetos donde se distribuyen los distintos elementos que se crean en el informe. A continuación se muestran dos imágenes con la visión general de la herramienta y el navegador de objetos.





Para crear un informe en Report Builder tenemos disponibles dos opciones: Manual o Asistida.



Elementos del Navegador de objetos

Los distintos objetos que se pueden incorporar en un informe de Oracle Reports se agrupan en el navegador de objetos del Report Builder de la siguiente forma:

- **Modelo de Datos**: en este grupo se incluyen los siguientes elementos:
  - *Parámetros del Sistema*: parámetros comunes a todos los informes.
  - *Parámetros del Usuario*: parámetros definidos por el usuario en el desarrollo del informe.
  - *Consultas*: las querys que se definen para la captura de los datos del informe.
  - *Grupos*: agrupamiento de los datos a mostrar en el informe.
  - *Columnas de Fórmula*: definición de aquellas columnas cuyo resultado se obtiene a través de una fórmula.
  - *Columnas de Resumen*: definición de aquellas columnas cuyo resultado se obtiene a través de una función de acumulación (resumen) de SQL. Entre las funciones se encuentran: count, sum, avg, etc.
  - *Columnas de Marcador de Posición*: son aquellas columnas para las que el desarrollador define un tipo de dato y un valor en PL/SQL y cuyo valor se puede cambiar mediante programación.
  - *Enlaces de Datos*: definición de aquellos enlaces de conexión a otras bases de datos.
- **Origen Web**: permite mostrar el informe en formato web (.jsp, .xml, .html).
- **Presentación en Papel**: es donde se define el diseño del informe. Está dividido en 3 partes:
  - *Sección de Cabecera*.
  - *Sección Principal*.
  - *Sección Final*.
- **Pantalla de Parámetros de Papel**: es el área donde se personaliza la entrada de parámetros por parte del usuario, y que hacen cambiar el resultado del informe.
- **Disparadores de Informes**: son los eventos (disparadores) que se permiten utilizar dentro de un informe.
  - *BEFORE PARAMETER FORM*: se ejecuta antes de entrar en la pantalla de parámetros de papel.
  - *AFTER PARAMETER FORM:* se ejecuta después de salir de la pantalla de parámetros de papel.
  - *BEFORE REPORT:* se ejecuta antes del informe.
  - *BETWEEN PAGES:* se ejecuta con cada cambio de página dentro de un informe.
  - *AFTER REPORT:* se ejecuta después de terminar el informe.
- **Unidades de Programa**: son los diversos bloques de código PL/SQL definidos por el desarrollador para la ejecución del informe. Pueden ser procedimientos, funciones o paquetes.

- **Bibliotecas Conectadas**: son las distintas bibliotecas conectadas al informe y que se pueden utilizar para la ejecución del informe.

APERTURA DE INFORMES REPORTS DESDE FORMS

Forms también tiene utilidades para realizar llamadas a listados en formato Oracle Reports. Para ello se pueden utilizar los siguientes tres métodos:

- **RUN_REPORT_OBJECT**: permite invocar la ejecución de un report, pasándole además el nombre de la plataforma y el servidor de Report Server.
- **HOST**: para invocar la ejecución de un report con este comando hay que utilizar el ejecutable RWWUN.EXE. La ejecución del report se hará del lado del servidor.
- **CLIENT_HOST:** para invocar la ejecución de un report con este comando hay que utilizar el ejecutable RWWUN.EXE. La ejecución del report se hará del lado del cliente, por lo que el resultado de la ejecución de report se le mostrará directamente al cliente. Para utilizar el comando CLIENT_HOST hay que tener asociado al formulario las librerías de Webutil.

MODELOS DE INFORMES QUE SE PUEDEN CONSTRUIR

A continuación se presenta de forma gráfica una serie de ejemplos de informes que se pueden obtener con Oracle Reports.

Informes tabulares


YOUR Inc. COMPANY

Company Departments

Department Id	Department Name	Manager Id	Location Id
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700

Son los más habituales donde la información se presenta en columnas.

Informes de tipo etiquetas para correos

Este tipo de listados permiten generar etiquetas para correos.

Informes de tipo carta

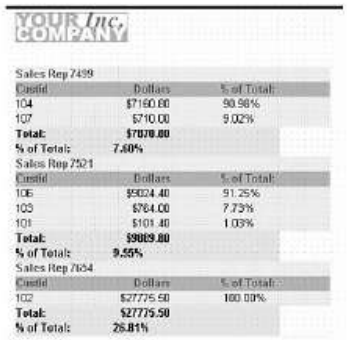
Este tipo de listados permiten generar una carta.

Informes de tipo maestro/maestro

Este tipo de listados muestran un listado en el que se visualizan, al menos, dos conjuntos de datos que no están directamente relacionados, a través de al menos dos consultas separadas.

Informes de tipo resumen de totalización

Este tipo de listados se utilizan para mostrar columnas de totalización de datos. Un informe de este tipo tiene que contener al menos una columna cuyo valor se obtenga a través de una fórmula resumen de totalización.


YOUR Inc. COMPANY

Sales Rep 7499

Custid	Dollars	% of Total:
104	$7160.80	90.98%
107	$710.00	9.02%
Total:	**$7870.80**	
% of Total:	**7.60%**	

Sales Rep 7521

Custid	Dollars	% of Total:
106	$9024.40	91.25%
103	$764.00	7.73%
101	$101.40	1.03%
Total:	**$9889.80**	
% of Total:	**9.55%**	

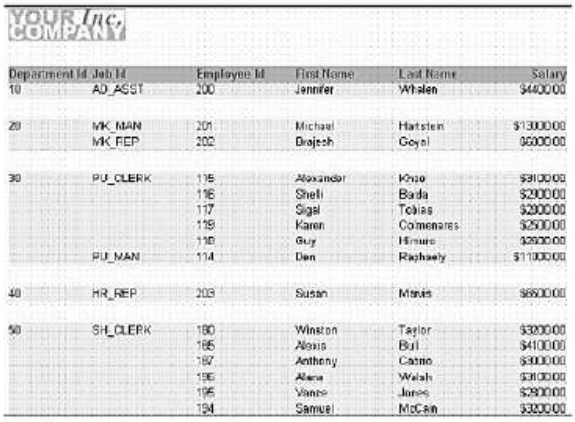
Sales Rep 7654

Custid	Dollars	% of Total:
102	$27775.50	100.00%
Total:	**$27775.50**	
% of Total:	**26.81%**	

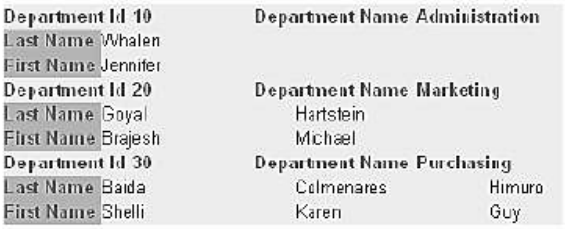
Entre las funciones de resumen se encuentran el sumatorio (SUM), conteo (COUNT), media (AVG), máximo (MAX), mínimo (MIN), etc.

Informes de consulta de grupo

Este tipo de listados utilizan al menos una consulta para agrupar información a la izquierda del listado o en la zona superior del mismo para una o más columnas.


YOUR Inc. COMPANY

Department Id	Job Id	Employee Id	First Name	Last Name	Salary
10	AD_ASST	200	Jennifer	Whalen	$4400.00
20	MK_MAN	201	Michael	Hartstein	$13000.00
	MK_REP	202	Brajesh	Goyal	$6000.00
30	PU_CLERK	115	Alexander	Khoo	$3100.00
		116	Shelli	Baida	$2900.00
		117	Sigal	Tobias	$2800.00
		118	Karen	Colmenares	$2500.00
		119	Guy	Himuro	$2600.00
	PU_MAN	114	Den	Raphaely	$11000.00
40	HR_REP	203	Susan	Mavris	$6500.00
50	SH_CLERK	180	Winston	Taylor	$3200.00
		185	Alexis	Bull	$4100.00
		187	Anthony	Cabrio	$3000.00
		196	Alana	Walsh	$3100.00
		195	Vance	Jones	$2800.00
		194	Samuel	McCain	$3200.00


Department Id 10 **Department Name Administration**
Last Name Whalen
First Name Jennifer
Department Id 20 **Department Name Marketing**
Last Name Goyal Hartstein
First Name Brajesh Michael
Department Id 30 **Department Name Purchasing**
Last Name Baida Colmenares Himuro
First Name Shelli Karen Guy

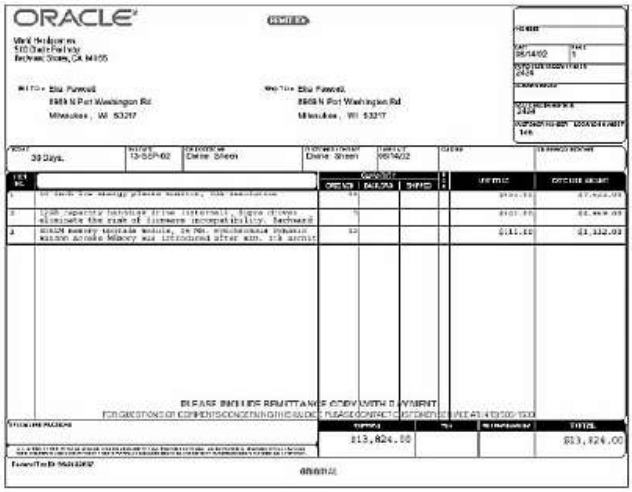
Informes con cabecera o pie de página


YOUR Inc. COMPANY **Employee Summary Report**

Department	First Name	Last Name	Employee Id	Salary
10	Jennifer	Whalen	200	$4400.00
	Total Salary for Department 10:	**$4400.00**		
20	Michael	Hartstein	201	$13000.00
	Pat	Fay	202	$6000.00
	Total Salary for Department 20:	**$19000.00**		
30	Den	Raphaely	114	$11000.00
	Alexander	Khoo	115	$3100.00
	Shelli	Baida	116	$2900.00
	Sigal	Tobias	117	$2800.00
	Guy	Himuro	118	$2600.00
	Karen	Colmenares	119	$2500.00
	Total Salary for Department 30:	**$24900.00**		
40	Susan	Mavris	203	$6500.00
	Total Salary for Department 40:	**$6500.00**		

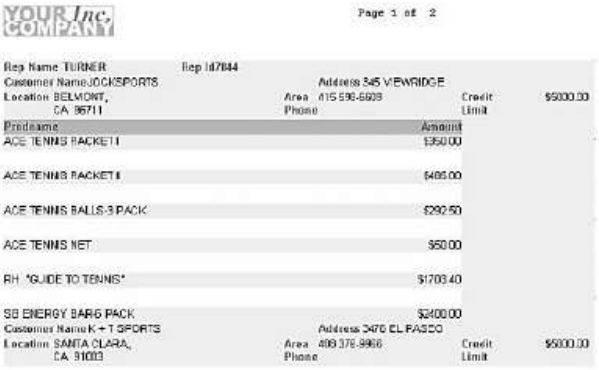
Este tipo de listados se utilizan para incorporar en la parte superior de la página una cabecera o en la parte inferior de la misma un pie de página. En ambos casos el texto o las imágenes incorporadas en la cabecera o el pie, se repetirán en todas las páginas.

Informes con gráficos, texto y color



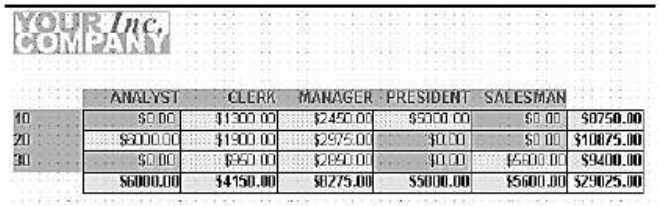
Este tipo de listados se utilizan para imprimir informes complejos que incorporan gráficos, texto y zonas en color. Normalmente se usan para emitir facturas o formularios preimpresos.

Informes con páginas numeradas



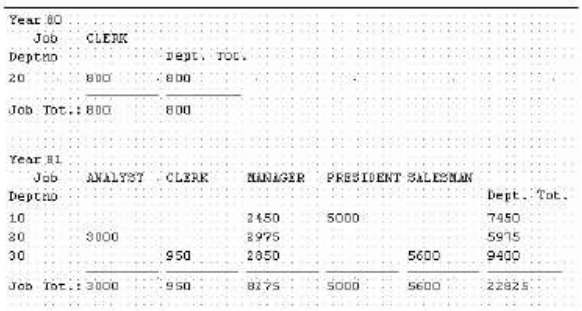
Sobre cualquiera de los listados se pueden incorporar funciones para la numeración de las páginas, como se muestra en el siguiente ejemplo.

Informes de tipo matriz de datos



Un informe de tipo matriz se asemeja a una hoja de cálculo, donde la información se presenta en formato de filas y columnas.

Informes combinados con matrices y funciones de grupo



Year 80
Job CLERK
Deptno Dept. Tot.
20 800 800
Job Tot.: 800 800

Year 81
Job ANALYST CLERK MANAGER PRESIDENT SALESMAN
Deptno Dept. Tot.
10 2450 5000 7450
20 3000 2975 5975
30 950 2850 5600 9400
Job Tot.: 3000 950 8275 5000 5600 22825

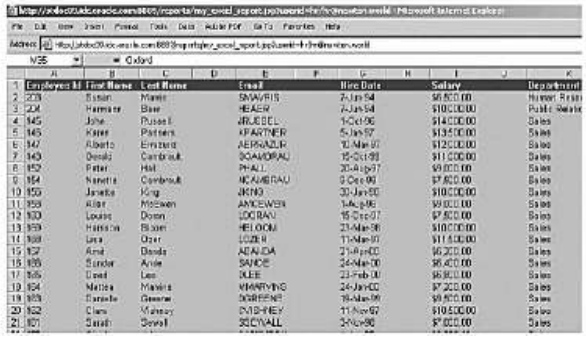
También es posible crear listados que combinen funciones de grupo y elementos de tipo matriz como en el ejemplo siguiente.

Informes de tipo cálculo de series temporales

Custid	Shipdate	Total	4-Month Moving Average
100	30-JUL-86	$3.40	$3.40
	15-AUG-86	$97.50	$50.45
	01-JAN-87	$730.00	$730.00
	12-MAR-87	$4,460.00	$2,590.00
101	08-JAN-87	$101.40	$101.40
102	05-JUN-86	$224.00	$224.00
	20-JUN-86	$56.00	$140.00
	11-JAN-87	$46.00	$46.00
	05-FEB-87	$23,940.00	$11,992.50
	06-MAR-87	$3,510.50	$9,165.17
103	10-FEB-87	$764.00	$764.00
104	18-JUL-86	$5.60	$5.60
	25-JUL-86	$35.20	$20.40

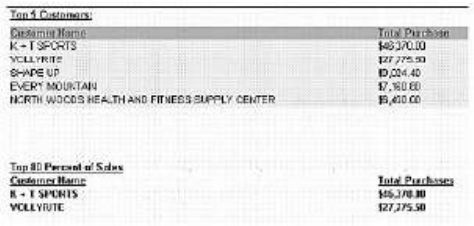
También es posible crear listados que realicen cálculos para series temporales, como en el ejemplo siguiente donde se obtiene la media de ventas de los últimos cuatro meses para cada cliente.

Informes emitidos con formato de salida hoja Excel



Este tipo de listados se diseñan como cualquier otro, pera la salida del mismo en vez de generarse en formato estándar de Report, se genera en formato Excel, que permite ser visualizado dentro de un navegador web.

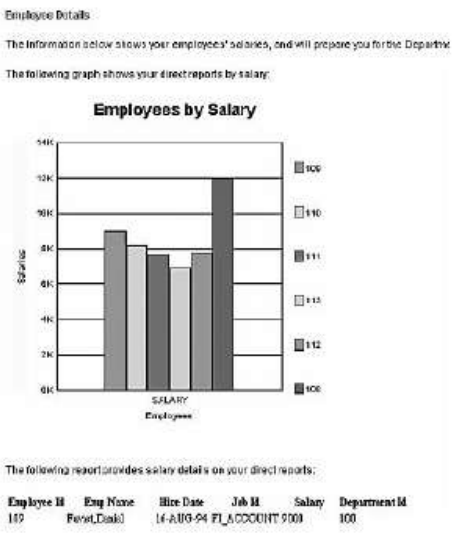
Informes de tipo ranking



Este tipo de listados se diseñan para obtener ranking de datos. En el ejemplo se están obteniendo un listado con dos rankings: uno por total de ventas y otro por porcentaje de ventas.

Informes de tipo formulario de parámetros para un JSP

Este tipo de listados permiten crear un formulario simple en HTML que después creará un formulario de parámetros JSP basado en un modelo de datos en Report Builder.



Informes con códigos de barras

Este tipo de listados permiten incorporar al mismo un código de barras a través de un Java bean.

CREANDO UN INFORME ASISTIDO 43

INTRODUCCIÓN

En este capítulo vamos a aprender a crear un informe asistido con la herramienta Report Builder.

CREANDO UN INFORME ASISTIDO

En primer lugar tenemos que abrir el programa de diseño de informe **Report Builder**.

Al hacerlo nos aparecerá un cuadro de diálogo como el que se muestra a continuación para seleccionar la opción que deseamos en la creación del informe. Dejaremos marcada la opción por defecto de **Usar el Asistente de Informes** y pulsaremos en **Aceptar.**

Bienvenida al asistente de informes

El primer cuadro de diálogo que se muestra dentro del asistente de informes es el de bienvenida.

Este cuadro de diálogo se puede deshabilitar para futuras ocasiones pulsando en el check **Mostrar esta página la próxima vez.**

En nuestro caso vamos a dejarle puesta la marca con el fin de que siga mostrándose y pulsamos el botón **Siguiente** para comenzar el proceso de creación del informe.

Seleccionando el tipo de disposición del informe

El primer paso del diseño de un informe es identificar para qué tipo de dispositivo queremos obtener el listado: para visualizar en pantalla (Diseño Web), para presentar en papel, o combinadas ambas opciones.

Esto es lo que se muestra en el cuadro de diálogo que se presenta al comenzar el asistente.

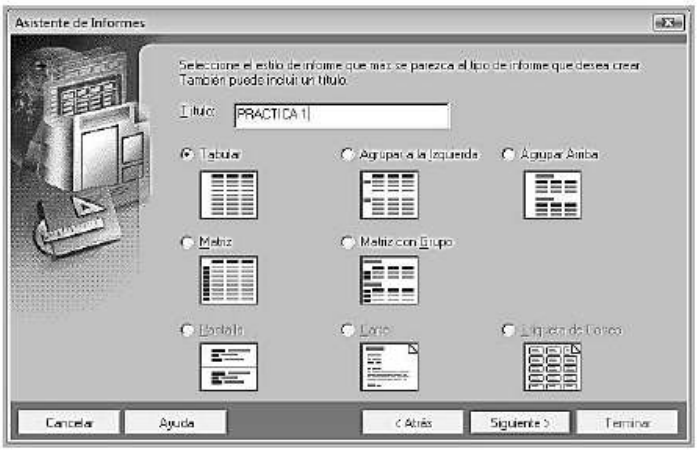
En nuestro ejemplo vamos a seleccionar la opción que viene por defecto: **Crear Diseño Web y Presentación en Papel**. A continuación pulsamos el botón **Siguiente.**

Seleccionando el estilo del informe

Seguidamente tenemos que seleccionar el estilo de presentación de datos dentro del informe entre las siguientes opciones:

- Tabular
- Agrupar a la Izquierda.
- Agrupar Arriba.
- Matriz.
- Matriz con Grupo.

- Pantalla.
- Carta.
- Etiqueta de Correo.



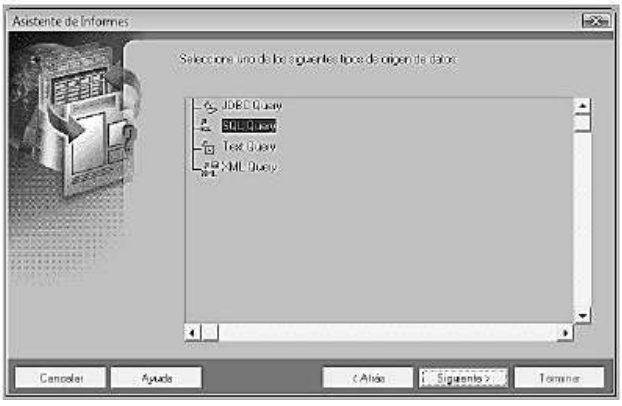
Las opciones Pantalla, Carta y Etiqueta de Correo no se podrán seleccionar si la disposición del listado es vía Web o combinada esta con papel. Solo se habilitarán cuando se seleccione una disposición única en papel.

En nuestro ejemplo dejaremos marcada la opción por defecto: **Tabular** y en el título del listado escribiremos **PRACTICA 1**. A continuación pulsaremos el botón **Siguiente**.

Seleccionando el origen de datos

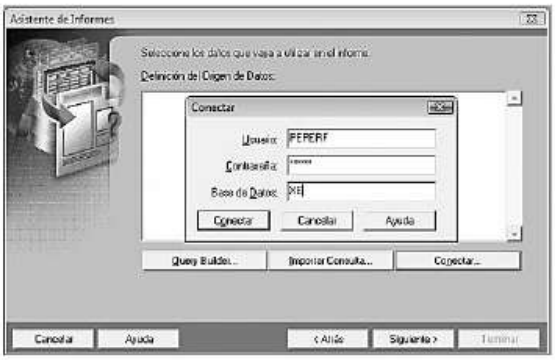
Seguidamente tenemos que seleccionar el origen de la información que aparecerá dentro del informe. Los posibles orígenes de datos son:

- Consulta JDBC.
- Consulta SQL.
- Consulta de texto.
- Consulta XML.



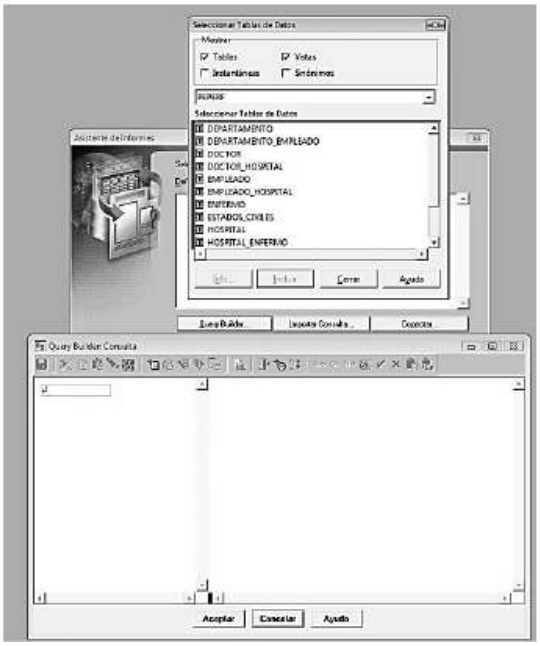
En nuestro ejemplo dejaremos marcada la opción por defecto: **SQL Query** y pulsaremos el botón **Siguiente**.

Diseñando una consulta SQL



Seguidamente tenemos que diseñar la consulta que obtendrá la información a presentar en el listado. Para ello antes hay que conectarse a la base de datos donde se encuentran las tablas necesarias, así que pulsaremos el botón **Conectar** que nos presentará esta pantalla.

En el usuario indicaremos **PEPERF**, en la contraseña **PEPITO,** y si hemos instalado la base de datos Oracle 11G Express Edition (como se indica en los anexos de este libro), entonces indicaremos en el apartado Base de Datos: **XE**. Una vez introducidos estos datos pulsaremos el botón **Conectar.**



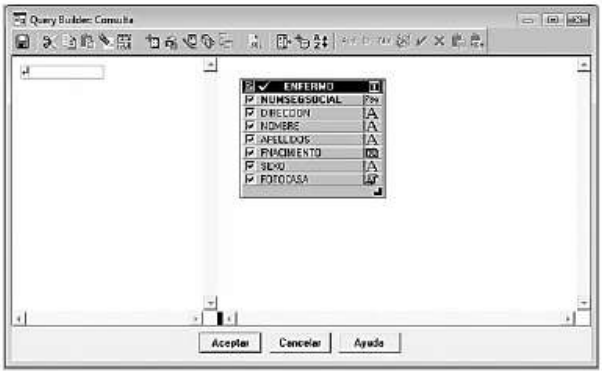
A continuación pulsamos el botón **Query Builder** para comenzar a diseñar la consulta. Al hacerlo se presentará una imagen como la que se muestra a continuación con una lista de las tablas a las que tiene permisos de selección el usuario PEPERF.

Como se puede observar en esta imagen, el proceso de diseño de la consulta asistida consta de dos pantallas diferenciadas:

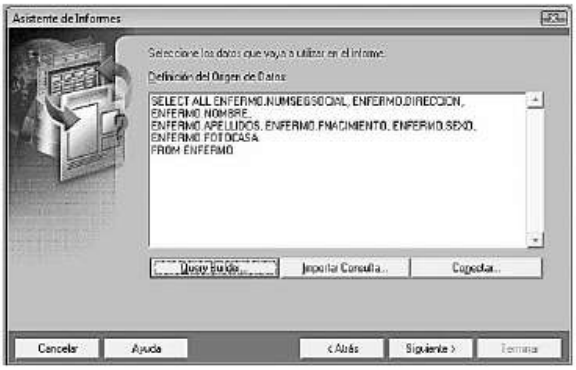
- **Seleccionar Tablas de Datos**: en este cuadro de diálogo se muestran los distintos elementos del usuario conectado a los que se tiene acceso de consulta (SELECT). Se pueden visualizar tablas, vistas, instantáneas e sinónimos. Por defecto aparece marcado que se visualicen solo tablas y vistas. Este cuadro de diálogo permite seleccionar las tablas que queremos incluir en la consulta.
- **Query Builder-Consulta:** en este espacio es donde se diseña la propia consulta: se indican criterios sobre las columnas, se marcan aquellas columnas que se quieren visualizar, etc.

Para seguir con nuestro ejemplo, en el cuadro de diálogo **Seleccionar Tablas de Datos** vamos a marcar la tabla **ENFERMO** y a continuación pulsamos el botón **Incluir**

y después pulsamos el botón **Cerrar** porque ya no vamos a incluir más tablas en la consulta.



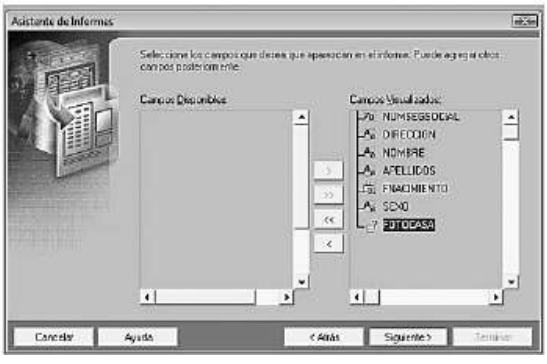
El resultado supone que se muestran, en el cuadro de diálogo **Query Builder**, las columnas de la tabla EMPLEADO, para que marquemos (en el check que aparece a la izquierda de cada columna), aquellas que queremos visualizar en la consulta. Vamos a marcar que queremos ver todas las columnas de la tabla. Para ello pulsaremos en el check que hay a la izquierda del nombre de la tabla, para que se marquen todas las columnas.



Para finalizar el proceso de diseño de la consulta pulsamos el botón **Aceptar,** lo que nos devolverá a la pantalla inicial del proceso de diseño, donde se mostrará la consulta SQL a la que se convierte el diseño que hemos realizado, como se muestra en la imagen de la página siguiente.

Para continuar con el proceso de creación del informe pulsaremos el botón **Siguiente**.

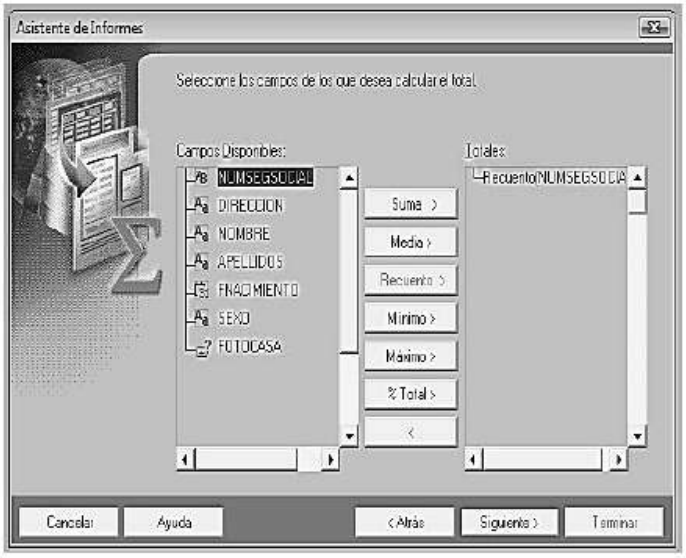
Mostrando columnas de la consulta en el listado



Seguidamente tenemos que seleccionar cuáles de las columnas de la consulta queremos que también se impriman en el listado.

En nuestro ejemplo seleccionamos que todos los campos de la consulta se van a imprimir en el listado y para ello pulsamos en el botón >> de la pantalla anterior, lo que provoca que se trasladen los campos de **Campos Disponibles** a **Campos Visualizados**. A continuación pulsamos el botón **Siguiente**.

Creando totales

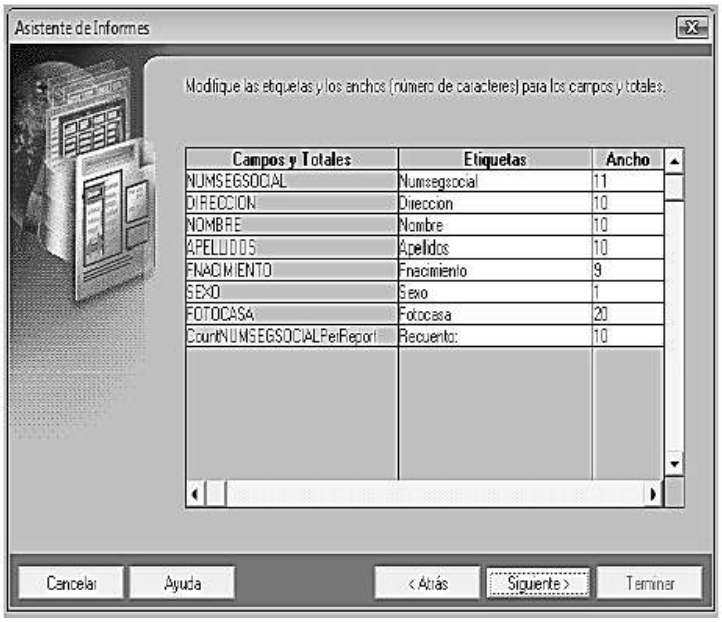


Seguidamente podemos crear aquellas funciones de totalización que consideremos oportunas para el resultado del listado.

En nuestro ejemplo vamos a crear un **Recuento** sobre la columna **NUMSEGSOCIAL** como se indica en esta pantalla.

Para conseguirlo primero pulsamos en la columna **NUMSEGSOCIAL** y luego en el botón **Recuento.** Para terminar este paso pulsamos el botón **Siguiente.**

Modificando los títulos de las columnas a mostrar

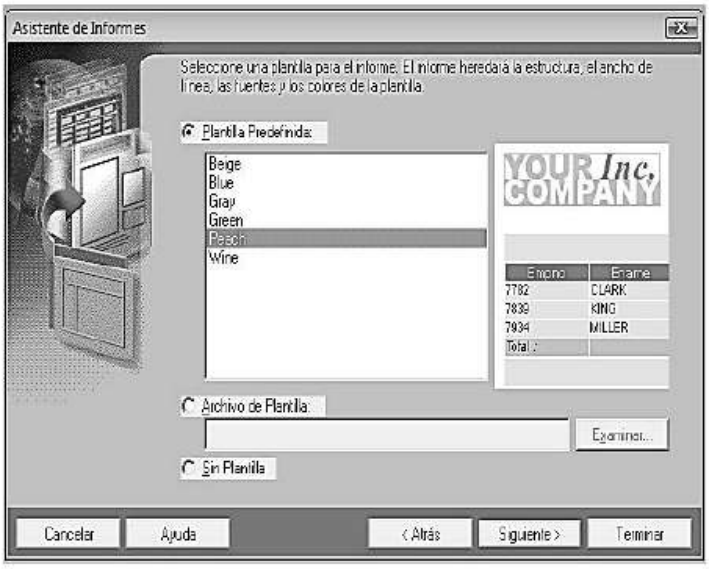


Seguidamente podemos modificar el título de cada una de las columnas que se van a visualizar en el listado.

En nuestro ejemplo mantenemos los mismos títulos que se muestran en la pantalla del asistente.

Pulsamos el botón **Siguiente** para continuar con el proceso de creación del listado.

Utilizando una plantilla de diseño para la tonalidad del listado

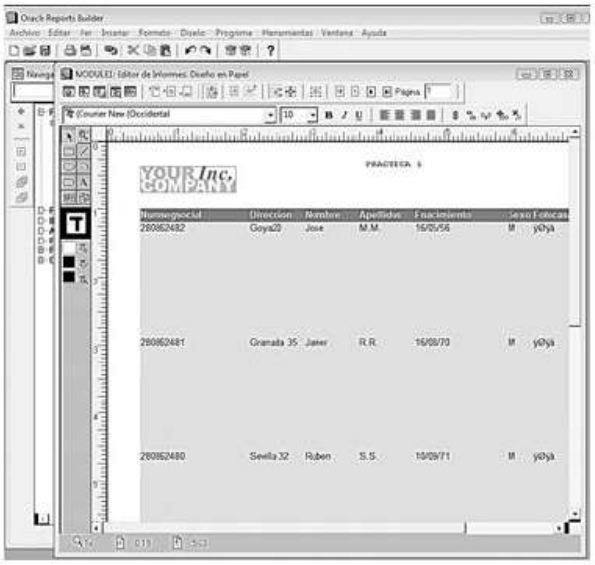


Seguidamente podemos seleccionar una de las plantillas preestablecidas para elegir la tonalidad y el estilo de presentación de los datos del listado, o bien seleccionar una que ya hayamos creado, e incluso no utilizar ninguna plantilla.

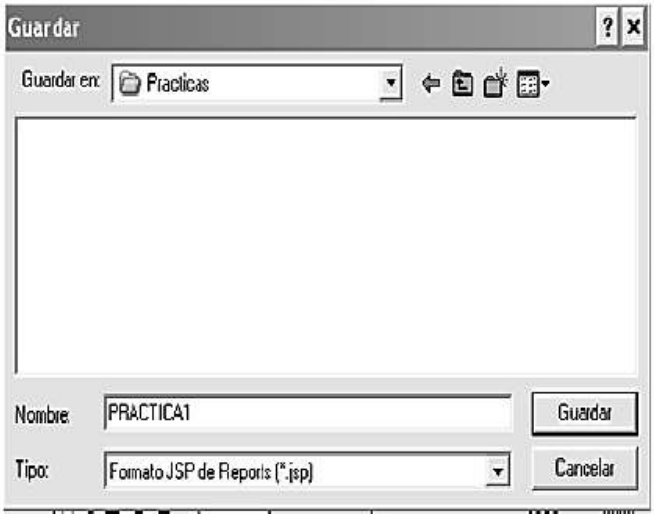
En nuestro ejemplo vamos a seleccionar la plantilla **Peach** (tonalidad melocotón).

Para continuar con el proceso pulsamos el botón **Siguiente** y habremos concluido el proceso de diseño del listado. Para ello el asistente nos lo recuerda con esta pantalla.

Pulsamos el botón de **Terminar** y seguidamente se genera el listado como se muestra en la siguiente pantalla.



Guardando el listado



Antes de salir de la herramienta Report Builder tenemos que guardar el trabajo, para ello pulsamos en el menú **Archivo**, y dentro de este en la opción **Guardar,** que nos presentará el siguiente cuadro de diálogo.

Guardamos la práctica con el nombre **PRACTICA1.JSP** y pulsamos el botón de **Guardar.**

Seguidamente volvemos a realizar la operación de guardado: **Archivo-Guardar Como**, pero cuando se presente el cuadro de diálogo anterior

Cambiamos el tipo de formato de almacenamiento del report y seleccionamos **Binario de reports (*.rdf)** y pulsamos el botón de **Guardar.**

A continuación podemos cerrar la herramienta de Report Builder y salir.

MÉTODOS DE EJECUCIÓN 44

INTRODUCCIÓN

Un informe (report) se puede ejecutar dentro del diseñador de informes (Report Builder), como hemos visto en la práctica anterior, pero también es posible ejecutar un informe desde fuera del Report Builder, utilizando comandos.

En este capítulo aprenderemos estas dos formas de ejecutar informes y completaremos una práctica de ejemplo de cada tipo.

FORMATOS DE EJECUCIÓN DEL REPORT BUILDER

Dentro de la aplicación para diseñar informes (Report Builder) disponemos de dos métodos de ejecución del informe, una vez finalizado el diseño del mismo:

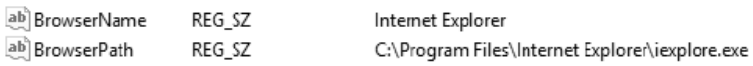
- Ejecución del diseño web.
- Ejecución de la presentación en papel.

Ejecución del diseño web

Para ejecutar un informe en modo diseño web, previamente tenemos que tener levantado el **WLS_REPORTS** local, como se indica en el capítulo 3 del libro.

Así mismo, tenemos que configurar el explorador que queremos utilizar para visualizar las ejecuciones de informes mediante esta herramienta. Para ello tenemos

que abrir desde Windows la utilidad **Regedit** (editor del Registro de Windows). Dentro de la misma nos situaremos en la carpeta HKEY_CURRENT_USER – SOFTWARE – ORACLE. Si en la carpeta Oracle no existe una denominada **Toolkit,** tendremos que crearla (con la opción **Nueva-Clave**). Dentro de la carpeta TOOLKIT comprobaremos si existe una carpeta **TKBrowser**, y si no existe la crearemos también. Por último dentro de la carpeta TKBROWSER tendremos que indicar los siguientes valores de configuración para asociar el navegador por defecto que queremos utilizar en Reports Builder:



El parámetro **BrowserName** contendrá el nombre del navegador y el parámetro **BrowserPath** la ruta completa y el nombre del ejecutable que lanza el navegador.

La ejecución en modo diseño web dentro de la herramienta Reports Builder, se realiza desde el menú **Programa – Ejecutar Diseño Web**, o bien pulsando el botón  que se encuentra en la barra de herramientas icónicas del Reports Builder.

Al ejecutar el informe de esta forma, se obtiene el resultado del mismo dentro del navegador que se haya indicado por defecto.

Ejecución de la presentación en papel

Para ejecutar un informe en modo presentación en papel tenemos que entrar en el menú **Programa – Ejecutar presentación en papel,** o bien pulsando el botón  que se encuentra en la barra de herramientas icónicas del Report Builder.

El ejecutar el informe de esta forma, se obtiene el resultado dentro del propio Report Builder pero con el aspecto con el que se imprimiría en papel dicho informe.

MÉTODOS DE EJECUCIÓN DE UN REPORT POR LÍNEA DE COMANDOS

Desde fuera del Report Builder también es posible ejecutar un informe report que se ha creado ya con extensión .rdf o .jsp. Para ello se dispone de dos métodos:

- Ejecución del informe en la máquina local.
- Ejecución del informe en un servidor remoto (Report Server).

En ambos casos la ejecución se realiza a través de la línea de comandos del sistema operativo, o bien se podrá generar un fichero ejecutable (por ejemplo .bat en Windows) donde se incluyan los comandos a ejecutar.

Ejecutar un informe en la máquina local

Para ejecutar un informe en la máquina local, fuera del Reports Builder, previamente tenemos que tener levantado el **WLS_REPORTS** local, como se indica en el capítulo 3 del libro.

Para ejecutar el informe en local tendremos que utilizar el comando **rwrun.exe** con los parámetros deseados para la salida del informe. A continuación se muestra una lista de los parámetros más utilizados:

Parámetro	Descripción	Valores
ACCESIBLE	Indica si están disponibles las características de accesibilidad ofrecidas a través de Oracle Reports	YES NO *(p. defecto)*
ARRAYSIZE	Especifica el tamaño en kb para usar con los arrays de Oracle que se estén procesando. Generalmente un valor mayor de tamaño de array hace que el informe se ejecute más rápido	1 a 9999 10 *(p. defecto)*
AUTHID	Especifica si el nombre de usuario y el password se utilizarán para restringir el acceso a Reports Server solo para usuarios autenticados.	*Username/Password*
AUTOCOMMIT	Especifica si los cambios en la b.d. deberían ser confirmados automáticamente.	YES NO *(p. defecto)*
BCC	Especifica los receptores del email de la Copia Cortesía Blindada. Es decir los receptores que aparecen aquí no pueden visualizar el nombre del resto de los receptores.	*Nombres de e-mails entre comillas y separados por comas.* *BCC="emailid" o BCC= ("emailid","emailid",...)*

BLANKPAGES	Especifica si se suprimen las páginas en blanco cuando se imprime un informe.	YES *(p. defecto)* NO
BUFFERS	Especifica el tamaño de la memoria caché virtual en kb.	1 .. 9999 640 *(p. defecto)*
CC	Especifica los receptores del email de la Copia de Cortesía.	*Nombres de e-mails entre comillas y separados por comas.* *BCC="emailid" o BCC= ("emailid","emailid",...)*
CELLWRAPPER	Especifica el/los caracteres a visualizar en celdas delimitadas en la salida de los informes. Los caracteres elegidos aparecerán a ambos lados de la celda, delimitándola.	- Cualquier carácter alfanumérico o conjunto de caracteres alfanuméricos. - " - ‘ - tab - space - return - none *(p.defecto)* - \t - \n
CMDFILE	Especifica un fichero que contiene una línea de argumentos para la ejecución de un report desde línea de comandos. El fichero debe estar en formato ascii.	*Nombre de fichero con su extensión.*
COLLATE	Especifica el método de intercalado de las copias en la salida por impresora del informe. Si se indica YES el método de intercalado de las copias sería 123 | 123 … En cambio si se indica NO, el método de intercalado de copias sería 111 | 222 | 333 …	YES *(p. defecto)* NO
COPIES	Especifica el número de copias de la salida del informe.	1 .. 9999
CUSTOMIZE	Especifica un fichero XML de Oracle Reports para ser ejecutado en el informe actual.	*Nombre de fichero.xml*

DATEFORMATMASK	Especifica la máscara de formato de salida de los campos de fecha.	*Máscara de formato*
DELIMITER	Especifica el carácter que se utiliza para separar las celdas en la salida del informe.	- Cualquier carácter alfanumérico o conjunto de caracteres alfanuméricos. - , - . - tab *(p. defecto)* - space - return - none - \t - \n
DESFORMAT	Especifica el formato de salida del informe.	**DFLT:** se envía a un fichero utilizando los drivers de la impresora por defecto (p.e: un PostScript driver genera un formato de salida PostScript).
		DELIMITED: se envía a un fichero con campos delimitados para que se pueda leer por hojas de cálculo como Excel. **DELIMITEDDATA:** la misma función que DELIMITED pero utilizada con informes de gran volumen. **HTML:** se envía a un fichero en formato HTML. **HTMLCSS:** se envía a un fichero que incluye una hoja de estilos HTML. **PDF:** se envía a un fichero PDF. **RTF:** se envía a un fichero RTF. **SPREADSHEET:** se envía a un fichero HTML que es compatible con aplicaciones de hojas de cálculo como Excel. **ENHANCEDSPREADSHEET:** la misma función que SPREADSHEET pero en un formato mejorado. Se utiliza también con informes de gran volumen. **XML:** se envía a un fichero XML.
		Definición de impresora: este formato se utiliza cuando DESTYPE=FILE y DESNAME=nombre_fich. Si

		MODE=BITMAP aquí aparecerá el nombre de la impresora, pero si el MODE=CHARACTER, aquí aparecerá el nombre del fichero de definición de la impresora (fichero .PRT).
DESNAME	Especifica el nombre de la caché, fichero, impresora, servidor WebDAV o identificador de email al que se envía el informe.	*Nombre destino válido.* (Ej: DESNAME=printer,LPT1:) Por defecto asume el valor del parámetro del sistema DESNAME que se haya indicado en Report Builder.
DESTI NATI ON	Especifica el nombre del fichero XML que define la distribución de la ejecución del informe.	*Nombre de fichero.xml*
DESTYPE	Especifica el tipo de dispositivo que recibirá la salida del informe.	**CACHE:** envía la salida a la caché del navegador web. **FILE:** envía la salida al fichero cuyo nombre se indique en el parámetro DESNAME. **PRINTER:** envía la salida a la impresora indicada en DESNAME. **MAIL:** envía la salida por mail a los usuarios indicados en DESNAME. **ORACLEPORTAL:** envía la salida a Oracle Portal. **FTP:** envía la salida a un servidor FTP.
		WEBDAV: envía la salida a un servidor WebDAV. **Nombre_destino_personal:** si se ha creado un destino con la herramienta Reports Destination API, aquí se puede indicar su nombre. Por defecto asume el valor del parámetro del sistema DESTYPE que se haya indicado en el Report Builder.
DI STRI BUTE	Especifica si se habilita (YES) o no la distribución del informe a múltiples destinos utilizando un fichero XML.	YES NO *(p. defecto)*
FROM	Especifica la dirección e-mail del origen de envío del informe.	*Dirección e-mail*

MODE	Especifica si el informe se ejecuta en modo carácter o bitmap.	**BITMAP** **DEFAULT** *(p. defecto se utiliza el modo del sistema indicado en el Report B.)* **CHARACTER**
MODULE| REPORT	Especifica el nombre del informe a ejecutar.	*Nombre del fichero con extensión .rep, .rdf, .jsp o .xml*
NONBLOCKSQL	Especifica si se permite a otros programas ejecutar los mismos datos que se están recuperando en la consulta de Base de Datos incluida en el informe.	YES *(p. defecto)* NO
NUMBERFORMATMASK	Especifica la máscara de formato de salida de los campos de tipo número.	*Máscara de formato*
ONFAI LURE	Especifica si se realiza un COMMIT o ROLLBACK cuando se produce un error al ejecutar el informe.	COMMIT ROLLBACK NOACTION *(p. defecto)*
ONSUCCESS	Especifica si se realiza un COMMIT o ROLLBACK cuando se ejecuta correctamente el informe.	COMMIT ROLLBACK NOACTION *(p. defecto)*
ORI ENTATI ON	Especifica la dirección de visualización del informe.	**DEFAULT** *(p. defecto usa la configuración de la orientación de la impresora).* **LANDSCAPE:** horizontal. **PORTRAIT:** vertical
OUTPUTI MAGI NEF ORMAT	Especifica el formato de las imágenes incluidas en el informe.	PNG JPEG *(p. defecto)* JPG GIF BMP
PAGESI ZE	Especifica las dimensiones (ancho y alto) de la página física donde se visualizará el informe.	*Ancho x alto.* *Por defecto si tiene MODE=BITMAP el tamaño será 8.5 x 11 pulgadas. Y si el MODE=CHARACTER el tamaño será de 80 x 66 caracteres.*
PARAMFORM	Especifica si se visualiza el formulario de parámetros cuando se ejecuta un report.	**YES:** se visualiza **NO:** *(p. defecto)* no se visualiza. **HTML:** se visualiza en formato HTML.
PDFCOMP	Especifica si se comprime el PDF resultante.	***Un valor de 0 a 9**, donde 0 es sin compresión y 9 la máxima*

		compresión. **YES** *(p. defecto* se comprime a 6) **NO** *(sin compresión)*
READONLY	Requiere consistencia de lectura en las consultas del informe.	**YES:** requiere consistencia **NO:** *(p. defecto)* no se requiere.
REPORT | MODULE	Especifica el nombre del informe a ejecutar.	*Nombre del fichero con extensión .rep, .rdf, .jsp o .xml*
ROLE	Especifica el ROL de base de datos que se utilizará para ejecutar el informe.	*Un nombre de rol de b.d[/password]*
RUNDEBUG	Especifica si se realiza un chequeo extra antes de la ejecución del informe para determinar posibles anomalías en el mismo (que no son errores).	YES *(p. defecto)* NO
SAVE_RDF	Especifica un nombre de fichero RDF donde se almacenará el resultado de combinar un informe Report XML con un fichero RDF ya existente.	*Nombre de fichero.rdf*
SQLTRACE	Especifica si se tracea el código SQL del informe.	YES NO *(p. defecto)*
SUBJECT	Especifica el asunto del e-mail.	*Texto del asunto*
TRACEFILE	Especifica el nombre del fichero donde se almacena el resultado de tracear el informe.	*Nombre del fichero de traza.* **rwserver.trc** o **rwEng-x.trc** *(p.defecto)*
TRACEMODE	Especifica cómo se añade la información a un fichero de traza existente.	**TRACE_APPEND:** *se añade al final de la información existente.* **TRACE_REPLACE:** *se reemplaza todo el contenido del fichero por el nuevo resultado de la traza.*
USERID	Especifica el nombre y opcionalmente la password y el enlace de base de datos para ejecutar el informe.	*Nombre usuario[/password][@enlace_baseda tos]*
USERSTYLES	Especifica si existe un fichero externo de estilos (.css) para el informe, cuando se genera en formato HTMLCSS.	YES *(p. defecto)* NO

EJEMPLOS DE EJECUCIÓN DE UN INFORME EN UNA MÁQUINA LOCAL

A continuación se muestran diversos ejemplos de ejecución de un informe mediante el comando rwrun.exe para una máquina local.

La mejor opción para probar cada ejemplo es editar el fichero **rwrun.bat** que se habrá creado en la máquina local junto con la instalación del software de Oracle Forms, y añadir el contenido que se muestra en cada caso, para posteriormente ejecutar dicho fichero.

En el siguiente ejemplo se genera un informe a partir del diseño **practica1.rdf**, en formato **PDF** y el resultado se almacena en un fichero denominado **d:\pruebas.pdf**.

```
rwrun.exe report=d:\practica1.rdf userid=PEPERF/PEPITO@XE
desformat=pdf destype=file desname="d:\pruebas.pdf"
```

En el siguiente ejemplo se genera un informe a partir del diseño **practica1.rdf**, en formato **PDF** y el resultado se envía a una impresora denominada **PDFCreator**, generándose 2 copias del informe.

```
rwrun.exe report=d:\practica1.rdf userid=PEPERF/PEPITO@XE
copies=2 desformat=pdf destype=printer desname=PDFCreator
```

En el siguiente ejemplo se genera un informe a partir del diseño **practica1.rdf**, en formato de texto **DELIMITADO** y el resultado se almacena en un fichero denominado **d:\pruebas.txt.** Como carácter delimitador entre los campos se utiliza el símbolo **;** (punto y coma).

```
rwrun.exe report=d:\practica1.rdf userid=PEPERF/PEPITO@XE
destype=file desformat=DELIMITED delimiter=;
desname="d:\pruebas.txt"
```

En el siguiente ejemplo se genera un informe a partir del diseño **practica1.rdf**, en formato **SPREADSHEET** (hoja de cálculo) y el resultado se almacena en un fichero denominado **d:\pruebas.xls**.

```
rwrun.exe report=d:\practica1.rdf userid=PEPERF/PEPITO@XE
destype=file desformat=SPREADSHEET desname="d:\pruebas.xls"
```

En el siguiente ejemplo se genera un informe a partir del diseño **practica1.rdf**, en formato **PDF** y el resultado se envía por email a los siguientes destinatarios: **emp1@comp.com** y **emp2@comp.com** con copia para **emp3@comp.com** y con cópia oculta para **mgr@comp.com**. El origen del correo partirá de **me@comp.com**.

```
rwrun.exe report=d:\practica1.rdf
userid=userid=PEPERF/PEPITO@XE desformat=PDF destype=MAIL
desname="emp1@comp.com","emp2@comp.com" cc="emp3@comp.com"
bcc="mgr@comp.com" from="me@comp.com"
```

En el siguiente ejemplo se genera un informe a partir del diseño **test.rdf** y se distribuye a los destinos indicados en el fichero xml **c:\mydistribute.xml**.

```
rwrun.exe report=test.rdf userid=scott/tiger@mydb
DISTRIBUTE=yes DESTINATION=c:\mydistribute.xml
```

En el siguiente ejemplo se genera un informe a partir del diseño **practica1.rdf**, en formato **RTF** y el resultado se almacena en un fichero denominado **d:\pruebas.rtf**.

```
rwrun.exe report=d:\practica1.rdf userid=PEPERF/PEPITO@XE
destype=file desformat=RTF desname="d:\pruebas.rtf"
```

En el siguiente ejemplo se genera un informe a partir del diseño **practica1.rdf**, utilizando el fichero de comandos **d:\prueba.cmd**.

```
rwrun.exe report=d:\practica1.rdf userid=PEPERF/PEPITO@XE
cmdfile="d:\prueba.cmd"
```

El fichero de comandos **d:\prueba.cmd** contiene la siguiente información, donde se indica que el formato del informe es **HTML** y el resultado se almacenará en **d:\pruebas.html**.

```
destype=file desformat=HTML desname="d:\pruebas.html"
```

Ejecutar un informe en un servidor remoto Report Server

Para ejecutar un informe en un servidor remoto de Report Server, hay que utilizar el comando **rwclient.exe,** con los mismos parámetros que se han indicado en el punto anterior para la ejecución de un informe contra la máquina local.

Los ejemplos vistos anteriormente serían también válidos para ejecutar en un servidor remoto, si sustituimos rwrun.exe por **rwclient.exe**, y además introducimos a continuación el parámetro SERVER=nombre_servidor. Por ejemplo:

```
rwclient.exe server=myrepserv report=d:\practica1.rdf
userid=PEPERF/PEPITO@XE desformat=pdf destype=file
desname="d:\pruebas.pdf"
```

PREFERENCIAS DE EJECUCIÓN DEL REPORT BUILDER 45

INTRODUCCIÓN

Oracle Reports Builder posee un cuadro de preferencias preestablecidas para la creación de cualquier nuevo report, y que afectan también a los parámetros de la línea de comandos (RWRUN o RWCLIENT), cuando no se indican expresamente.

En este capítulo aprenderemos a modificar estas preferencias.

APERTURA DEL CUADRO DE PREFERENCIAS

Para modificar las preferencias del Report Builder, en primer lugar tenemos que abrir el propio Report Builder. A continuación dentro del menú seleccionaremos la opción **Editar – Preferencias**, lo que hará que se muestre este cuadro de diálogo.

Las preferencias del Report Builder se agrupan en 5 solapas:

- General.
- Acceso.
- Asistentes.
- Valores de Ejecución.
- Parámetros de Ejecución.

Preferencias de la solapa General

En la solapa **General** nos encontramos con las siguientes opciones de configuración del Report Builder:

- **Suprimir Editor de Informes al Abrir**: por defecto aparece desmarcada para permitir que el editor de informes se abra automáticamente al abrir un nuevo informe.
- **Suprimir Inspector de Propiedades al Crear Objetos**: por defecto aparece marcada para que no se abra el editor de propiedades automáticamente cuando se crea un nuevo objeto dentro del informe.
- **Suprimir Recuperación de Listas al Acceder a Recuadros de Diálogo**: por defecto aparece desmarcado para permitir que las listas se visualicen cuando se recupera información de la base de datos mediante un cuadro de diálogo.
- **Guardar Automáticamente**: por defecto aparece marcada para que se almacene automáticamente cualquier cambio efectuado en el informe en un archivo temporal.
- **Unidad de Medida**: por defecto la unidad de medida de trabajo de los informes son las pulgadas.
- **Modo de Color**: por defecto es “Solo Lectura – Compartido” para que se añadan las paletas de colores del sistema al Report Builder hasta que no haya más sitio físico para mostrarlas.

Preferencias de la solapa Acceso

En la solapa **Acceso** nos encontramos con las siguientes opciones de configuración del Report Builder:

- **Acceso**: por defecto aparece marcada la opción “Archivo” para limitar que solo se carguen informes almacenados en fichero.
- **Mostrar**: por defecto aparece marcada la opción “Informes” para que por defecto se muestren en los cuadros de diálogo, solo aquellos objetos de tipo informes que se pretendan abrir o guardar.

Preferencias de la solapa Asistentes

En la solapa **Asistentes** nos encontramos con las siguientes opciones de configuración del Report Builder:

- **Recuadro de Diálogo de Bienvenido**: por defecto aparece marcada para mostrar un cuadro de diálogo de bienvenida al entrar en Reports Builder.
- **Página de Bienvenida del Asistente de Informes**: por defecto aparece marcada para mostrar un cuadro de diálogo cuando se comienza la creación de un informe asistido.
- **Página de Bienvenida del Asistente de Datos**: por defecto aparece marcada para mostrar un cuadro de diálogo cuando se comienza la creación de una consulta de datos asistida.
- **Página de Bienvenida del Asistente de Web**: por defecto aparece marcada aunque es una opción obsoleta que no se utiliza en Reports.
- **Página de Bienvenida del Asistente de Gráficos**: por defecto aparece marcada para mostrar un cuadro de diálogo cuando se comienza la creación de un gráfico de manera asistida.
- **Espaciado de Objetos del Asistente de Informes (Sin Plantilla)**: por defecto los cuatro valores (intervalo horizontal / vertical y Espacio horizontal / vertical) tienen valor 0, para que no se deje ninguna separación entre los objetos.

Preferencias de la solapa Valores de Ejecución

En la solapa **Valores de Ejecución** nos encontramos con las siguientes opciones de configuración del Report Builder:

- **Destino - Tipo**: por defecto aparece “En Definición de Informe”. Permite configurar las opciones del parámetro DESTYPE indicado en el capítulo anterior.
- **Destino – Nombre**: por defecto aparece en blanco y permite configurar las opciones del parámetro DESNAME indicado en el capítulo anterior.
- **Destino – Formato:** por defecto aparece en blanco y permite configurar las opciones del parámetro DESFORMAT indicado en el capítulo anterior.
- **Copias:** por defecto aparece en blanco y permite configurar el parámetro COPIES indicado en el capítulo anterior.
- **Orientación:** por defecto aparece “En Definición de Informe”. Permite configurar las opciones del parámetro ORIENTATION indicado en el capítulo anterior.
- **Modo:** por defecto aparece “En Definición de Informe”. Permite configurar las opciones del parámetro MODE indicado en el capítulo anterior.

- **Trabajo de Impresión:** por defecto aparece “En Definición de Informe”. Permite indicar si se muestra o no el cuadro de diálogo propio de la impresora cuando se imprime el informe.

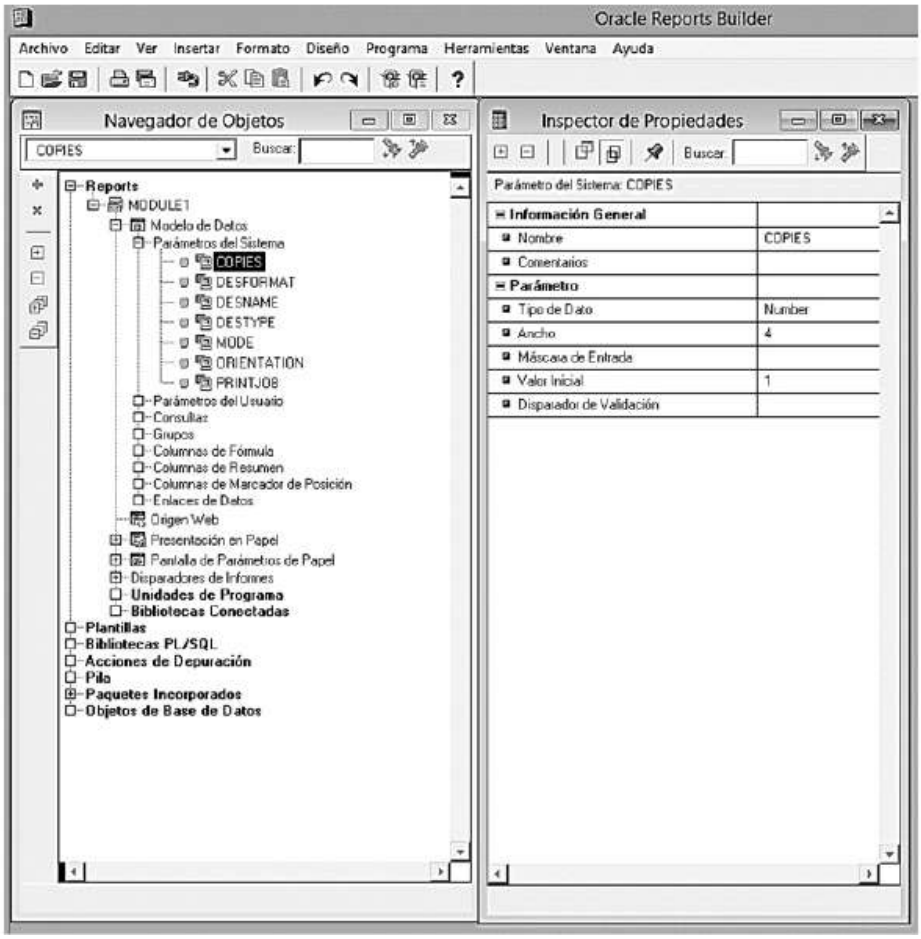
Preferencias de la solapa Parámetros de Ejecución

En la solapa **Parámetros de Ejecución** nos encontramos con las siguientes opciones de configuración del Report Builder:

- **Tamaño de la Matriz**: por defecto aparece 10. Permite configurar las opciones del parámetro ARRAYSIZE indicado en el capítulo anterior.
- **Buffers**: por defecto aparece 640. Permite configurar las opciones del parámetro BUFFERS indicado en el capítulo anterior.
- **Fragmento Long**: por defecto aparece 10. Permite especificar el tamaño en kb del incremento en el que Reports Builder recupera una columna de tipo LONG.
- **Anchura x Altura de Página**: por defecto aparecen en blanco. Permite configurar las opciones del parámetro PAGESIZE indicado en el capítulo anterior.
- **Ejecución Correcta**: por defecto aparece “Confirmar”. Permite configurar las opciones del parámetro ONSUCCESS indicado en el capítulo anterior.
- **Ejecución Fallida:** por defecto aparece “Rollback”. Permite configurar las opciones del parámetro ONFAILURE indicado en el capítulo anterior.
- **Rol:** por defecto aparece en blanco. Permite configurar las opciones del parámetro ROLE indicado en el capítulo anterior.
- **Pantalla de Parámetros:** por defecto aparece marcado. Permite configurar las opciones del parámetro PARAMFORM indicado en el capítulo anterior.
- **Sólo lectura:** por defecto aparece desmarcado. Permite configurar las opciones del parámetro READONLY indicado en el capítulo anterior.
- **Ejecutar en Modo de Depuración:** por defecto aparece desmarcado. Permite configurar las opciones del parámetro RUNDEBUG indicado en el capítulo anterior.
- **Confirmación Automática:** por defecto aparece desmarcado. Permite configurar las opciones del parámetro AUTOCOMMIT indicado en el capítulo anterior.
- **SQL no Bloqueante:** por defecto aparece marcado. Permite configurar las opciones del parámetro NONBLOCKSQL indicado en el capítulo anterior.
- **Páginas en Blanco:** por defecto aparece marcado. Permite configurar las opciones del parámetro BLANKPAGES indicado en el capítulo anterior.

PARÁMETROS POR DEFECTO DEL DISEÑO DE UN REPORT

Cuando creamos un nuevo informe (report) en el diseñador (Report Builder), los parámetros que hemos configurado en las distintas solapas de las Preferencias, quedan reflejados en el propio informe dentro de la sección **Parámetros del Sistema** como se indica en la siguiente imagen:



INFORMES BASADOS EN CONSULTAS JDBC 46

INTRODUCCIÓN

Oracle Reports Builder permite diseñar consultas SQL que no solo obtengan los datos de una base de datos Oracle, sino también de otro tipo de fuentes como pueden ser: DB2, ACCESS, MYSQL, SQLSERVER, etc. Para ello se utilizan las consultas JDBC dentro del Report Builder.

En este capítulo aprenderemos a crear este tipo de consultas, tomando como ejemplo una base de datos de tipo ACCESS.

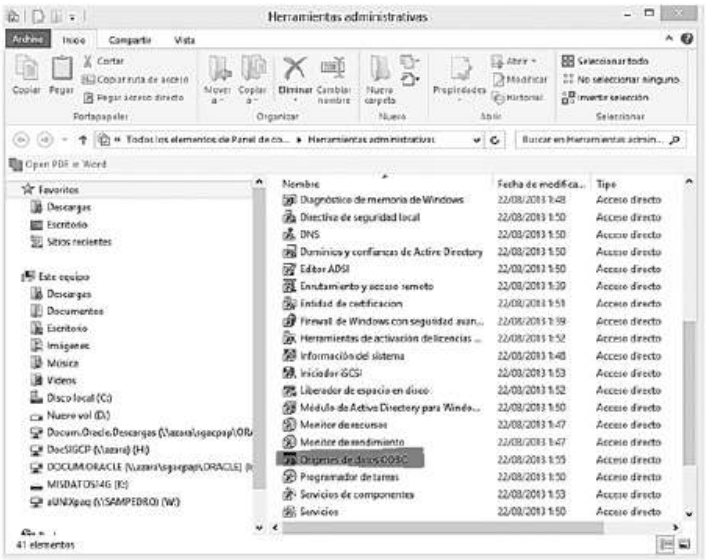
SELECCIONAR EL ORIGEN DE CONSULTAS JDBC

En primer lugar abrimos el Report Builder y seguidamente indicamos que el informe estará destinado **al Diseño Web y presentación en papel**.

Seguidamente indicamos el formato del informe, que para nuestro caso será **Tabular**, indicando además un título para el mismo: **Consulta de Países.** A continuación se nos presentan los orígenes de datos disponibles para la obtención de la consulta. En nuestro caso seleccionaremos la opción **JDBC Query,** como se muestra en esta imagen.

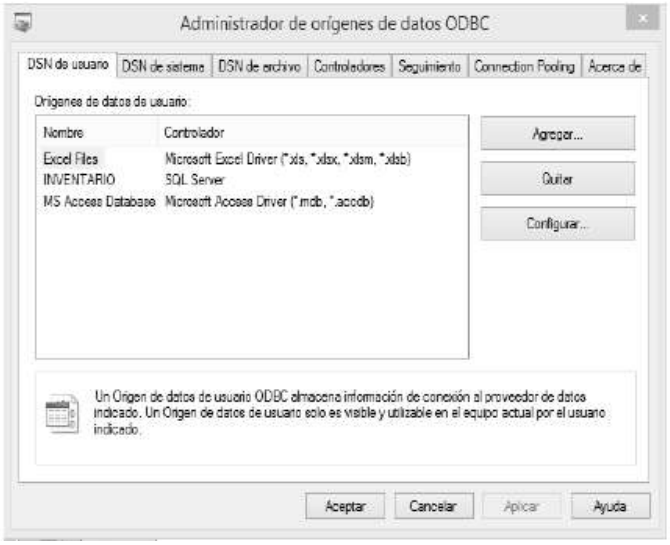
Antes de pulsar el botón **Siguiente** de esta pantalla, debemos configurar para el caso práctico que estamos realizando la conexión ODBC contra Access. Para ello hay que seguir las indicaciones del siguiente apartado del capítulo.

CONFIGURAR ODBC



La configuración de las conexiones ODBC a las distintas bases de datos se realiza con la herramienta de Windows **Orígenes de datos ODBC.** Esta herramienta se encuentra dentro del **Panel de Control – Herramientas administrativas**, como se muestra en esta imagen.

Crear una conexión con Access en ODBC



Una vez abierta la aplicación **Orígenes de datos ODBC** se muestra una pantalla como esta (solapa DSN de usuario).

Para crear la conexión contra una base de datos (en nuestro caso Access), tenemos que situarnos en la solapa **DSN de sistema** y a continuación debemos de pulsar el botón **Agregar,** lo que nos presentará una pantalla como la que se muestra a continuación:

En esta pantalla es donde seleccionamos el origen de la base de datos a la que nos vamos a conectar. En nuestro caso práctico tenemos que seleccionar como origen: **Microsoft Access Driver (*.mdb)**, y tras ello pulsaremos el botón **Finalizar**.

Seguidamente se presentará una nueva pantalla (como la que se muestra en la imagen de la página siguiente) donde se deben de incorporar los datos de conexión contra la base de datos. Los valores que se solicitan en la misma son:

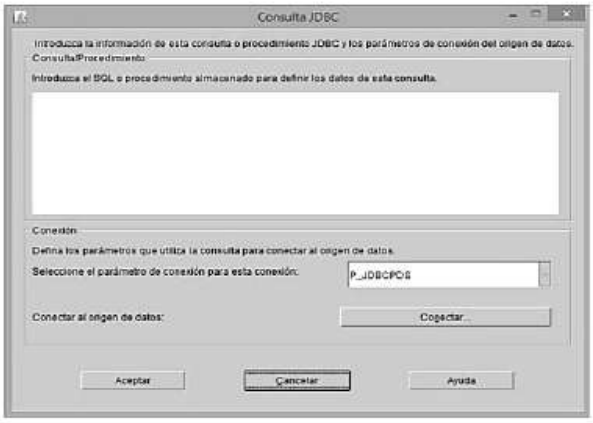
- **Nombre del origen de datos**: es el nombre lógico con el que identificaremos la conexión contra la base de datos.
- **Descripción**: es un texto identificativo de la conexión.
- **Base de datos**: en este apartado seleccionaremos la base de datos (en nuestro caso de tipo Access .mdb con la que nos vamos a conectar).

Una vez introducidos los datos que se muestran en la imagen, seleccionando la base de datos **Practica9.mdb** que se suministra con los ejemplos del libro, pulsaremos el botón **Aceptar** y la conexión quedará almacenada en el administrador ODBC para poder ser utilizada por cualquier aplicación.

CREAR LA CONSULTA JDBC

Una vez definida la conexión ODBC contra la base de datos correspondiente (en nuestro caso ACCESS), podemos continuar con la creación de nuestro informe report basado en una consulta JDBC, en el punto donde lo dejamos.

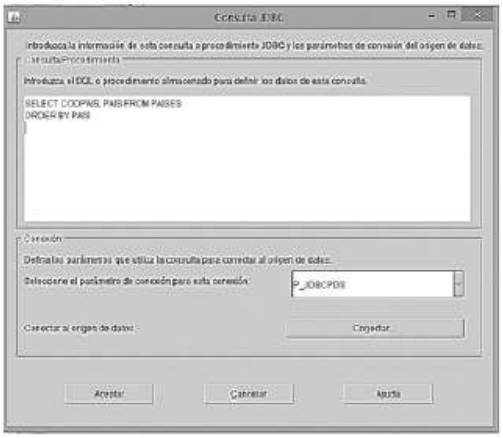
Ahora sí podemos pulsar el botón **Siguiente**, que nos presentará una pantalla donde debemos seleccionar el origen de datos que hemos definido en al apartado anterior dentro de la aplicación ODBC. Para ello pulsamos el botón **Definición de Consulta** que se presenta en la imagen siguiente.



Una vez pulsado se presenta una nueva pantalla para realizar la conexión, en la que pulsaremos el botón **Conectar.**

Seguidamente se muestra una nueva pantalla para introducir los datos concretos de la conexión (usuario, contraseña, base de datos), que rellenaremos como se indica en la imagen siguiente teniendo en cuenta que el controlador para nuestro ejemplo será **jdbc-ldbc** y la base de datos **practica9 (**el nombre con el que creamos la conexión ODBC anteriormente**).**

A continuación pulsamos el botón **Conectar** que nos devolverá a la pantalla anterior para poder crear ya la consulta concreta. En nuestro caso práctico vamos a crear la siguiente consulta:

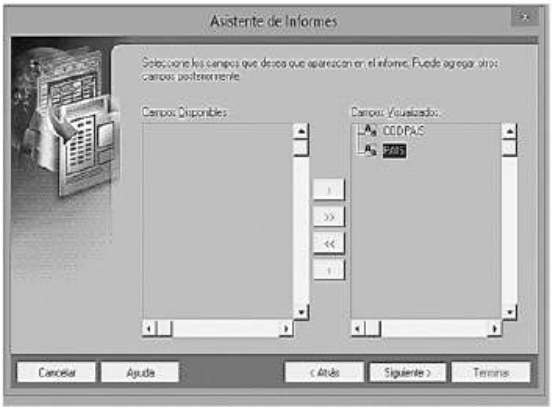



```
SELECT CODPAIS, PAIS FROM PAISES
ORDER BY PAIS
```

Por tanto, la pantalla quedará de esta forma.

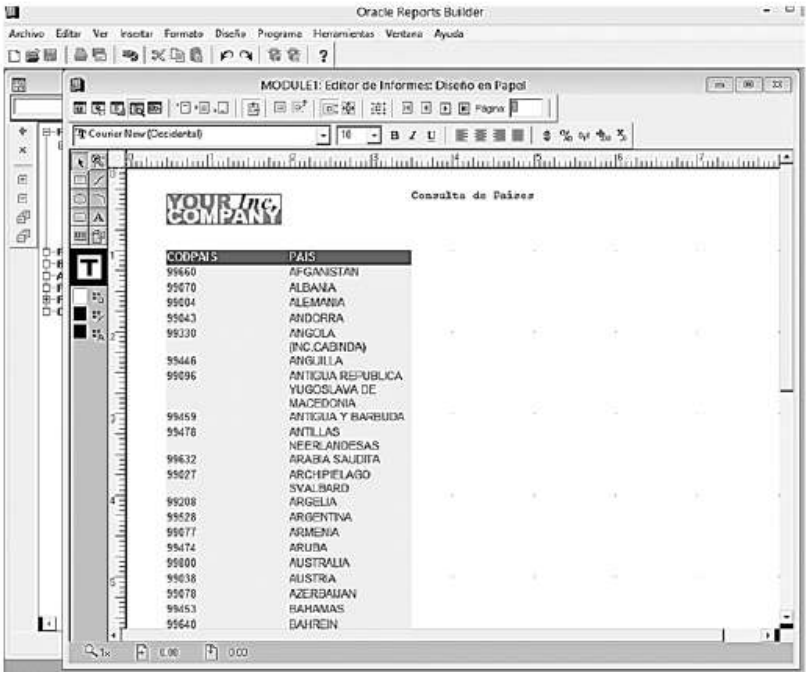
Seguidamente pulsaremos el botón **Aceptar** y el propio report compondrá con esta información la consulta definitiva en el asistente de la siguiente forma:

Seguidamente en esta pantalla pulsaremos el botón **Siguiente** e indicaremos que queremos visualizar en el listado todas las columnas disponibles de la consulta como se muestra en la imagen siguiente.



Seguidamente pulsamos el botón **Siguiente** y en la pantalla de totalización no indicamos nada, volviendo a pulsar el botón **Siguiente** hasta la pantalla donde se solicita la plantilla predefinida para mostrar el resultado, en la que seleccionaremos **Blue**.

A continuación pulsamos **Siguiente** y **Terminar**, lo que nos mostrará en el editor de diseño en papel el resultado que obtendríamos similar al que se muestra en la imagen de la página siguiente.



Para finalizar podemos ejecutar este listado en modo Web y veremos que se abre el navegador con los resultados obtenidos, así como salvar el resultado en los distintos formatos que hemos aprendido (PDF, RTF, XLS…).

Antes de cerrar el Report Builder almacenaremos este caso práctico como **PRACTICA46.JSP.**

INFORMES BASADOS EN CONSULTAS DE FICHEROS TXT 47

INTRODUCCIÓN

Oracle Reports Builder permite diseñar también consultas que utilizan como origen de datos un fichero TXT o un fichero LOG, separado por comas o con campos de longitud fija.

Esta opción solo se utilizará en casos muy especiales ya que el acceso a archivos de este tipo es una tarea computacional costosa.

SELECCIONAR COMO ORIGEN UN FICHERO TEXTO

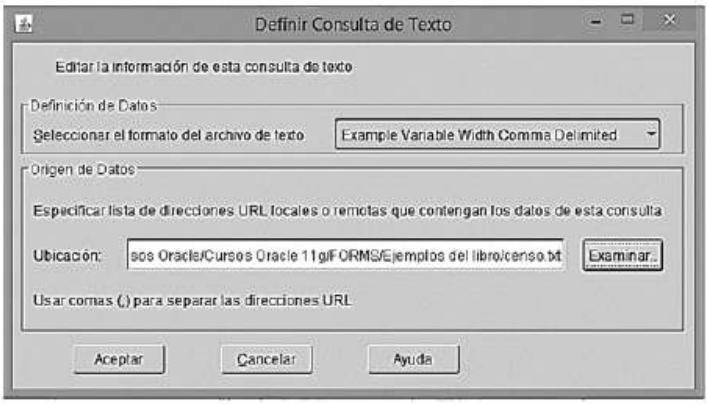
En primer lugar abrimos el Report Builder y en la ventana que se muestra seleccionamos que queremos crear un informe asistido.

Seguidamente indicamos que el informe estará destinado al **Diseño Web y presentación en papel**.

A continuación indicamos el formato del informe, que para nuestro caso será **Tabular**, indicando además un título para el mismo: **Censo.**

Seguidamente se nos presentan los orígenes de datos disponibles para la obtención de la consulta. En nuestro caso seleccionaremos la opción **Text Query**.

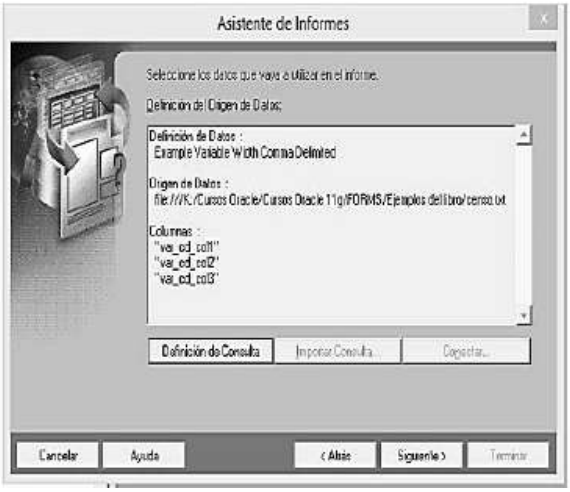
A continuación pulsamos el botón **Siguiente**, que nos presentará una pantalla donde debemos pulsar el botón **Definición de Consulta.**



Una vez pulsado se presenta una nueva pantalla para seleccionar el fichero que utilizaremos como fuente de datos como se indica en la imagen de la siguiente página.

En esta pantalla debemos indicar en la sección **Definición de Datos** el tipo de fichero que se va a utilizar para la obtención de datos. En nuestro caso práctico el fichero que se utilizará tiene una estructura de registro de tamaño variable con campos delimitados por comas, por lo que la opción a elegir será **Example Variable Width Comma Delimited**.

Además, habrá que indicar el nombre y la ruta del fichero que se van a utilizar dentro del cuadro **Ubicación**. En nuestro caso práctico debemos seleccionar el fichero **censo.txt** que se suministra junto con el resto de ejemplos de este libro.

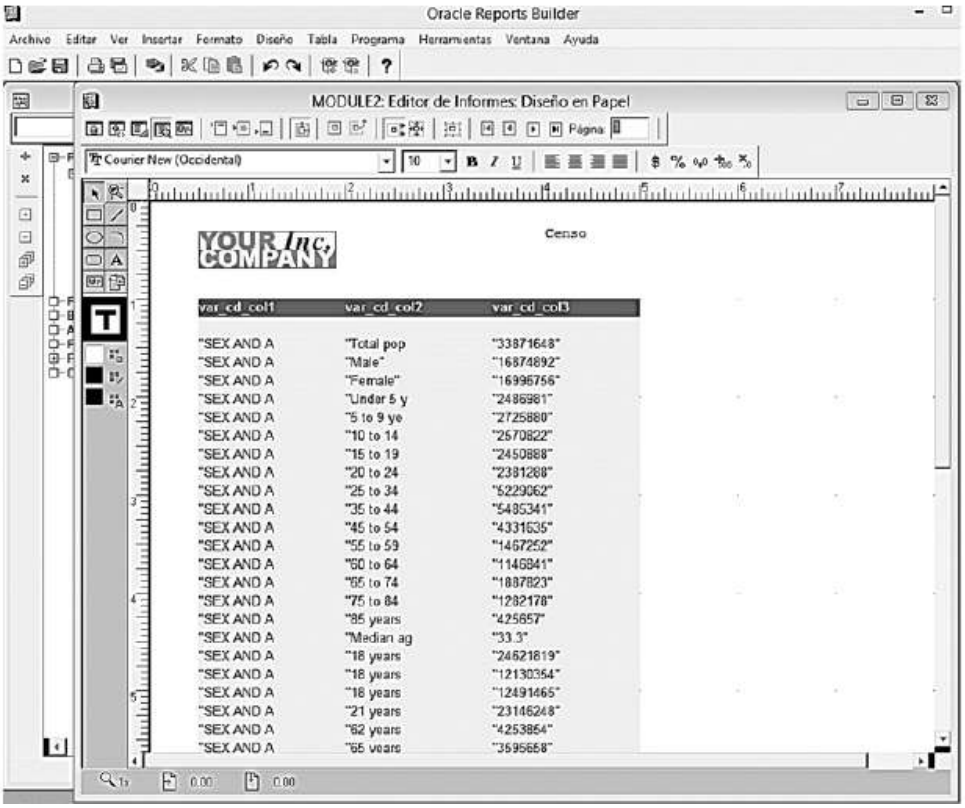


Seguidamente pulsamos **Aceptar** y Report Builder compone la consulta como se muestra en esta imagen.

En esta pantalla pulsaremos **Siguiente** e indicaremos que queremos visualizar en el listado todas las columnas disponibles de la consulta.

Seguidamente pulsamos el botón **Siguiente** sucesivamente hasta llegar a la última pantalla donde se nos solicita la plantilla predefinida para mostrar el resultado, en la que seleccionaremos **Blue**.

A continuación pulsamos **Siguiente** y **Terminar**, lo que nos mostrará en el editor de diseño en papel el resultado que obtendríamos similar al que se muestra en la imagen de la página siguiente.



Para finalizar podemos ejecutar este listado en modo Web y veremos que se abre el navegador con los resultados obtenidos, así como salvar el resultado en los distintos formatos que hemos aprendido (PDF, RTF, XLS…).

Antes de cerrar el Report Builder almacenaremos este caso práctico como **PRACTICA47.JSP.**

INFORMES BASADOS EN CONSULTAS XML 48

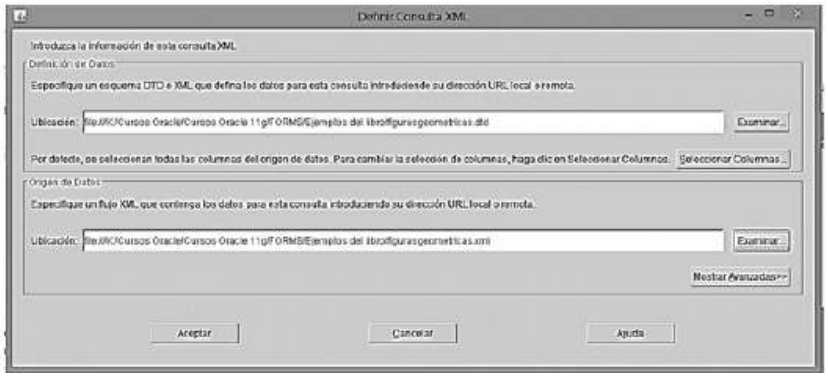
INTRODUCCIÓN

Oracle Reports Builder permite diseñar también consultas que utilizan como origen de datos un fichero XML.

SELECCIONAR COMO ORIGEN UN FICHERO XML

En primer lugar abrimos el Report Builder y en la ventana que se muestra seleccionamos que queremos crear un informe asistido.

Seguidamente indicamos que el informe estará destinado al **Diseño Web y presentación en papel** y a continuación indicamos el formato del informe, que para nuestro caso será **Tabular**, indicando además un título para el mismo: **Figuras Geométricas.** Seguidamente se nos presentan los orígenes de datos disponibles para la obtención de la consulta. En nuestro caso seleccionaremos la opción **XML Query**. A continuación pulsamos el botón **Siguiente**, que nos presentará una pantalla donde debemos pulsar el botón **Definición de Consulta.**



Una vez pulsado se presenta una nueva pantalla para seleccionar el fichero XML que utilizaremos como fuente de datos como se indica esta imagen.

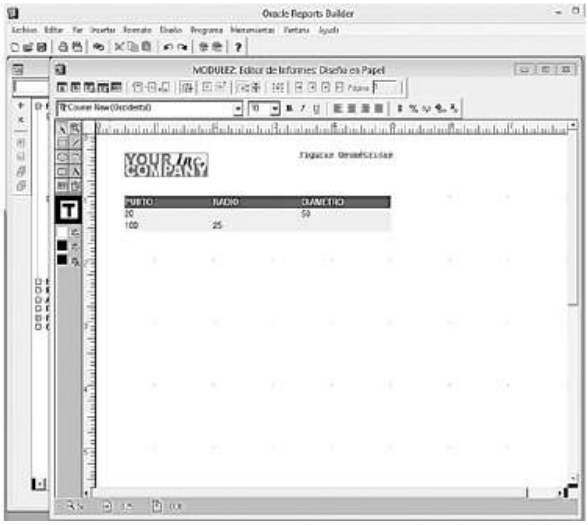
En la pantalla anterior debemos indicar en la sección **Definición de Datos** el nombre del archivo que contiene la definición de los datos XML (archivo con extensión DTD) En nuestro caso indicaremos el fichero **figurasgeometricas.dtd**.

Además habrá que indicar en la sección **Origen de Datos** el nombre del archivo que contiene los datos en XML. En nuestro caso indicaremos el fichero **figurasgeometricas.xml**.

Seguidamente pulsamos **Aceptar** y Report Builder compondrá la consulta. En la pantalla que se muestra con la consulta compuesta, pulsaremos **Siguiente** e indicaremos que queremos visualizar en el listado todas las columnas disponibles.

A continuación pulsamos el botón **Siguiente** sucesivamente hasta llegar a la última pantalla donde se nos solicita la plantilla predefinida para mostrar el resultado, en la que seleccionaremos **Blue**.

Por último pulsamos **Siguiente** y **Terminar**, lo que nos mostrará en el editor de diseño en papel el resultado que obtendríamos similar al que se muestra a continuación:



Para finalizar podemos ejecutar este listado en modo Web y veremos que se abre el navegador con los resultados obtenidos, así como salvar el resultado en los distintos formatos que hemos aprendido (PDF, RTF, XLS…).

Antes de cerrar el Report Builder almacenaremos este caso práctico como **PRACTICA48.JSP.**

DISEÑO MANUAL DE UN REPORT 49

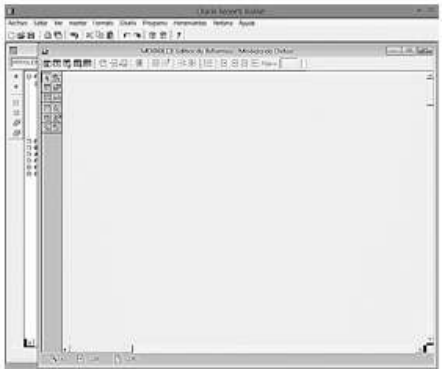
INTRODUCCIÓN

En los capítulos anteriores dedicados a la herramienta Oracle Reports Builder hemos tratado únicamente la creación de informes asistidos. En este capítulo vamos a abordar la creación de un informe de forma manual para aprender los distintos elementos que podemos utilizar.

CREAR UN INFORME NUEVO MANUALMENTE

En primer lugar abrimos el Report Builder y en la ventana que se muestra seleccionamos que queremos crear un informe **Nuevo Manualmente**.

Después pulsaremos el botón **Aceptar** lo que generará que la herramienta Report Builder nos abra el **Editor de Informes** donde podemos pasar a diseñar nuestro informe, como se muestra en la imagen inferior de esta página.



Como se puede comprobar en esta imagen, el **Editor de Informes** aparece superpuesto al propio navegador de objetos de la herramienta Report Builder, pero podremos trabajar indistintamente en cualquiera de las dos áreas de la herramienta.

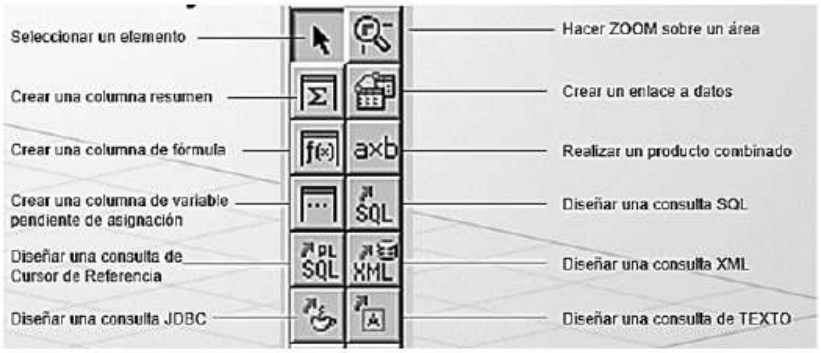
EL MODELO DE DATOS

Es el área donde se define la consulta que permitirá mostrar los datos del informe.

Dentro de esta área también se definen las columnas resumen y las columnas basadas en fórmulas que se quieren aplican sobre el listado.

Barra de herramientas del modelo de datos

El área del modelo de datos dispone de una barra de herramientas para facilitarnos la creación de la consulta y de los elementos de cálculo. A continuación se muestra el significado de los elementos que forman parte de dicha barra de herramientas.



COLUMNAS RESUMEN

Las columnas resumen, dentro de un informe, permiten definir las siguientes operaciones:

- Recuento de elementos.
- Suma de elementos.
- Media de elementos.
- Valor máximo y mínimo.
- Varianza de una serie de elementos.
- Deviación estándar de una serie de elementos.
- Primer y último elemento de una serie.
- % sobre el total.

COLUMNAS FÓRMULA

Las columnas fórmula, dentro de un informe, permiten añadir bloques de código PL/SQL para la definición de procedimientos y funciones adicionales, que pueden ser necesarias para el cálculo de operaciones específicas.

El valor que devuelvan estas columnas también puede ser mostrado en el informe.

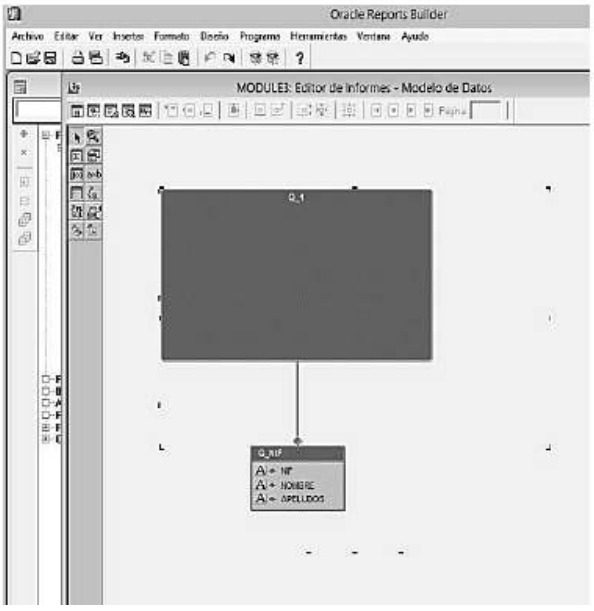
CREANDO UNA CONSULTA MANUAL

Volviendo al **Editor de Informes** que teníamos abiertos, vamos a pulsar el botón  y dibujaremos una cuadrícula sobre la pantalla de diseño, para crear una consulta manual donde obtengamos los datos que queremos presentar en el informe.

Una vez pulsado el botón y generada la cuadrícula, se mostrará una nueva ventana en la que podemos utilizar la herramienta **Query Builder** para construir la consulta (como hemos indicado en capítulos anteriores) o bien escribir directamente la misma en la cuadrícula **Sentencia de Consulta SQL.**

En nuestro caso práctico vamos a escribir directamente la siguiente consulta:

```
SELECT DISTINCT NIF, NOMBRE, APELLIDOS FROM EMPLEADO;
```



A continuación pulsaremos **Aceptar** y en ese instante la herramienta Report Builder revisa la sintaxis de la sentencia y si es correcta desaparecerá dicha ventana y volveremos al **Editor de Informes** donde aparecerá la consulta diseñada como se muestra en esta imagen.

Una vez finalizada la consulta no es suficiente para poder ejecutar el informe porque nos falta un diseño del mismo, por tanto si pulsamos en **Programa – Ejecutar Presentación en Papel**, nos aparecerá el siguiente error:

CREANDO UN DISEÑO

En el mensaje anterior pulsaremos el botón **SÍ** para crear un diseño, lo que provocará que se muestre el **Asistente de Informes**.

Seleccionamos la opción **Crear Diseño Web y Presentación en Papel** y pulsaremos el botón **Siguiente**.

A continuación en la ventana que nos solicita el **Estilo** del informe, seleccionaremos la opción **Tabular** y como título del informe: **Relación de empleados**, pulsando a continuación el botón **Siguiente**, sucesivamente, hasta que nos aparezca la ventana **Campos** donde incluiremos todos los campos mostrados en el informe. Seguidamente pulsaremos de nuevo **Siguiente** hasta la última ventana (**Plantilla**) en la que pulsaremos el botón **Terminar,** lo que hará que se muestre en el **Editor de Informes** nuestro diseño del mismo con un aspecto similar al que se muestra en esta imagen.

Antes de cerrar el Report Builder almacenaremos este caso práctico como **PRACTICA49a.JSP.**

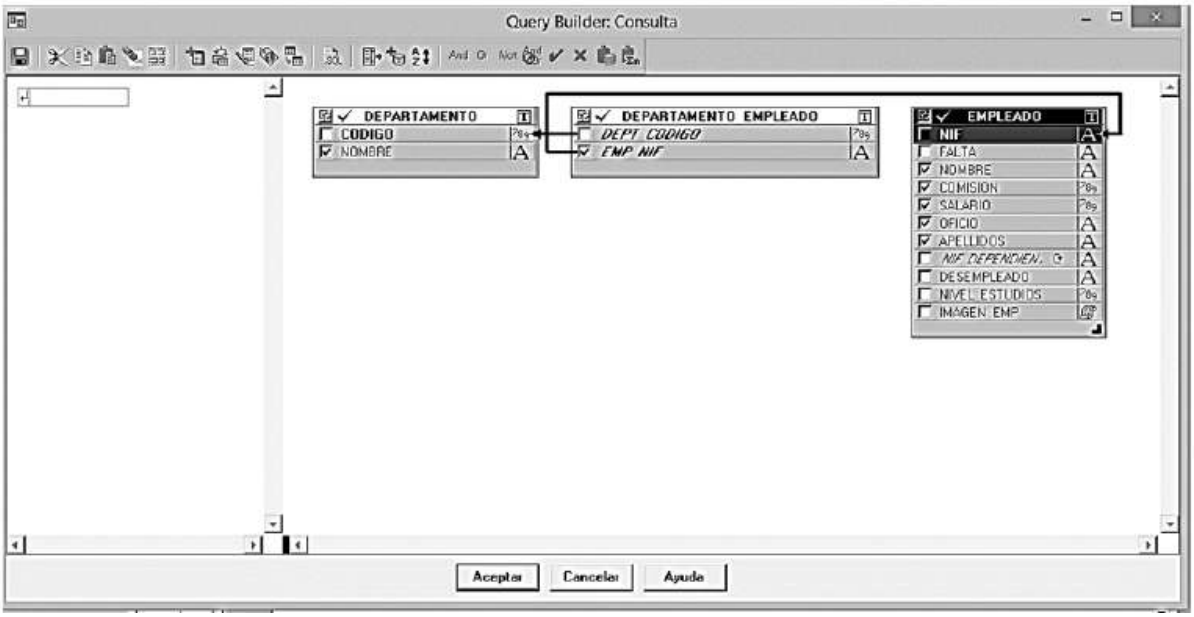
CREAR UNA CONSULTA CON QUERY BUILDER

Una vez cerrada la práctica anterior, vamos a abrir un nuevo informe desde el Report Builder que también **crearemos manualmente**.

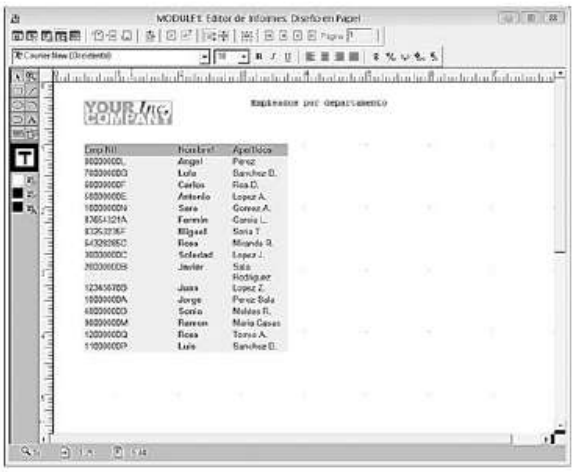
Dentro del Editor de Informes creamos una nueva consulta SQL y entramos en el constructor de consultas **Query Builder** para definirla. En el caso práctico que vamos a diseñar tenemos que seleccionar, como tablas para la consulta, las siguientes: **DEPARTAMENTO, EMPLEADO** y **DEPARTAMENTO_EMPLEADO** como se indica en esta imagen.

A continuación pulsamos los botones **Incluir** y después **Cerrar,** lo que provoca que se trasladen las tablas al Query Builder para continuar su edición. Nos aparecerá una imagen donde podemos observar por las flechas, cuál es la relación que existe entre las 3 tablas.

Seguidamente marcamos los campos que queremos mostrar en nuestro informe de cada una de las tablas. Para nuestro caso práctico tenemos que incluir en el informe todas las columnas marcándolas en sus casillas correspondientes (aparecerá el símbolo ☑). Las columnas a presentar son las que se muestran en esta imagen:



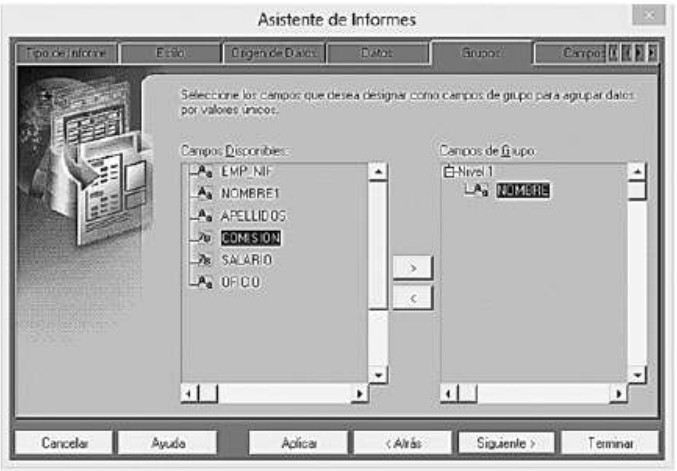
A continuación pulsamos **Aceptar** para cerrar el editor de consultas y nuevamente **Aceptar** para cerrar la sentencia de consulta. Tras estos pasos volveremos al Editor de Informes, donde iniciaremos el **Asistente de Informes** desde el menú **Herramientas** para indicar los campos que queremos visualizar en el listado. En nuestro caso práctico seleccionaremos como campos a pintar los siguientes: **NIF, NOMBRE1** y **APELLIDOS.**



Seguidamente pulsaremos el botón **Siguiente** hasta finalizar (**Terminar**) la creación del listado, lo que mostrará el mismo de la siguiente forma.

MODIFICANDO UN DISEÑO DE LISTADO

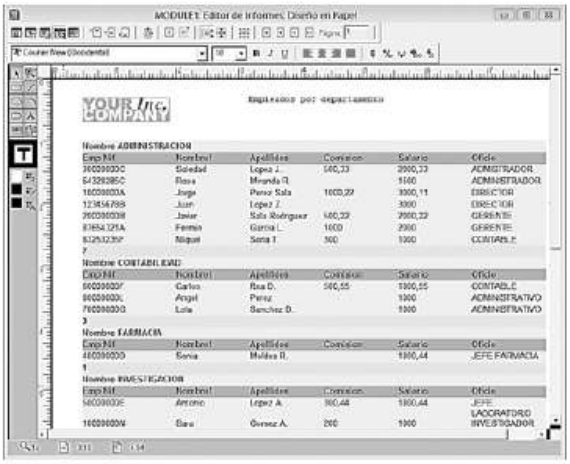
Aunque se haya diseñado un listado con un formato concreto de presentación, este también se puede modificar con posterioridad. En este apartado y continuación con el caso práctico anterior, vamos a modificar el formato de presentación del listado anterior. Para ello volvemos a pulsar en el menú **Herramientas** – **Asistente de Informes** y nos vamos hasta la pestaña **Estilo** donde vamos a modificar el formato del listado de estilo **Tabular** a estilo **Agrupar Arriba**, indicando también un título para el listado con el siguiente nombre: **Empleados por departamento**.



Al cambiar el formato de listado a un estilo agrupado, es necesario también indicar el método de agrupamiento mediante la selección de la/s columna/s por las que se hará. En nuestro caso práctico vamos a seleccionar como campo de agrupamiento **NOMBRE** como se indica en esta imagen.

También vamos a modificar los campos a visualizar y vamos a añadir a la lista el resto que faltaban.

Por último vamos a añadir un nuevo campo de totales (**suma**) que se obtenga a partir del campo **EMP_NIF** como se muestra en la imagen siguiente:



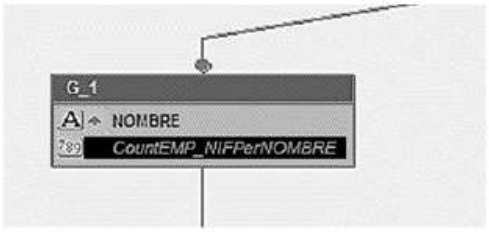
A continuación pulsamos el botón **Terminar** para mostrar el resultado del listado como aparece en la imagen de la página siguiente.

MODIFICANDO COLUMNAS DE TOTALIZACIÓN

En este apartado vamos a ver cómo se realiza la modificación de una columna de totalización ya incluida en el listado tanto en su sentido como en su origen. Volviendo a nuestro caso práctico nos situamos en el **Modelo de Datos.**

La interpretación de los elementos que aparecen en el modelo de datos es la siguiente:

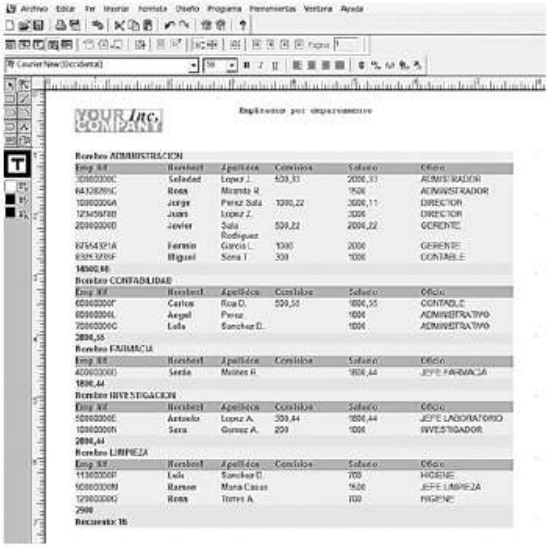
	Es una columna resumen. En nuestro caso **CountEMP_NIFPerReport**
	Es la consulta SELECT.
	Es el grupo superior. En nuestro caso existe un agrupamiento por la columna **NOMBRE** y además se muestra una columna resumen **CountEMP_NIFPerReport.**
	Es el grupo inferior o detalle de la consulta que contiene el resto de elementos a mostrar en el listado.



Para modificar la columna resumen, primero tenemos que marcarla dentro del modelo de datos, así que nos situamos en **CountEMP_NIFPerReport** dentro del grupo superior **G1** como se muestra en esta imagen.

Seguidamente dentro del menú **Herramientas** pulsamos en la opción **Inspector de Propiedades,** lo que nos mostrará las propiedades de la columna resumen. Vamos a modificar las siguientes propiedades para ajustar los valores que se indican a continuación:

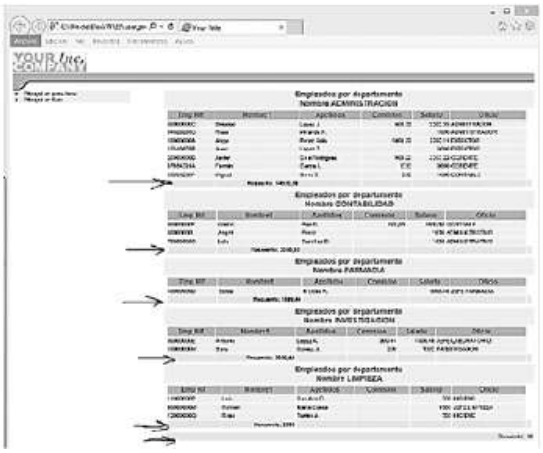
- Ancho: **10**
- Valor si el nulo: **0**
- Origen: **SALARIO**
- Función: **Suma**

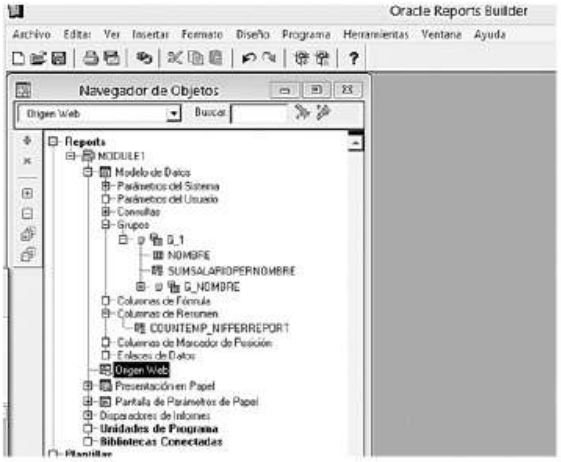


A continuación cerramos la ventana de propiedades y visualizamos el resultado del listado en modo Papel (pulsando el botón ) como se muestra en esta imagen.

Si ahora ejecutamos el listado en modo **Presentación en Web** pulsando sobre el menú **Programa – Ejecutar Diseño Web,** nos aparecerá el siguiente resultado:

Como se puede comprobar en las indicaciones que se han realizado en el listado (las marcadas con las flechas), el resultado presenta una pequeña anomalía, dado que si bien el acumulado mostrado es la suma de los salarios, el título que se muestra a la izquierda del mismo (**Recuento**) no refleja el resultado, por lo que se debe modificar este literal por "**Suma Salario**".

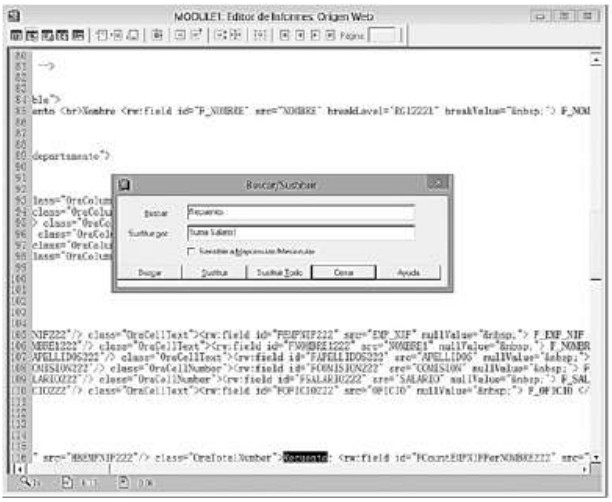




Para realizar este cambio nos tenemos que situar dentro del Navegador de Objetos de Report Builder en la sección **Origen Web** como se muestra en esta imagen.

Una vez situados en dicho elemento, hacemos doble clic con el ratón y se nos presenta la página HTM que contiene todo lo necesario para visualizar el informe en modo web. A continuación pulsamos en el menú **Editar** y la opción **Buscar/Sustituir** donde indicaremos que se busque el texto **"Recuento:"** y se sustituya por **"Suma Salario:"** como se indica en esta imagen.

Seguidamente pulsamos el botón **Sustituir Todo** y después el botón **Cerrar.**

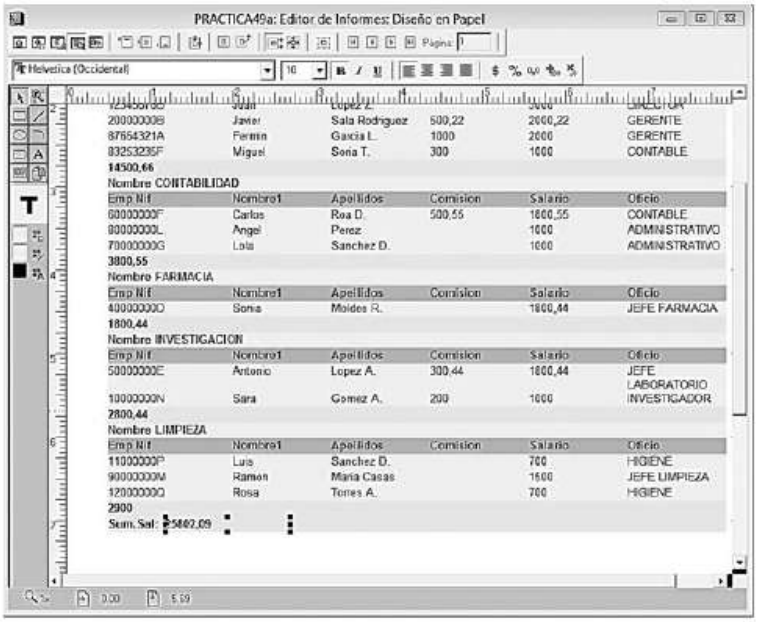


A continuación tenemos que situarnos el **Editor de Informes** y pulsamos en el botón del **Diseño en Papel,** donde modificaremos para la presentación en papel, el literal Recuento y la función de acumulación.

En primer lugar nos posicionamos en el literal "Recuento" y lo modificamos por "Sum.Sal.". A continuación nos posicionamos en la función de agrupamiento **F_CountEMP_NIFPerReport** realizando un doble clic para editar las propiedades del mismo, en la que cambiaremos las siguientes propiedades:

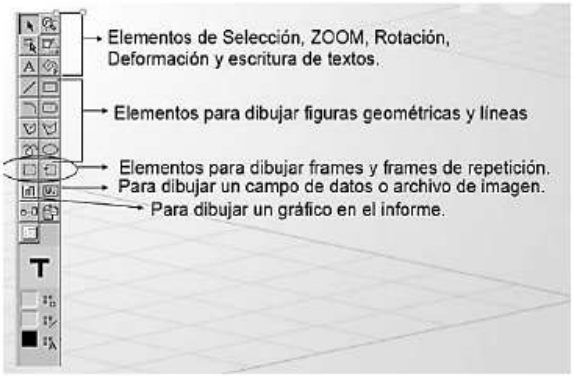
- Origen: **SALARIO**
- Función: **Suma**
- Ancho: **10**

Si ahora ejecutáramos la presentación en papel, el resultado obtenido sería el mostrado en la imagen de la página siguiente.



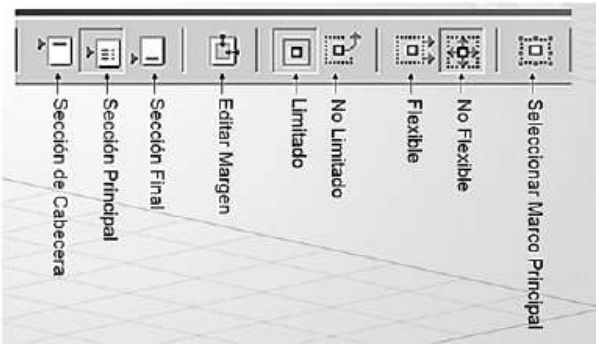
BARRA DE HERRAMIENTAS DEL EDITOR DE INFORMES/PRESENTACIÓN EN PAPEL

El **Editor de Informes** – **Presentación en Papel** posee una barra de herramientas que permite el manejo de los elementos del formulario en las diversas secciones que tiene: cabecera, principal y final. En este apartado vamos a identificar cada uno de los elementos de dicha barra.



La barra de herramientas vertical consta de los siguientes elementos.

La barra de herramientas horizontal consta de los siguientes elementos.

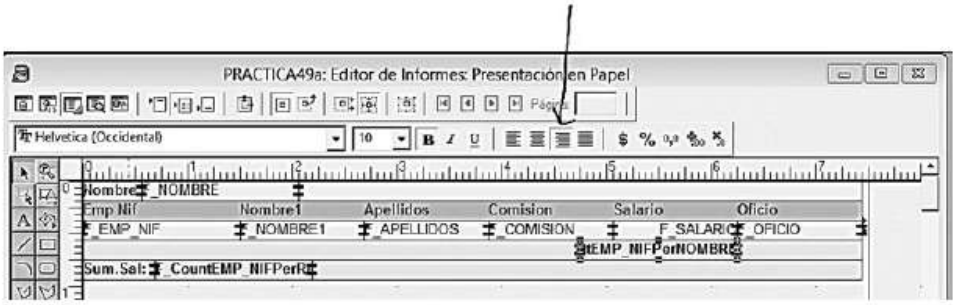


- La **Sección de Cabecera** corresponde a la zona de la página que se imprime antes de la página de datos (cabecera).
- La **Sección Principal** corresponde con la zona de la página donde se imprimen los datos del informe (zona de detalle).
- La **Sección Final** corresponde con la zona de la página que se imprime después de la principal (pie de página).
- La herramienta **Editar Margen** permite observar el formato de la página que se va a imprimir con toda su información: logos, títulos, etc.
- La herramienta **Limitado** impide que un objeto pueda desplazarse fuera de su marco contenedor.
- La herramienta **No Limitado** permite que un objeto pueda desplazarse fuera de su marco contenedor.
- La herramienta **Flexible** permite que el marco se ensanche a la vez que se mueve alguno de sus objetos que se están ensanchando también.
- La herramienta **No Flexible** permite que únicamente se ensanche, a la vez que se mueve, alguno de sus objetos que se están ensanchando también.
- La herramienta **Seleccionar Marco Principal** permite seleccionar aquel marco que contiene al objeto seleccionado o bien al marco actual, por tanto siempre selecciona el marco superior al que nos encontramos.

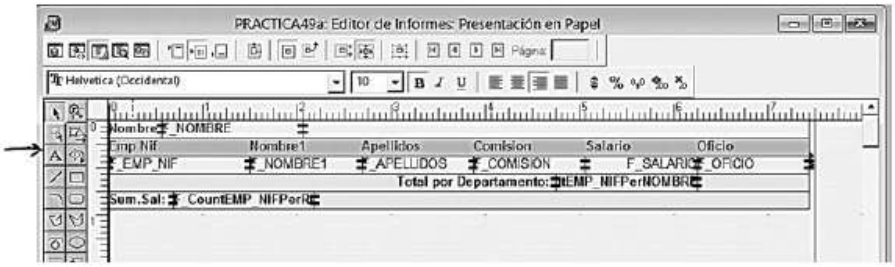
MODIFICANDO ELEMENTOS DENTRO DE LOS MARCOS

Una vez aprendido el significado y funcionamiento de los elementos de la barra de herramientas de un informe para presentar en papel, vamos a retomar la práctica que estamos diseñando en este capítulo y nos situamos de nuevo en el **Editor de Informes** – **Presentación en Papel.** A continuación vamos a mover el sumatorio de cada departamento justo debajo de la columna **Salario**. Para ello en primer lugar vamos a marcar las opciones **Limitado** y **No Flexible** en la barra de herramientas como se muestra en la imagen siguiente:

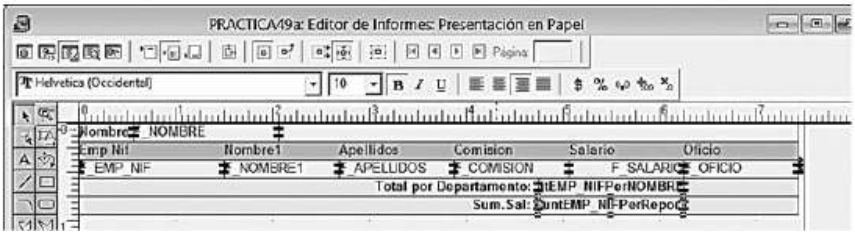
Ahora seleccionamos el objeto **F_CountEMP_NIFPerNOMBRE** correspondiente al sumatorio del departamento y lo movemos justo alineado debajo de la columna Salario. Es recomendable realizar esta operación con las teclas de movimiento del teclado porque así se realiza con mayor precisión. Además, indicaremos que el contenido del campo quedará alineado a la derecha tanto del sumatorio como del propio salario. El resultado que debemos de obtener es el que aparece en esta imagen.



A continuación vamos a añadir un texto delante del sumatorio del departamento, que etiquetaremos con el texto "**Total por Departamento:**". Para ello seleccionaremos en la barra de herramientas vertical el objeto **Texto** y pulsaremos sobre una zona colindante al sumatorio. Después de escribir la etiqueta la arrastraremos y colocaremos con las teclas de movimiento hasta que el resultado sea similar al siguiente:

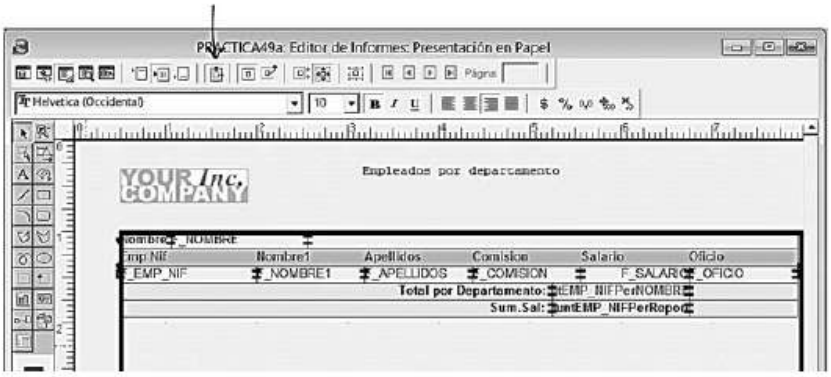


Seguidamente vamos a realizar el mismo proceso con el sumatorio total del informe para desplazarlo (junto con su etiqueta) por debajo de la columna salario y ajustado a la derecha, como se muestra en la imagen siguiente:



A continuación cambiamos el color de las letras del sumatorio por departamento y el sumatorio total. Al primero lo ponemos en rojo y el total en verde.

Seguidamente vamos a cambiar el logo del listado y el título del mismo, para ello seleccionamos en la barra de herramienta horizontal la opción **Editar Margen** lo que nos mostrará la imagen siguiente:



En primer lugar seleccionamos el logo haciendo clic con el ratón sobre el mismo y lo eliminamos. Seguidamente vamos a añadir un nuevo logo pulsando sobre la opción del menú **Insertar – Imagen**. En el cuadro de diálogo que se presenta vamos a indicar las siguientes opciones:

- Archivo: **logo_hospital.jpg** (se suministra con los ejemplos del libro)
- Formato: **JFIF – JPEG**
- Calidad: **Buena**

A continuación pulsamos **Aceptar** y colocamos la imagen en la posición que se encontraba el logo anterior que eliminamos, recortando previamente el tamaño de la misma para que sea más pequeña. Seguidamente cambiamos el título del listado haciendo clic sobre el texto "Empleados por departamento". El nuevo título será **Listado de Empleados por Departamento** que deberá estar en negrita y subrayado con letra Arial en color azul de 14 puntos. El resultado será similar al siguiente:

A continuación podemos ejecutar de nuevo el listado en Papel y obtendremos el siguiente resultado.

MODIFICANDO ELEMENTOS EN LA PRESENTACIÓN EN PAPEL

De la misma forma que se permite la modificación de elementos en el Diseño en Papel, también es posible modificar campos y elementos en la propia presentación del resultado en papel.

Volviendo a nuestro ejemplo y el resultado de la ejecución en papel que hemos obtenido anteriormente, vamos a modificar la columna **Salario** para agregarle un formato de máscara. Para ello hacemos un clic en la columna y dentro del menú seleccionamos **Herramientas – Inspector de Propiedades**. Dentro de la propiedad **Máscara de Formato** vamos a introducir **NNNGNN0DNN.** Este mismo formato lo vamos a indicar sobre la columna total por departamento y sumatorio del listado.

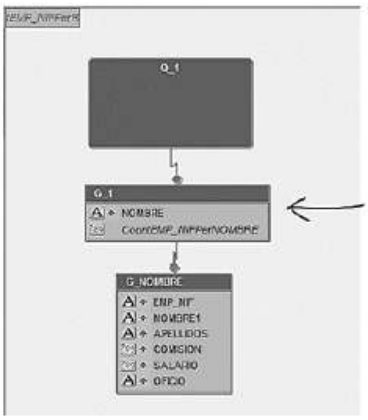
Seguidamente vamos a darle formato al resultado de cada columna de la siguiente forma:

- Las columnas **Nif, Nombre y Apellidos** las vamos a alinear los datos a la izquierda.
- La columna **Oficio** la alinearemos al centro.
- La columna **Comisión** la alinearemos a la derecha con formato de máscara **NNNGNN0DNN**.

Ejecutamos el listado para ver el resultado.

AÑADIENDO COLUMNAS DE FÓRMULA MANUALMENTE CON EL ASISTENTE DE DISEÑO

En este apartado vamos a aprender a añadir de manera manual columnas de fórmula a un informe ya diseñado. Para ejemplificarlo vamos a retornar al informe que llevamos diseñando en este capítulo. Para ello nos situamos en el **modelo de datos**.

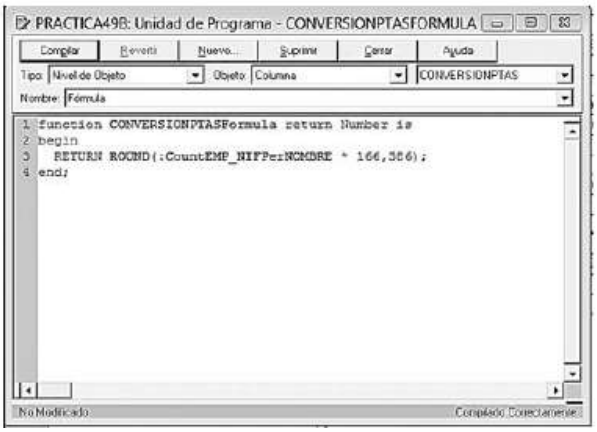


Dentro del modelo de datos la primera fase para la creación de una columna de fórmula es identificar a qué nivel vamos a crear la columna. En nuestro caso práctico deseamos crear una columna de fórmula que realice una conversión de moneda por cada cambio de departamento, por tanto el nivel en el que deberemos crear la columna es el grupo 1.

Por tanto el primer paso a realizar es marcar el grupo **G_1** donde vamos a crear la columna, haciendo un clic en el marco que rodea al elemento de grupo **G_1.** A continuación seleccionamos en la barra de herramientas vertical el botón  que permite crear una columna de fórmula y hacemos clic dentro del cuadro de grupo **G_1** lo que provocará la creación de la columna de fórmula (**CF_1**) con un nombre genérico como se muestra en esta imagen.

Seguidamente abrimos las propiedades de la nueva columna desde el menú **Herramientas – Inspector de Propiedades** y cambiamos la siguiente propiedad:

- Nombre: **CONVERSIONPTAS**



A continuación dentro también del panel de propiedades de la columna resumen pulsamos en la propiedad Fórmula PL/SQL, que nos abre un editor PL/SQL donde debemos introducir la siguiente fórmula de cálculo para nuestra columna: "**RETURN ROUND(:CountEMP_NIFPerNOMBRE * 166,386);**"

Para validar la sintaxis pulsaremos el botón **Compilar** y si no hay errores a continuación el botón **Cerrar** que nos retornará al cuadro de propiedades de la columna. A continuación cerramos también el cuadro de propiedades y volveremos al **Modelo de Datos.** Desde aquí podemos ya ejecutar el listado desde el menú **Programa – Ejecutar Presentación en Papel**, pero vemos que en el resultado no aparece la nueva columna creada esto es debido a que aún tenemos que añadir dicha columna al diseño del listado. Para hacerlo abrimos desde el menú **Herramientas – Asistente de Informes** y pulsamos sobre la solapa **Campos** para a continuación añadir la nueva columna **CONVERSIONPTAS** a la lista de campos visualizados.

Seguidamente pulsamos el botón **Terminar** y abrimos con el botón  el **Editor de Informes en Papel**, que nos mostrará una imagen similar a la siguiente:

Como podemos ver por la indicación de la flecha, el nuevo campo creado aparece junto con su etiqueta en la parte superior del listado. Ahora lo que debemos hacer es colocarlo justo por debajo. En primer lugar seleccionamos las siguientes opciones para mover la etiqueta y el nuevo campo a la izquierda del total departamento:

- **No Limitado.** 
- **No Flexible.** 

Con ello obtendremos un diseño como el que se muestra a continuación:

Seguidamente marcamos la opción **Flexible**  y movemos los elementos una línea por debajo para que obtengamos un diseño como el siguiente:

Por último marcamos las opciones **No Flexible** y **No Limitado** y movemos los elementos justo alineados por debajo del total departamento para obtener el siguiente diseño:

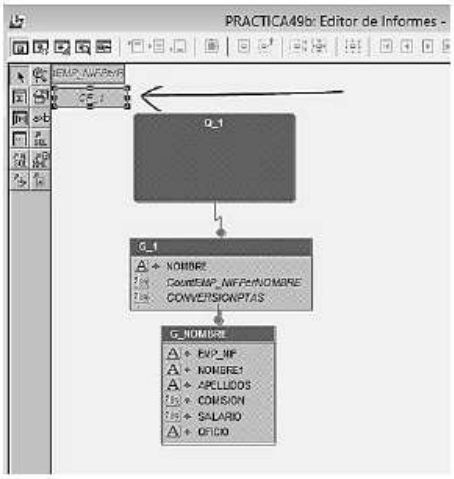
Para completar las modificaciones ajustamos el contenido del nuevo campo de fórmula para que el resultado se alinee a la derecha y además le introducimos una máscara de formato igual a **NNNGNNNGNN0**. También modificamos el texto de la etiqueta para que sea **Total por Departamento (ptas).**

Por último lanzamos la ejecución del listado desde el menú **Programa – Ejecutar Presentación en Papel** para ver el nuevo resultado.

Es posible que después de haber incluido la nueva columna de fórmula y ejecutado el listado, alguna de las etiquetas de las columnas puede haber variado, así como que aparezca el logo antiguo que habíamos eliminado, esto se debe a que hemos incluido la nueva columna resumen a través del **Asistente de Informes**, y esta herramienta no se actualiza con los cambios realizados directamente en el diseño del informe. Para solucionar esto hay que crear la columna y pintarla en diseño del informe sin pasar por el asistente. Es lo que veremos en el siguiente apartado.

AÑADIENDO COLUMNAS DE FÓRMULA MANUALMENTE SIN EL ASISTENTE DE DISEÑO

En este apartado vamos a aprender a añadir de manera manual columnas de fórmula a un informe ya diseñado e incluirlo dentro del diseño sin utilizar el asistente. Para ejemplificarlo vamos a retornar al informe que llevamos diseñando en este capítulo. Para ello nos situamos en el **modelo de datos**.



Dentro del modelo de datos vamos a crear una nueva columna de fórmula a nivel de informe. El resultado que debemos obtener es el siguiente.

A continuación tenemos que definir las propiedades del nuevo campo de fórmula indicando los siguientes valores:

- Nombre: **LISCONVERPTAS**
- Fórmula PL/SQL: **RETURN ROUND((:CONVERSIONPTAS + :COMISION) / 12);**

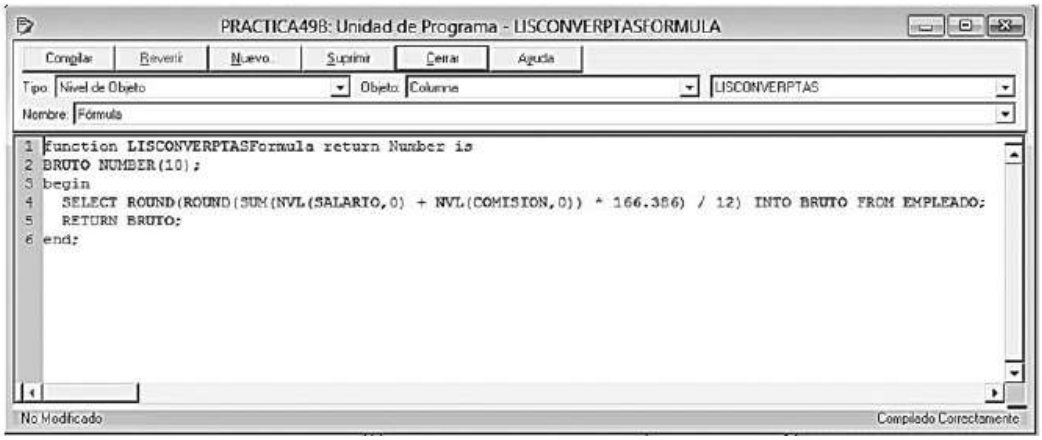
Seguidamente cerramos la paleta de propiedades y entramos en el **Editor de Informes Presentación en Papel**. A continuación seleccionamos en la barra de herramientas vertical el elemento **Campo** y hacemos un clic justo a la izquierda del sumatorio final del listado. Seguidamente cambiamos las propiedades del nuevo elemento insertado en el diseño, con los siguientes valores:

- Nombre: **F_LISCONVERPTAS**
- Origen: **LISCONVERPTAS**
- Máscara de formato: **NNNGNNNGNN0**

También tenemos que añadir una etiqueta a la izquierda del campo con el texto "**Total mensual (12 pagas) en ptas**". El resultado que debemos obtener es un diseño similar al siguiente:

Si ahora lanzamos la ejecución del listado desde **Programa – Ejecutar Presentación en Papel**, aparecerá el siguiente mensaje de error:

Este error es debido a que la fórmula PL/SQL que se ha utilizado hace referencia a un elemento que no pertenece al grupo (en este caso el campo **:COMISION**). Para solventar este problema modificamos la propiedad PL/SQL del campo **LISCONVERPTAS** e indicamos la sentencia que se muestra en la imagen de la página siguiente:



Ahora sí podemos ejecutar el listado desde **Programa – Ejecutar Presentación en Papel** y vemos el resultado sin errores.

Para terminar este capítulo y la práctica que hemos ido desarrollando, tenemos que guardar el report con el nombre **practica49b.jsp.**

DISEÑO DE CONSULTAS ENLAZADAS 50

INTRODUCCIÓN

En este capítulo vamos a aprender a crear varias consultas en el Modelo de Datos de forma manual y cómo se pueden enlazar entre sí a través de campos con semántica equivalente.

CREANDO VARIAS CONSULTAS

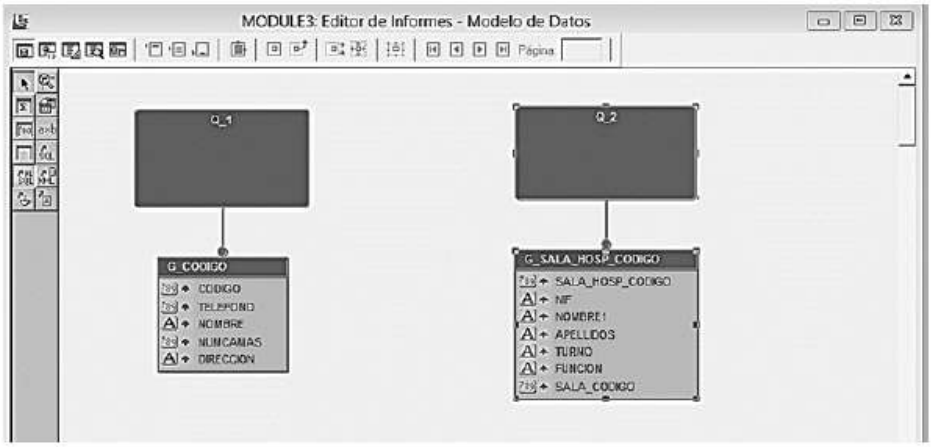
En primer lugar abrimos el Report Builder y en la ventana que se muestra seleccionamos que queremos crear un informe manual. A continuación nos vamos al **Modelo de Datos** y creamos una primera consulta con la siguiente sentencia:

```
SELECT * FROM HOSPITAL
```

Seguidamente creamos una segunda consulta con la siguiente sentencia:

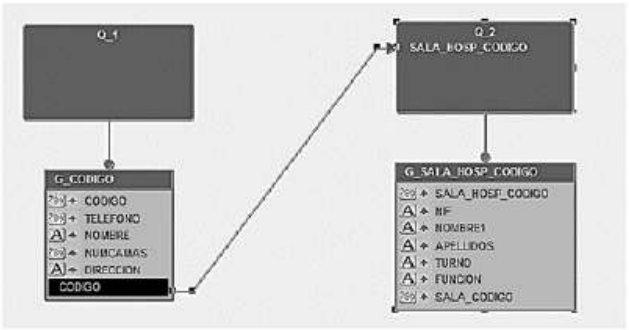
```
SELECT DISTINCT PLANTILLA_SALA.SALA_HOSP_CODIGO,
PLANTILLA.NIF, PLANTILLA.NOMBRE, PLANTILLA.APELLIDOS,
PLANTILLA.TURNO, PLANTILLA.FUNCION,
PLANTILLA_SALA.SALA_CODIGO
FROM PLANTILLA, PLANTILLA_SALA
WHERE (PLANTILLA_SALA.PLAN_NIF = PLANTILLA.NIF)
```

El resultado que tenemos que obtener después de la creación de estas dos consultas es el que se muestra en la imagen de la página siguiente:



ENLAZANDO CONSULTAS

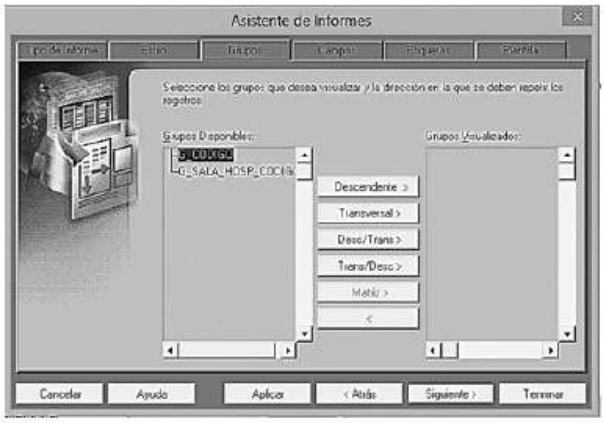
Para enlazar dos consultas hay que situarse en el **Modelo de Datos** y seleccionar el botón  para poder enlazar las consultas conrrespondientes. Seguidamente hay que seleccionar el campo de la tabla origen y desplazar con el ratón hasta el campo de la tabla destino lo que provoca que se enlacen las mismas. En nuestro caso práctico vamos a seleccionar el campo **CODIGO** de la primera consulta y vamos a arrastrarlo hasta el campo **SALA_HOSP_CODIGO** de la segunda consulta, obteniendo un resultado como el que se muestra a continuación:



A continuación tenemos que definir las propiedades de la relación creada. Para ello hay que hacer doble clic sobre la flecha que une las consultas.

En nuestro caso práctico pulsaremos sobre la única flecha que existe en la consulta y se nos abrirá el inspector de propiedades que tiene un aspecto como el mostrado en la imagen de la derecha.

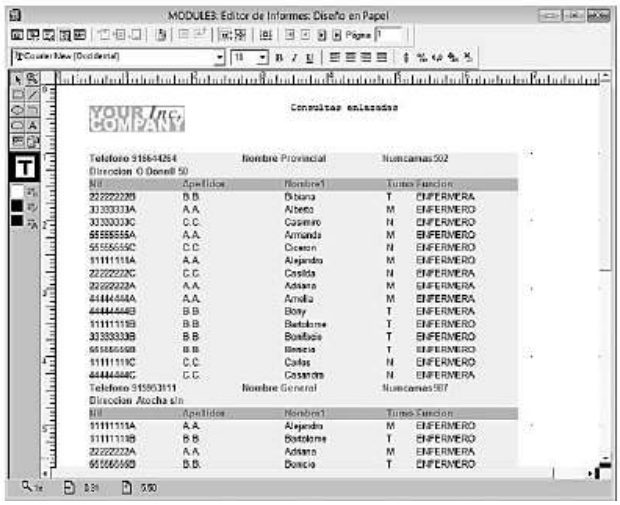
Continuamos nuestra práctica diseñando el formato del informe y para ello seleccionamos en el menú **Herramientas – Asistente de Informes** y mantenemos la opción **Crear Diseño Web y Presentación en Papel**. Seguidamente nos situamos en la solapa **Estilo** y seleccionamos **Agrupar Arriba** e indicamos como título del listado "**Consultas enlazadas".**



Seguidamente pasamos a la solapa **Grupos** y podemos observar en la imagen que viene a continuación que el formato es diferente al que se presenta en esta solapa cuando solo hay una consulta.

Los botones que se presentan en esta solapa en el caso de varias consultas enlazadas son: **Descendente, Transversal, Descendente/Transversal** y **Transversal/Descendente.** Estos valores indican la disposición que queremos para el agrupamiento.

Volviendo a nuestro caso práctico vamos a seleccionar en esta pantalla de agrupamiento la opción **Descendente** para los dos campos propuestos. Luego pasaremos a la solapa **Campos** donde visualizaremos todos los campos menos los que se indican a la izquierda de la imagen.



Tras esto podemos pulsar **Terminar** para ver el resultado del diseño, que deberá ser similar al que se muestra a continuación.

Para concluir almacenamos el informe con el nombre **practica50.jsp.**

DISEÑO MANUAL DEL FORMATO DEL INFORME 51

INTRODUCCIÓN

En este capítulo vamos a practicar el diseño manual de los distintos frames (marcos de recubrimiento de objetos) de los que se compone un informe, sin utilizar el asistente.

Los pasos iniciales para comenzar las prácticas que vamos a realizar en este capítulo comienzan con la apertura de un nuevo Report desde la herramienta Report Builder, que sea manual. A continuación dentro del **Modelo de Datos** diseñaremos una consulta con la siguiente sentencia:

```
SELECT * FROM HOSPI TAL
```

Seguidamente nos situaremos en el  el **Editor de Informes en Papel**, para comenzar el diseño de nuestro informe.

AÑADIENDO ELEMENTOS A UN DISEÑO

En primer lugar vamos a añadir los elementos a la **Sección de Cabecera** del informe y para ello pulsamos el botón  .

Comenzaremos añadiendo una imagen al diseño, para ello desde el menú seleccionaremos **Insertar – Imagen** y añadimos la imagen **logo_hospital.jpg** que se suministra con este manual dentro de los ejemplos prácticos del libro.

A continuación insertaremos el título del listado pulsando en la barra de herramientas vertical el botón y escribiendo el texto "**LISTADO CREADO MANUALMENTE**".

Seguidamente vamos a crear un campo de contenido en la cabecera del informe y para ello pulsamos el botón en la barra de herramientas vertical, para añadirlo en la posición que se indica en la imagen siguiente:

Ahora vamos a indicar las propiedades que definen el nuevo campo desde el menú **Herramientas – Inspector de Propiedades**. Los valores a indicar en las propiedades del campo son los siguientes:

- Nombre: **F_FECHADIA**
- Origen: **Fecha Actual**
- Tipo de Dato Origen: **Date**
- Visible: **SI**
- Máscara de Formato: **DD/MM/YYYY**

Después de introducir estas propiedades cerramos la ventana y al volver al Editor de Informes, ajustamos el contenido del nuevo campo al centro pulsando el botón de la barra horizontal y después añadimos una etiqueta a la izquierda del campo con el literal "Fecha Actual:", con lo que obtendremos el siguiente resultado en el diseño:

A continuación vamos a crear dos campos más de contenido con las siguientes propiedades:

Campo 1:

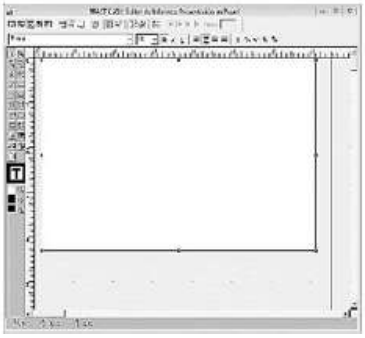
- Nombre: **F_NUMPAG**
- Origen: **Número de Pagina**
- Tipo de Dato Origen: **Number**
- Visible: **SÍ**

Campo 2:

- Nombre: **F_PAGTOTAL**
- Origen: **Total de Páginas**
- Tipo de Dato Origen: **Number**
- Visible: **SÍ**

Después de crear estos campos, el resultado del diseño debe ser similar al siguiente:

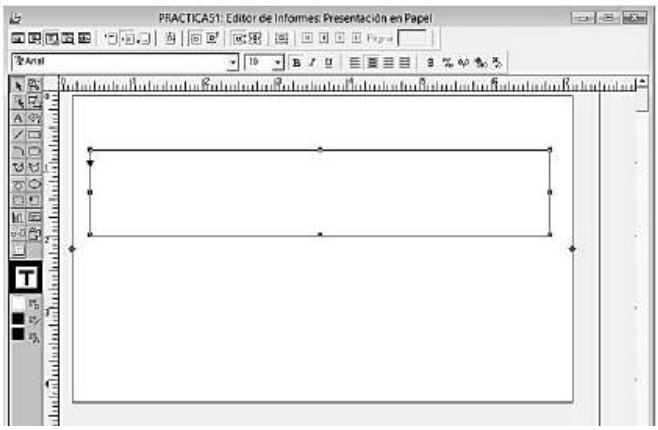
Con esto concluimos el diseño de la cabecera y vamos a pasar a diseñar el pie de página del listado. Para ello ahora pulsamos el botón  .



Ahora vamos a diseñar la sección principal o detalle del listado. Para ello pulsamos el botón . En esta sección vamos a comenzar añadiendo un elemento de **marco**, pulsando para ello sobre el botón que se muestra en la barra de herramientas vertical. El tamaño del marco será aproximadamente el que se muestra en esta imagen.

Una vez pintado el marco abrimos las propiedades del mismo desde el menú **Herramientas – Inspector de Propiedades**. Los valores a indicar en las propiedades del marco son los siguientes:

- Nombre: **MARCO_CONTENEDOR**
- Elasticidad Vertical: **Variable**



Después de cerrar las propiedades del marco, creamos un **marco de repetición** pulsando en el botón en la barra de herramientas vertical con un tamaño similar al que se muestra en esta imagen.

Una vez pintado el marco de repetición debemos de indicar las siguientes propiedades del mismo:

- Nombre: **MARCO_REPETICION_HOSPITAL**
- Origen: **G_CODIGO**
- Imprimir Objeto en: **Todas las Páginas**

Después de cerrar las propiedades del marco de repetición vamos a insertar 4 campos de contenido dentro del marco de repetición con las siguientes propiedades:

Campo 1:

- Nombre: **F_CODHOSPITAL**
- Origen: **CODIGO**
- Visible: **SÍ**

Campo 2:

- Nombre: **F_NOMBREHOSPITAL**
- Origen: **NOMBRE**
- Visible: **SÍ**

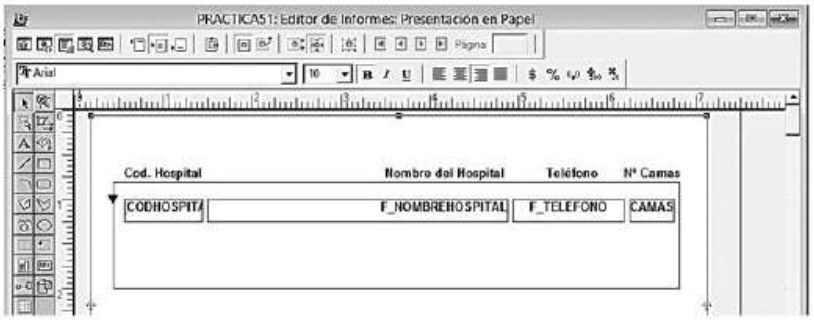
Campo 3:

- Nombre: **F_TELEFONO**
- Origen: **TELEFONO**
- Visible: **SÍ**

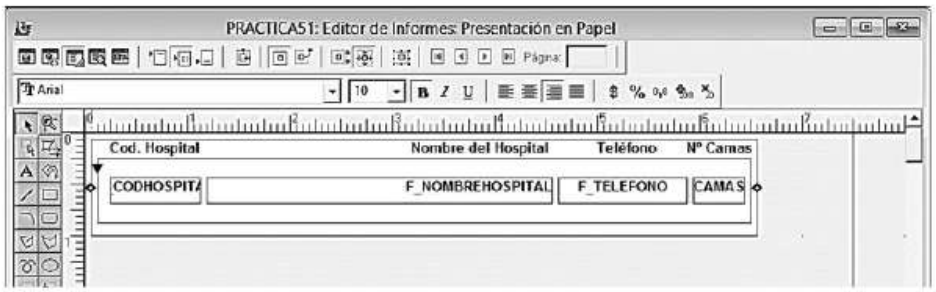
Campo 4:

- Nombre: **F_NUMCAMAS**
- Origen: **NUMCAMAS**
- Visible: **SÍ**

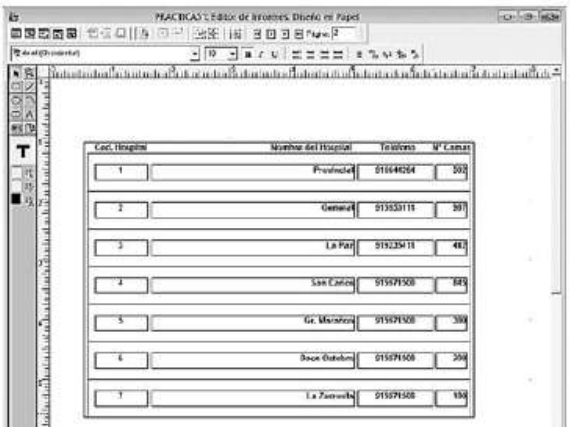
Después de hacerlo debemos de tener un diseño similar al que se muestra en la imagen siguiente (incluidas las etiquetas de los nuevos campos que hemos creado fuera del marco contenedor):



A continuación vamos a ajustar el diseño del listado (incluidos los tamaños de los marcos) para dejar un aspecto similar al que se muestra a continuación:



Ahora vamos a ejecutar el listado para ver cómo se obtiene el resultado en modo papel, para ello desde el menú **Programa** pulsamos en la opción **Ejecutar Presentación en Papel**. El resultado que obtendremos será de 2 páginas como las que se muestran a continuación:



MOVIENDO/COPIANDO OBJETOS ENTRE SECCIONES

Como hemos podido observar en el resultado de la ejecución del listado, aparecen 2 páginas diferenciadas con las 3 secciones diseñadas: cabecera, cuerpo.

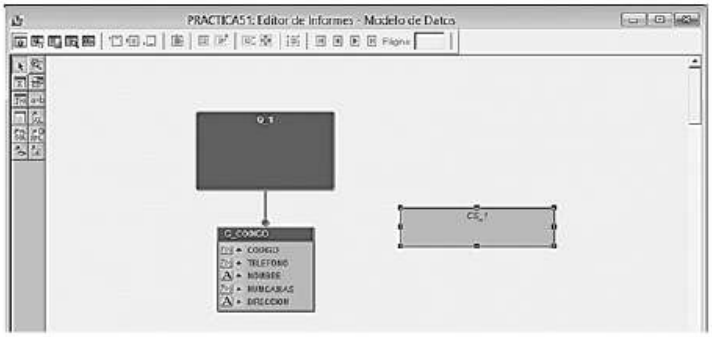
En este apartado vamos a explorar las posibilidades que ofrece Report Builder para mover/copiar elementos entre las secciones con el fin de unificar cabecera y pie dentro de la hoja de detalle (sección principal).

En primer lugar vamos a mover todos los elementos de la sección de cabecera a la sección principal, para ello nos situamos en la **sección de cabecera** y desde el menú **Editar** pulsamos la opción **Seleccionar Todo**. A continuación desde el mismo menú **Editar** pulsamos la opción **Cortar.** A continuación nos situamos en la **Sección Principal** y pulsamos en el botón **Editar Margen.** Tras realizar esta operación volveremos al menú **Editar** y pulsaremos la opción **Pegar**, lo que provocará que los elementos de la cabecera se incrusten en la sección principal como se muestra a continuación:

Ahora vamos a ejecutar el listado para ver el resultado de las modificaciones efectuadas, para ello desde el menú **Programa** pulsamos en la opción **Ejecutar Presentación en Papel**. El resultado que obtendremos será de 1 página como la que se muestra a continuación:

AÑADIENDO UNA COLUMNA RESUMEN AL LISTADO

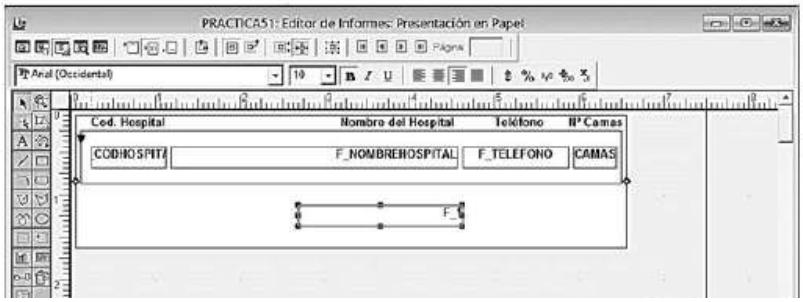
Finalizadas las modificaciones del aspecto y diseño del listado, vamos a introducir una columna resumen al diseño que se ubicará en la sección principal del listado. Para ello nos situamos en el **Modelo de Datos**. A continuación pulsamos en el botón  de la barra de herramientas vertical y añadimos la columna resumen como se muestra en la imagen siguiente:



Seguidamente tenemos que indicar las siguientes valores a las propiedades que se indican del elemento resumen:

- Nombre: **RECUENTO_HOSPITAL**
- Tipo de columna: **Resumen**
- Tipo de Dato: **Number**
- Ancho: **10**
- Función: **Recuento**
- Origen: **CODIGO**

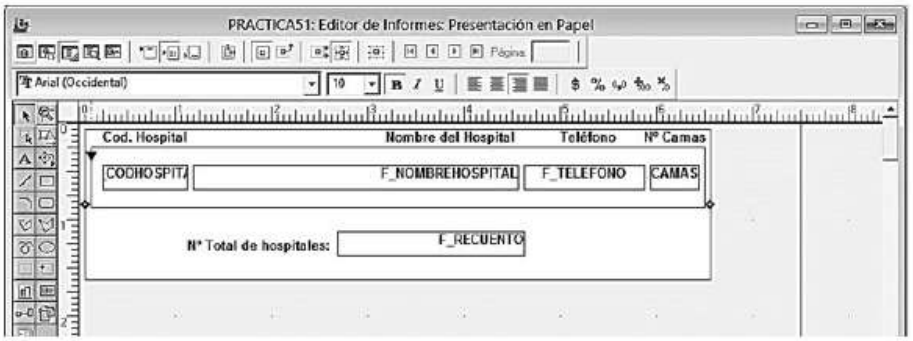
A continuación nos vamos al **Editor de Informes (Presentación en Papel)** y añadimos un nuevo campo a la Sección Principal, dentro del marco contenedor como se muestra en la siguiente imagen:



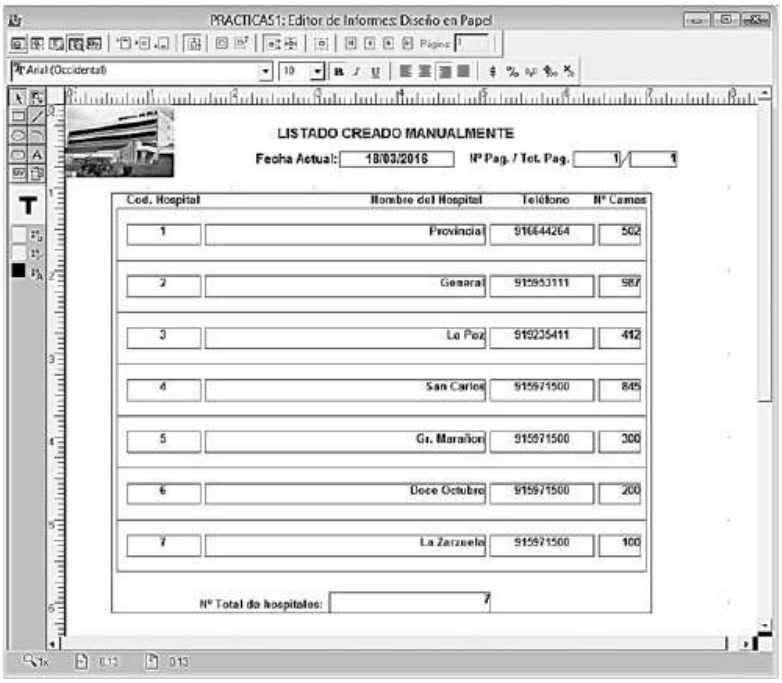
Una vez creado, abrimos las propiedades del campo e introducimos los siguientes valores de propiedad:

- Nombre: **F_RECUENTO**
- Origen: **RECUENTO_HOSPITAL**

Por último añadimos una etiqueta para el campo con el literal "**Nº Total de hospitales:**" y el resultado obtenido en el diseño debería ser similar al siguiente:



Seguidamente ejecutamos el listado para ver el resultado, para ello desde el menú **Programa** pulsamos en la opción **Ejecutar Presentación en Papel**. El resultado que obtendremos será de 1 página como la que se muestra a continuación:



Para finalizar el capítulo guardamos el informe con el nombre **practica51.jsp.**

LA PANTALLA DE PARÁMETROS EN REPORTS 52

INTRODUCCIÓN

Una pantalla de parámetros en Reports es aquella que se presenta antes de ejecutar el listado, y mediante la cual podremos alterar el resultado del mismo de acuerdo a los valores que introduzcamos en los parámetros que se visualicen en la misma.

Por tanto, la pantalla de parámetros del Reports permite personalizar al usuario los resultados que pretende obtener en el listado, según las alternativas que le presente el programador en la misma.

En este capítulo vamos a diseñar un informe asistido y posteriormente vamos a añadirle una pantalla de parámetros para aprender cómo se crean.

CREANDO UN INFORME ASISTIDO

En primer lugar abrimos Reports Builder y en la ventana de bienvenida seleccionamos la creación de un nuevo informe utilizando el **Asistente de Informes** y pulsamos **Aceptar.**

Seguidamente seleccionamos la opción **Crear Diseño Web y Presentación en Papel** y pulsamos el botón **Siguiente.** A continuación en la pantalla para seleccionar el tipo de diseño del informe elegimos el estilo **Agrupar Arriba** e indicamos el título "**Empleados por departamento".**

A continuación pulsamos el botón **Siguiente**, que nos mostrará la pantalla para indicar el tipo de origen de datos. En nuestro caso seleccionaremos la opción **SQL Query** y pulsaremos el botón **Siguiente.**

A continuación se mostrará la pantalla donde tenemos que introducir la consulta que queremos diseñar para el listado.

Una vez introducida la consulta, pulsamos el botón **Siguiente.** En la pantalla que se muestra a continuación, tenemos que seleccionar el campo que se utilizará para el agrupamiento. En nuestro caso tenemos que elegir el campo **DEPT_CODIGO.** Después de introducir el campo de agrupamiento pulsamos el botón **Siguiente.**

En la siguiente pantalla que se muestra hay que seleccionar los campos que se quieren visualizar en el informe. Indicaremos que se muestren los siguientes campos: **DEPT_CODIGO, NOMBRE, APELLIDOS, OFICIO, SALARIO.**

Cuando hayamos concluido con la selección de campos a visualizar pulsaremos el botón **Siguiente.** En la pantalla que se muestra a continuación tenemos que indicar los campos sobre los que queremos mostrar un total del agrupamiento. En este caso marcaremos que queremos realizar un **recuento** del **NIF** y la **suma** del **SALARIO**.

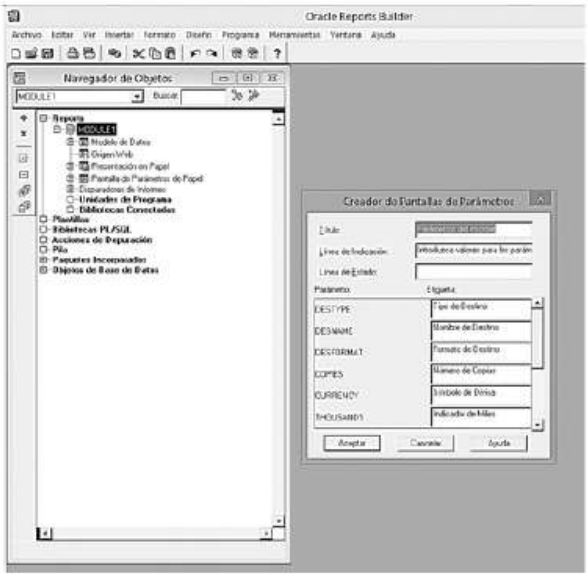
Después de realizar esta operación pulsaremos el botón **Siguiente** y en la pantalla que se muestre con las etiquetas que se van a dar a los campos, pulsaremos de nuevo el botón **Siguiente** sin realizar cambios. Después se nos mostrará otra pantalla para seleccionar la plantilla que se utilizará para visualizar los datos:

marcaremos por defecto la opción mostrada (**Beige**) y pulsaremos el botón **Siguiente.**

Para finalizar el diseño pulsamos el botón **Terminar**, lo que provocará que se muestre el resultado del listado.

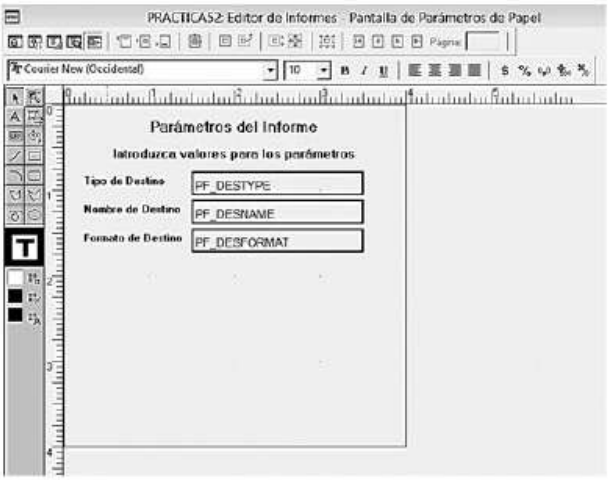
DISEÑANDO UNA PANTALLA DE PARÁMETROS

Una vez concluido el diseño del listado, vamos a crear una pantalla de parámetros para que el usuario pueda obtener diferentes resultados según los criterios de selección que le mostremos en la pantalla.



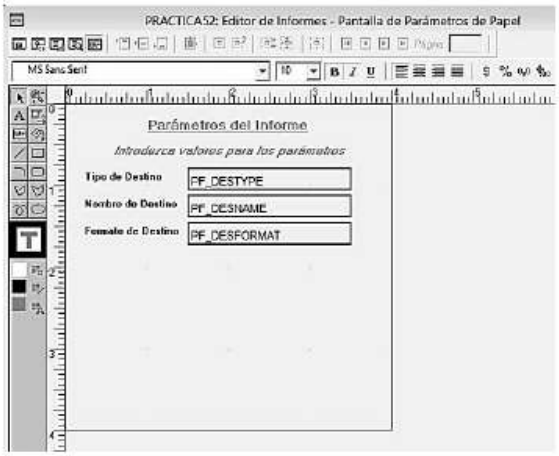
Para crear una pantalla de parámetros tenemos que entrar en el menú **Herramientas** y elegimos la opción **Creador de Pantallas de Parámetros.** Al realizar esta operación nos aparecerá la pantalla de parámetros donde se muestran automáticamente una serie de parámetros por defecto.

Para seleccionar los parámetros que verdaderamente queremos visualizar en la pantalla de parámetros tenemos que ir señalando cada uno de ellos (se marcarán en negro al hacerlo). En nuestro caso vamos a marcar los parámetros **DESTYPE, DESNAME y DESFORMAT**.

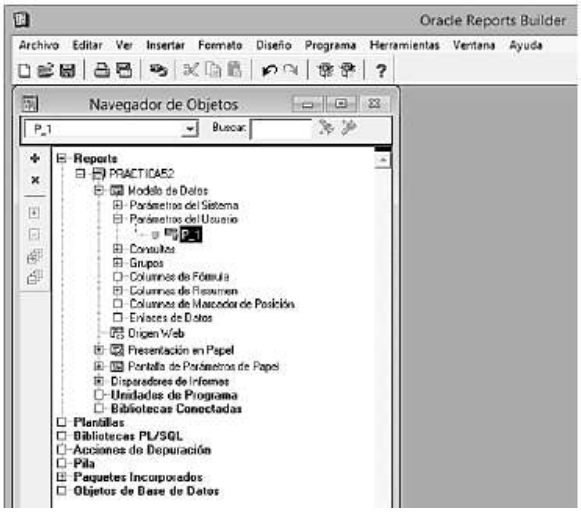


Una vez seleccionados los parámetros a visualizar, pulsamos el botón **Aceptar** y se mostrará el diseñador de la pantalla de parámetros como se observa a continuación.

Vamos a realizar dos cambios en esta pantalla, para poner en color azul y subrayado el título de la pantalla y en rojo e itálica el texto para introducir los valores, tal como se muestra a continuación.



Ahora salimos del diseñador de la pantalla de parámetros y nos vamos al navegador de objetos, donde desplegamos el grupo **Modelo de Datos** y dentro del grupo **Parámetros del Usuario** creamos uno nuevo pulsando el símbolo **+**, lo que dará de alta el parámetro **P_1**, como se muestra debajo en la imagen.



Seguidamente abrimos las propiedades del nuevo parámetro para modificar las siguientes:

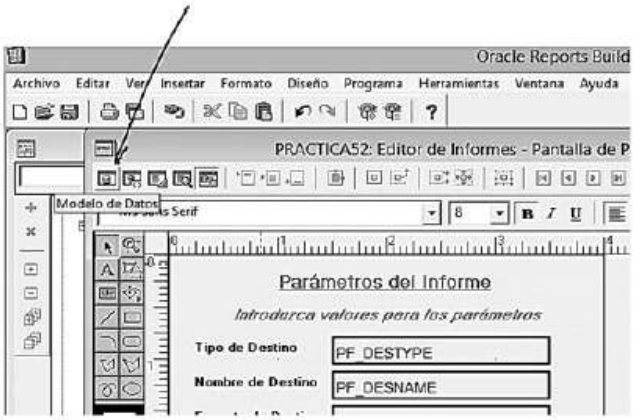
- Nombre: **P_CODEPARTAMENTO**
- Lista de Valores: **SELECT ALL DEPARTAMENTO.CODIGO FROM DEPARTAMENTO**

Cuando hayamos concluido la modificación de las propiedades pulsamos el botón **Aceptar** para salir de ellas y volver al navegador de objetos. Dentro del navegador desplegamos el grupo **Pantalla de Parámetros de Papel** y hacemos un doble clic sobre este nombre para que vuelva a abrir el diseñador de pantalla de parámetros.

Una vez dentro del diseñador pulsamos en la barra vertical, sobre el elemento, para crear un nuevo campo dentro de la pantalla, como se muestra en la siguiente imagen. A continuación abrimos las propiedades del nuevo elemento (pulsando doble clic sobre el mismo) y modificamos las siguientes:

- Nombre: **PF_CODPTO**
- Origen: **P_CODEPARTAMENTO**

Cerramos las propiedades del elemento una vez modificadas y al volver al diseñador de los parámetros de pantalla, vamos a crear un literal para el nuevo campo como se muestra en esta imagen. Para ello tendremos que pulsar sobre la barra de herramientas vertical en el botón y luego escribir el texto deseado.



Finalizado el diseño de la pantalla de parámetros vamos a abrir el Modelo de Datos, pulsando para ello donde se indica en la imagen de la izquierda.

Dentro del modelo de datos abrimos la consulta **Q_1** haciendo un doble clic sobre el nombre de la misma, lo que presentará el diseñador de consultas SQL. Dentro del mismo vamos a cambiar la consulta para añadir un nuevo condicionante: **AND DEPARTAMENTO_EMPLEADO.DEPT_CODIGO = : P_CODEPARTAMENTO**

El resultado debe ser el mostrado en la imagen de la derecha.

De esta forma permitimos que cuando el usuario introduzca un valor para el código del departamento en la pantalla de parámetros, la consulta solo muestre los resultados de dicho código. Pulsamos **Aceptar** para cerrar la consulta.

Ahora ejecutamos el listado en modo **Dispositivo Papel**, para ello pulsamos en el menú la opción **Programa** y a continuación **Ejecutar Presentación en Papel**, lo que hará que se muestre en primer lugar la pantalla de parámetros como se observa en esta imagen.

Pulsamos el botón  para lanzar el listado con los valores por defecto mostrados. El resultado obtenido será el siguiente:

Ahora podemos realizar distintas ejecuciones modificando los valores de la pantalla de parámetros para observar cómo cambia el resultado.

Para finalizar el capítulo, almacenamos el informe con el nombre **PRACTICA52.jsp**.

INSERTAR GRÁFICOS EN UN REPORT 53

INTRODUCCIÓN

Un informe de Reports permite adjuntar un gráfico de Oracle Graphics en el mismo. En este capítulo vamos a aprender a realizarlo.

CREANDO UN INFORME ASISTIDO

En primer lugar abrimos Reports Builder y en la ventana de bienvenida seleccionamos la creación de un nuevo informe utilizando el **Asistente de Informes** y pulsamos **Aceptar.** Seguidamente seleccionamos la opción **Crear Diseño Web y Presentación en Papel** y pulsamos el botón **Siguiente.** A continuación en la pantalla para seleccionar el tipo de diseño del informe elegimos el estilo **Agrupar a la izquierda** e indicamos un título del mismo: "Enfermos por hospitales". A continuación pulsamos el botón **Siguiente**, que nos mostrará la pantalla para indicar el tipo de origen de datos. En nuestro caso seleccionaremos la opción **SQL Query** y pulsaremos el botón **Siguiente.**

Una vez introducida la consulta pulsamos el botón **Siguiente.** En la pantalla que se muestra a continuación, tenemos que seleccionar el campo que se utilizará para el agrupamiento. En nuestro caso tenemos que elegir el campo **HOSP_CODIGO.**

Después de introducir el campo de agrupamiento pulsamos el botón **Siguiente.** En la siguiente pantalla que se muestra hay que seleccionar TODOS los campos para visualizar en la pantalla. Cuando hayamos concluido con la selección de campos a visualizar pulsaremos el botón **Siguiente.** En la pantalla que se muestra a continuación tenemos que indicar los campos sobre los que queremos mostrar un total del agrupamiento. En nuestro caso práctico, marcaremos que queremos realizar un **recuento** del campo **INSCRIPCION**.

Después de realizar esta operación pulsaremos el botón **Siguiente** y en la pantalla que se muestre con las etiquetas que se van a dar a los campos, pulsaremos de nuevo el botón **Siguiente** sin realizar cambios. Después se nos mostrará otra pantalla para seleccionar la plantilla que se utilizará para visualizar los datos: marcaremos por defecto la opción mostrada (**Beige**) y pulsaremos el botón **Siguiente.**

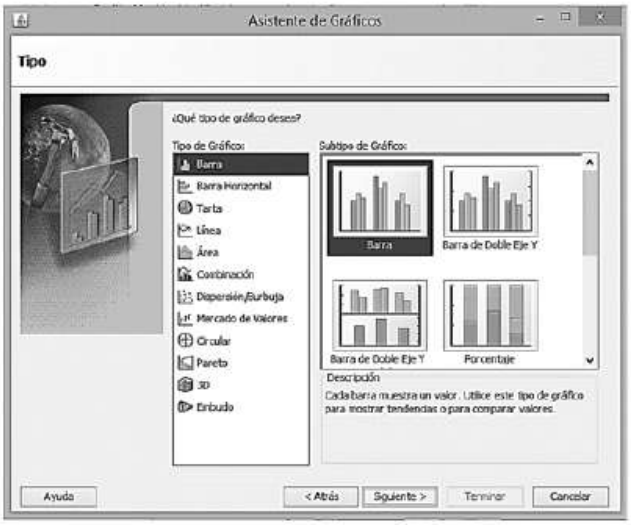
Para finalizar el diseño pulsamos el botón **Terminar**, lo que provocará que se muestre el resultado del listado.

DISEÑANDO UN GRÁFICO DENTRO DE UN REPORT

En primer lugar nos tenemos que situar en la sección donde deseamos pintar el gráfico. Para nuestro caso práctico vamos a elegir la **sección Final** dentro del Editor de Informes, pulsando sobre el botón siguiente:

Una vez dentro de la sección final insertamos un elemento gráfico seleccionando desde el menú **Insertar** la opción **Gráfico**.

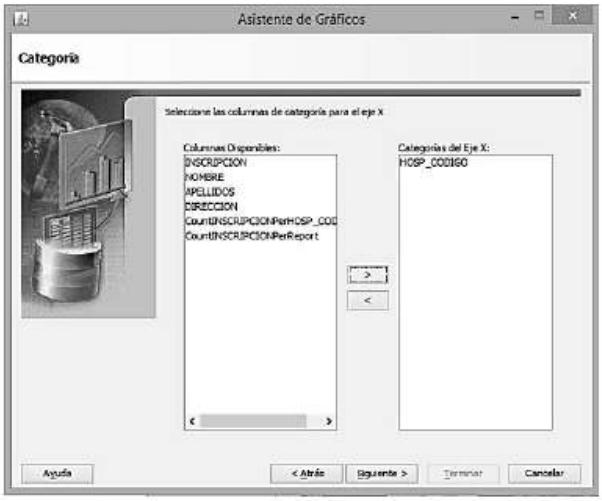
Desde la pantalla de bienvenida para la creación de gráficos, pulsamos el botón **Siguiente.**

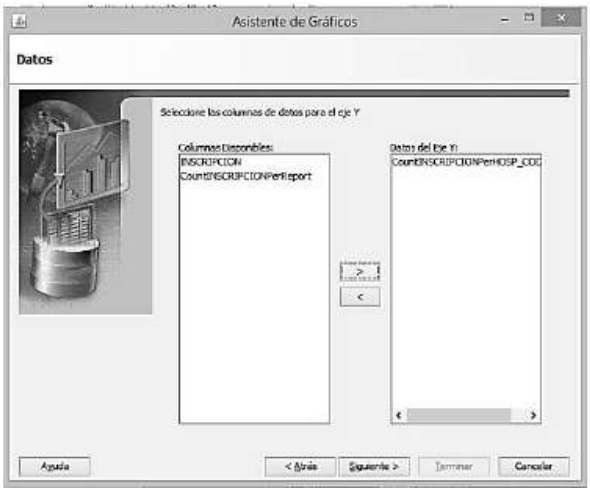


En la pantalla que se muestra a continuación hay que seleccionar el tipo de gráfico que se quiere pintar en el informe. En nuestro caso práctico vamos a elegir un gráfico de **Barras.**

Pulsamos el botón **Siguiente** y en la siguiente pantalla tenemos que seleccionar la ubicación del gráfico en el listado. En nuestro caso vamos a seleccionar la opción **al final del informe**.

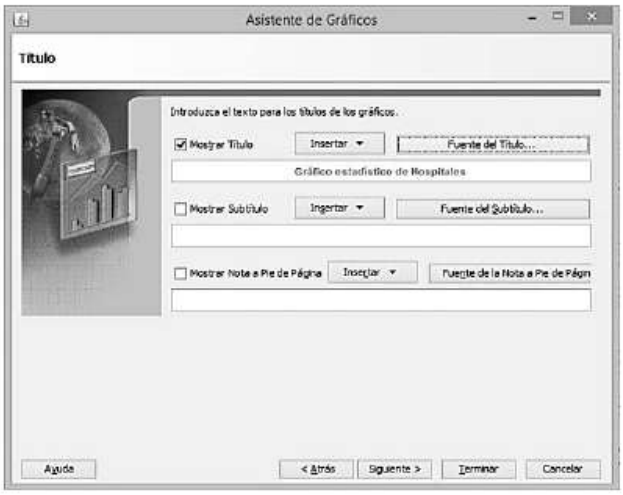
A continuación en la siguiente pantalla comienza el diseño de los campos que se van a utilizar para pintar cada uno de los ejes del gráfico. El primer elemento a seleccionar es el campo que formará parte del eje X. En nuestro caso práctico seleccionamos como campo **HOSP_CODIGO** como se muestra en esta imagen.





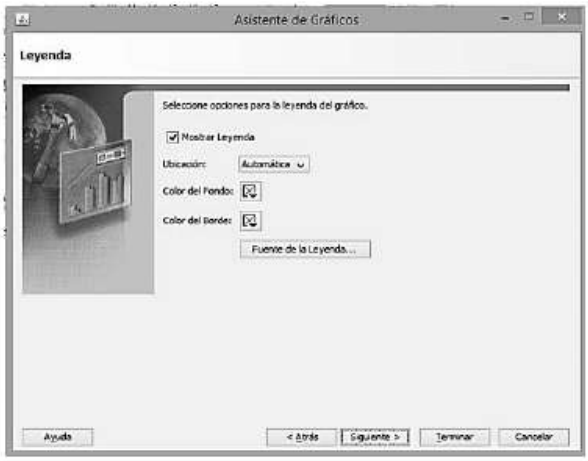
Pulsamos **Siguiente** y seleccionamos como eje Y el campo **CountINSCRIPCIONPerHOSP_CODIGO**.

A continuación en la siguiente pantalla no hacemos modificaciones en los elementos del gráfico que aparecen por defecto y pulsamos el botón **Siguiente.**

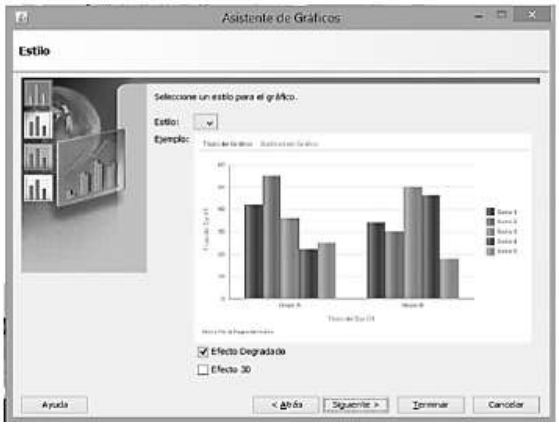


Seguidamente en la pantalla que se muestra en la página siguiente, podemos seleccionar opciones sobre las leyendas que aparecerán en el gráfico: título, subtítulo, nota a pie de página. En nuestro caso vamos a marcar la opción de **Mostrar Título** y vamos a introducir la leyenda: "**Gráfico estadístico de Hospitales".**

Pulsamos en la misma el botón **Siguiente.** A continuación se muestra otra pantalla para reconfigurar las leyendas del gráfico.



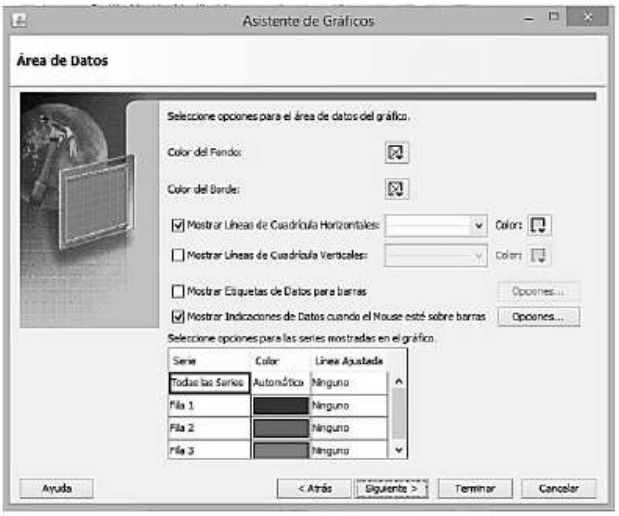
Mantendremos la opciones que se muestran por defecto y pulsaremos el botón **Siguiente.** Seguidamente se muestra otra pantalla donde podemos configurar el estilo de visualización de los gráficos.



Mantendremos las opciones que se muestran por defecto y pulsaremos el botón **Siguiente.** Seguidamente se muestra otra pantalla donde podemos personalizar la leyenda que se quiere mostrar en el eje X. En nuestro caso práctico vamos a marcar la casilla **Mostrar Título del Eje X** e introduciremos la leyenda "**Hospitales".**

Pulsamos **Aceptar** y al volver a la pantalla de configuración de la leyenda del eje X, mantenemos el resto de opciones por defecto.

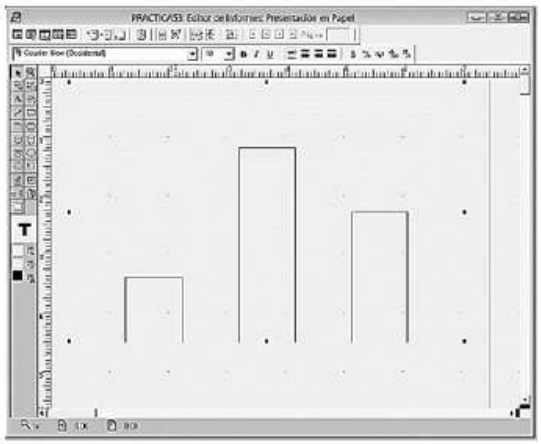
Pulsamos **Siguiente** y realizamos la misma operación pero ahora para el eje Y. En este caso la leyenda será **"Nº de Pacientes"** con el mismo formato de fuente que se utilizó en el eje X.

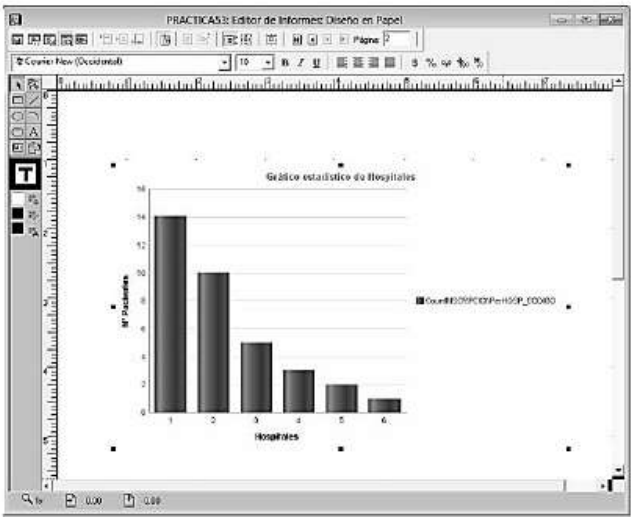


Pulsamos **Siguiente** y se mostrará una nueva pantalla con las opciones de colores del área de datos del gráfico. Mantendremos todas las opciones por defecto.

Pulsamos el botón **Siguiente** y en la siguiente pantalla que se muestra donde se permite enlazar áreas del gráfico con páginas Web, pulsamos directamente el botón **Terminar.**

Al finalizar el proceso de diseño del gráfico, retornaremos al **Editor de diseño** donde se mostrará el área que ocupará el gráfico. En este momento podemos redimensionar el tamaño del gráfico a nuestro gusto. En nuestro caso debemos obtener un tamaño de gráfico similar al que se muestra en esta imagen.





Concluida esta operación, podemos ejecutar el report para ver el resultado y la gráfica que hemos diseñado, obteniendo un resultado similar al siguiente.

Ahora almacenamos el report con el nombre **practica53.jsp.**

USO DE TRIGGERS EN UN REPORT 54

INTRODUCCIÓN

Un informe de Reports puede ser controlado en tiempo de ejecución mediante el uso de triggers (eventos).

El número de triggers que se pueden utilizar en Reports es mucho más limitado que los existentes en Oracle Forms. En concreto existen los siguientes tipos en Reports:

- Triggers para ejecutar antes/después de la pantalla de parámetros.
- Triggers para ejecutar antes/después del informe.
- Trigger para ejecutar entre páginas.
- Triggers de validación.
- Triggers de formato.
- Triggers de acción.

Trigger para ejecutar antes de la pantalla de parámetros

Se dispara antes de visualizar la pantalla de parámetros. Desde este trigger se puede acceder y cambiar:

- Los valores de los parámetros.
- Las variables globales PL/SQL
- Columnas a nivel de Report.

Aunque se suprima la pantalla de parámetros, el trigger de este tipo continuará ejecutándose. Consecuentemente se pueden usar para validar parámetros de la línea de comandos.

Trigger para ejecutar después de la pantalla de parámetros

Se dispara después de visualizar la pantalla de parámetros. Consecuentemente se puede usar para validar parámetros de la línea de comandos u otros datos.

Trigger para ejecutar antes del informe

Se dispara antes de ejecutar el informe pero después de que las consultas asociadas al informe hayan sido cargadas.

Trigger para ejecutar después del informe

Se dispara después de que se haya ejecutado el informe y se haya enviado al destino especificado (tal como un fichero, impresora, email, etc.), o bien se haya salido del previsualizador de informes.

Este trigger se puede utilizar para arreglar, confirmar, limpiar o rechazar cualquier proceso inicial que se haya ejecutado, tal como el borrado de tablas. Sin embargo, hay que tener en cuenta que este trigger siempre se ejecutará aunque el report se haya completado o no satisfactoriamente.

Trigger para ejecutar entre páginas

Se dispara antes de que cada página del informe sea formateada, con excepción de la primera página.

Este trigger se puede utilizar para personalizar el formato de página. Hay que tener en cuenta que si se está previsualizando el informe, este trigger solo se disparará la primera vez que se vaya a la página. Si se retorna con posterioridad a la misma no se volverá a disparar el trigger.

Triggers de validación

Este tipo de triggers son funciones PL/SQL que se ejecutan cuando los valores de los parámetros se especifican en la línea de comandos y cuando se aceptan los valores de la pantalla de parámetros.

Se suelen utilizar para validar los valores iniciales de los parámetros.

Triggers de formato

Este tipo de triggers son funciones PL/SQL que se ejecutan antes de que el objeto sea formateado.

Este tipo de triggers se pueden utilizar dinámicamente para cambiar los atributos de formato de los objetos.

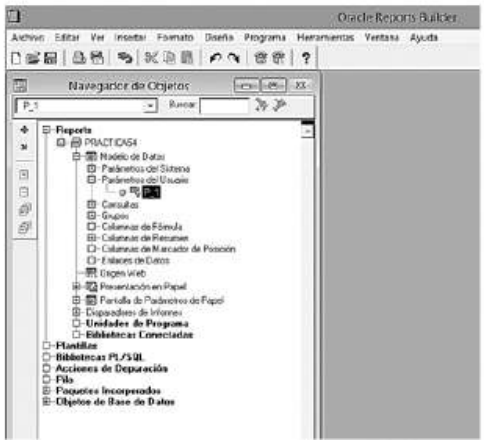
Triggers de acción

Este tipo de triggers son procedimientos PL/SQL que se ejecutan cuando un botón es seleccionado en el previsualizador de informes.

Este tipo de triggers se pueden utilizar dinámicamente para llamar a otro report o ejecuta cualquier otro código PL/SQL.

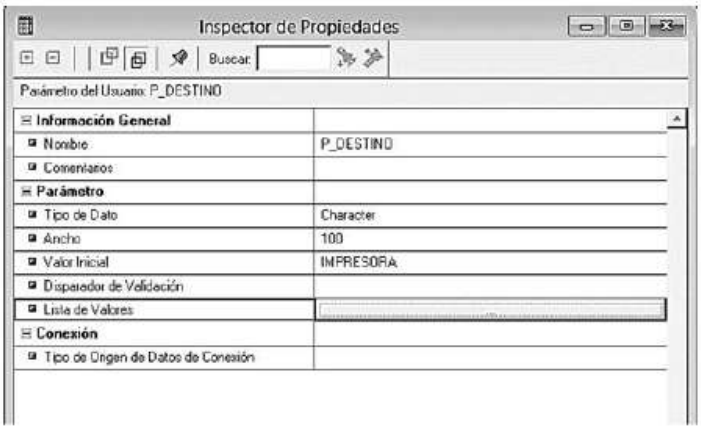
MODIFICANDO UN INFORME EXISTENTE

En primer lugar abrimos Reports Builder y en la ventana de bienvenida seleccionamos la apertura de **un informe existente**. En este supuesto práctico vamos a abrir el report **PRACTICA53.JSP.**



Una vez abierto lo vamos a guardar con el nombre **PRACTICA54.JSP.**

A continuación nos situamos en el navegador de objetos y desplegamos el **Modelo de Datos** y dentro de la sección **Parámetros de Usuario** vamos a añadir uno nuevo (**P_1**) como se muestra en esta imagen.



Seguidamente abrimos las propiedades del nuevo parámetro y ajustamos sus valores a los que se muestran en la siguiente imagen modificando también el nombre del parámetro a **P_DESTINO.**

Sin abandonar aún la pantalla de propiedades del parámetro nos vamos a situar en la propiedad **Lista de Valores** y vamos a pulsar el botón que aparece a su derecha para configurar los valores disponibles para dicho parámetro como se muestra en esta imagen.

Para añadir valores estáticos a una lista de valores únicamente hay que introducir el texto del valor en el campo **Valor** y a continuación pulsar el botón **Agregar**. Así con cada uno de los valores que queramos que estén disponibles para dicho parámetro.

Un vez finalizada la introducción de valores pulsamos el botón **Aceptar** que nos llevará de nuevo a la ventada de propiedades del parámetro P_DESTINO. A continuación cerramos también la ventana de propiedades del parámetro.

CREANDO UNA PANTALLA DE PARÁMETROS

En este punto vamos a crear una pantalla de parámetros personalizada para el informe, pulsando en el menú de Report la opción **Herramientas – Creador de Pantalla de Parámetros.**

Dentro de la pantalla de creación de parámetros, vamos a marcar únicamente para visualizar los parámetros siguientes: **ORIENTATION** y **P_DESTINO.**

CONFIGURANDO TRIGGERS ASOCIADOS A UNA VENTANA DE PARÁMETROS

Cerramos la pantalla de parámetros pulsando el botón **Aceptar** y volvemos al navegador de objetos para desplegar el grupo de elementos **Disparadores de Informes.**

Vamos a configurar el trigger **BEFORE PARAMETER FORM**, para ello hacemos doble clic en el trigger con dicho nombre, lo que hará que se muestra una ventana para introducir el código PL/SQL asociado al trigger. Dentro del trigger añadiremos únicamente la siguiente línea delante de **return (TRUE):**

```
SRW.MESSAGE(1000,'Este listado muestra agrupado los enfermos de
cada hospital.');
```

Una vez incluida esta línea de código, cerramos la ventana del trigger y volvemos al navegador de objetos dentro de la sección **Disparadores de Informes**, para configurar un nuevo trigger. En este caso vamos a configurar el trigger **AFTER PARAMETER FORM** realizando la misma operación que antes, es decir, pulsando doble clic sobre el nombre del parámetro para que se muestre la ventana de código PL/SQL asociada. Dentro del trigger añadiremos las siguientes líneas delante de **return (TRUE):**

```
IF :P_DESTINO = 'IMPRESORA' THEN :DESTYPE := 'PRINTER';
ELSE :DESTYPE := 'SCREEN';
END IF;
```

A continuación cerramos la ventana de código del trigger y nos situamos de nuevo en el navegador de objetos.

CREANDO UN TRIGGER DE VALIDACIÓN

Nos situamos en la sección **Parámetros de Usuario** y la desplegamos. Ahora vamos a marcar el parámetro **P_DESTINO.** A continuación pulsaremos dentro del menú la opción **Herramientas – Editor PL/SQL**, lo que nos permitirá crear un trigger asociado a un parámetro, y se mostrará la ventana que permite configurar su código. Dentro del trigger añadiremos las siguientes líneas delante de **return (TRUE):**

```
SRW.MESSAGE(1000,'Ha seleccionado '||:P_DESTINO||' como destino
del listado');
```

Cerramos la ventana del código asociado al trigger del parámetro y volvemos al navegador de objetos.

CREANDO UN TRIGGER DE INFORME

Nos situamos en la sección **Disparadores de informes** y hacemos doble clic sobre el trigger **BEFORE REPORT** para que se muestre la ventana de código asociada. Dentro del trigger añadiremos las siguientes líneas delante de **return (TRUE):**

```
SRW.MESSAGE(1000,'Recuerde que los datos seleccionados para
el listado son (orientacion,destino): '||
:ORIENTATION||','||:P_DESTINO);
```

Cerramos la ventana de código del trigger.

COMPROBANDO RESULTADOS

Una vez configurados todos los triggers del informe vamos a comprobar el resultado ejecutando el listado, para ello en primer lugar almacenamos los cambios del listado (**Archivo – Guardar**), seguidamente compilamos el informe (**Programa – Compilar – Compilar Todo)** y a continuación lo ejecutamos (**Programa – Ejecutar Presentación en Papel**). En cuanto comience la ejecución se mostrará el primer mensaje que hemos programado en el trigger BEFORE REPORT:

Al pulsar el botón **Aceptar** se mostrará el siguiente mensaje que hemos programado perteneciente al trigger BEFORE PARAMETER FORM.

Pulsamos el botón **Aceptar** y se mostrará la ventana de parámetros.

Modificamos el parámetro P_DESTINO para que contemple la opción **PANTALLA** en vez de IMPRESORA y pulsamos el botón de ejecución del listado, lo que provocará que se muestre un nuevo mensaje, en este caso correspondiente al trigger de validación del parámetro P_DESTINO.

Pulsamos el botón **Aceptar** y justo cuando comienza la composición del listado se muestra un nuevo mensaje correspondiente al trigger programado BEFORE REPORT.

Pulsamos **Aceptar** y se concluye la ejecución mostrándose el resultado del listado.

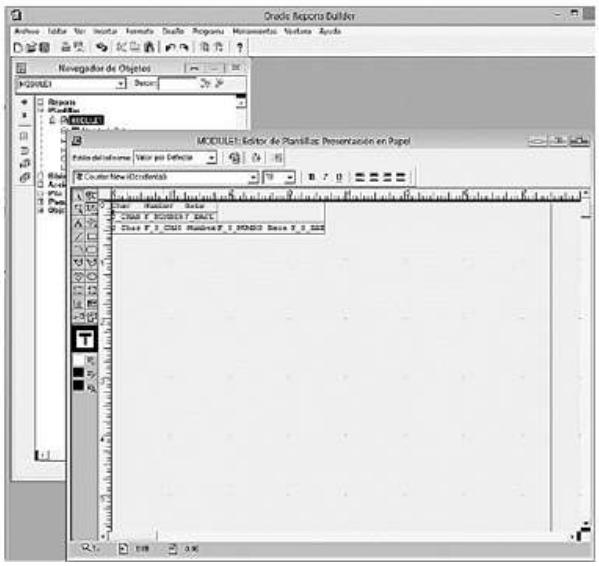
CREAR UNA PLANTILLA PARA REPORT 55

INTRODUCCIÓN

Como hemos podido ver durante los diseños asistidos de los informes, Oracle Report Builder proporciona una serie de plantilla predefinidas para el diseño de los informes (Beige, Blue, etc.). Pero además de estas plantillas predefinidas, podemos también crear una plantilla personalizada que podamos aplicar a los nuevos informes que se creen.

En este capítulo aprenderemos de manera práctica a crear una plantilla para Report.

CREANDO UNA PLANTILLA



En primer lugar abrimos Reports Builder y en la ventana de bienvenida pulsamos el botón **Cancelar** para que se visualice el navegador de objetos.

Una vez mostrado nos situamos en el menú **Archivo** y seleccionamos la opción **Nuevo – Plantilla.** Esto provocará que se muestre automáticamente el **Editor de Plantillas : Presentación en Papel**, como se muestra en esta imagen.

MODIFICANDO UNA PLANTILLA

Ahora vamos a configurar los elementos que deseamos personalizar en la plantilla. En primer lugar vamos a modificar el **Estilo del Informe**, para ello cambiamos el valor de dicha casilla (**Valor por Defecto**) por **Agrupar a la izquierda**.

Esto provoca también una modificación automática del diseño de elementos del informe dentro del Editor, como se muestra en la imagen de la página siguiente.

Seguidamente vamos a editar el margen, para ello pulsamos en el botón 

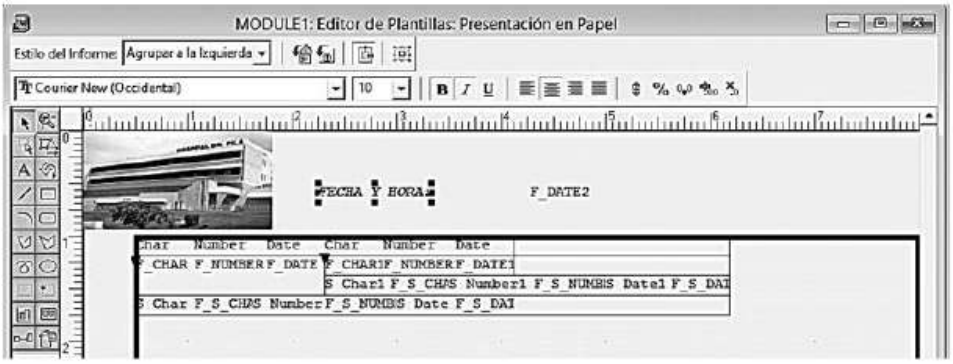
lo que permite mostrar el espacio alrededor del margen del listado. Una vez mostrado dicho margen, vamos a añadir una imagen (como logo del listado) en la parte superior izquierda del margen.

Podemos elegir como imagen la que deseemos. En este caso práctico hemos seleccionado el archivo **logo_hospital.jpg** que se adjunta con los ejemplos del libro. Con independencia de la imagen seleccionada, lo que habrá que conseguir es redimensionarla para que ocupe la zona exterior del margen y no se superponga a los elementos que se incluyen dentro del listado (por dentro de la línea gruesa que corresponde al margen).

Seguidamente vamos a añadir una fecha y hora del listado, pulsando para ello sobre el botón 

Al hacerlo se mostrará una pantalla para seleccionar el formato de fecha que queremos añadir al listado. En nuestro caso práctico seleccionaremos la opción de formato **XX-MES-XX H24:MIN:SG** y mantendremos la opción de ubicación de la misma en la zona **Superior-Centro** como se muestra en esta imagen.

A continuación añadimos un texto por delante de la fecha insertada como se muestra en la imagen siguiente:



Pulsamos de nuevo en botón del Margen para finalizar la edición del mismo y volver a los elementos propios del listado.

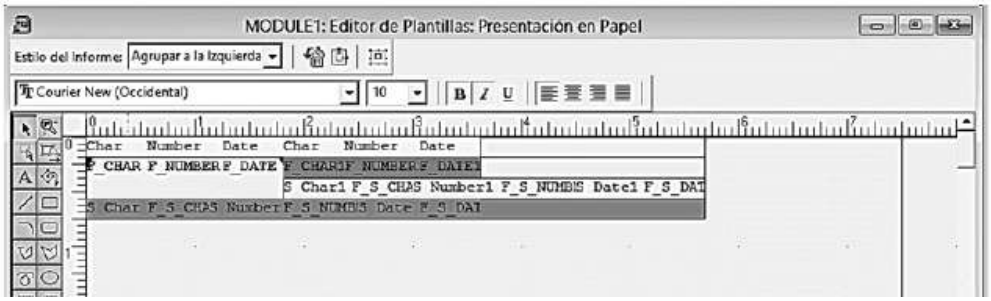
En este punto vamos a configurar los colores de fondo de los distintos elementos a presentar en un listado basados en esta plantilla.

En primer lugar marcamos los elementos F_CHAR, F_NUMBER y F_DATE y los ponemos en negrita.

Seguidamente marcamos los elementos que aparecen justo a la derecha de los anteriores (F_CHAR1, F_NUMBER1 y F_DATE1) y les indicamos como color de fondo **Green**.

Por último seleccionamos los elementos del final (S_CHAR F_S_CHAR, etc.) y les indicamos como color de fondo **Yellow.**

El resultado de estas operaciones será similar al siguiente:



GUARDANDO LA PLANTILLA

Concluidas las modificaciones de la plantilla, es el momento de almacenarla, para ello pulsamos en el menú sobre **Archivo** – **Guardar**, lo que abrirá el administrador de archivos para indicar la ubicación y el nombre con el que se almacenará dicha plantilla. En nuestro caso práctico seleccionaremos como nombre, manteniendo como extensión la que se ofrece por defecto ***.tdf.**

Una vez almacenada podemos cerrarla (**Archivo – Cerrar**).

USO DE PLANTILLAS PERSONALIZADAS

Para asociar una plantilla personalizada a nuestro nuevo informe debemos crear un informe asistido, y llegar hasta la pantalla donde se solicita indicar una plantilla, como se muestra en esta imagen.

En los ejemplos anteriores que hemos realizado en este curso, siempre seleccionábamos una plantilla predefinida. Cuando se disponga de una personalizada, habrá que pulsar sobre la opción **Archivo de Plantilla** y pulsando el botón **Examinar** localizaremos la plantilla personalizada que queremos asociar a nuestro informe, y proseguiremos el diseño del mismo.

GUÍA DE INSTALACIÓN DE ORACLE 11g XE RELEASE 2

INTRODUCCIÓN

Para el mejor aprovechamiento de este curso se aconseja la instalación de la base de datos de Oracle 11g Express Edition, dado que es una versión reducida de libre distribución que consume pocos recursos de máquina y que permite llevar a cabo todas las prácticas y supuestos que se especifican en este libro.

Puede descargar este producto desde el siguiente enlace de la Web de Oracle previo registro gratuito:

http://www.oracle.com/technetwork/products/express-edition/downloads/index.html

A la finalización de la redacción de este libro, la última versión que ofrece Oracle sobre la base de datos Express Edition es la 11g Release 2.

En este anexo podrá encontrar una guía práctica para la instalación de este producto Oracle en su PC.

REQUERIMIENTOS MÍNIMOS

Los requerimientos de máquina para la instalación de Oracle 11g XE Release 2 son los siguientes:

- Sistema operativo Windows 32 bits:
  - Windows 2000 Service Pack 4 o posterior.
  - Windows Server 2003.
  - Windows XP Professional Service Pack 1 o posterior.
  - Compatible con Windows Vista y Windows 7.
- Protocolo TCP/IP:
  - El equipo debe de disponer de una tarjeta de red fija o inalámbrica.
- Espacio en disco necesario:
  - 1,6 GB.
- Memoria RAM:
  - 265 MB mínimo. Se recomienda 512 MB.
- Microsoft Windows Installer (MSI) disponible en el equipo:
  - MSI versión 2.0 o posterior.
  - Si no dispone de este componente, lo puede descargar desde la página web http://msdn.microsoft.com.

TUTORIAL DE INSTALACIÓN

A continuación, se muestran los pasos a seguir para la instalación de Oracle 11g XE Release 2.

Paso 1: Descarga del producto

Descargue Oracle 11g XE Release 2 desde la Web de Oracle.

Paso 2: Configuración del equipo para la instalación

Antes de comenzar la instalación, debe de seguir estos pasos para preparar su equipo para la instalación:

- Cree una carpeta con el nombre *OracleXE11gR2*.
- Extraiga el fichero de instalación *win32_11gR2_OracleXE.zip* en la carpeta *OracleXE11gR2* creada en el punto anterior.

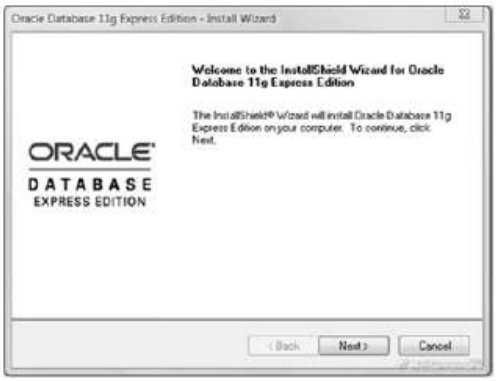
Paso 3: Ejecute la instalación

Para comenzar la instalación debe de ejecutar el fichero *setup.exe* que encontrará en la carpeta *\OracleXE11gR2\DISK1*.

Paso 4: Comienza el proceso de instalación

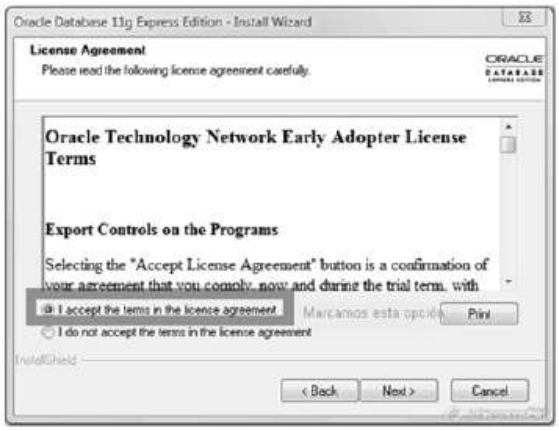
Una vez ejecutado el fichero de instalación se mostrará una pantalla con el progreso de la preparación para la instalación.

Paso 5: Pantalla de bienvenida



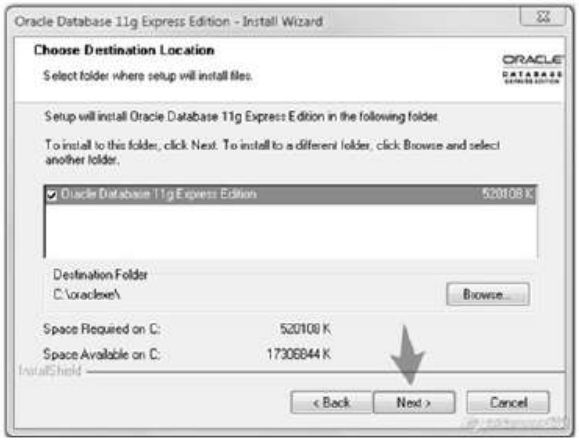
Concluida la preparación para comenzar la instalación se mostrará una pantalla de bienvenida en la que deberá pulsar el botón ***Next*** para comenzar el proceso de instalación de la base de datos en su equipo.

Paso 6: Términos de la licencia



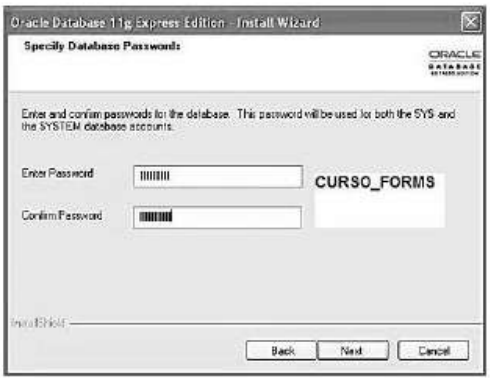
A continuación, se le mostrará los términos de la licencia de uso del producto para su aceptación. Debe marcar la opción *I accept the terms in the license agreement* y posteriormente pulsar el botón ***Next***.

Paso 7: Ubicación para la instalación



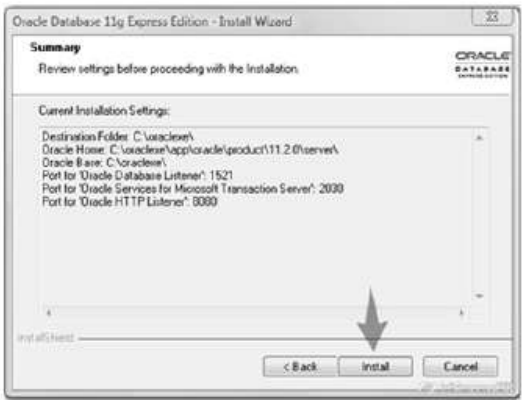
Mantenga la ubicación por defecto de la instalación y pulse el botón ***Next***.

Paso 8: Contraseña para los usuarios SYS y SYSTEM



En este punto de la instalación el sistema nos solicita que introduzcamos una contraseña para el acceso a la base de datos de los usuarios administradores SYS y SYSTEM. Debe introducir la misma contraseña 2 veces. Se recomienda para el seguimiento de este curso que la contraseña que introduzca sea *CURSO_FORMS* y a continuación pulse el botón ***Next.***

Paso 9: Resumen previo a la instalación



Justo antes de comenzar la instalación física de acuerdo a los parámetros introducidos en las pantallas previas, se presenta una pantalla resumen con las opciones que se van a instalar, así como los puertos que se habilitarán después de la instalación, para poder acceder a la base de datos. Pulse el botón ***Install***.

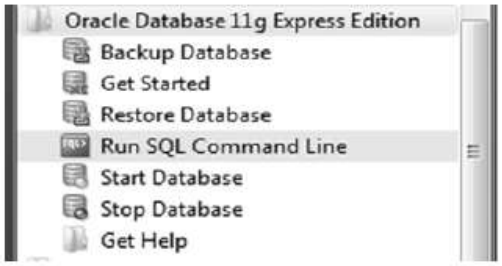
Paso 10: Progreso de la instalación

En esta fase el usuario recibirá en pantalla una serie de imágenes con el progreso de la instalación en su equipo. No debe pulsar ningún botón hasta que concluya la misma.

Paso 11: Finalización de la instalación

La instalación finalizará cuando se presente una pantalla como la que se muestra a continuación, con un botón ***Finish*****,** que deberá pulsar para concluirla.

Paso 12: Familiarizándose con los elementos instalados



Finalizada la instalación se encontrará con el siguiente grupo de elementos instalado en su equipo. E igualmente le aparecerá el siguiente icono en su escritorio.

El funcionamiento básico de estos elementos es el siguiente:

BACKUP DATABASE

Permite realizar un backup o salvaguarda de la base de datos.

GET STARTED

Abre el sistema de gestión administrativa de la base de datos, a través del explorador de su equipo.

RESTORE DATABASE

Permite realizar un restore o recuperación de una salvaguarda que se haya realizado de la base de datos, con la utilidad backup comentada anteriormente.

RUN SQL COMMAND LINE

Permite ejecutar sentencias y scripts contra la base de datos. Es en esta herramienta donde deberá probar la ejecución de las sentencias SQL que se han explicado durante este curso, al igual que la usará para la resolución de los supuestos prácticos que se le han sugerido en el libro.

START DATABASE

Permite arrancar la base de datos para poderse conectar con ella. Normalmente esta utilidad queda configurada en el equipo para que se ejecute automáticamente cuando arranque el ordenador.

STOP DATABASE

Permite parar la base de datos.

ÍNDICE ALFABÉTICO

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W