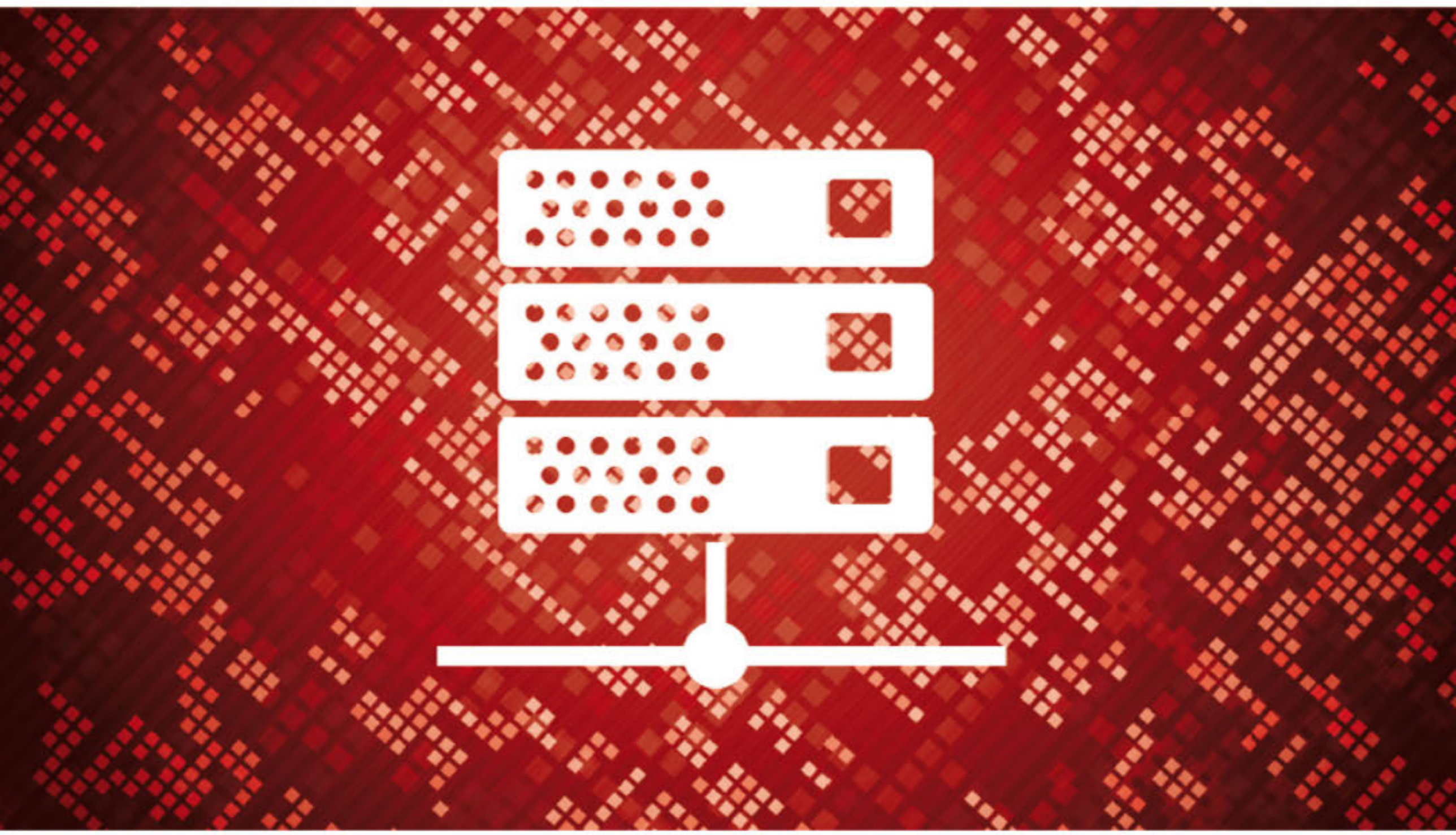
Administración básica de bases de datos con

ORACLE 12c SQL

Prácticas y ejercicios



María Pérez Marqués

Alfaomega

libros R

Administración básica de bases de datos con

ORACLE 12c SQL

Prácticas y ejercicios

Administración básica de bases de datos con ORACLE 12c SQL

Prácticas y ejercicios

María Pérez Marqués

Alfaomega

libros R

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

Datos catalográficos

Pérez, María
Administración básica de bases de datos con
Oracle 12c SQL
Primera Edición
Alfaomega Grupo Editor, S.A. de C.V., México

ISBN: 978-607-622-730-5

Formato: 17 x 23 cm Páginas: 256

Administración básica de bases de datos con Oracle 12c SQL
María Pérez Marqués
ISBN: 978-84-944650-1-7 edición original publicada por RC Libros, Madrid, España.
Derechos reservados © 2016 RC Libros

Primera edición: Alfaomega Grupo Editor, México, agosto 2016

© 2016 Alfaomega Grupo Editor, S.A. de C.V.
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: 978-607-622-730-5

Derechos reservados:
Esta obra es propiedad intelectual de su autor y los derechos de publicación en lengua española han sido legalmente transferidos al editor. Prohibida su reproducción parcial o total por cualquier medio sin permiso por escrito del propietario de los derechos del copyright.

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.

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. – Paraguay 1307 P.B. Of. 11, C.P. 1057, Buenos Aires, Argentina, – Tel./Fax: (54-11) 4811-0887 y 4811 7183 – E-mail: ventas@alfaomegaeditor.com.ar

ÍNDICE

CAPÍTULO 1: INSTALACIÓN E INICIO DE ORACLE 12C 1

PROCESO DE INSTALACIÓN DE ORACLE 12C 1

- CONFIGURAR ACTUALIZACIONES DE SEGURIDAD 2
- ACTUALIZACIONES DE SOFTWARE 3
- OPCIONES DE INSTALACIÓN 5
- OPCIONES DE INSTALACIÓN DE GRID 8
- INSTALAR TIPO DE BASE DE DATOS DE CLUSTER (RAC) 8
- SELECCIONAR TIPO DE INSTALACIÓN 9
- CONFIGURACIÓN DE INSTALACIÓN TÍPICA 10
- CONFIGURACIÓN DE INSTALACIÓN AVANZADA 16
- REALIZAR COMPROBACIONES DE REQUISITOS 36
- RESUMEN 37
- INSTALACIÓN DEL PRODUCTO 39
- TERMINAR 41

INICIO DE ORACLE 12C 41

CAPÍTULO 2: BASES DE DATOS RELACIONALES. DISEÑO BÁSICO. CREACIÓN DE TABLAS E INSERCIÓN DE REGISTROS CON SQL 45

EL MODELO RELACIONAL: CONCEPTOS BÁSICOS 45

EL LENGUAJE SQL 47

- OBJETIVOS ESENCIALES DE SQL 49
- CARACTERÍSTICAS DE SQL 51

DISEÑO DE UNA BASE DE DATOS RELACIONAL 54

- LAS REGLAS DE CODD 55
- INTEGRIDAD 58
- ÍNDICES 62
- UN EJEMPLO DE DISEÑO DE UNA BASE DE DATOS RELACIONAL 67

CREAR TABLAS 76

- AGREGAR DATOS A TABLAS. LA SENTENCIA INSERT 77
- INSERTAR UNA FILA MEDIANTE INSERT INTO...VALUES 78
- INSERTAR VALORES NULL 79
- INSERTAR TODAS LAS COLUMNAS 79
- INSERTAR VARIAS FILAS MEDIANTE INSERT INTO...SELECT 80

CAPÍTULO 3: ELEMENTOS DE ORACLE SQL. FORMATOS, TIPOS DE DATOS, OPERADORES Y FUNCIONES..... 93

UTILIZANDO ORACLE SQL 93

ELEMENTOS BÁSICOS DE ORACLE SQL.....94

FORMATOS EN ORACLE SQL 96

FORMATOS DE ELEMENTOS NUMÉRICOS.....96
FORMATOS DE FECHAS 98

TIPOS DE DATOS EN ORACLE SQL 100

TIPOS DE DATOS ANSI, DB2 Y SQL/DS 102
CARACTERES ASCII Y EBCDIC EN ORACLE SQL 103
CONVERSIÓN DE TIPOS DE DATOS.....104
CONVERSIÓN ENTRE DISTINTOS TIPOS DE DATOS MEDIANTE FUNCIONES SQL.....107
CREACIÓN DE LAS TABLAS DE LA BASE DE DATOS DE EJEMPLO.....107

OPERADORES.....109

OPERADORES ARITMÉTICOS 109
OPERADOR CONCATENACIÓN.....110
OPERADORES DE COMPARACIÓN 110
OPERADORES DE CONJUNTOS 111
OPERADORES LÓGICOS 113
ORDEN DE PRIORIDAD DE LOS OPERADORES 113
FUNCIONES 114
FUNCIONES NUMÉRICAS 116
FUNCIONES CARÁCTER QUE DEVUELVEN VALORES CARÁCTER 117
FUNCIONES CARÁCTER QUE DEVUELVEN VALORES NUMÉRICOS.....118
FUNCIONES DE FECHA.....118
FUNCIONES DE CONVERSIÓN.....118
FUNCIONES AGREGADAS 119

CAPÍTULO 4: LENGUAJE DE CONSULTA DE DATOS. SENTENCIA SELECT 123

INTRODUCCIÓN 123

LA SENTENCIA SELECT 124

EJEMPLOS RELATIVOS A LA SENTENCIA SELECT 129
SELECCIÓN DE TODAS LAS TABLAS DE UNA INSTANCIA DE BASE DE DATOS Y DE TODAS LAS COLUMNAS DE UNA TABLA 132
SELECCIÓN DE FILAS 133
FILAS DUPLICADAS Y LA CLÁUSULA DISTINCT 134
ORDENACIÓN DE LOS RESULTADOS DE UNA CONSULTA 135
COLUMNAS CALCULADAS 138
RENOMBRAR COLUMNAS: CLÁUSULA AS.....139

© Alfaomega - RC Libros

SYNAPS PREMIUM

CONSULTAS CON CONDICIONES DE BÚSQUEDA 140

CONTRASTES DE COMPARACIÓN 141
CONTRASTE DE RANGO: CLÁUSULA BETWEEN...AND 142
CONTRASTE DE PERTENENCIA A UN CONJUNTO: CLÁUSULA IN 143
CONTRASTE DE CORRESPONDENCIA CON PATRÓN: CLÁUSULA LIKE 144
CONTRASTE DE VALOR NULO: CLÁUSULA IS NULL 146
CONTRASTES COMPUESTOS: CLÁUSULAS AND, OR Y NOT 147
COMBINACIÓN DE CONSULTAS: CLÁUSULA UNION 149

CAPÍTULO 5: CONSULTAS MULTITABLA (JOINS) Y CONSULTAS RESUMEN 165

INTRODUCCIÓN 165

JOIN DE DOS TABLAS 166

JOINS CON CRITERIOS DE SELECCIÓN DE FILAS 169
JOINS CON MÚLTIPLES COLUMNAS DE EMPAREJAMIENTO 170

JOINS DE TRES O MÁS TABLAS 171

JOIN CON CONDICIONES DE BÚSQUEDA SIN IGUALDAD 173

JOINS INTERNOS Y EXTERNOS 174

JOIN EXTERNO COMPLETO 175
JOIN EXTERNO IZQUIERDO Y DERECHO 176

INNER JOIN 177

LEFT JOIN, RIGHT JOIN 179

CASOS ESPECIALES EN CONSULTAS MULTITABLA 182

CONSULTAS RESUMEN 185

FUNCIONES DE AGREGADO EN LA LISTA DE SELECCIÓN 185
FUNCIÓN COUNT(*) 188
VALORES NULL EN LAS FUNCIONES DE AGREGADO 190
TRATAMIENTO DE FILAS DUPLICADAS CON DISTINCT 191
AGRUPAR FILAS CON GROUP BY 192

RESUMIR DATOS 198

RESUMIR DATOS CON CUBE 198
UTILIZAR GROUPING PARA DISTINGUIR VALORES NULL 199
RESUMIR DATOS CON ROLLUP 200

CAPÍTULO 6: SUBCONSULTAS Y USO AVANZADO DE FUNCIONES EN ORACLE SQL 207

SUBCONSULTAS 207

REGLAS DE LAS SUBCONSULTAS 211
SUBCONSULTAS CON IN 212

SYNAPS PREMIUM

SUBCONSULTAS CON OPERADORES DE COMPARACIÓN 213
SUBCONSULTAS CON EXISTS 214
SUBCONSULTAS CON ANY, SOME O ALL 216
SUBCONSULTAS USADAS EN CUALQUIER LUGAR DE UNA EXPRESIÓN 217
SUBCONSULTAS CON MÚLTIPLES NIVELES DE ANIDAMIENTO 218

SUBCONSULTAS CORRELACIONADAS 219

SUBCONSULTAS CORRELACIONADAS CON ALIAS 219
SUBCONSULTAS CORRELACIONADAS CON OPERADORES DE COMPARACIÓN 220
SUBCONSULTAS CORRELACIONADAS CON UNA CLÁUSULA HAVING 221
UNION, INTERSECT Y MINUS EN SUBCONSULTAS 221

USO AVANZADO DE FUNCIONES Y VARIABLES 223

FUNCIONES DE CADENA 223
FUNCIONES NUMÉRICAS 226
FUNCIONES DE FECHAS 231

FUNCIONES DE CONVERSIÓN Y TRANSFORMACIÓN 235

CONVERSIONES AUTOMÁTICAS 236
FUNCIONES TRANSLATE Y DECODE 237

ÍNDICE ANALÍTICO 243

© Alfaomega - RC Libros

SYNAPS PREMIUM

CAPÍTULO 1

INSTALACIÓN E INICIO DE ORACLE 12c

PROCESO DE INSTALACIÓN DE ORACLE 12C

Para instalar Oracle 12c se introduce el DVD del programa en la unidad correspondiente y, en caso de que la instalación no se inicie automáticamente, se hace clic en *setup*. Se obtiene la pantalla temporal de recopilación de datos del sistema de la Figura 1-1.

Figura 1-1

SYNAPS PREMIUM

Configurar actualizaciones de seguridad

A continuación se obtiene la pantalla de *Configurar Actualizaciones de Seguridad* de la Figura 1-2.

Figura 1-2

Para proporcionar un contacto de seguridad, seleccione una de las siguientes opciones:

- Proporcione una dirección de correo electrónico para recibir la información de seguridad para la instalación.
- Proporcione una dirección de correo electrónico de My Oracle Support o nombre de cuenta para recibir la información de seguridad de la instalación y para inscribir su sistema para las actualizaciones de seguridad. Puede recibir información sobre las alertas a través de My Oracle Support.

Puede optar por no proporcionar esta información, pero Oracle recomienda que configure un contacto para notificaciones de seguridad.

La información recogida por las actualizaciones de seguridad se limita a la información de configuración. Los datos recopilados no incluyen información identificable personalmente (con la excepción de un nombre de contacto local en el caso de que se produzcan problemas de transmisión). Podrá seguir utilizando todas las funciones con licencia de Oracle si decide no activar las actualizaciones de seguridad. Para no recibir notificaciones de seguridad, deje en blanco todos los campos de esta pantalla y haga clic en *Siguiente* para continuar.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Si proporciona las credenciales de My Oracle Support, la función de actualizaciones de seguridad recopilará automáticamente la información de configuración de los productos Oracle instalados y la carga en los sistemas de soporte de Oracle. Puede acceder a la información recopilada a través de la cuenta de My Oracle Support y revisar las recomendaciones de comprobación de estado y las recomendaciones de parches, así como otras recomendaciones para su sistema, además de las alertas de seguridad.

Si proporciona la dirección de correo electrónico de My Oracle Support, deberá seleccionar *Deseo recibir actualizaciones de seguridad a través de My Oracle Support* para introducir su contraseña de My Oracle Support.

Actualizaciones de software

Al hacer clic en *Siguiente*, se obtiene la pantalla de la Figura 1-3 relativa a *Descargar Actualizaciones de Software*.

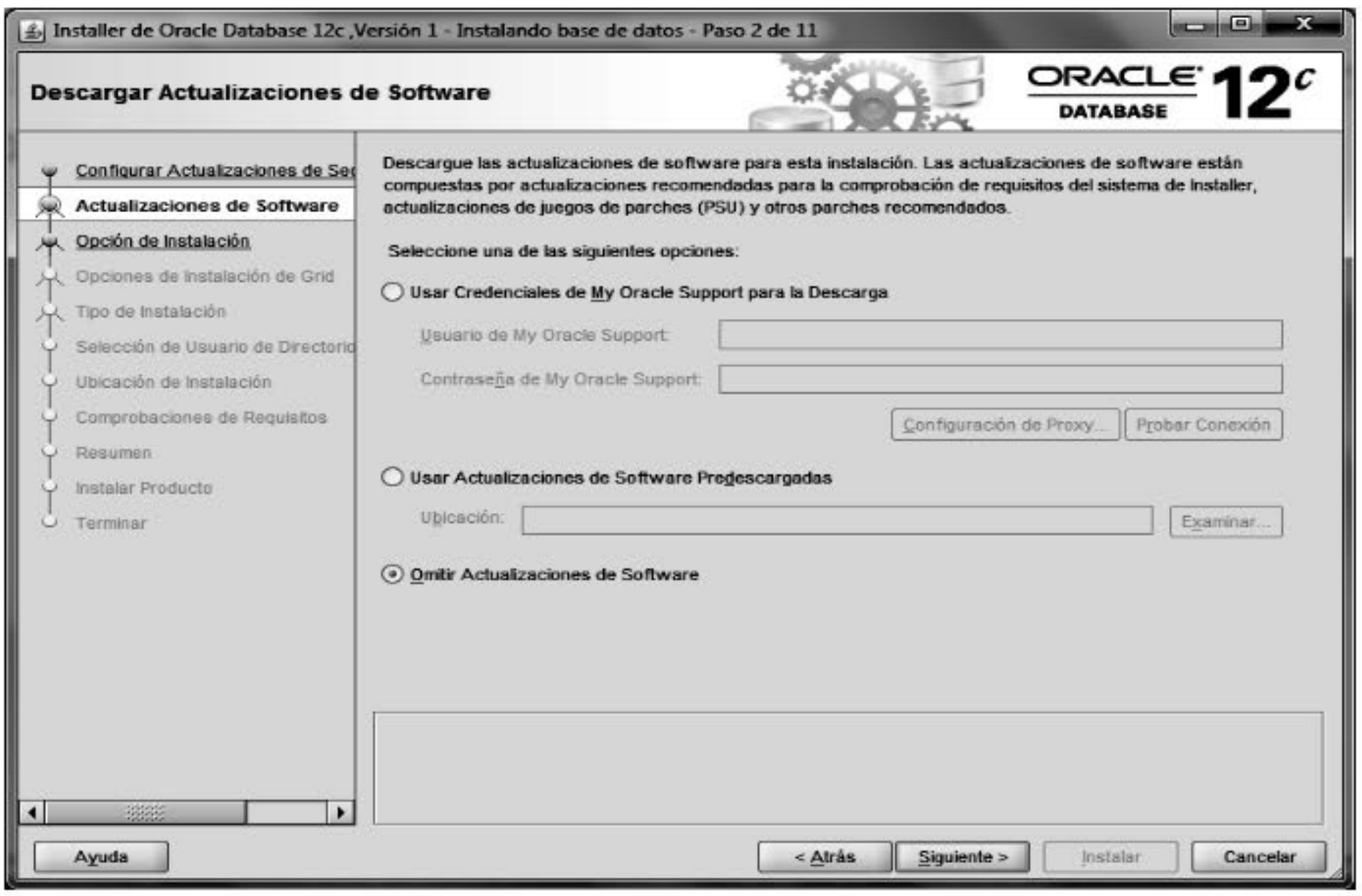


Figura 1-3

Proporcione la información necesaria para permitir el acceso a My Oracle Support y activar las conexiones de proxy. Proporcione esta información para activar las actualizaciones de software durante esta sesión de instalación.

En cuanto a *Usar Credenciales de My Oracle Support para la Descarga*, proporcione su nombre de usuario y contraseña de My Oracle Support o el nombre

SYNAPS PREMIUM

de usuario y la contraseña de la persona cuyas credenciales de Oracle Support desee utilizar con esta instalación.

Las credenciales de la cuenta de usuario que introduzca deben ser de una cuenta con privilegios de descarga de My Oracle Support.

En cuanto a *Configuración de Proxy y Prueba de Conexión*, para configurar el proxy que Oracle Universal Installer utilizará para conectarse a Internet, haga clic en *Configuración de Proxy* y proporcione la información del servidor de proxy necesaria para su sitio. Por ejemplo:

Servidor proxy: www-proxy.ejemplo.com

Número de puerto del proxy: 80

Usuario del proxy: Scott

Contraseña del proxy: Tiger112

Dominio de Proxy: dominio.ejemplo.com

La información de dominio de proxy es sensible a mayúsculas/minúsculas. Si no dispone de dominio de proxy, no tendrá que rellenar los campos *Usuario de Proxy*, *Contraseña de Proxy* ni *Dominio de Proxy*.

Haga clic en *Aceptar* para introducir la información de valores del proxy.

Haga clic en *Probar Conexión* para comprobar que los valores de configuración del proxy introducidos son correctos y que Installer puede descargar actualizaciones. Si se supera la prueba del proxy, aparece una ventana con un mensaje de confirmación que certifica que se ha establecido la conexión de prueba con My Oracle Support. Si no se supera la prueba del proxy, aparece una ventana con un mensaje que indica que el acceso a My Oracle Support está bloqueado debido a un fallo en la solicitud de acceso al proxy.

Verifique que ha introducido la información de usuario, contraseña y dominio de proxy correcta, o bien, póngase en contacto con el administrador de red para determinar la causa del fallo de la solicitud de acceso al proxy.

© Alfaomega - RC Libros

SYNAPS PREMIUM

En cuanto a *Usar Actualizaciones de Software Predescargadas*, seleccione esta opción si ya ha descargado las actualizaciones de software mediante el comando de Oracle Universal *Installer ./runInstaller -downloadUpdates* (en Linux y UNIX) o *setup.exe -downloadUpdates* (en Windows). Por ejemplo, puede iniciar una instalación desde un servidor que tenga acceso a un gateway público, descargar las actualizaciones de software a un dispositivo de almacenamiento portátil en dicho servidor y transferir las actualizaciones de software a una ubicación de servidor seguro sin acceso a gateway.

También puede iniciar una instalación, pararla para solucionar un fallo en un requisito de instalación e iniciar una nueva instalación utilizando las actualizaciones de software descargadas en la instalación anterior. La opción de predescarga funcionará únicamente si ha utilizado Oracle Universal Installer para descargar las actualizaciones, ya sea durante la instalación o antes de esta mediante la opción *-downloadUpdates*. No se pueden descargar las actualizaciones manualmente y, a continuación, suministrar la ruta de acceso de descarga manual al Installer.

Seleccione *Omitir Actualizaciones de Software* para ignorar las actualizaciones de software disponibles y continuar con la instalación.

Opciones de instalación

Al hacer clic en *Siguiente* en la Figura 1-3 se obtiene la pantalla de *Seleccionar Opción de instalación* (Figura 1-4). Seleccione cualquiera de las siguientes opciones de instalación y haga clic en *Siguiente*.

Crear y Configurar Base de Datos: seleccione esta opción para crear una base de datos nueva junto con esquemas de ejemplo.

Instalar solo Software de Base de Datos: seleccione esta opción para instalar en exclusiva los binarios de la base de datos. Para configurar la base de datos, debe ejecutar el Asistente de Configuración de Bases de Datos después de instalar el software.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Actualizar Base de Datos Existente: seleccione esta opción para actualizar una base de datos existente. Esta opción instala los binarios del software en un nuevo directorio raíz de Oracle. Al final de la instalación, puede actualizar la base de datos existente.

Figura 1-4

Si elegimos *Crear y configurar base de datos* obtenemos la pantalla de *Clase de sistema* de la Figura 1-5 en la que se pueden seleccionar las siguientes opciones:

Escritorio: seleccione esta opción si realiza la instalación en una computadora portátil o un sistema de escritorio. Esta opción incluye una base de datos inicial y permite una configuración mínima. Asimismo es idónea para quienes desean tener en marcha y ejecución la base de datos con gran rapidez.

Clase de Servidor: seleccione esta opción si realiza la instalación en un sistema de clase de servidor, como el que utilizaría para el despliegue de Oracle en un centro de datos de producción. Esta opción permite opciones de configuración más avanzadas. Entre estas están Oracle RAC, la gestión automática de almacenamiento, la configuración de copia de seguridad y recuperación, la integración con Enterprise Manager Cloud Control y el ajuste de memoria más preciso.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Figura 1-5

Si se elige *Clase Escritorio* en la Figura 1-5, aparece la pantalla de *Selección de Usuario de Directorio* (Figura 1-6) en la que se elige un usuario existente de Windows con su contraseña, se crea un usuario de Windows o se usa una cuenta incorporada de Windows. A continuación se especificará la *Ubicación de instalación*.

Figura 1-6

SYNAPS PREMIUM

Opciones de Instalación de Grid

Si se elige *Clase Servidor* en la Figura 1-5, será necesario seleccionar el tipo de instalación de base de datos que se desea realizar y hacer clic en *Siguiente*. Se elegirá uno de los siguientes tipos:

Instalación de Base de Datos de Instancia Única: esta opción instala la base de datos y el listener.

Instalación de Base de Datos de Oracle Real Application Clusters: esta opción instala Oracle Real ApplicationClusters.

Instalación de Base de Datos Oracle RAC OneNode: esta opción instala la base de datos Oracle RAC OneNode.

Instalar Tipo de Base de Datos de Cluster (RAC)

Seleccione el tipo de gestión de recursos de base de datos Oracle RAC que desea configurar:

Gestionada por Administrador: una base de datos gestionada por administrador es una base de datos creada en nodos que no forman parte de un pool de servidores, sino que se gestionan de forma manual por el administrador de la base de datos o clusterware. Los administradores gestionan la capacidad del servidor y la asignación de recursos. Este es el formato tradicional para las bases de datos Oracle.

Gestionada por Políticas: una base de datos gestionada por políticas se crea con un pool de servidores.

Los pools de servidores dividen el cluster en grupos de servidores que alojan los mismos recursos o similares. Distribuyen una carga de trabajo uniforme (un juego de recursos de Oracle Clusterware) entre varios servidores del cluster.

La gestión de la base de datos se define por la configuración de los pools de servidores para gestionar los recursos. Estas opciones de configuración incluyen el uso de cardinalidad para un pool de servidores para indicar en qué servidores puede ejecutarse la base de datos y el uso de atributos del pool de servidores para indicar el número de instancias de la base de datos necesario para soportar la carga de trabajo esperada. Oracle Clusterware asigna o reasigna capacidad entre los servidores disponibles según las políticas definidas por el usuario para los pools de servidores, lo que permite failover más rápido de los recursos y la asignación de capacidad dinámica.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Durante la instalación, se crea un pool de servidores y todas las instancias de la base de datos Oracle RAC son miembros de este pool de servidores. Después de la instalación, puede modificar este pool de servidores y crear nuevos pools de servidores de base de datos para gestionar los servicios de base de datos.

Si selecciona *Gestionada por Administrador*, haga clic en *Siguiente*.
Si selecciona *Gestionada por Políticas*, proporcione la siguiente información:

Nombre de Pool de Servidores: proporcione el nombre del pool de servidores inicial creado durante la instalación.

Cardinalidad: indica el número de servidores del cluster en el que se pueden ejecutar la base de datos u otros recursos gestionados. Por defecto, la cardinalidad se define en todos los nodos del cluster. Oracle recomienda aceptar el valor por defecto y realizar los cambios después de la instalación.

Después de proporcionar la información solicitada para el pool de servidores inicial, haga clic en *Siguiente*.

Seleccionar Lista de Nodos

Haga clic en la casilla de los nodos candidatos del cluster en el que desea instalar Oracle Real Application Clusters (Oracle RAC) u Oracle RAC One. Con un cluster gestionado por el administrador, Oracle RAC o RAC One solo se ejecuta en los nodos seleccionados.

Haga clic en *Seleccionar Todo* para seleccionar todos los nodos candidatos. Haga clic en *Anular la Selección de Todo* para anular la selección de todos los nodos del cluster en la lista de candidatos.

Haga clic en *Conectividad SSH* para comprobar que hay claves SSH disponibles y configuradas de manera que Installer pueda copiar los archivos.

Después de seleccionar los nodos y de comprobar que la conectividad SSH está configurada, haga clic en *Siguiente*.

Seleccionar Tipo de Instalación

Seleccione el tipo de instalación que desea realizar y haga clic en *Siguiente*. Se puede elegir entre:

SYNAPS PREMIUM

Instalación Típica: es el método de instalación seleccionado por defecto. Permite instalar rápidamente Oracle Database con una intervención mínima del usuario. Instala el software y, de manera opcional, crea una base de datos de uso general con la información especificada en este cuadro de diálogo.

Instalación Avanzada: seleccione este método si desea realizar instalaciones más complejas, por ejemplo, si desea crear contraseñas individuales para las diferentes cuentas, crear tipos específicos de bases de datos iniciales (para el procesamiento de transacciones o para sistemas de almacenes de datos, por ejemplo), utilizar distintos grupos de idiomas, especificar notificaciones de correo electrónico, etc.

Configuración de Instalación Típica

Proporcione información de configuración para la instalación típica de Oracle Database y haga clic en *Siguiente*. Los campos de configuración son los siguientes (Figura 1-7):

Directorio Base de Oracle

Ubicación del Software

Tipo de Almacenamiento (Clase de Servidor, Solo para Instalación Típica)

Ubicación de Archivo de Base de Datos

Contraseña de ASMSNMP (Clase de Servidor, Solo para Instalación Típica)

Edición de Base de Datos

- Enterprise Edition
- Standard Edition
- Standard EditionOne (Solo para Instalaciones de Escritorio e Instancia Única)
- Personal Edition (Solo para Sistemas Operativos Microsoft Windows)

Juego de Caracteres (Clase de Escritorio, Solo para Instalación Típica)

Grupo OSDBA

Nombre de la Base de Datos Global

Contraseña Administrativa (SYS)

© Alfaomega - RC Libros

SYNAPS PREMIUM

Confirmación de Contraseña

Nombre de Servicio (con Oracle RAC OneNode)

Asegúrese de que la ruta de acceso del directorio raíz de Oracle para el directorio raíz de la base de datos y la ruta de acceso del directorio base de Oracle utilizan solo caracteres ASCII. En esta versión, el uso de caracteres no ASCII para un directorio raíz de Oracle Database o un directorio base de Oracle no está soportado.

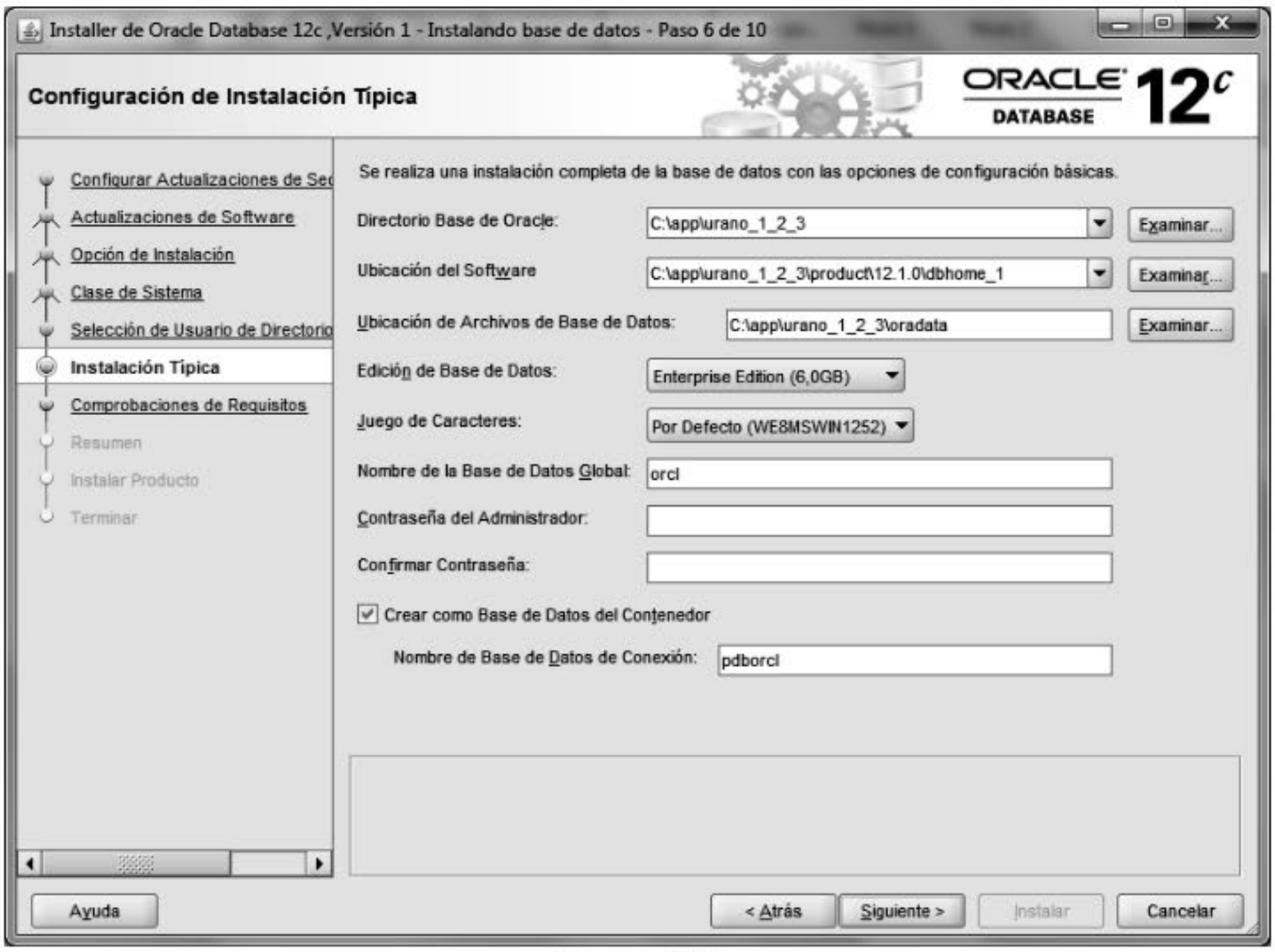


Figura 1-7

Directorio Base de Oracle

El directorio base de Oracle es el directorio de nivel superior para las instalaciones de software de Oracle. La ruta de acceso al directorio base de Oracle aparece por defecto. Puede cambiar la ruta de acceso según sus requisitos.

Si ha creado una ruta de acceso para la base de Oracle de acuerdo con las reglas de la Arquitectura Flexible Óptima para entornos de software de Oracle bien estructurados, OUI incluye esta ruta de acceso como ruta de acceso por defecto del

SYNAPS PREMIUM

directorio base de Oracle. Para que OUI reconozca la ruta de acceso como una ruta de acceso de software de Oracle, debe tener el formato u0[1-9]/app y debe poder escribirla cualquier miembro del grupo oraInventory (normalmente oinstall).

Los miembros de este grupo tienen permiso para modificar el archivo oraInventory, que es el inventario central de todas las instalaciones de software de Oracle. Oracle recomienda crear manualmente una ruta de acceso al directorio base de Oracle. La ruta de acceso de la Arquitectura Flexible Óptima del directorio base de Oracle es /u01/app/usuario, donde usuario es el nombre de la cuenta de usuario que desea que sea propietaria del software de Oracle Database.

Si es necesario, haga clic en *Examinar* para seleccionar otra ruta de acceso al directorio base de Oracle.

Ubicación del Software

Es la ruta de acceso al directorio raíz de Oracle, en el que se colocan los binarios de Oracle Database para esta instalación. Si la ruta de acceso al directorio base de Oracle es una ruta de acceso de Arquitectura Flexible Óptima estándar, la ruta de acceso al directorio raíz de Oracle se rellenará con la ruta de acceso por defecto. Si la ruta de acceso al directorio base de Oracle es */u01/app/oracle,* OUI crea por defecto la siguiente ruta de acceso al directorio raíz de Oracle: */u01/app/oracle/product/12.1.0/dbhome*_1

En este campo, acepte los valores por defecto o introduzca la ruta de acceso del directorio raíz de Oracle en el que desea instalar los componentes de Oracle. La ruta de acceso al directorio no puede contener espacios.

Si es necesario, haga clic en *Examinar* para seleccionar otra ruta de acceso al directorio base de Oracle.

Tipo de Almacenamiento (Clase de Servidor, Solo para Instalación Típica)

Los archivos de Oracle Database se pueden almacenar en un sistema de archivos o en grupos de discos de Gestión Automática de Almacenamiento. No se pueden colocar directamente en dispositivos raw ni dispositivos de bloque.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Ubicación de Archivo de Base de Datos

La ubicación de archivos de base de datos es el lugar en el que se almacenan los archivos de Oracle Database. La ubicación de los archivos de datos por defecto es $ORACLE_BASE/oradata para las instalaciones de Real ApplicationCluster (Oracle RAC) de un único nodo, de una instancia única y de clase de escritorio. Para las instalaciones de Oracle RAC de varios nodos, la ubicación por defecto se encuentra bajo cualquier punto de montaje compartido detectado en el juego de nodos seleccionado.

Si es necesario, haga clic en *Examinar* para seleccionar otra ubicación para los archivos de la base de datos.

Contraseña de ASMSNMP (Clase de Servidor, Solo para Instalación Típica)

Este campo se activa al seleccionar grupos de discos de Gestión Automática de Almacenamiento como tipo de almacenamiento. Es la contraseña que necesita la instancia existente de Gestión Automática de Almacenamiento para configurar la base de datos en ASM.

Edición de Base de Datos

Es el tipo de base de datos que desea instalar. Podemos elegir entre:

Enterprise Edition: este tipo de instalación está diseñado para aplicaciones a nivel de empresa. Está diseñado para el Procesamiento de Transacciones en Línea (OLTP) de alta seguridad y de importancia crítica y para entornos de almacenes de datos. Si selecciona este tipo de instalación, se instalan todas las opciones de Enterprise Edition con licencias independientes.

Standard Edition: este tipo de instalación está diseñado para aplicaciones a nivel de departamento o grupo de trabajo y para pequeñas y medianas empresas. Está diseñado para proporcionar las opciones y servicios de gestión de bases de datos relacionales esenciales. Instala un juego integrado de herramientas de gestión, distribución completa, replicación y funciones web para crear aplicaciones críticas.

Standard EditionOne (Solo para Instalaciones de Escritorio e Instancia Única): este tipo de instalación está diseñado para aplicaciones a nivel de departamento o grupo de trabajo, o para aplicaciones web. Desde entornos de un único servidor para pequeñas empresas hasta entornos de sucursales distribuidas, Oracle Database Standard EditionOne incluye todas las utilidades necesarias para crear aplicaciones críticas.

SYNAPS PREMIUM

Personal Edition (Solo para Sistemas Operativos Microsoft Windows): este tipo de instalación instala el mismo software que el tipo de instalación Enterprise Edition, con excepción de los paquetes de gestión. Sin embargo, solo soporta un entorno de desarrollo y despliegue monousuario que debe ser totalmente compatible con Enterprise Edition y Standard Edition. Oracle RAC no se instala con Personal Edition.

Juego de Caracteres (Clase de Escritorio, Solo para Instalación Típica)

Esta opción permite almacenar los datos de caracteres de la base de datos con uno de los siguientes métodos:

Usar Valor por Defecto: esta opción utiliza los valores del idioma del sistema operativo.

Usar Unicode: esta opción permite almacenar varios grupos de idiomas.

Grupo OSDBA

El grupo OSDBA es el grupo del sistema operativo a cuyos miembros se les otorga el privilegio SYSDBA para la base de datos mediante la autenticación del sistema operativo. El nombre que se utiliza para este grupo en los ejemplos de código de Oracle es dba. El administrador del sistema debe haber creado grupos y usuarios que pertenezcan a los grupos adecuados antes de iniciar la instalación.

Nombre de la Base de Datos Global

Es el nombre que se da a una base de datos para identificarla de forma única con respecto al resto de bases de datos de una red. El nombre de la base de datos global se compone de un nombre único de base de datos (nombre_único_bd), un delimitador (punto) y un nombre único de dominio de base de datos (dominio_bd). Se representa como: nombre_único_bd.dominio_bd.

El nombre único de la base de datos es el nombre de la base de datos. Puede contener un máximo de 30 caracteres, siempre que los primeros ocho caracteres sean únicos (alfanuméricos, carácter de subrayado [_], dólar [$] y almohadilla numérica [#]). El nombre de dominio de base de datos es el entorno de computación utilizado para la base de datos. No puede contener más de 128 caracteres (alfanuméricos, carácter de subrayado [_] y almohadilla numérica [#]), incluidos todos los puntos. Asegúrese de que la combinación de nombre de base de datos (primeros ocho caracteres únicos del nombre único de la base de datos), delimitador y nombre de dominio de la base de

© Alfaomega - RC Libros

SYNAPS PREMIUM

datos no supera los 128 caracteres. Por ejemplo: *ventas.us.example.com*. El nombre_único_bd es *ventas* y el dominio_bd es es *ejemplo.com*.

Contraseña Administrativa (SYS)

La contraseña administrativa es la contraseña para el privilegio SYS de la base de datos.

La instalación no podrá continuar si no se cumplen los siguientes requisitos para la contraseña:

- La contraseña no puede superar los 30 caracteres.
- No se aceptan contraseñas vacías.
- La contraseña no podrá ser el nombre de usuario.
- La contraseña de la cuenta SYS no podrá ser change_on_install. (no sensible a mayúsculas/minúsculas)

Cuando se le solicite introducir una contraseña, Oracle recomienda que la contraseña que especifique siga estas directrices:

- Contenga al menos una letra en minúscula.
- Contenga al menos una letra en mayúscula.
- Contenga al menos un dígito.
- Tenga como mínimo 8 caracteres de longitud.
- Utilice un juego de caracteres de base de datos que incluya el guion bajo (_), símbolo de dólar ($) y símbolo de almohadilla (#).
- Si contiene caracteres especiales, también en el caso en el que la contraseña empiece por un número o por un símbolo, incluya la contraseña entre comillas dobles.
- No debe ser una palabra real.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Será necesario volver a introducir la misma contraseña administrativa para el privilegio de base de datos SYS en el campo *Confirmar Contraseña*.

Nombre de Servicio (con Oracle RAC OneNode)

Asegúrese de que proporciona un nombre de servicio para la base de datos Oracle RAC OneNode. Los nombres de servicio permiten a los clientes conectar a un servicio en lugar de conectar a una instancia concreta de Oracle Database. Los nombres de servicio permiten a los clientes estar conectados constantemente incluso si la base de datos Oracle RAC OneNode sufre un failover o si se reubica en otro nodo.

Crear una base de datos como Base de Datos de Contenedor

Seleccione la opción *Crear como Base de Datos de Contenedor* para crear la base de datos como una base de datos de contenedor (CDB) que puede soportar una o varias bases de datos de conexión (PDB). Si desea que Oracle Universal Installer cree una PDB en el momento de crear la CDB, especifique el nombre de PDB en el campo Nombre de Base de Datos de Conexión.

Configuración de Instalación Avanzada

Si en *Seleccionar tipo de instalación* se elige *Instalación Avanzada*, será necesario especificar características adicionales de idioma del producto, edición de base de datos, ubicación de instalación, inventario, tipo y opciones de configuración, identificador de la base de datos, opciones de gestión y almacenamiento de la base de datos, grupos de discos ASM, opciones de recuperación, contraseñas de esquemas de bases de datos, grupos de sistema operativo con privilegios y otras características.

Seleccionar Idiomas del Producto

Seleccione los idiomas en los que desea ejecutar el producto. Con esta selección, se cambia el idioma de los productos que se van a instalar, pero no el idioma de la instalación.

Seleccione el elemento adecuado en la lista *Idiomas Disponibles*, transfiéralo a la lista *Idiomas Seleccionados* y haga clic en *Siguiente*. Puede seleccionar más de un idioma. Si desea desplazar el idioma seleccionado a la lista *Idiomas Seleccionados* de uno en uno, haga clic en el botón ">"; de lo contrario, haga clic en el botón ">>" para desplazarlos todos a la vez. Están seleccionados por defecto el inglés y el idioma de la configuración regional de la sesión del sistema operativo.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Seleccionar Edición de Base de Datos

Seleccione la edición de la base de datos que desea instalar y haga clic en *Siguiente*. Se puede elegir entre:

Enterprise Edition: este tipo de instalación está diseñado para aplicaciones a nivel de empresa. Está diseñado para el Procesamiento de Transacciones en Línea (OLTP) de alta seguridad y de importancia crítica y para entornos de almacenes de datos. Si selecciona este tipo de instalación, se instalarán todas las opciones de Enterprise Edition con licencias independientes.

Standard Edition: este tipo de instalación está diseñado para aplicaciones a nivel de departamento o grupo de trabajo y para pequeñas y medianas empresas. Está diseñado para proporcionar las opciones y servicios de gestión de bases de datos relacionales esenciales. Instala un juego integrado de herramientas de gestión, distribución completa, replicación y funciones web para crear aplicaciones críticas.

Standard EditionOne (Solo Instalaciones de Escritorio e Instancia Única): este tipo de instalación está diseñado para aplicaciones a nivel de departamento o grupo de trabajo o para aplicaciones web. Desde entornos de un único servidor para pequeñas empresas hasta entornos de sucursales distribuidas, Oracle Database Standard EditionOne incluye todas las utilidades necesarias para crear aplicaciones críticas.

Personal Edition (Solo para Sistemas Operativos Microsoft Windows): este tipo de instalación instala el mismo software que el tipo de instalación Enterprise Edition, con excepción de los paquetes de gestión. Sin embargo, solo soporta un entorno de desarrollo y despliegue monousuario que debe ser totalmente compatible con Enterprise Edition y Standard Edition. Oracle RAC no se instala con Personal Edition.

Especificar Ubicación de Instalación

Especifique el directorio base de Oracle y la ubicación del software y, a continuación, haga clic en *Siguiente*. Asegúrese de que la ruta de acceso del directorio raíz de Oracle para el directorio raíz de la base de datos y la ruta de acceso del directorio base de Oracle utilizan solo caracteres ASCII. En esta versión, el uso de caracteres no ASCII para un directorio raíz de Oracle Database o un directorio base de Oracle no está soportado.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Directorio Base de Oracle: el directorio base de Oracle es un directorio de nivel superior para instalaciones de software de Oracle en una cuenta de propietario de instalaciones de Oracle.

En sistemas basados en UNIX: la ruta de acceso del directorio base de Oracle por defecto es punto de montaje/*app/usuario*, donde usuario será la cuenta de usuario que ejecuta la instalación.

En sistemas Microsoft Windows: la ruta de acceso del directorio base de Oracle por defecto es DRIVE_LETTER*:\app\user*. Puede cambiar esta ruta en función de sus necesidades.

Ubicación del Software: en este campo, acepte los valores por defecto o introduzca la ruta de acceso del directorio raíz de Oracle en el que desea instalar el software de Oracle. La ruta de acceso al directorio no puede contener espacios.

Crear Inventario

Especifique la siguiente información y haga clic en *Siguiente*.

- La ruta de acceso completa del directorio en el que Installer debe colocar la información sobre el inventario (directorio de inventario de Oracle)

- El nombre del grupo del sistema operativo que será propietario del directorio de inventario de Oracle (grupo de inventario de Oracle).

Nota: Asegúrese de que la ruta de acceso del directorio raíz de inventario de Oracle solo contiene caracteres ASCII. En el momento del lanzamiento de este producto, no está soportado el uso de caracteres que no sean ASCII para un inventario de Oracle.

Directorio de Inventario de Oracle

El directorio de inventario de Oracle (oraInventory) estará en la ubicación de inventario central de todo el software de Oracle instalado en un servidor. Especifique la ruta de acceso completa del directorio de inventario de Oracle (también denominado directorio oraInventory). Installer utiliza este directorio para almacenar información sobre todos los productos Oracle instalados en este sistema.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Nota: Installer creará el directorio oraInventory que se especifica solo si no existe. Asegúrese de que no crea el directorio de inventario de Oracle en el directorio base o en el directorio raíz de Oracle.

Utilice las siguientes instrucciones de Arquitectura Flexible Óptima (OFA) al seleccionar el directorio:

La primera vez que instala software de Oracle en una computadora, Oracle Universal Installer crea un directorio oraInventory con el formato u[01-99]/app como, por ejemplo, /u01/app. Si el usuario que ejecuta la instalación tiene permisos de escritura en el archivo oraInventory, Installer crea un directorio de inventario de Oracle en la ruta /u[01-99]/app/oraInventory. Por ejemplo*: /u01/app/oraInventory*

Por defecto, el directorio de inventario de Oracle no se instala en el directorio base de Oracle. Esto se debe a que todas las instalaciones de software de Oracle comparten un inventario de Oracle común, de manera que solo hay un inventario de Oracle para todos los usuarios, mientras que cada usuario que realiza una instalación de software utiliza un directorio base de Oracle.

Si proporciona la ruta de acceso para el directorio base de Oracle cuando se le solicita durante la instalación o define la variable de entorno ORACLE_BASE para el usuario que realiza la instalación, OUI sugerirá la siguiente ruta de acceso por defecto:

ORACLE_BASE/../oraInventory

Si no ha introducido ninguna ruta de acceso ni ha definido ORACLE_BASE, el directorio de inventario de Oracle estará en el directorio raíz del usuario que realiza la instalación. Por ejemplo: *HOME/oraInventory*

En este ejemplo, HOME es el directorio raíz de conexión del usuario que ha iniciado Installer. A menos que instale el software en un subdirectorio del directorio raíz, Oracle recomienda que cambie este valor para especificar el valor recomendado anteriormente.

Grupo de Inventario de Oracle

Seleccione el nombre del grupo del sistema operativo que será propietario del directorio de inventario de Oracle. Todos los miembros de este grupo tienen acceso de escritura en el directorio de inventario de Oracle (oraInventory), que es el registro de inventario central de todas las instalaciones de software de Oracle en un servidor.

SYNAPS PREMIUM

Por defecto, este grupo se denomina oinstall. El grupo de inventario de Oracle debe ser el grupo primario para los propietarios de instalaciones de software de Oracle. Los usuarios de UNIX que pertenecen a este grupo son los únicos que pueden actualizar, instalar o eliminar el software de Oracle mediante Oracle Universal Installer.

Seleccionar Tipo de Configuración

Seleccione el tipo de base de datos que desea crear durante la instalación y haga clic en *Siguiente.* Puede seleccionar una de las siguientes configuraciones de base de datos:

Uso General/Procesamiento de Transacciones: seleccione este tipo de configuración para crear una base de datos preconfigurada adecuada para una serie de usos, desde transacciones sencillas ejecutadas por un número elevado de usuarios simultáneos hasta consultas complejas. Esta configuración proporciona soporte para los siguientes tipos de usos:

- Acceso rápido a los datos de varios usuarios simultáneos, habitual en los entornos de procesamiento de transacciones.
- Largas consultas en ejecución de datos históricos complejos a través de un pequeño número de usuarios, habituales en los sistemas de toma de decisiones (DSS).
- Alta disponibilidad y rendimiento de procesamiento de transacciones.
- Capacidad de recuperación de gran volumen.

Almacenes de Datos: seleccione este tipo de configuración para crear una base de datos preconfigurada que sea adecuada para ejecutar consultas complejas sobre temas específicos. Normalmente, las bases de datos de Almacenes de Datos se utilizan para almacenar datos históricos necesarios para responder a preguntas comerciales estratégicas sobre temas relativos a pedidos de clientes, llamadas al servicio de soporte, perspectivas de ventas y patrones de adquisición. Esta configuración proporciona un soporte óptimo para los entornos de base de datos con los siguientes requisitos:

- Acceso rápido a grandes volúmenes de datos
- Soporte para procesamiento analítico en línea (OLAP).

© Alfaomega - RC Libros

SYNAPS PREMIUM

Nota: Los tipos de bases de datos preconfiguradas varían solo en los valores especificados para parámetros de inicialización concretos. Se crean los mismos archivos de datos y se utilizan para cada tipo y los requisitos de espacio en disco serán los mismos.

Especificar Identificadores de Base de Datos

Una base de datos se identifica mediante un nombre de base de datos global y la instancia asociada a la base de datos se identifica mediante un SID (Identificador del Sistema Oracle).

Nombre de la Base de Datos Global: nombre que se da a una base de datos para identificarla de forma única con respecto al resto de bases de datos de una red. El nombre de la base de datos global se compone de un nombre único de base de datos (*nombre_único_bd*), un delimitador (punto) y un nombre de dominio de base de datos (*dominio_bd*). Se representa como: *nombre_único_bd.dominio_bd*.

Donde:

nombre_único_bd es el nombre de la base de datos. Puede contener un máximo de 30 caracteres, siempre que los ocho primeros sean únicos y empiecen por un carácter alfanumérico. Los caracteres soportados serán alfanuméricos, carácter de subrayado (_), símbolo de dólar ($) y almohadilla numérica (#).

dominio_bd es el entorno de computación utilizado para la base de datos. No puede contener más de 128 caracteres (alfanuméricos, carácter de subrayado [_] y almohadilla numérica [#]), incluidos todos los puntos.

Nota: Asegúrese de que la combinación de nombre de base de datos (primeros ocho caracteres únicos del nombre único de la base de datos), delimitador y nombre de dominio de la base de datos no supera los 128 caracteres.

Por ejemplo: *sales.us.example.com*

Donde:

nombre_único_bd es *ventas*

dominio_bd es *es.ejemplo.com*

© Alfaomega - RC Libros

SYNAPS PREMIUM

Identificador del Sistema Oracle: es un nombre único para una instancia de Oracle Database en un host concreto. El identificador del sistema Oracle (SID) ayuda a identificar el archivo de control, así como a ubicar los archivos necesarios para abrir la base de datos. Al introducir el nombre de la base de datos global, Oracle Universal Installer rellena automáticamente el campo *Identificador del Sistema Oracle* con el nombre de la base de datos. Puede modificar este nombre en la instalación avanzada. Oracle Universal Installer limita el SID a 12 caracteres alfanuméricos para bases de datos de instancia única. Para las bases de datos Oracle RAC, el prefijo SID, que está compuesto por los ocho primeros caracteres del SID, debe ser un nombre único para cada base de datos. El SID no puede contener el carácter de subrayado (_), el signo del dólar ($) ni la almohadilla numérica (#). Seleccione la opción *Crear como Base de Datos de Contenedor* para crear la base de datos como una base de datos de contenedor (CDB) que puede soportar una o varias bases de datos de conexión (PDB). Si desea que Oracle Universal Installer cree una PDB en el momento de crear la CDB, especifique el nombre de PDB en el campo Nombre de Base de Datos de Conexión. El nombre de PDB debe ser único y cumplir las reglas de nomenclatura de bases de datos. Para crear PDB adicionales y gestionarlas, utilice el *Asistente de Configuración de Oracle Database*. Haga clic en *Siguiente*.

Nombre de Servicio (con Oracle RAC OneNode): asegúrese de que proporciona un nombre de servicio para la base de datos Oracle RAC OneNode. Los nombres de servicio permiten a los clientes conectar a un servicio en lugar de conectar a una instancia concreta de Oracle Database. Los nombres de servicio permiten a los clientes estar conectados constantemente incluso si la base de datos Oracle RAC OneNode sufre un failover o si se reubica en otro nodo.

Especificar Opciones de Configuración

Especifique la información de configuración para la base de datos que desea crear y, a continuación, haga clic en *Siguiente*. Las opciones de configuración disponibles son las siguientes: Memoria, Juegos de Caracteres y Esquemas de Ejemplo.

Memoria: Especifique la cantidad de memoria física (RAM) que desea asignar a la base de datos. Oracle Universal Installer calcula y muestra un valor por defecto para la asignación de memoria en el selector cíclico de memoria asignada y ajusta la barra de desplazamiento según corresponda. Los valores de la barra de desplazamiento y el selector cíclico están también limitados a los valores máximo y mínimo según la memoria física disponible. Puede utilizar la barra de desplazamiento o el selector cíclico para especificar un valor (MB) para la cantidad de memoria que desee asignar a la base de datos según los requisitos de la base de datos. Para Uso

© Alfaomega - RC Libros

SYNAPS PREMIUM

General/Procesamiento de Transacciones suele utilizarse Destino de SGA: 75 y Destino de PGA Agregada: 25. Para Almacenes de Datos suele utilizarse Destino de SGA: 60 y Destino de PGA Agregada: 40. Los números muestran el porcentaje de memoria física disponible que ha asignado para la base de datos.

Juegos de Caracteres: esta opción permite almacenar los datos de caracteres de la base de datos con uno de los siguientes métodos:

Usar Valor por Defecto: esta opción utiliza los valores del idioma del sistema operativo.

Usar Unicode: esta opción permite almacenar varios grupos de idiomas. Esta opción activa la lista desplegable *Seleccionar Juego de Caracteres de la Base de Datos*. Para determinar el juego de caracteres que se utilizará, debe decidir los grupos de idiomas que desea soportar en la base de datos. Debe aplicar estas directrices:

- Si desea soporte para el símbolo del Euro, seleccione WE8ISO8859P1, WE8MSWIN1252 o Unicode (AL32UTF8 o UTF8).

- Si solo desea soporte para inglés, seleccione el juego de caracteres por defecto mostrado por Oracle Universal Installer.

- Si desea soporte para el inglés y otro idioma, seleccione el juego de caracteres adecuado de la lista.

- Si el sistema operativo está configurado para utilizar el otro idioma, el juego de caracteres por defecto mostrado por Oracle Universal Installer puede ser el correcto.

- Si desea soporte para inglés y dos o más idiomas adicionales, o un idioma que no aparece en la lista, seleccione Unicode (AL32UTF8 o UTF8).

Esta es la lista de idiomas y juegos de caracteres comunes para un grupo de idiomas:

- *Grupo de Idiomas*

  Europeo Occidental

SYNAPS PREMIUM

Idiomas: Catalán, Danés, Holandés, Finlandés, Francés, Alemán, Islandés, Indonesio, Italiano, Malayo, Noruego, Portugués, Español, Sueco

Juego de Caracteres Comunes: WE8ISO8859P1, WE8ISO8859P15, WE8MSWIN1252

Europeo Oriental

Idiomas: Croata, Checo, Húngaro, Polaco, Rumano, Eslovaco, Esloveno

Juego de Caracteres Comunes: EE8ISO8859P2, EE8MSWIN1250

Báltico

Idiomas: Estonio, Letón, Lituano

Juego de Caracteres Comunes: NEE8ISO8859P4, BLT8MSWIN1257, BLT8ISO8859P13

Cirílico

Idiomas: Búlgaro, Ruso, Ucraniano

Juego de Caracteres Comunes: CL8ISO8859P5, CL8MSWIN1251

Árabe

Idioma: Árabe

Juego de Caracteres Comunes: AR8ISO8859P6, AR8MSWIN1256

Griego

Idioma: Griego

Juego de Caracteres Comunes: EL8ISO8859P7, EL8MSWIN1253

Hebreo

Idioma: Hebreo

© Alfaomega - RC Libros

SYNAPS PREMIUM

Juego de Caracteres Comunes: IW8ISO8859P8, IW8MSWIN1255

Japonés

Idioma: Japonés

Juego de Caracteres Comunes: JA16EUC, JA16SJIS

Coreano

Idioma: Coreano

Juego de Caracteres Comunes: KO16KSC5601, KO16MSWIN949

Chino Simplificado

Idioma: Chino Simplificado

Juego de Caracteres Comunes: ZHS16GBK, ZHS16CGB231280, ZHS32GB18030

Tailandés

Idioma: Tailandés

Juego de Caracteres Comunes: TH8TISASCII

Chino Tradicional

Idioma: Chino Tradicional

Juego de Caracteres Comunes: ZHT32EUC, ZHT16MSWIN950, ZHT16HKSCS, ZHT16BIG5

Turco

Idioma: Turco

Juego de Caracteres Comunes: WE8ISO8859P9, TR8MSWIN1254

© Alfaomega - RC Libros

SYNAPS PREMIUM

Vietnamita

Idioma: Vietnamita

Juego de Caracteres Comunes: VN8MSWIN1258

Esquemas de Ejemplo: especifique si desea incluir los esquemas de ejemplo en la base de datos. Oracle proporciona esquemas de ejemplo para utilizarlos con ejemplos de productos y documentación. La opción *Crear Base de Datos con Esquemas de Ejemplo* no está seleccionada por defecto. Sin embargo, puede seleccionarla si desea crear la base de datos inicial con esquemas de ejemplo.

Nota: Si instala los esquemas de ejemplo, el Asistente de Configuración de Oracle Database creará el tablespace EXAMPLES en la base de datos, que necesita 150 MB de espacio en disco. Si está utilizando dispositivos raw para el almacenamiento de base de datos, debe crear y especificar un dispositivo raw diferente para este tablespace. Si no instala los esquemas de ejemplo, podrá crearlos manualmente en la base de datos después de la instalación.

Especificar Opciones de Gestión: gestionar la base de datos con Oracle Enterprise Manager Cloud Control

Esta pantalla le ofrece la opción de gestionar la base de datos con Oracle Enterprise Manager Cloud Control. Seleccione *Registrar* en Enterprise Manager (EM) Cloud Control y especifique lo siguiente para la configuración de Oracle Enterprise Manager Cloud Control. A continuación, haga clic en *Siguiente*:

- *Host de OMS*: este es el nombre del sistema en el que se está ejecutando el repositorio de gestión.

- *Puerto de OMS*: este es el número de puerto de Oracle Enterprise Manager para recibir solicitudes desde el servicio de gestión.

- *Nombre de Usuario de Administrador de EM*: nombre de usuario para conectarse a Oracle Enterprise Manager.

- *Contraseña de Administrador de EM*: contraseña para conectarse a Oracle Enterprise Manager.

© Alfaomega - RC Libros

SYNAPS PREMIUM

- *Especificar Contraseña de Usuario ASMSNMP*: contraseña para el usuario ASMSNMP configurado en Oracle ASM; solo es necesario si selecciona Oracle ASM como opción de almacenamiento de la base de datos.

Nota: Oracle Enterprise Manager Database Express está siempre instalado y configurado por defecto independientemente de si registra Oracle Enterprise Manager Cloud Control o no.

Especificar Opciones de Almacenamiento de Base de Datos

Seleccione el método que desea utilizar para almacenar archivos de base de datos y haga clic en *Siguiente*.

Las opciones de almacenamiento disponibles son las siguientes: Sistema de Archivos y Oracle Automatic Storage Management.

Sistema de Archivos: seleccione esta opción si desea almacenar los archivos de base de datos en un sistema de archivos. Para instalaciones de Oracle Real ApplicationClusters (Oracle RAC), el sistema de archivos que seleccione debe ser un sistema de archivos de cluster. También puede estar en un dispositivo de almacenamiento conectado a red (NAS) certificado. Para almacenar los archivos de base de datos en un sistema de archivos:

- Seleccione *Sistema de Archivos*.

- En el campo *Especificar Ubicación de Archivo de Base de Datos*, especifique el directorio principal para los archivos de base de datos.

Nota: Oracle recomienda que cambie la ruta de acceso por defecto seleccionada por Oracle Universal Installer.

Oracle Universal Installer proporciona una ruta de acceso de directorio por defecto, seleccionada del siguiente modo:

En sistemas UNIX:

- Si la variable de entorno ORACLE_BASE está definida cuando inicie Oracle Universal Installer, la ruta de acceso de directorio por defecto será $ORACLE_BASE/oradata.

© Alfaomega - RC Libros

SYNAPS PREMIUM

- Si la variable de entorno ORACLE_BASE no está definida, pero tiene permisos de escritura en el directorio principal del directorio raíz de Oracle, la ruta de acceso del directorio por defecto será directorio_raíz_oracle_directorio_principal/oradata, donde directorio_raíz_oracle_directorio_principal es la ruta de acceso al directorio principal del directorio raíz de Oracle que ha especificado.

- En todos los demás casos, la ruta de acceso del directorio por defecto es directorio_raíz_oracle/oradata, donde directorio_raíz_oracle es la ruta de acceso del directorio raíz de Oracle que ha especificado.

En sistemas Microsoft Windows:

La ruta de acceso del directorio por defecto es ORACLE_BASE\oradata, donde ORACLE_BASE es el directorio principal del directorio raíz de Oracle en el que ha instalado el producto.

Aplique las siguientes directrices al especificar este directorio:

- El directorio que especifique debe estar en un sistema de archivos diferente del sistema operativo y del directorio raíz de Oracle.

- En sistemas UNIX, las directrices de Arquitectura Flexible Óptima (OFA) recomiendan que la ruta de acceso sea similar a la siguiente, donde punto_montaje es un directorio de puntos de montaje del sistema de archivos: */mount_point/oradata.* Por ejemplo: */u02/oradata.*

- El directorio que especifique debe existir y el usuario que ha iniciado Oracle Universal Installer debe tener permisos de escritura en él.

- El sistema de archivos que contiene el directorio debe tener al menos 1,2 GB de espacio libre en disco. También es necesario espacio libre en disco adicional para las bases de datos de producción, dependiendo de la cantidad de datos que planee almacenar.

- Oracle Universal Installer instala los archivos de base de datos en un subdirectorio de la ruta de acceso que especifique, a la que asigna un nombre utilizando el valor especificado para el nombre de la base de datos. Por ejemplo, si especifica ventas.es.ejemplo.com para el nombre de la base de datos global y /u02/oradata para la ubicación del archivo

© Alfaomega - RC Libros

SYNAPS PREMIUM

de base de datos, Oracle Universal Installer coloca los archivos de base de datos en el directorio /u02/oradata/sales.

Oracle Automatic Storage Management: seleccione esta opción si ha instalado la infraestructura de grid y desea almacenar los archivos de base de datos en un grupo de discos de Oracle Automatic Storage Management. Los grupos de discos de Oracle Automatic Storage Management se crean especificando uno o más dispositivos de disco que gestionarán una instancia diferente de Oracle Automatic Storage Management.

Seleccionar Grupo de Discos de ASM

- Proporcione el nombre del grupo de discos inicial que configure para los archivos de la base de datos en el campo *Nombre del Grupo de Discos*.

- En la tabla *Discos* se muestran los grupos de discos existentes. Seleccione el grupo de discos que desea utilizar como almacenamiento de los archivos de la base de datos. Estos necesitan un mínimo de 3,3 MB de espacio disponible.

- Si no aparece el grupo de discos que desea utilizar, haga clic en *Refrescar* para asegurarse de que se muestran todos los grupos de discos disponibles.

¿Qué es un Grupo de Discos de ASM?

Cada grupo de discos de ASM consta de varias particiones de disco. Los grupos de discos se configuran durante la instalación de Oracle GridInfrastructure y, después de la instalación, con el privilegio SYSASM mediante asmcmd o los comandos SQL de creación de grupos de discos. El grupo de discos es el objeto fundamental que gestiona ASM. Cada grupo de discos contiene los metadatos necesarios para la gestión de su espacio. Los archivos se asignan desde los grupos de discos. Cualquier archivo de ASM se encuentra entero en un solo grupo de discos. Sin embargo, un grupo de discos puede contener archivos que pertenecen a varias bases de datos y una sola base de datos puede utilizar archivos de varios grupos de discos. En la mayoría de las instalaciones, solo se necesita un número reducido de grupos de discos, por lo general, dos y, en contadas ocasiones, más de tres. Los componentes de los grupos de discos incluyen discos, archivos y unidades de asignación.

Opciones de Redundancia

La redundancia o duplicación es el proceso que consiste en replicar el contenido de un archivo en otros archivos. La redundancia protege la integridad de los datos

SYNAPS PREMIUM

almacenando copias de ellos en varios discos. El tipo del grupo de discos determina los niveles de duplicación con los que Oracle crea los archivos en el grupo. Al crear un grupo de discos de ASM, especifique el tipo basándose en uno de los tres niveles siguientes de redundancia:

- *Alta*: duplicación en tres direcciones.

- *Normal*: duplicación en dos direcciones.

- *Externa*: no se utiliza la duplicación de archivos con ASM, normalmente cuando se realiza por medio de otra tecnología como una matriz redundante de discos independientes (RAID).

Grupo de Discos ASM

El tipo del grupo de discos determina los niveles de duplicación con los que Oracle crea los archivos en el grupo. El nivel de redundancia controla cuántos fallos de disco se toleran sin desmontar el grupo de discos ni perder datos.

Especificar Opciones de Recuperación

Seleccione *Activar Recuperación* para activar la recuperación mediante una de las siguientes opciones y haga clic en *Siguiente*:

- Seleccione *Sistema de Archivos* para utilizar un directorio del sistema de archivos para el área de recuperación rápida y, a continuación, especifique la ruta de acceso del área de recuperación rápida en el campo *Ubicación del Área de Recuperación*.

- Seleccione *Oracle Automatic Storage Management* para utilizar un grupo de discos de Automatic Storage Management para el área de recuperación rápida.

Uso del Sistema de Archivos para el Área de Recuperación Rápida

Aplique las siguientes instrucciones cuando decida utilizar o no el sistema de archivos para el área de recuperación rápida:

- Al crear una base de datos preconfigurada, debe especificar un directorio del sistema de archivos con al menos 2 GB de espacio libre en disco.

© Alfaomega - RC Libros

SYNAPS PREMIUM

- Como instrucción general, el espacio disponible para el área de recuperación rápida debe duplicar el tamaño de la base de datos.

- Para mejorar el rendimiento y asegurarse de que un único punto de fallo no provoca la pérdida de datos, Oracle recomienda que seleccione un sistema de archivos individual para el área de recuperación rápida. Seleccione un sistema de archivos independiente del sistema operativo, el software de Oracle y los archivos de base de datos. Nota: Por este motivo, Oracle recomienda que cambie la ruta de acceso por defecto seleccionada por Oracle Universal Installer.

- Asegúrese de que el usuario del sistema operativo que está utilizando para ejecutar Oracle Universal Installer tiene permisos de escritura en el directorio que ha especificado.

- Para instalaciones de Oracle Real ApplicationClusters (Oracle RAC), el sistema de archivos que seleccione debe ser un sistema de archivos de cluster o un sistema de archivos NFS en un dispositivo NAS certificado. Esta es la opción por defecto si utiliza el almacenamiento de sistema de archivos para los archivos de base de datos. También puede utilizar Oracle Automatic Storage Management para los archivos de copia de seguridad incluso si no ha seleccionado ese mecanismo para los archivos de base de datos.

Directorio por Defecto para el Área de Recuperación Rápida

Oracle Universal Installer proporciona una ruta de acceso de directorio por defecto, seleccionada del siguiente modo:

En sistemas basados en UNIX:

- Si la variable de entorno ORACLE_BASE está definida cuando inicie Oracle Universal Installer, la ruta de acceso de directorio por defecto será $ORACLE_BASE/área_recuperación.

- Si la variable de entorno ORACLE_BASE no está definida, pero tiene permisos de escritura en el directorio principal del directorio raíz de Oracle, la ruta de acceso del directorio por defecto será directorio_raíz_oracle_directorio_principal/área_recuperación, donde directorio_raíz_oracle_directorio_principal es la ruta de acceso al directorio principal del directorio raíz de Oracle que ha especificado.

© Alfaomega - RC Libros

SYNAPS PREMIUM

- En todos los demás casos, la ruta de acceso del directorio por defecto es directorio_raíz_oracle/área_recuperación, donde directorio_raíz_oracle es la ruta de acceso del directorio raíz de Oracle que ha especificado.

En sistemas Microsoft Windows: la ruta de acceso del directorio por defecto es ORACLE_BASE\área_recuperación, donde ORACLE_BASE es el directorio principal del directorio raíz de Oracle en el que ha instalado el producto.

Uso de Oracle Automatic Storage Management para el Área de Recuperación Rápida

Siga estas instrucciones para decidir si utilizar o no Oracle Automatic Storage Management para el área de recuperación rápida.

- Al crear una base de datos preconfigurada, debe seleccionar o crear un grupo de discos con espacio en disco suficiente para la base de datos y los archivos de copia de seguridad. El espacio en disco efectivo necesario para los archivos de copia de seguridad es de 2 GB. Sin embargo, según el nivel de redundancia utilizado por el grupo de discos, el espacio físico en disco necesario será dos o tres veces superior a este valor si ha seleccionado los niveles de redundancia Normal o Alta.

- Como instrucción general, el espacio disponible para el área de recuperación rápida debe duplicar el tamaño de la base de datos.

- Para mejorar el rendimiento y asegurar la recuperación ante un fallo de disco, Oracle recomienda que utilice un grupo de discos con redundancia Normal o Alta.

- Para instalaciones de Oracle RAC, el grupo de discos que seleccione o cree debe utilizar dispositivos de almacenamiento que compartan todos los nodos del cluster.

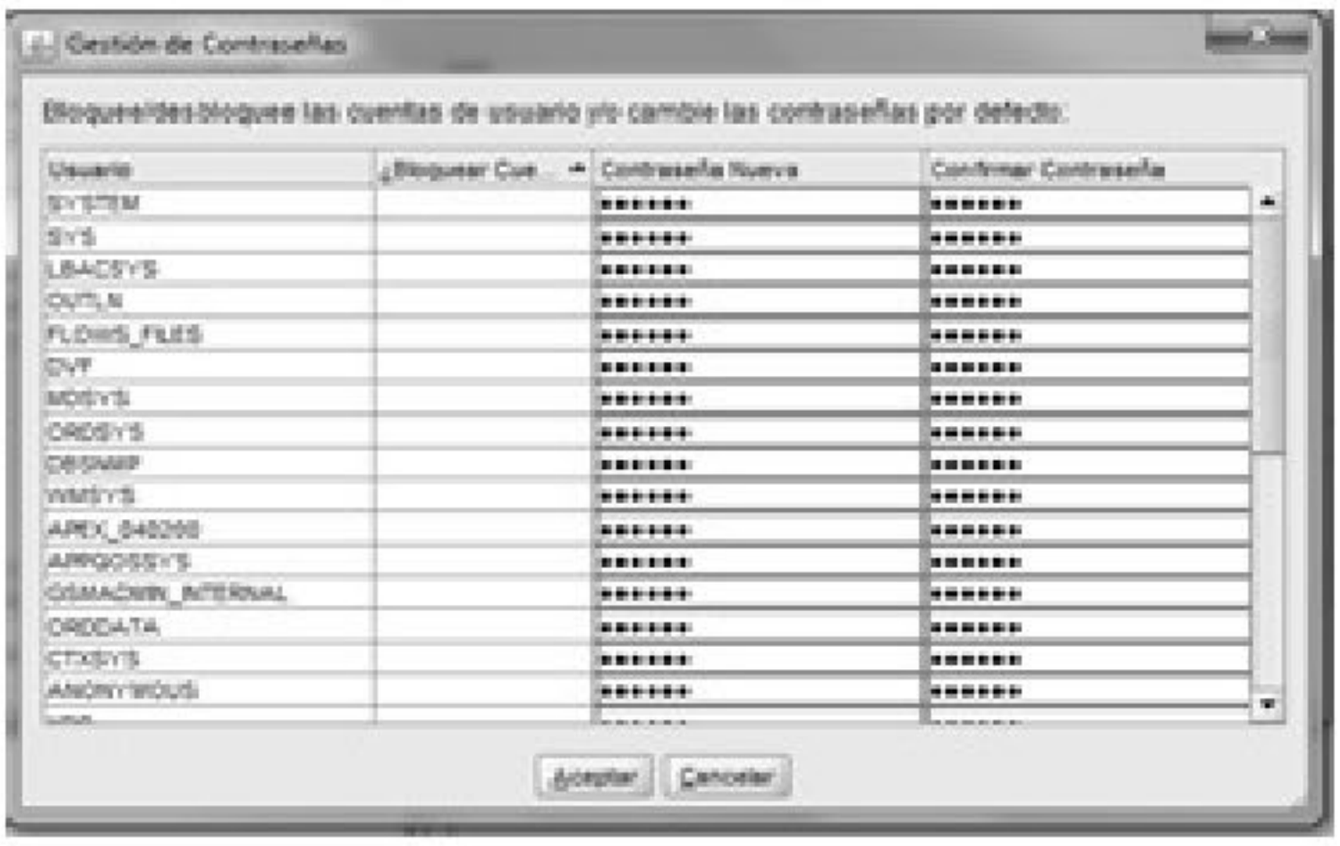
- Es la opción por defecto si opta por utilizar Oracle Automatic Storage Management o el almacenamiento de dispositivos raw para los archivos de base de datos. También puede utilizar el sistema de archivos para los archivos de copia de seguridad incluso si no ha seleccionado ese mecanismo para los archivos de base de datos.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Especificar Contraseñas de Esquemas de Base de Datos

Especifique las contraseñas para las siguientes cuentas de base de datos administrativas (esquemas): SYS, SYSTEM y DBSNMP. A continuación, haga clic en *Siguiente*. Aunque el software permite utilizar la misma contraseña para todas las cuentas, Oracle recomienda especificar distintas para cada cuenta. Para especificar una contraseña diferente para cada cuenta seleccione *Usar diferentes contraseñas para estas cuentas* y para cada cuenta mostrada en la tabla, introduzca una contraseña en la columna *Contraseña* y vuelva a introducir la misma contraseña en la columna *Confirmar Contraseña*. Para especificar la misma contraseña para todas las cuentas seleccione *Usar la Misma Contraseña para Todas las Cuentas*, introduzca una contraseña en el campo *Contraseña* y vuelva a introducir la misma contraseña en el campo *Confirmar Contraseña*. Nota: Estas cuentas son cuentas con privilegios de la base de datos. Debe recordar las contraseñas que ha especificado.



Requisitos Mínimos de Contraseña

La instalación no podrá continuar si no se cumplen los siguientes requisitos:

- La contraseña no puede superar los 30 caracteres.
- La contraseña no puede contener caracteres no válidos: ! @ % ^ & * ( ) + = \ | ` ~ [{] } ; : ' " , < > ?
- La contraseña no podrá ser el nombre de usuario.

© Alfaomega - RC Libros

SYNAPS PREMIUM

- No se aceptan contraseñas vacías.
- La contraseña de la cuenta SYS no podrá ser change_on_install (no sensible a mayúsculas/minúsculas).
- La contraseña de la cuenta SYSTEM no podrá ser manager (no sensible a mayúsculas/minúsculas).
- La contraseña de la cuenta DBSNMP o podrá ser dbsnmp (no sensible a mayúsculas/minúsculas).
- Si decide utilizar la misma contraseña para todas las cuentas, no debe ser change_on_install, manager o dbsnmp. (no sensible a mayúsculas/minúsculas).

Recomendaciones para Contraseñas

Cuando se le solicite introducir una contraseña, siga estas directrices. Oracle recomienda que la contraseña que especifique:

- Contenga al menos una letra en minúscula.
- Contenga al menos una letra en mayúscula.
- Contenga al menos un dígito.
- Tenga como mínimo 8 caracteres de longitud.
- Utilice un juego de caracteres de base de datos que incluya el guion bajo (_), símbolo de dólar ($) y símbolo de almohadilla (#).
- Si contiene caracteres especiales, también en el caso en el que la contraseña empiece por un número o por un símbolo, incluya la contraseña entre comillas dobles.
- No debe ser una palabra real.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Grupos del Sistema Operativo con Privilegios

Especifique los nombres de grupo UNIX para el grupo de administradores de base de datos (OSDBA) y, opcionalmente, el grupo de operadores de base de datos (OSOPER) y haga clic en *Siguiente*. Los grupos que especifique deben existir. Para instalaciones de Oracle Real ApplicationClusters, los identificadores de grupo (gid) de cada grupo deben ser los mismos en todos los nodos del cluster. Estos grupos tienen el siguiente significado:

Grupo OSDBA (normalmentedba): debe crear este grupo al instalar por primera vez el software de Oracle Database en el sistema. Este grupo identifica a las cuentas de usuario del sistema operativo que tienen privilegios administrativos de base de datos (privilegio SYSDBA). El nombre que se utiliza para este grupo en los ejemplos de código de Oracle es dba.

Grupo OSOPER para Oracle Database (normalmente, oper): este grupo es opcional. Cree este grupo si desea que un grupo independiente de usuarios del sistema operativo tenga un juego limitado de privilegios administrativos de la base de datos (privilegio SYSOPER). Este grupo no se puede conectar directamente como SYSOPER a menos que se haya otorgado explícitamente. Sin embargo, tienen los privilegios otorgados por el privilegio SYSOPER. Por defecto, los miembros del grupo OSDBA tienen otorgados todos los privilegios con el privilegio SYSOPER. El nombre habitual para este grupo es oper.

Grupo OSBACKUPDBA para Oracle Database (normalmente, backupdba): cree este grupo si desea que un grupo independiente de usuarios del sistema operativo tenga un juego limitado de privilegios administrativos relacionados con la copia de seguridad y recuperación de la base de datos (privilegio SYSBACKUP). El nombre habitual para este grupo es backupdba.

Grupo OSDGDBA para Oracle Data Guard (normalmente, dgdba): cree este grupo si desea que un grupo independiente de usuarios del sistema operativo tenga un juego limitado de privilegios para administrar y supervisar Oracle Data Guard (privilegio SYSDG). El nombre habitual para este grupo es dgdba.

Grupo OSKMDBA para gestión de clave de cifrado (normalmente, kmdba): Cree este grupo si desea que un grupo independiente de usuarios del sistema operativo tenga

SYNAPS PREMIUM

un juego limitado de privilegios para gestión de claves de cifrado como la gestión de Oracle Wallet Manager (privilegio SYSKM). El nombre habitual para este grupo es kmdba. Esta pantalla aparece solo cuando el usuario de UNIX que ha iniciado Installer no es miembro del grupo dba.

Realizar Comprobaciones de Requisitos

Las comprobaciones de requisitos garantizan que se cumplan los requisitos mínimos del sistema para llevar a cabo la instalación de la base de datos (Figura 1-8).

- Si hace clic en *Volver a Comprobar*, se vuelve a ejecutar la comprobación de requisitos a fin de determinar si se cumplen los requisitos mínimos para continuar con la instalación de la base de datos.

- Haga clic en *Corregir y Volver a Comprobar* si desea que Installer corrija el problema y vuelva a comprobar los requisitos del sistema. La función *Corregir y Volver a Comprobar* genera un script (solo en UNIX) que define algunos de los parámetros del sistema en los valores recomendados por Oracle. Debe ejecutar el script como usuario raíz. Oracle recomienda no modificar el contenido de este script.

  En la lista, seleccione una de las siguientes opciones:

- *Mostrar Fallos*: para obtener la lista de los requisitos fallidos.

- *Mostrar Todo*: para obtener la lista de todas las comprobaciones de requisitos ejecutadas.

- *Mostrar Correctos*: para obtener la lista de las comprobaciones de requisitos correctas.

Nota: Active *Ignorar Todo* si desea ignorar todos los errores y continuar con la instalación de la base de datos.

© Alfaomega - RC Libros

SYNAPS PREMIUM

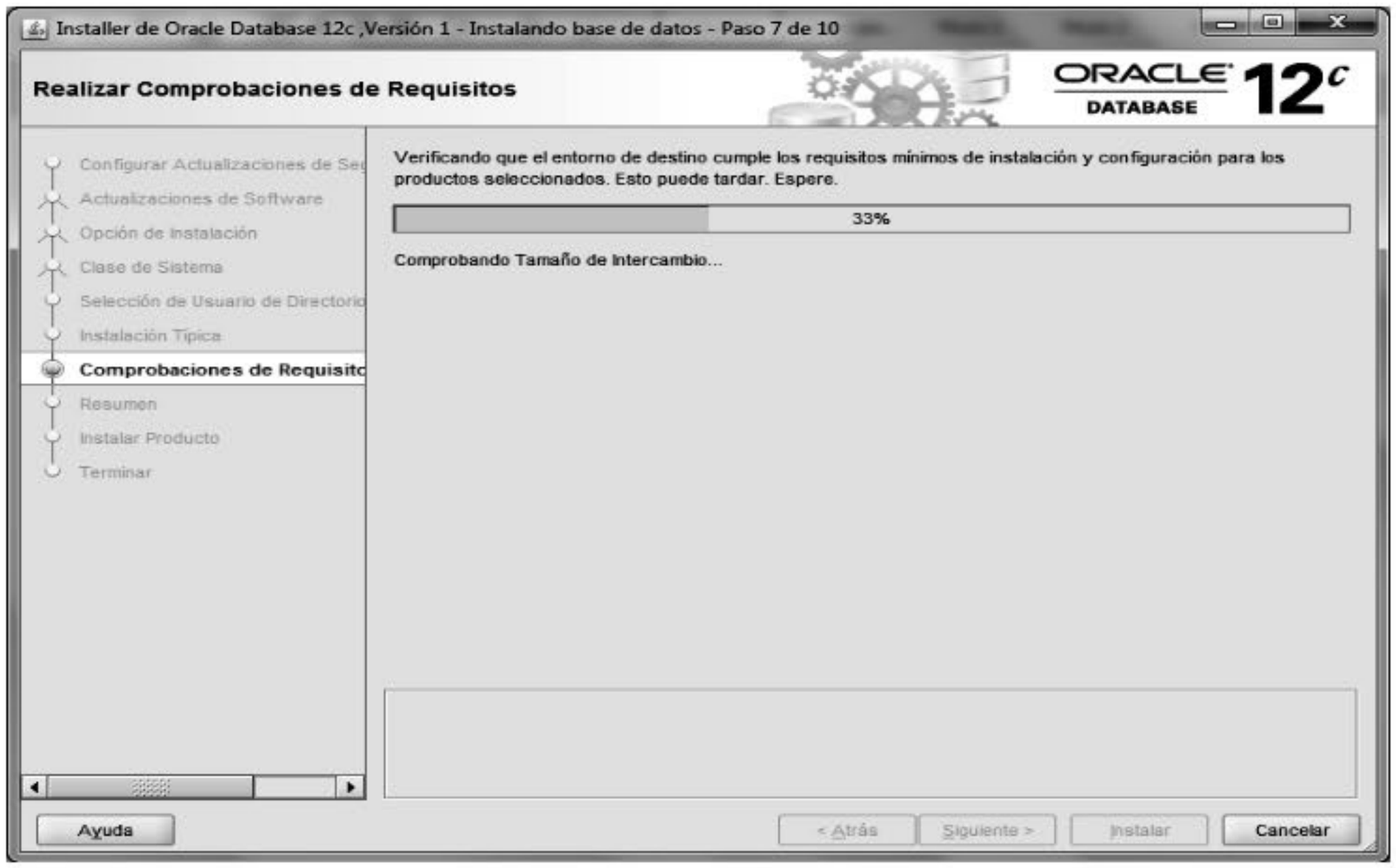


Figura 1-8

Resumen

En este cuadro de diálogo (Figura 1-9) se presenta un resumen de las opciones seleccionadas durante el proceso de instalación. Según la instalación, la información puede incluir:

- Valores globales
- Espacio en disco
- Usuario de servicio de Oracle (solo para sistemas operativos Microsoft Windows)
- Ubicación de origen
- Método de instalación
- Edición de base de datos
- Directorio base de Oracle
- Ubicación del software

SYNAPS PREMIUM

- Grupos del Sistema Operativo con Privilegios
- Información de Base de Datos
- Configuración
- Nombre de base de datos global
- Identificador del Sistema Oracle (SID)
- Memoria asignada
- Opción Gestión Automática de Memoria
- Juego de caracteres de la base de datos
- Mecanismo de almacenamiento de base de datos
- Ubicación de archivos de base de datos
- Copia de seguridad automática

Haga clic en *Guardar Archivo de Respuesta* si desea guardar todos los pasos de la instalación en un archivo de respuesta. Haga clic en *Instalar*.

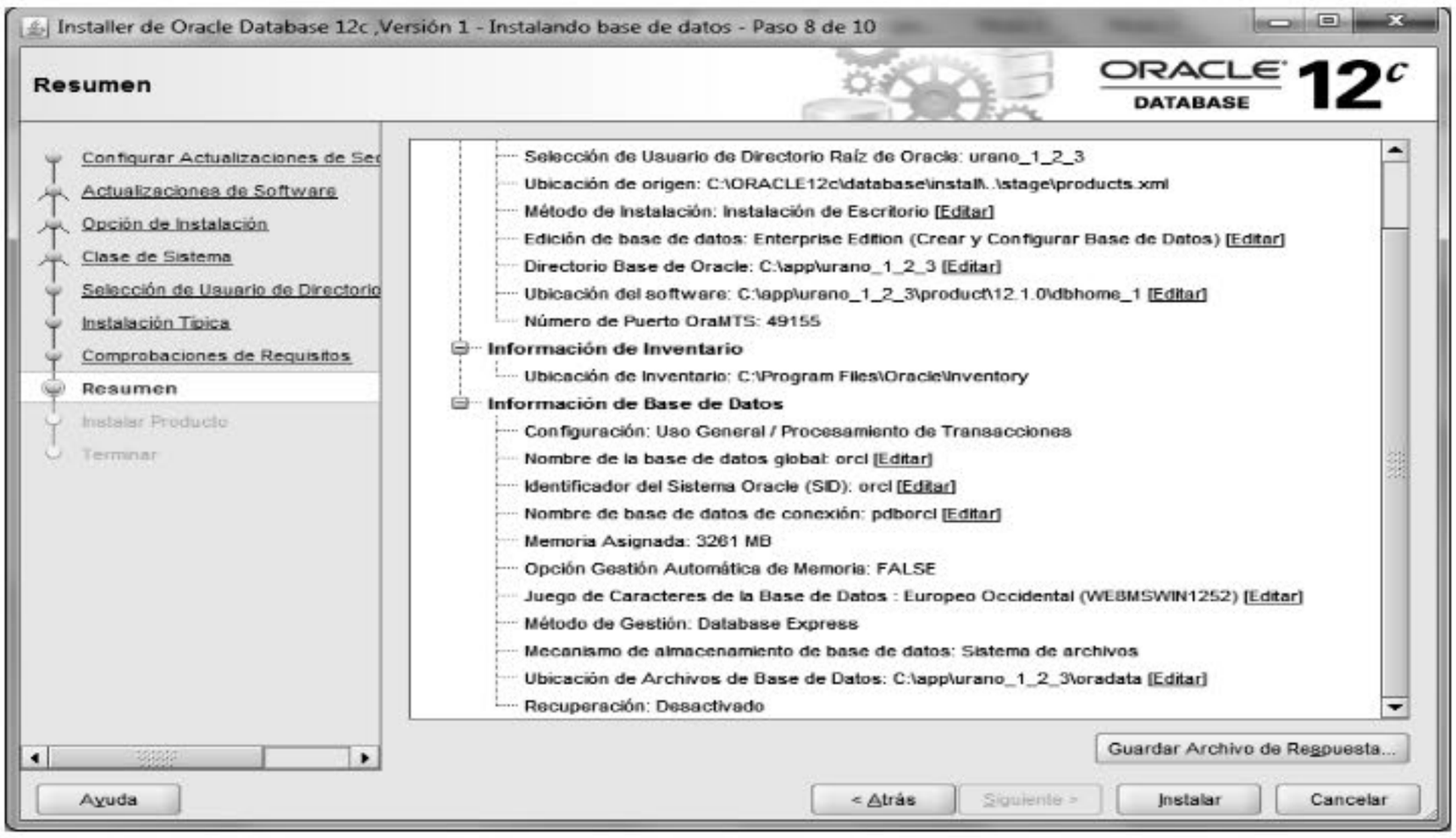


Figura 1-9

© Alfaomega - RC Libros

SYNAPS PREMIUM

Instalación del Producto

La pantalla *Instalar Producto* (Figura 1-10) aparece al instalar el producto. Las operaciones de instalación incluyen la ejecución de acciones como copiar archivos, agregar servicios en sistemas Windows, enlazar con sistemas UNIX y ejecutar cálculos y decisiones. Una barra de progreso muestra el estado de la instalación de componentes según el tamaño del componente. El tamaño está relacionado con el tamaño total de la instalación.

Nota: En las actualizaciones y reinstalaciones, OUI desinstala las versiones anteriores del producto antes de iniciar la instalación.

Haga clic en *Detalles* para obtener más información acerca de la instalación de la base de datos. Si no dispone de los asistentes de configuración opcionales o necesarios para la instalación, aparecerá la pantalla *Terminar* al finalizar la instalación. Después de instalar la base de datos, se le solicita que ejecute algunos script de configuración de raíz para el nuevo inventario como usuario raíz. Haga clic en *Siguiente*.

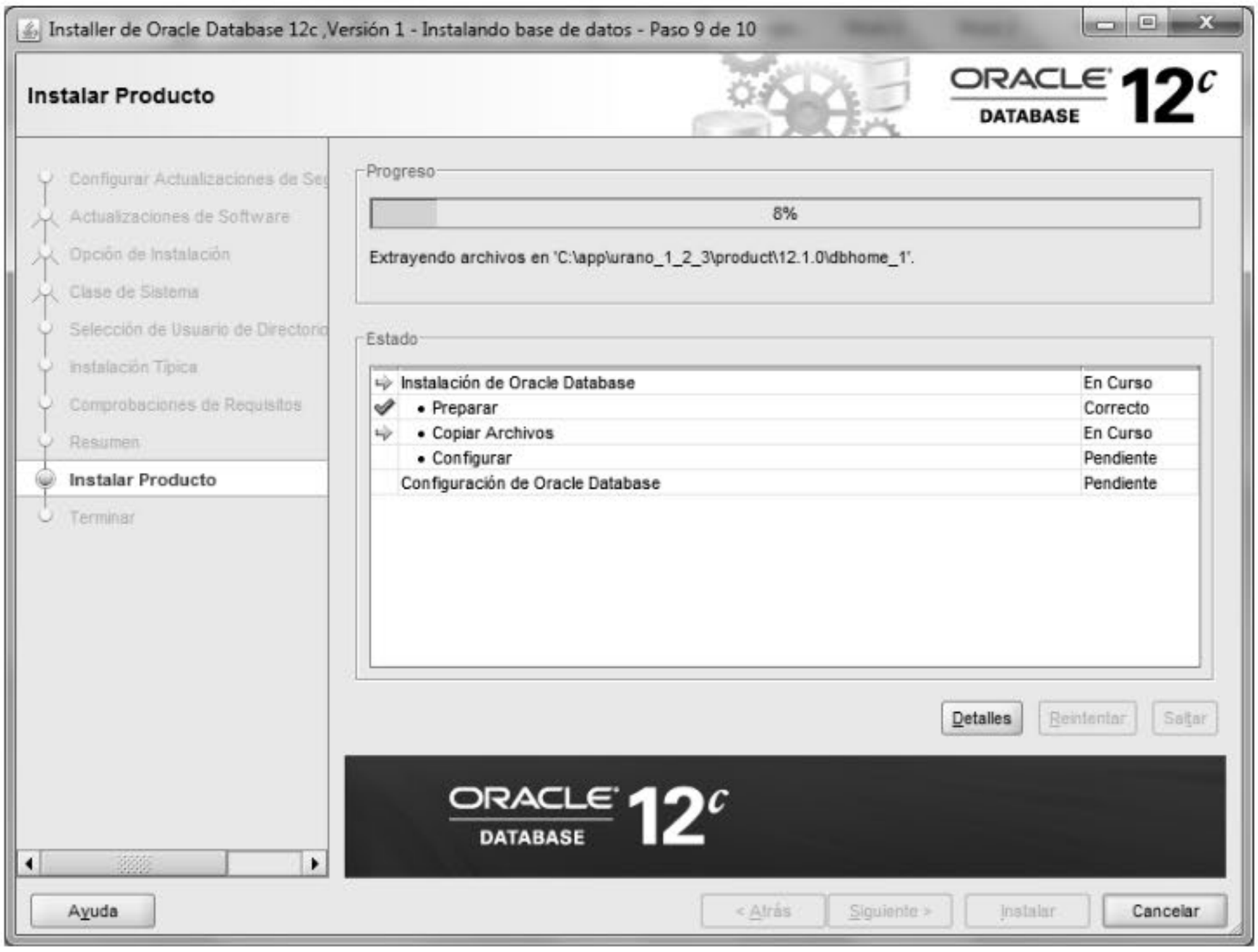


Figura 1-10

SYNAPS PREMIUM

Durante el proceso de extracción de archivos se realiza el proceso de Configuración de Oracle Database. Una vez finalizado este proceso, se obtiene la pantalla de la Figura 1-11, que muestra, entre otras cosas, la información de la base de datos y la URL de Enterprise Manager Database, necesaria para trabajar con Oracle Enterprise Manager.

Figura 1-11

A continuación se realiza el proceso de extracción de archivos hasta su finalización (Figura 1-12).

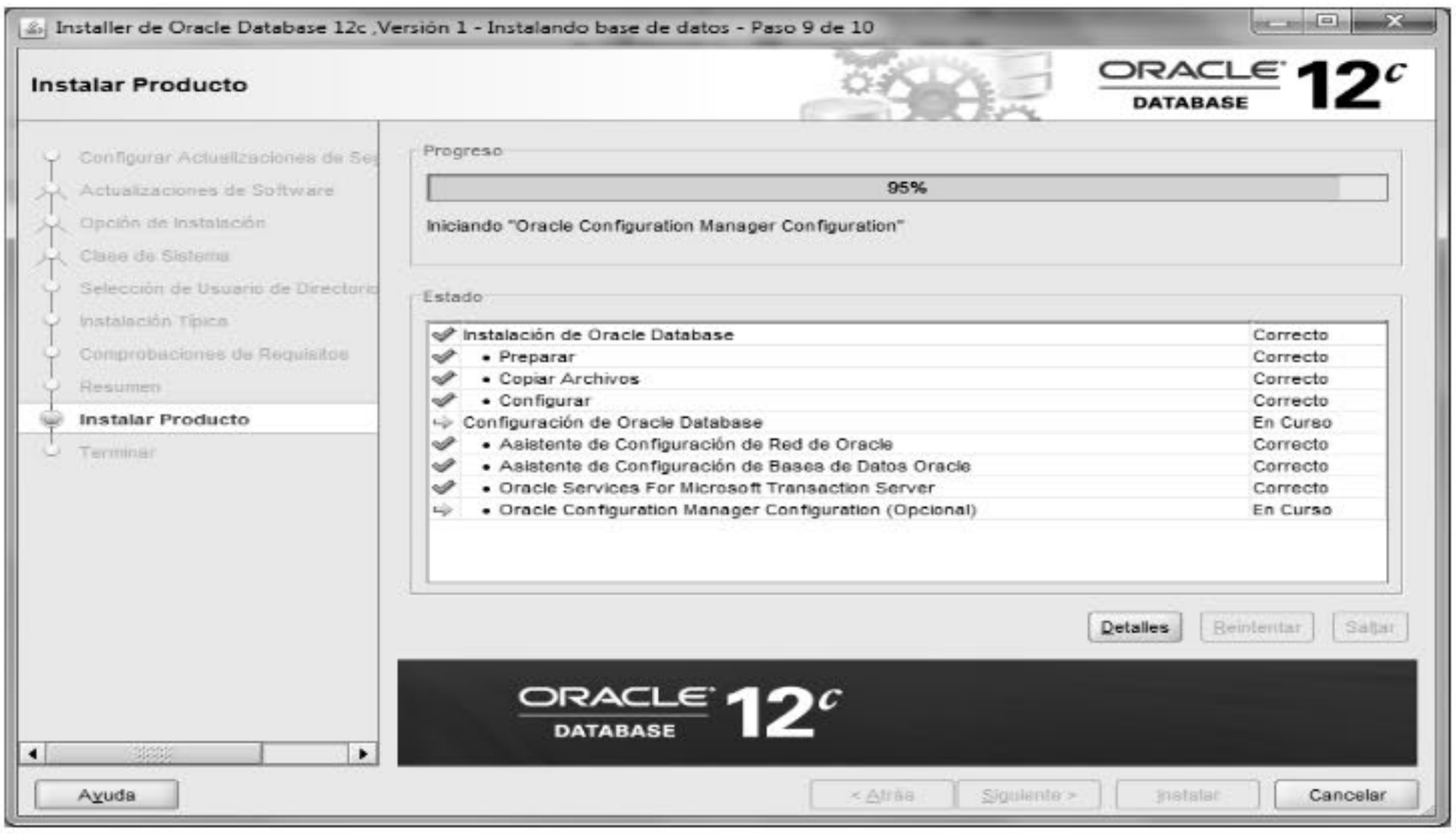


Figura 1-12

© Alfaomega - RC Libros

SYNAPS PREMIUM

Terminar

Este cuadro de diálogo (Figura 1-13) muestra que la instalación de Oracle Database se ha realizado correctamente. Haga clic en *Cerrar*.

Figura 1-13

Finalizada la instalación, ya disponemos de Oracle 12c en el menú *Inicio* de Windows (Figura 1-14).

INICIO DE ORACLE 12C

Para empezar a trabajar con Oracle es necesario arrancar el servidor de base de datos, lo que implica iniciar una instancia de base de datos, asociar la base de datos a la instancia y abrir la base de datos. En Oracle 12c estas operaciones pueden realizarse eligiendo *Inicio → Todos los programas → Oracle-Ora12gHome1 → Herramientas de conFiguración y migración → Asistente de administración para Windows* (Figura 1-14). A continuación se abre el árbol *Oracle Administration Assistant for Windows* y se hace clic en los subniveles *Oracle Managed Objects, Computers, Nombre del servidor que contiene la base de datos* (en nuestro caso, AULAMASTER01) y *Databases* (Figura 1-15).

Se observa que al hacer doble clic en *Databases* aparecen las instancias de bases de datos disponibles (en nuestro caso, solo ORCL). Debemos tener presente que en la terminología de Oracle, una *instancia* es el conjunto de procesos y áreas de memoria del servidor que utiliza Oracle para administrar el acceso a una base datos. En Windows una instancia de base de datos Oracle 12c es un *servicio de Windows*. Para abrir la instancia de base de datos adecuada (en nuestro caso, ORCL) hacemos clic con el botón secundario del ratón sobre ella y en el menú emergente resultante elegimos ***Conectar a***

SYNAPS PREMIUM

base de datos (Figura 1-15). A continuación puede ser necesario definir el método de autenticación con el nombre de usuario y la contraseña. Finalmente se produce la conexión a la instancia (Figura 1-16). Para realizar la desconexión de una instancia de base de datos hacemos clic con el botón secundario del ratón sobre ella y elegimos la opción ***Desconectar de base de datos*** (Figura 1-17).

Una vez iniciada una instancia de base de datos, se puede establecer una conexión al servidor de base de datos Oracle y comenzar el trabajo con la información de la misma. Lo más habitual es utilizar la herramienta ***SQL*Plus*** para realizar este trabajo. No obstante, la instancia de base de datos previamente iniciada vigila automáticamente el trabajo en la base de datos con *SQL*Plus* preservando la integridad de la base de datos y la gestión compartida, entre otras muchas tareas. Para realizar la conexión con Oracle 12c a través de SQL*Plus se elige *Inicio → Todos los programas → Oracle-Ora12cHome1 → Desarrollo de aplicaciones → SQL Plus* (Figura 1-13). Se obtiene el cuadro de diálogo *Conectar* en el que se introducen el nombre de usuario y la contraseña para conectarse a la base de datos. No olvidemos que, según lo indicado en el proceso de instalación, se puede utilizar como usuarios SYS, SYSTEM, etc., con las claves que ya se han definido en el proceso de instalación. Al pulsar *Aceptar* se obtiene el entorno de trabajo de SQL*Plus (Figura 1-18).

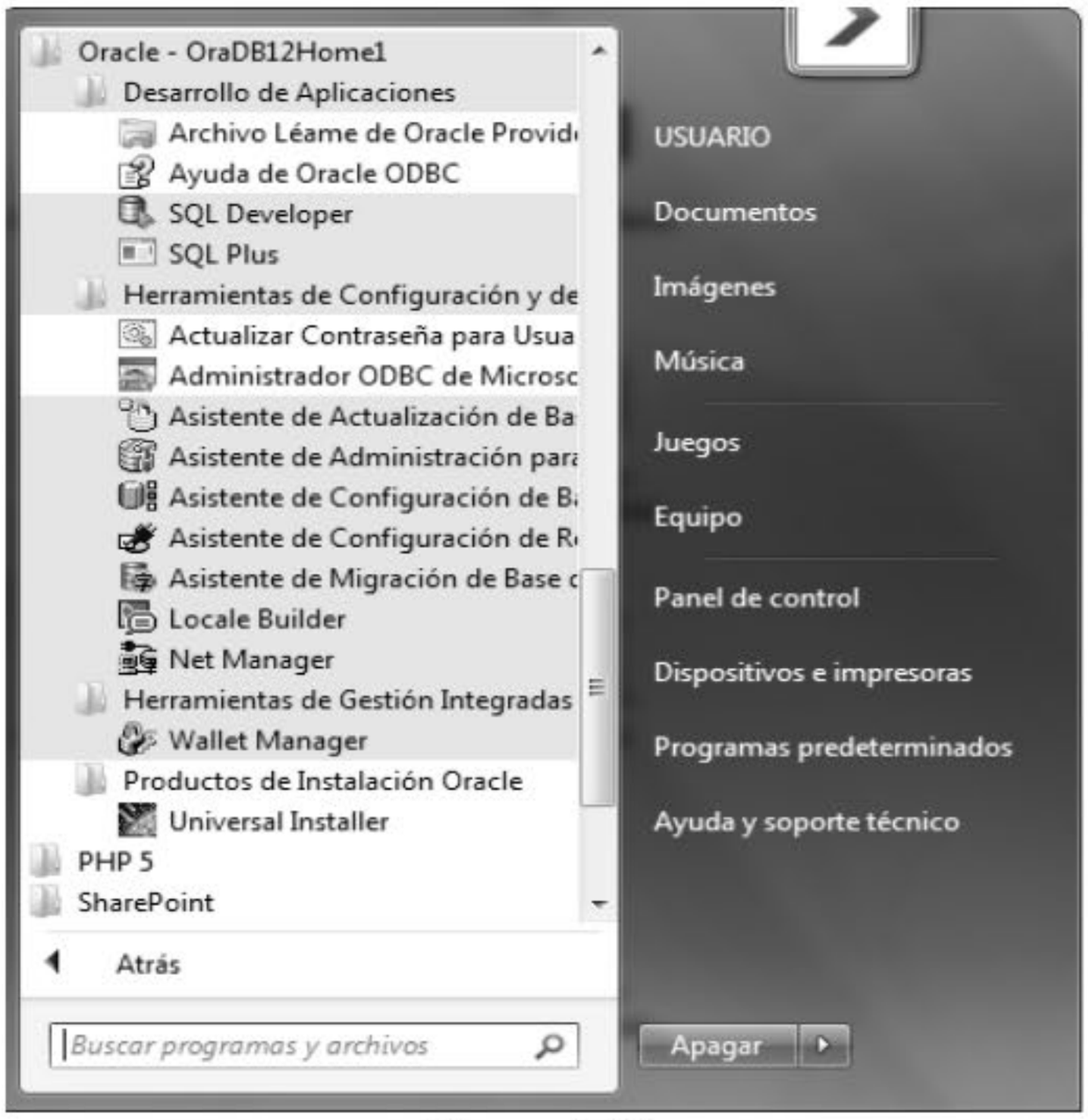


Figura 1-14

© Alfaomega - RC Libros

SYNAPS PREMIUM

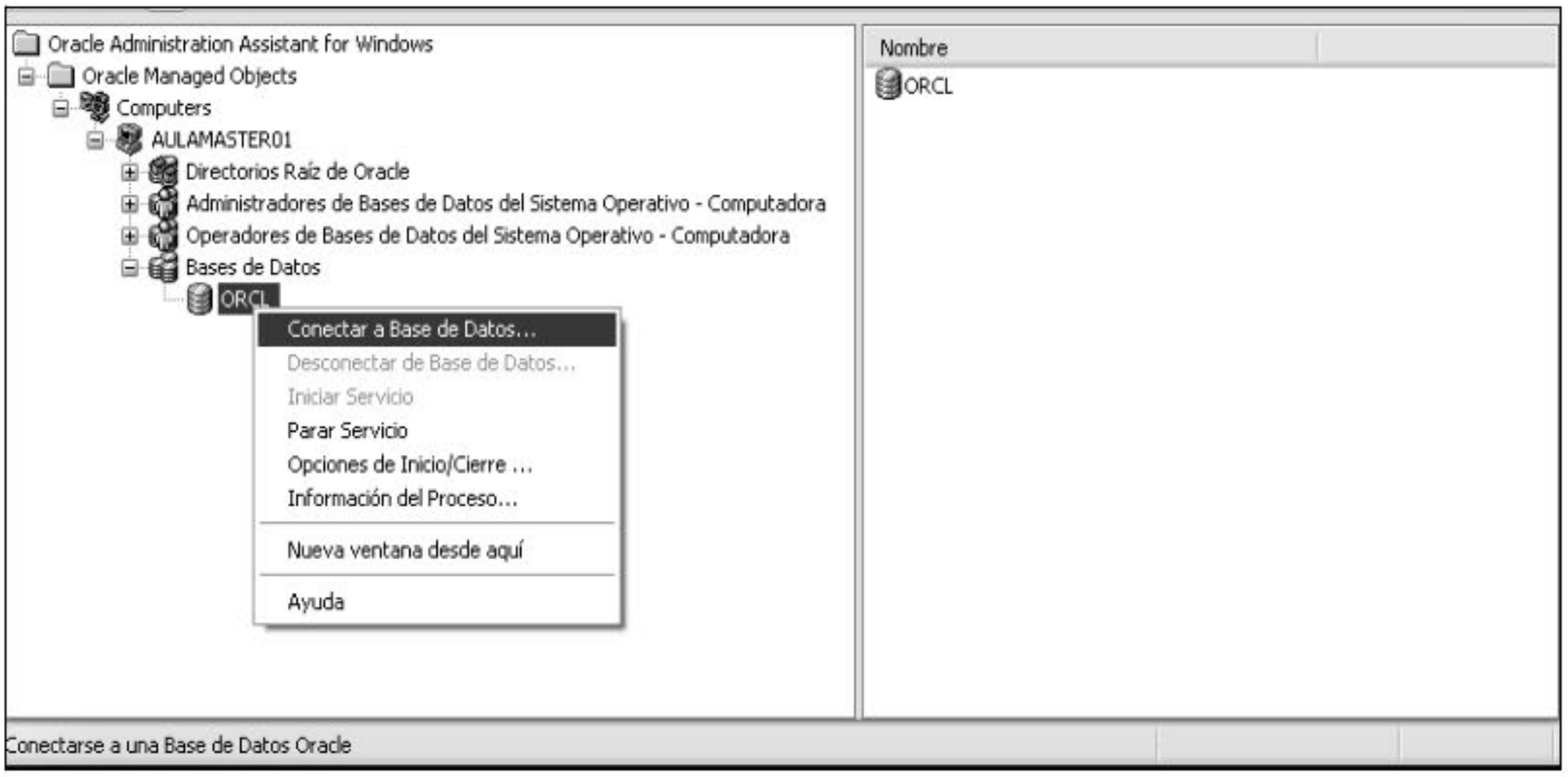


Figura 1-15

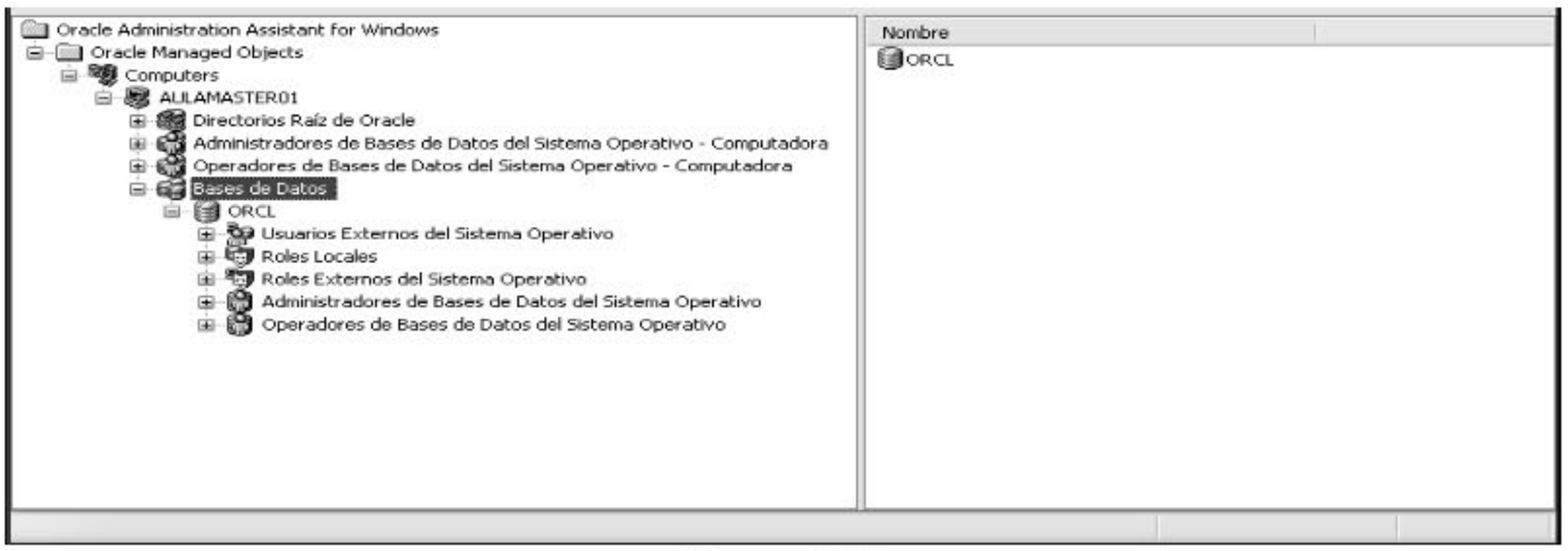


Figura 1-16

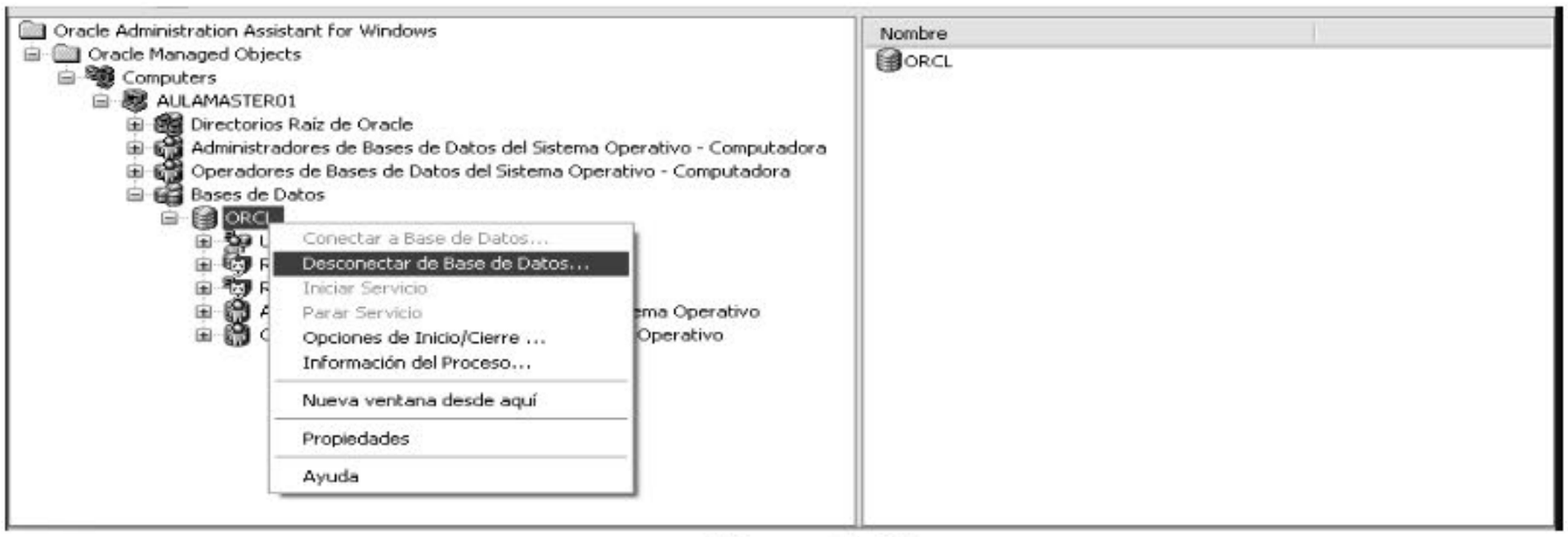


Figura 1-17

SYNAPS PREMIUM

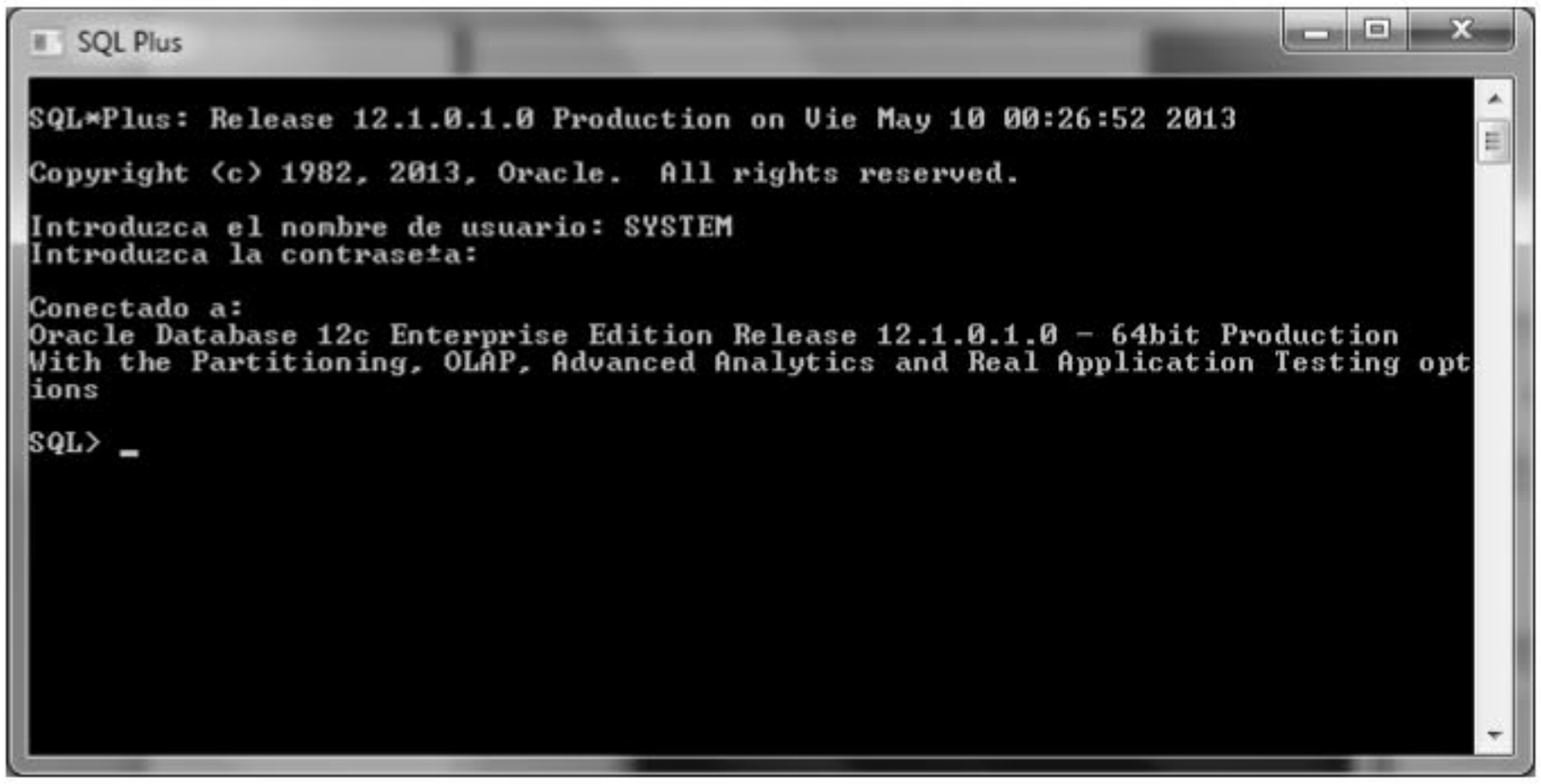


Figura 1-18

© Alfaomega - RC Libros

SYNAPS PREMIUM

CAPÍTULO 2

BASES DE DATOS RELACIONALES. DISEÑO BÁSICO. CREACIÓN DE TABLAS E INSERCIÓN DE REGISTROS CON SQL

EL MODELO RELACIONAL: CONCEPTOS BÁSICOS

El modelo relacional de base de datos fue propuesto por E.F. Codd en 1970. Sin embargo, los productos comerciales no aparecieron en el mercado hasta los primeros años de la década de los ochenta.

Las desventajas de los modelos jerárquico y en red condujeron a un intenso interés en el nuevo modelo de datos *relacional* cuando fue escrito por primera vez por el Dr. Codd. El modelo relacional era un intento de simplificar la estructura de las bases de datos. Eliminaba las estructuras explícitas padre/hijo de la base de datos, y en su lugar representaba todos los datos como sencillas tablas fila/columna de valores.

Una base de datos relacional ha de cumplir al menos dos objetivos. El primero es que el sistema debe presentar al usuario los datos en forma de tabla. Una sola fila en una tabla corresponde a un registro, y una columna en una tabla corresponde a un solo campo en un registro. De hecho, a menudo, utilizaremos con frecuencia los

SYNAPS PREMIUM

términos fila y registro como sinónimos. De la misma forma, emplearemos indistintamente los términos columna y campo. Hay que destacar que, a diferencia del anterior diseño de red, no existe un diagrama esquemático explícito entre las tablas. Esta observación podría conducir a la conclusión errónea de que una base de datos relacional es simplemente una presentación simplificada para el usuario del concepto clásico de archivo.

Sin embargo, esto no es así. La segunda característica importante de las bases de datos relacionales distingue claramente los sistemas de base de datos relacional de los sistemas de archivo y de los sistemas anteriores de base de datos tipo red.

Otra característica esencial de cualquier base de datos relacional es que acepte el lenguaje de consulta "set-at-a-time". Con los sistemas tradicionales, el lenguaje de consulta estaba orientado "record-at-a-time". Esto implicaba que el lenguaje de consulta solo era capaz de expresar una pregunta para obtener un solo registro. Por lo tanto, si el programador deseaba obtener un conjunto de registros, tenía que escribir la orden para ejecutar la pregunta varias veces. Por ejemplo, si quisiéramos obtener datos sobre todos los empleados del departamento de proceso de datos y existieran doscientos empleados que cumplieran esta condición, la consulta tendría que ser ejecutada 200 veces. Cada ejecución aportaría un registro y lo devolvería al programa de aplicaciones.

Con un sistema relacional el proceso es bastante más sencillo, ya que el lenguaje de consulta es mucho más potente. El lenguaje es suficientemente potente para pedir cualquier subconjunto de filas y columnas. Cuando el usuario asigna una pregunta, el sistema recoge todos los datos y los devuelve al usuario. El usuario realiza solo una pregunta (sin iteraciones múltiples) y el sistema devuelve todos los datos deseados.

Otro punto importante es que los lenguajes relacionales para los productos de bases de datos son relativamente fáciles de aprender. No es necesario ser un programador profesional para acceder a la base de datos. Un usuario iniciado en informática puede utilizar un terminal o un microordenador e introducir una orden en el lenguaje de consulta. El sistema extraerá las filas deseadas y hará que aparezcan en pantalla.

Se ha desarrollado un gran número de lenguajes de consulta de base de datos. El más conocido de estos lenguajes es el SQL. Se trata de un lenguaje de base de datos "set-at-a-time" simple pero muy completo.

© Alfaomega - RC Libros

SYNAPS PREMIUM

EL LENGUAJE SQL

El SQL (*Structured Query Language*) fue definido inicialmente por D. D. Chamberlin y otros durante la última parte de la década de los setenta. Al principio se le denominó SEQUEL (*Structured English Query Language*) y fue desarrollado junto con el prototipo de base de datos relacional de IBM: el System R. El System R es el predecesor del DB2 y del SQL/DS, mientras que el SEQUEL es el precursor del SQL.

En los últimos años, numerosas firmas comerciales han introducido en el mercado productos de bases de datos relacionales orientados al SQL. Las versiones anteriores de estos productos también estaban orientadas al SQL, pero eran bastante idiosincráticas, es decir, desconectadas o independientes. En respuesta a esta situación, el Instituto de Normalización Americano ANSI (*American National Standards Institute*) ha establecido una normativa oficial.

Esta normalización ha permitido que la mayoría de los productos actuales aceptan el *SQL embebido* (o inmerso). Esto permite que un programa de aplicaciones escrito en COBOL, FORTRAN, PL/I, C, Ensamblador u otro lenguaje de programación estructurado contenga código SQL. Esto significa que un sistema de proceso de datos con trabajo por lotes (batch) y/o con programas de transacción puede también interactuar con el sistema gestor de base de datos. El SQL interactivo y el SQL embebido son similares. Sin embargo, se pueden destacar algunas diferencias importantes. Por ejemplo, el SQL embebido debe identificar de alguna forma la posición en la memoria donde se van a emplazar los datos después de ser extraídos de la base de datos. Además, debido a que los lenguajes de programación tradicionales están orientados al registro, se requiere un código especial SQL para procesar una pregunta que devuelva múltiples filas.

La Figura 2.1 muestra cómo funciona SQL. El sistema informático de la figura tiene una base de datos que almacena información importante. Si el sistema informático estuviera en una empresa comercial, la base de datos podría almacenar datos de inventario, producción, ventas o nómina. En un computador personal, la base de datos podría almacenar datos referentes a los cheques que usted haya firmado, listas de personas y sus números de teléfono, o datos extraídos de un sistema informático mayor. El programa informático que controla la base de datos se denomina *sistema de gestión de base de datos (Database Management System)* o DBMS (en castellano SGBD).

© Alfaomega - RC Libros

SYNAPS PREMIUM

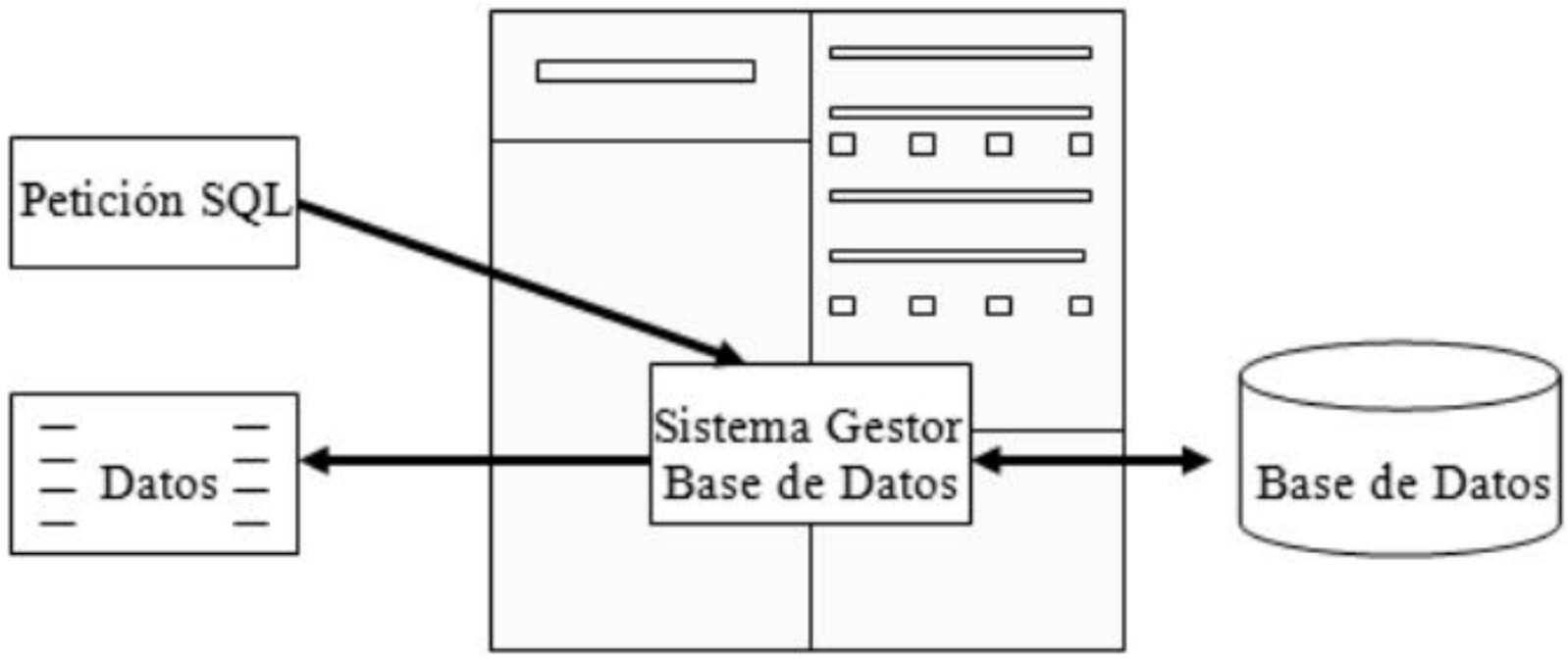


Figura 2-1

Cuando se necesita recuperar datos de la base de datos, se utiliza el lenguaje SQL para efectuar la petición. El DBMS procesa la petición SQL, recupera los datos solicitados de la base de datos y los devuelve. Este proceso de solicitar datos de la base de datos y de recibir los resultados se denomina *consulta (query) a la base de datos*; de aquí el nombre Structured *Query* Language. El nombre Structured Query Language es realmente y en cierta medida inapropiado. En primer lugar, SQL es mucho más que una herramienta de consulta, aunque ese fue su propósito original y recuperar datos sigue siendo una de sus funciones más importantes. SQL se utiliza para controlar todas las funciones que un DBMS proporciona a sus usuarios, incluyendo:

- **Definición de datos:** SQL permite a un usuario definir la estructura y organización de los datos almacenados y de las relaciones entre ellos.
- **Recuperación de datos:** SQL permite a un usuario o a un programa de aplicación recuperar los datos almacenados de la base de datos y utilizarlos.
- **Manipulación de datos:** SQL permite a un usuario o a un programa de aplicacion actualizar la base de datos añadiendo nuevos datos, suprimiendo datos antiguos y modificando datos previamente almacenados.
- **Control de acceso:** SQL puede ser utilizado para restringir la capacidad de un usuario para recuperar, añadir y modificar datos, protegiendo así los datos almacenados frente a accesos no autorizados.
- **Compartición de datos:** SQL se utiliza para coordinar la compartición de datos por parte de usuarios concurrentes, asegurando que no interfieren unos con otros.

© Alfaomega - RC Libros

SYNAPS PREMIUM

- **Integridad de datos:** SQL define restricciones de integridad en la base de datos, protegiéndola contra corrupciones debidas a actualizaciones inconsistentes o a fallos del sistema.

Por tanto, SQL es un lenguaje completo de control e interactuación con un sistema de gestión de base de datos.

En segundo lugar, SQL no es realmente un lenguaje informático completo como COBOL, FORTRAN o C. SQL no dispone de la sentencia IF para examinar condiciones, ni de la sentencia GOTO para bifurcaciones, ni de las sentencias DO o FOR para iteraciones. En vez de ello, SQL es un *sublenguaje* de base de datos, consistente en unas treinta sentencias especializadas para tareas de gestión de bases de datos. Estas sentencias SQL se *incorporan* a otro lenguaje (SQL embebido), como COBOL, FORTRAN o C para extender ese lenguaje y permitirle utilizar el acceso a la base de datos. Actualmente existe un lenguaje extensión de SQL denominado PLSQL, que incorpora a SQL los condicionales, las bifurcaciones, las iteraciones y otros elementos de los lenguajes de programación estructurados.

Finalmente, SQL no es un lenguaje particularmente estructurado, especialmente cuando se compara con lenguajes altamente estructurados tales como C o Pascal. En vez de ello, las sentencias SQL se asemejan a frases en inglés, completadas con "palabras de relleno" que no añaden nada al significado de la frase pero que hace que se lea más naturalmente. Hay unas cuantas inconsistencias en el lenguaje SQL, y también existen algunas reglas especiales para impedir la construcción de sentencias SQL que parecen perfectamente legales, pero que no tienen sentido. A pesar de la imprecisión de su nombre, SQL ha emergido como el lenguaje estándar para la utilización de bases de datos relacionales. SQL es a la vez un potente lenguaje y un lenguaje fácil de aprender.

Objetivos esenciales de SQL

SQL no es en sí mismo un sistema de gestión de base de datos ni un producto autónomo. No se puede ir a una tienda de informática y "comprar SQL". En su lugar, SQL es parte integral de un sistema de gestión de base de datos, un lenguaje y una herramienta para comunicarse con el DBMS. La Figura 2.2 muestra algunos de los componentes de un DBMS típico y cómo SQL actúa como el "adhesivo" que los une.

© Alfaomega - RC Libros

SYNAPS PREMIUM

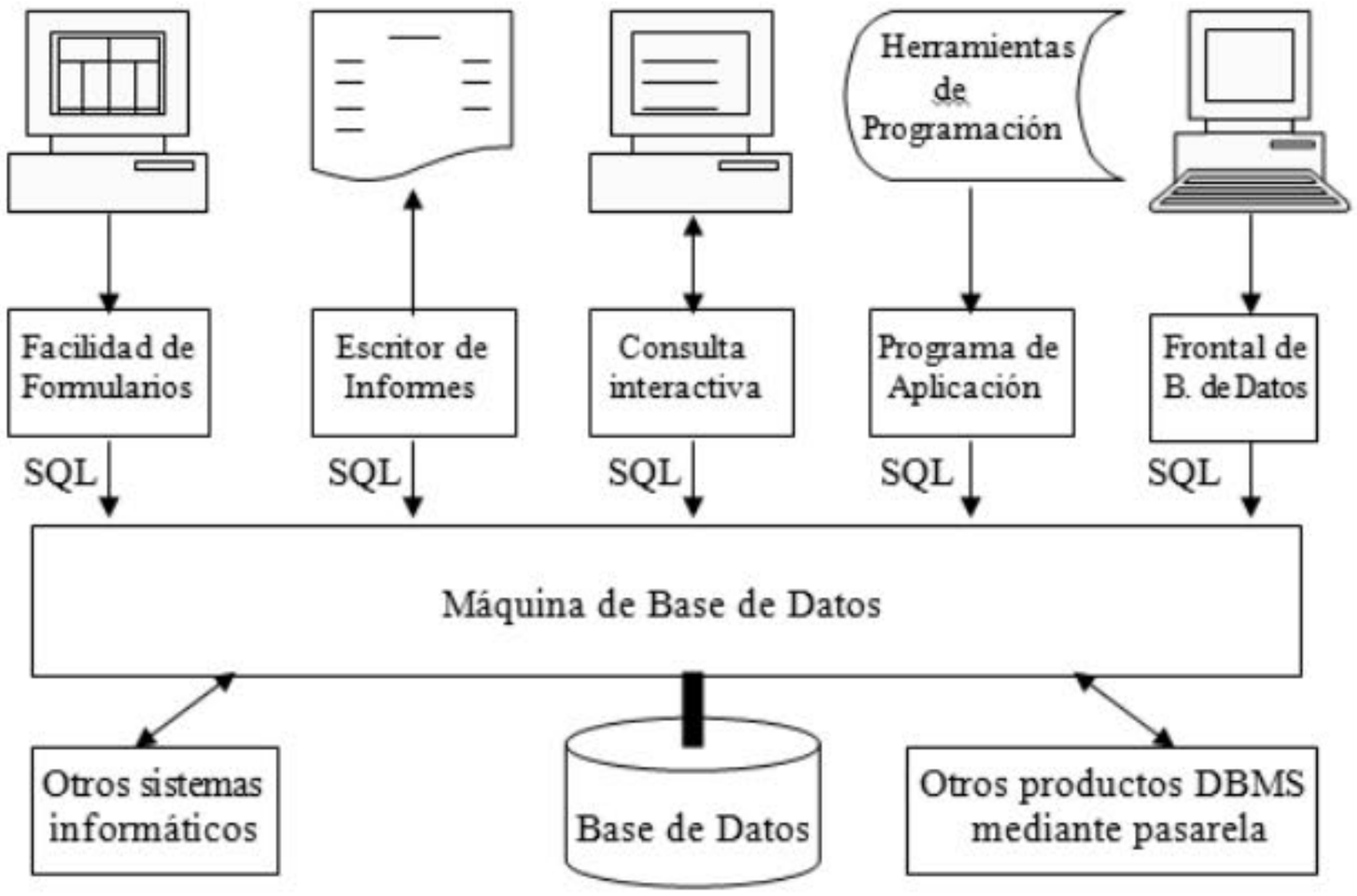


Figura 2-2

La máquina de base de datos es el corazón del DBMS, responsable de estructurar, almacenar y recuperar realmente los datos en el disco. Acepta peticiones SQL procedentes de otros componentes DBMS, tales como una facilidad de formularios, un escritor de informes o una facilidad de consultas interactivas, desde los programas de aplicación escritos por el usuario e incluso desde otros sistemas informáticos. Como muestra la figura, SQL juega muchos papeles diferentes. A saber:

- **SQL es un lenguaje de consultas interactivas:** los usuarios escriben órdenes SQL en un programa SQL interactivo para recuperar datos y mostrarlos en la pantalla, proporcionando una herramienta conveniente y fácil de utilizar para consultas *ad hoc* de la base de datos.

- **SQL es un lenguaje de programación de base de datos:** los programadores insertan órdenes SQL en sus programas de aplicación para acceder a los datos de la base. Tanto los programas escritos por el usuario como los programas de utilidad de la base de datos (tales como los escritores de informes y las herramientas de entrada de datos) utilizan esta técnica para acceso a la base de datos.

- **SQL es un lenguaje de administración de base de datos:** el administrador de la base de datos responsable de gestionar una base de datos en un minicomputador o en un maxicomputador utiliza SQL para definir la estructura de la base de datos y para controlar el acceso a los datos almacenados.

© Alfaomega - RC Libros

SYNAPS PREMIUM

- **SQL es un lenguaje cliente/servidor:** los programas de computador personal utilizan SQL para comunicarse sobre una red de área local con servidores de base de datos que almacenan los datos compartidos. Muchas aplicaciones novedosas están utilizando esta arquitectura de cliente/servidor, que minimiza el tráfico por la red y permite que tanto los PCs como los servidores efectúen mejor su trabajo.

- **SQL es un lenguaje de base de datos distribuidas:** los sistemas de gestión de base de datos distribuidos utilizan SQL para ayudar a distribuir datos a través de muchos sistemas informáticos conectados. El software DBMS de cada sistema utiliza SQL para comunicarse con los otros sistemas, enviando peticiones para acceso a datos.

- **SQL es un lenguaje de pasarela de base de datos:** en una red informática con una mezcla de diferentes productos DBMS, SQL se utiliza a menudo en una *pasarela (gateway)* que permite que nuestro producto de DBMS se comunique con otro producto.

SQL ha emergido, por tanto, como una herramienta potente y útil para enlazar personas y sistemas informáticos a los datos almacenados en una base de datos relacional.

Características de SQL

SQL es a la vez un lenguaje fácil de entender y una herramienta completa para gestionar datos. He aquí algunas de las principales características de SQL y las fuerzas del mercado que le han hecho tener éxito:

Independencia de los vendedores: SQL es ofertado por todos los principales vendedores de DBMS, y ningún producto nuevo de base de datos puede tener éxito sin el soporte de SQL. Una base de datos basada en SQL y los programas que la utilizan pueden transferirse de un DBMS al DBMS de otro vendedor con mínimo esfuerzo de conversión y poco reentrenamiento del personal. Herramientas de base de datos basadas en SQL que funcionan con varios productos de DBMS, tales como escritores de informe y generadores de aplicación, están presentes en el mercado. La independencia del vendedor proporcionada de esta manera por SQL es una de las razones más importantes de su popularidad.

Portabilidad a través de sistemas informáticos: los vendedores de DBMS en SQL ofertan sus productos sobre sistemas informáticos que van desde computadores personales y estaciones de trabajo hasta redes de área local, minicomputadores y

SYNAPS PREMIUM

maxicomputadores. Las aplicaciones basadas en SQL que comienzan en sistemas monousuario pueden ser transferidas a sistemas mayores de minicomputadores o maxicomputadores cuando crecen. Los datos procedentes de bases de datos corporativas basadas en SQL pueden ser extraídas y remitidas a bases de datos departamentales o personales. Finalmente, los económicos computadores personales pueden ser utilizados para construir prototipos de aplicaciones de bases de datos basadas en SQL antes de transferirlas a un sistema multiusuario costoso.

Estándares SQL: el *American National Standards Institute* (ANSI) y la *International Standards Organization* (ISO) han publicado conjuntamente un estándar oficial para SQL. Este se ha convertido también en un estándar del U.S. *Federal Information Processing Standard* (FIPS), lo que le convierte en un requerimiento esencial para los grandes contratos informáticos del gobierno. En Europa, X/OPEN, un estándar para un entorno de aplicación por cable basado en UNIX, ha añadido SQL como el estándar para acceso a base de datos. La *Open Software Foundation* (OSF), un grupo de vendedores de UNIX, planea basar su estándar de acceso a base de datos en SQL. Estos estándares sirven como sello oficial de aprobación para SQL y han acelerado su aceptación en el mercado.

Fundamento relacional: SQL es un lenguaje para base de datos relacionales, y se ha popularizado juntamente con el modelo de base de datos relacional. La estructura tabular de filas y columnas de una base de datos relacional es intuitiva para los usuarios y hace que el lenguaje SQL se mantenga simple y fácil de entender. El modelo relacional tiene también un fuerte fundamento teórico que ha guiado la evolución y la implementación de las bases de datos relacionales. Cabalgando sobre una ola de aceptación provocada por el éxito del modelo relacional, SQL se ha convertido en el lenguaje de base de datos para las bases de datos relacionales.

Claridad y sencillez de sintaxis: las sentencias SQL parecen sencillas frases en inglés, lo que hace que SQL sea fácil de aprender y entender. Esto es en parte debido a que las sentencias de SQL describen los datos a recuperar, en lugar de especificar cómo hallar los datos. Las tablas y columnas de una base de datos SQL pueden tener nombres largos y descriptivos. Como consecuencia, la mayoría de las sentencias SQL "dicen lo que significan" y pueden ser leídas como frases claras y naturales.

Consultas interactivas ad hoc: SQL es un lenguaje de consultas interactivas que proporciona a los usuarios acceso *ad hoc* a los datos almacenados. Utilizando SQL interactivamente, un usuario puede obtener respuestas incluso a cuestiones complejas en minutos o segundos, en fuerte contraste con los días o semanas que le

© Alfaomega - RC Libros

SYNAPS PREMIUM

llevaría a un programador escribir un programa de informe a medida. Debido a la potencia de consulta *ad hoc* de SQL, los datos son más accesibles y pueden ser utilizados para ayudar a una organización a tomar decisiones mejores y más informadas.

Acceso a la base de datos mediante programas: SQL es también un lenguaje de base de datos utilizado por los programadores para escribir aplicaciones que acceden a una base de datos. Las mismas sentencias SQL se utilizan para acceso interactivo y programado, de modo que las partes de acceso a base de datos de un programa pueden ser comprobadas primero con SQL interactivo y luego insertadas dentro del programa. En contraste, las bases de datos tradicionales proporcionan un conjunto de herramientas para acceso mediante programas y una facilidad de consulta aparte para peticiones *ad hoc*, sin ninguna sinergia entre los dos modos de acceso.

Vistas múltiples de datos: utilizando SQL, el creador de una base de datos puede dar a diferentes usuarios de la base de datos distintas vistas (diferentes en su estructura y contenido). Por ejemplo, la base de datos puede ser construida de modo que cada usuario solo vea datos de su propio departamento o de su propia región de ventas. Además, los datos procedentes de diferentes partes de la base de datos pueden combinarse y presentarse al usuario como una simple fila/columna de una tabla. Las vistas de SQL pueden ser utilizadas de este modo para mejorar la seguridad de una base de datos y para acomodarla a las necesidades particulares de los usuarios individuales.

Lenguaje completo de base de datos: SQL fue inicialmente desarrollado como un lenguaje de consulta *ad hoc*, pero su potencia va ahora más allá de la recuperación de datos. SQL proporciona un lenguaje complejo y consistente para crear una base de datos, gestionar su seguridad, actualizar sus contenidos, recuperar los datos y compartirlos entre muchos usuarios concurrentes. Los conceptos de SQL que son aprendidos en una parte del lenguaje pueden ser aplicados a otras órdenes de SQL, haciendo más productivos a sus usuarios.

Definición dinámica de datos: utilizando SQL, la estructura de una base de datos puede ser modificada y ampliada dinámicamente, incluso mientras los usuarios están accediendo a los contenidos de la base de datos. Este es un avance importante sobre los lenguajes de definición de datos estáticos, que impiden el acceso a la base de datos mientras su estructura está siendo modificada. SQL proporciona de este modo: máxima flexibilidad, permitiendo que una base de datos se adapte a exigencias cambiantes mientras continúan sin ser interrumpidas las aplicaciones en línea.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Arquitectura cliente/servidor: SQL es un vehículo natural para implementar aplicaciones utilizando una arquitectura cliente/servidor distribuida. En este papel, SQL sirve como enlace entre los sistemas informáticos "frontales" *(front-end)* optimizados para interacción con el usuario y los sistemas "de apoyo" *(back-end)* especializados para gestión de bases de datos, permitiendo que cada sistema rinda lo mejor posible. SQL también permite que los computadores personales funcionen como frontales de bases de datos mayores, dispuestas en minicomputadores y maxicomputadores, proporcionando acceso a datos corporativos desde aplicaciones informáticas personales.

Estas son esencialmente las razones por las que SQL ha emergido como la herramienta estándar para gestionar datos en computadores personales, minicomputadores y maxicomputadores.

DISEÑO DE UNA BASE DE DATOS RELACIONAL

Lo primero que hay que tener presente a la hora de diseñar una base de datos relacional es el propio concepto de modelo relacional, que organiza los datos en una base de datos como una colección de tablas teniendo presente inicialmente lo siguiente:

- Cada tabla tiene un nombre que la identifica unívocamente.
- Cada tabla tiene una o más columnas nominadas, que están dispuestas en un orden específico de izquierda a derecha.
- Cada tabla tiene cero o más filas, conteniendo cada una un único valor en cada columna. Las filas están desordenadas.
- Todos los valores de una columna determinada tienen el mismo tipo de datos, y estos están extraídos de un conjunto de valores legales llamado el dominio de la columna.

A su vez, las tablas están relacionadas unas con otras por los datos que contienen. El modelo de datos relacional utiliza claves primarias y claves secundarias (externas o foráneas) para representar estas relaciones entre tablas. A la hora de definir las claves primarias y secundarias es necesario tener presente de entrada lo siguiente:

- Una clave primaria es una columna o combinación de columnas dentro de una tabla cuyo (s) valor (es) identifica (n) unívocamente a cada fila de la tabla. Cada tabla tiene una única clave primaria.

© Alfaomega - RC Libros

SYNAPS PREMIUM

- Una clave secundaria es una columna o combinación de columnas en una tabla cuyo (s) valor (es) es (son) un valor de clave primaria para alguna otra tabla. Una tabla puede contener más de una clave secundaria, enlazándola a una o más tablas.

- Una combinación clave primaria/clave secundaria crea una relación padre/hijo entre las tablas que las contienen.

En el capítulo anterior ya se han tratado de modo amplio todos estos conceptos de forma aislada. No obstante, en este capítulo veremos cómo se trabaja de modo conjunto con todos estos conceptos para diseñar ordenadamente una base de datos.

Las reglas de Codd

Ted Codd presentó en 1985 en un artículo publicado en *Computerworld* las doce reglas que una base de datos debe obedecer para ser considerada verdaderamente como relacional. Estas reglas se han convertido desde entonces en una definición semioficial de una base de datos relacional. Las citadas reglas son las siguientes:

Regla de información: toda la información de una base de datos relacional está representada explícitamente a nivel lógico y exactamente de un modo mediante valores en tablas.

Regla de acceso garantizado: todos y cada uno de los datos (valores atómicos) de una base de datos relacional se garantiza que sean lógicamente accesibles recurriendo a una combinación de nombre de tabla, valor de clave primaria y nombre de columna.

Tratamiento sistemático de valores nulos: los valores nulos (distintos de la cadena de caracteres vacía o de una cadena de caracteres en blanco y distinta del cero o de cualquier otro número) se soportan en los DBMS completamente relacionales para representar la falta de información y la información inaplicable de un modo sistemático e independiente del tipo de datos.

Catálogo en línea dinámico basado en el modelo relacional: la descripción de la base de datos se representa a nivel lógico del mismo modo que los datos ordinarios, de modo que los usuarios autorizados puedan aplicar a su interrogación el mismo lenguaje relacional que aplican a los datos regulares.

Regla de sublenguaje completo de datos: un sistema relacional puede soportar varios lenguajes y varios modos de uso terminal (por ejemplo, el modo de rellenar con blancos). Sin embargo, debe haber al menos un lenguaje cuyas sentencias sean

© Alfaomega - RC Libros

SYNAPS PREMIUM

expresables mediante alguna sintaxis bien definida, como cadenas de caracteres, y que sea completa en cuanto al soporte de todos los puntos siguientes:

- Definición de datos.
- Definición de vista.
- Manipulación de datos (interactiva y por programa).
- Restricciones de integridad.
- Autorización.
- Fronteras de transacciones (comienzo, cumplimentación y vuelta atrás).

Regla de actualización de vista: todas las vistas que sean teóricas actualizables son también actualizables por el sistema.

Inserción, actualización y supresión de alto nivel: la capacidad de manejar una relación de base de datos o una relación derivada como un único operando se aplica no solo a la recuperación de datos, sino también a la inserción, actualización y supresión de los datos.

Independencia física de los datos: los programas de aplicación y las actividades terminales permanecen lógicamente inalterados cualesquiera que sean los cambios efectuados; ya sea a las representaciones de almacenamiento o a los métodos de acceso.

Independencia lógica de los datos: los programas de aplicación y las actividades terminales permanecen lógicamente inalterados cuando se efectúen sobre las tablas de base cambios preservadores de la información de cualquier tipo que teóricamente permita alteraciones.

Independencia de integridad: las restricciones de integridad específicas para una base de datos relacional particular deben ser definibles en el sublenguaje de datos relacional y almacenables en el catálogo, no en los programas de aplicación.

Independencia de distribución: un DBMS relacional tiene independencia de distribución.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Regla de no subversión: si un sistema relacional tiene un lenguaje de bajo nivel (un solo registro cada vez), ese bajo nivel no puede ser utilizado para subvertir o suprimir las reglas de integridad y las restricciones expresadas en el lenguaje relacional de nivel superior (múltiples registros a la vez).

Como comentarios a las doce reglas de Codd podemos empezar diciendo que es muy difícil que un DBMS satisfaga al pie de la letra todas las reglas. No obstante, se intenta aproximarse lo más posible a su contenido.

La Regla 1 es básicamente la definición informal de una base de datos relacional presentada ya anteriormente.

La Regla 2 refuerza la importancia de las claves primarias para localizar datos en la base de datos. El nombre de la tabla localiza la tabla correcta, el nombre de la columna encuentra la columna correcta y el valor de clave primaria encuentra la fila que contiene un dato individual de interés.

La Regla 3 requiere soporte para falta de datos mediante el uso de valores NULL.

La Regla 4 requiere que una base de datos relacional sea autodescriptiva. En otras palabras, la base de datos debe contener ciertas *tablas de sistema* cuyas columnas describan la estructura de la propia base de datos.

La Regla 5 ordena la utilización de un lenguaje de base de datos relacional, tal como SQL, aunque no se requiera específicamente SQL. El lenguaje debe ser capaz de soportar todas las funciones básicas de un DBMS (creación de una base de datos, recuperación y entrada de datos, implementación de la seguridad de la base de datos, etc.).

La Regla 6 trata de las vistas, que son *tablas virtuales* utilizadas para dar a diferentes usuarios de una base de datos distintas vistas de su estructura. Es una de las reglas más difíciles de implementar en la práctica, y ningún producto comercial suele satisfacerla totalmente hoy día.

La Regla 7 refuerza la naturaleza orientada a conjuntos de una base de datos relacional. Requiere que las filas sean tratadas como conjuntos en operaciones de inserción, supresión y actualización. La regla está diseñada para prohibir implementaciones que solo soportan la modificación o el recorrido fila a fila de la base de datos.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Las Reglas 8 y 9 aíslan al usuario o al programa de aplicación de la implementación de bajo nivel de la base de datos. Especifican que las técnicas de acceso a almacenamiento específicas utilizadas por el DBMS, e incluso los cambios a la estructura de las tablas en la base de datos, no deberían afectar a la capacidad del usuario de trabajar con los datos.

La Regla 10 dice que el lenguaje de base de datos debería soportar las restricciones de integridad que restringen los datos que pueden ser introducidos en la base de datos y las modificaciones que puedan ser efectuadas en ésta. Esta es otra de las reglas que no soportan la mayoría de los productos comerciales DBMS.

La Regla 11 dice que el lenguaje de base de datos debe ser capaz de manipular datos distribuidos localizados en otros sistemas informáticos.

Finalmente, la Regla 12 impide "otros caminos" en la base de datos que pudieran subvertir su estructura relacional y su integridad.

Integridad

El término *integridad de datos* se refiere a la corrección y completitud de los datos en una base de datos. Cuando los contenidos de una base de datos se modifican con sentencias INSERT (insertar), DELETE (borrar) o UPDATE (actualizar), la integridad de los datos almacenados puede perderse de muchas maneras diferentes. Por ejemplo:

- Pueden añadirse datos no válidos a la base de datos, tales como un pedido que especifica un producto no existente.
- Pueden modificarse datos existentes tomando un valor incorrecto, como por ejemplo, si se reasigna un vendedor a una oficina no existente.
- Los cambios a la base de datos pueden perderse debido a un error del sistema o a un fallo en el suministro de potencia.
- Los cambios pueden ser aplicados parcialmente, como por ejemplo si se añade un pedido de un producto sin ajustar la cantidad disponible para vender.

Una de las funciones importantes de un DBMS relacional es preservar la integridad de los datos almacenados en la mayor medida posible.

© Alfaomega - RC Libros

SYNAPS PREMIUM

¿Qué es la integridad de datos?

Para preservar la consistencia y corrección de los datos almacenados, un DBMS relacional impone generalmente una o más *restricciones de integridad de datos.* Estas restricciones restringen los valores que pueden ser insertados en la base de datos o creados mediante una actualización de la base de datos. Varios tipos diferentes de restricciones de integridad de datos suelen encontrarse en las bases de datos relacionales, incluyendo:

Datos requeridos: algunas columnas en una base de datos deben contener un valor de dato válido en cada fila, es decir, no se permite que contengan valores NULL o que falten. Pueden especificarse columnas requeridas cuando se crea una tabla, en cuyo caso, el DBMS impedirá los valores NULL en esas columnas.

Chequeo de validez: cada columna de una base de datos tiene un *dominio,* o sea, un conjunto de valores que son legales para esa columna. El DBMS puede ser preparado para impedir otros valores de datos en estas columnas.

Integridad de entidad: la clave primaria de una tabla debe contener un valor único en cada fila, diferente de los valores de todas las filas restantes. Los valores duplicados son ilegales y el DBMS puede ser preparado para forzar esta restricción de valores únicos. Las restricciones de integridad de entidad aseguran que la clave primaria identifique unívocamente a cada entidad representada en la base de datos.

Integridad referencial: una clave secundaria (externa o foránea) en una base de datos relacional enlaza cada fila de la tabla hijo que contiene la clave foránea con la fila de la tabla padre que contiene el valor de clave primaria correspondiente. El DBMS puede ser preparado para forzar esta restricción de clave foránea/clave primaria. Las restricciones de integridad referencial aseguran que las relaciones entre entidades en la base de datos se preserven durante las actualizaciones. En particular la integridad referencial debe incluir reglas que indiquen cómo manejar la supresión de filas que son referenciadas mediante otras filas.

Reglas comerciales: las actualizaciones de una base de datos pueden estar restringidas por reglas comerciales que gobiernan las transacciones en el mundo real que están representadas por las actualizaciones. El DBMS puede ser preparado para comprobar cada nueva fila añadida a cada tabla asegurándose que el valor en la columna no viole las reglas comerciales. Estas reglas comerciales pueden ser forzadas por el DBMS mediante los **disparadores**, que permiten emprender acciones complejas en respuesta a eventos tales como sentencias INSERT (insertar), DELETE (borrar) o UPDATE (actualizar).

SYNAPS PREMIUM

Consistencia: muchas transacciones del mundo real producen múltiples actualizaciones a una base de datos. Las sentencias INSERT y UPDATE deben todas ejecutarse, de modo que la base de datos continúe en un estado correcto y consistente. El DBMS puede ser preparado para forzar este tipo de regla de consistencia o para soportar aplicaciones que implementen tales reglas.

El estándar SQL ANSI/ISO especifica algunas de las restricciones de integridad de datos más simples. Por ejemplo, la restricción de datos requeridos está soportada por el estándar ANSI/ISO e implementada de manera uniforme en casi todos los productos SQL comerciales. Las restricciones más complejas, tales como las restricciones de reglas comerciales, no están especificadas por el estándar ANSI/ISO y existe una amplia variación en las técnicas y las sintaxis SQL utilizadas para soportarlas.

Problemas de integridad referencial

Existen cuatro tipos de actualizaciones de bases de datos que pueden corromper la integridad referencial de las relaciones padre/hijo de una base de datos.

La inserción de una nueva fila hijo: cuando se inserta una nueva fila en la tabla hijo, su valor de clave foránea debe coincidir con uno de los valores de clave primaria en la tabla padre. Si el valor de clave foránea no coincide con ninguna clave primaria, la inserción de la fila corromperá la base de datos, ya que habrá un hijo sin un padre (un "huérfano"). Observe que insertar una fila en la tabla padre nunca representa un problema; simplemente se convierte en un padre sin hijos.

La actualización de la clave foránea en una fila hijo: esta es una forma diferente del problema anterior. Si la clave foránea se modifica mediante una sentencia UPDATE, el nuevo valor deberá coincidir con un valor de clave primaria en la tabla padre. En caso contrario la fila actualizada será huérfana.

La supresión de una fila padre: si una fila de la tabla padre que tiene uno o más hijos se suprime, las filas hijo quedarán huérfanas. Los valores de clave foránea en estas filas ya no se corresponderán con ningún valor de clave primaria en la tabla padre. Observe que suprimir una fila de la tabla hijo nunca representa un problema; el padre de esta fila simplemente tendrá un hijo menos después de la supresión.

La actualización de la clave primaria en una fila padre: esta es una forma diferente del problema anterior. Si la clave primaria de una fila en la tabla padre se modifica,

© Alfaomega - RC Libros

SYNAPS PREMIUM

todos los hijos actuales de esa fila quedarán huérfanos, puesto que sus claves foráneas ya no corresponden con ningún valor de clave primaria.

Las características de integridad referencial del estándar SQL ANSI/ISO se ocupan de cada una de estas cuatro situaciones. El primer problema (INSERT en la tabla hijo) se maneja comprobando los valores de las columnas de clave foránea antes de permitir ejecutar la sentencia INSERT. Si no coincide con un valor de clave primaria, la sentencia INSERT se rechazará con un mensaje de error.

El segundo problema (UPDATE de la tabla hijo) es análogamente tratado mediante comprobación del valor de clave foránea actualizado. Si no hay un valor de clave primaria correspondiente, la sentencia UPDATE se rechaza acompañada de un mensaje de error.

El tercer problema (DELETE de una fila padre) es más complejo, pudiendo tomarse distintas medidas según cada caso particular.

El cuarto problema (UPDATE de la clave primaria en la tabla padre) también se maneja por prohibición. Antes de que la clave primaria pueda ser modificada, el DBMS efectúa comprobaciones para asegurar que no hay filas hijo que tengan valores de clave foránea correspondientes. Si existen tales filas hijo, la sentencia UPDATE se rechazará junto con un mensaje de error.

Ciclos referenciales

Supongamos que en una base de datos la tabla T1 contiene la columna C1 que es clave foránea para la tabla T2. Supongamos además que la tabla T2 contiene la columna C2 que es clave foránea para la tabla T1. Estas dos relaciones forman un *ciclo referencial*. En esta situación, cualquier fila determinada de la tabla T1 se refiere a una fila de la tabla T2, la cual a su vez se refiere a una fila de la tabla T1, y así sucesivamente. Este ciclo afecta únicamente a dos tablas, pero también es posible construir ciclos de tres o más tablas.

Con independencia de su número de tablas implicadas, los ciclos referenciales presentan problemas especiales respecto a las restricciones de integridad referencial. Para impedir interbloqueo de inserción, al menos una de las claves foráneas en un ciclo referencial debe permitir valores NULL. Debido a esta circunstancia sería conveniente que las restricciones de integridad referencial no fueran comprobadas hasta después de que se efectuaran una serie de actualizaciones interrelacionadas. Los ciclos referenciales también restringen las reglas de supresión que pueden

© Alfaomega - RC Libros

SYNAPS PREMIUM

especificarse para las relaciones que forman el ciclo. Se tratará de evitar las *supresiones en cascada* incontroladas mediante la adopción de determinadas reglas de supresión (RESTRICT o SET NULL).

Disparadores

El concepto de disparador (definido inicialmente por Sybase) es relativamente sencillo. Para cualquier evento que provoca un cambio en el contenido de una tabla, un usuario puede especificar una acción asociada que el DBMS debería efectuar automáticamente a la realización del evento. Los tres eventos que pueden disparar una acción automática asociada son ejecuciones de INSERT, DELETE o UPDATE sobre filas de la tabla. La acción disparada por uno de estos eventos se especifica mediante una secuencia de sentencias SQL.

Los disparadores proporcionan un modo alternativo de implementar las restricciones de integridad referencial proporcionadas por claves foráneas y claves primarias. Incluso los disparadores pueden ser utilizados para proporcionar formas ampliadas de integridad referencial no ofrecidas por el estándar SQL ANSI/ISO.

La principal ventaja de los disparadores es que las reglas comerciales pueden almacenarse en las bases de datos y ser forzadas consistentemente con cada actualización de la base de datos. Esto puede reducir sustancialmente la complejidad de los programas de aplicación que acceden a la base de datos. Los disparadores tienen también algunas desventajas, siendo las más importantes las siguientes:

Complejidad de la base de datos: cuando las reglas se trasladan al interior de la base de datos, preparar la base de datos pasa a ser una tarea más compleja. Los usuarios que razonablemente podían esperar crear pequeñas aplicaciones propias con SQL, encontrarán que la lógica de programación de los disparadores hace la tarea mucho más difícil.

Reglas ocultas: con las reglas ocultas dentro de la base de datos, programas que parecen efectuar sencillas actualizaciones de la base de datos pueden, de hecho, generar una cantidad enorme de actividad en la base de datos. En este caso, el programador ya no controla totalmente lo que sucede en la base de datos.

Índices

Un índice es una estructura interna que el sistema puede usar para encontrar uno o más registros en una tabla de forma rápida. En efecto, un índice de base de datos es, conceptualmente, similar a un índice encontrado al final de cualquier libro de

© Alfaomega - RC Libros

SYNAPS PREMIUM

texto. De la misma forma que el lector de un libro acudiría a un índice para determinar en qué páginas se encuentra un determinado tema, un sistema de base de datos leerá un índice para determinar las posiciones de registros seleccionados por una consulta SQL. En otras palabras, la presencia de un índice puede ayudar al sistema a procesar algunas consultas de un modo más eficiente.

Un índice de base de datos se crea para una columna o grupo de columnas. La Figura 2.3 muestra un índice (XCNOMBRE) basado en la columna CNOMBRE de la tabla CURSO definida en el capítulo anterior. Observemos que el índice, a diferencia de la tabla CURSO, representa valores CNOMBRE en orden. Además, el índice es pequeño en relación con el tamaño de la tabla. Por lo tanto, es, probablemente, más fácil que el sistema busque el índice para localizar un registro con un valor CNOMBRE dado, a que explore toda la tabla en busca de ese valor. Por ejemplo, el índice XCNOMBRE podría ser muy útil al sistema cuando ejecute la siguiente sentencia SELECT:

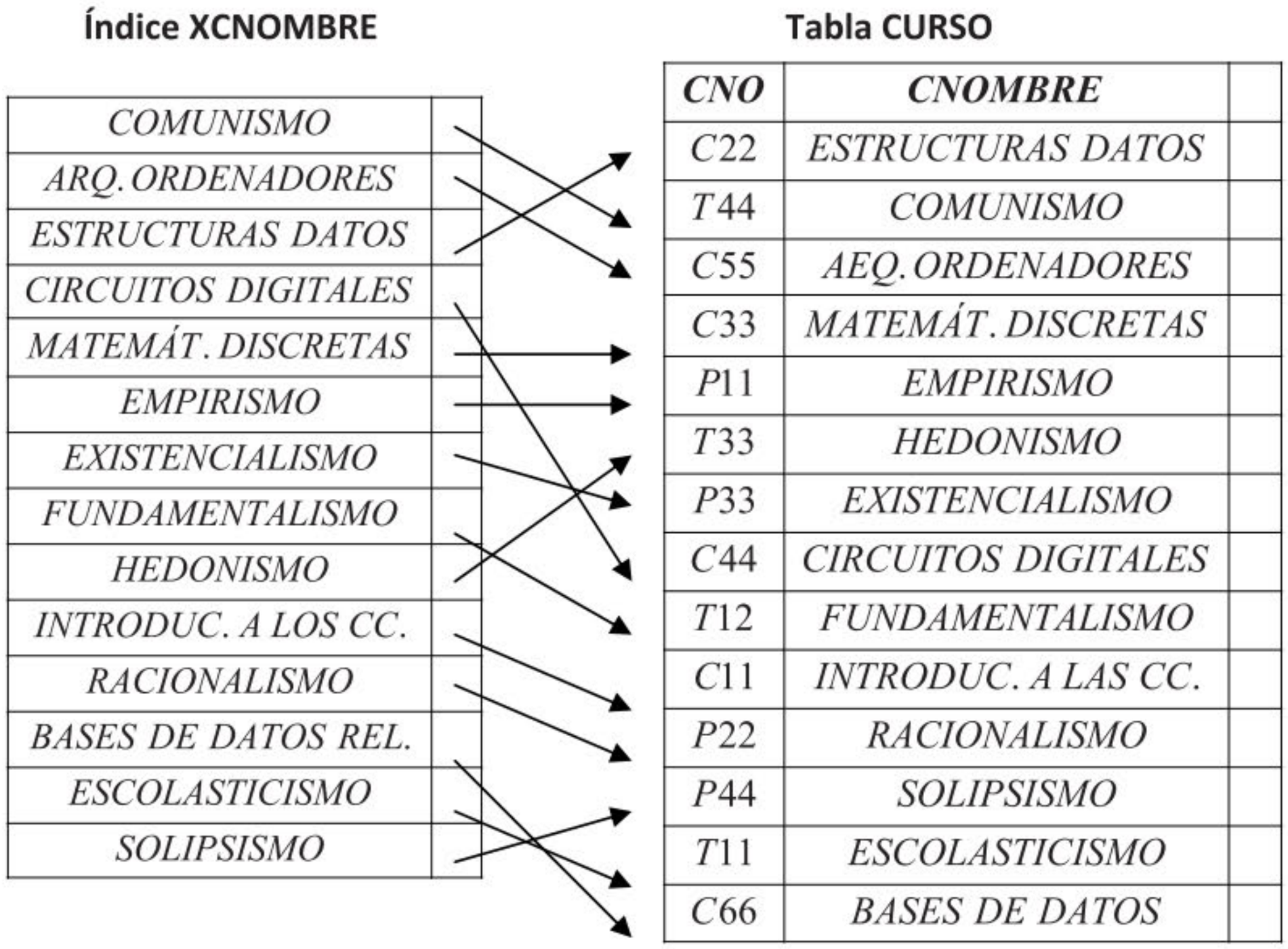


Figura 2-3

A un índice se le asigna un nombre de acuerdo con las mismas reglas que se aplican para los nombres de tablas. El nombre debe comenzar con una letra que puede ir seguida de una o más letras, dígitos o caracteres de subrayado. Su longitud

© Alfaomega - RC Libros

SYNAPS PREMIUM

no puede exceder de un determinado número de caracteres. Suele seguirse una convención común de elegir "X" como la primera letra seguida del nombre de columna para formar el nombre del índice. Sin embargo, esta convención no es indispensable para el sistema. Para crear un índice suele utilizarse la sentencia SQL CREATE INDEX, tal y como veremos a la hora de analizar la sintaxis SQL.

En el ejemplo de la Figura 2-3, la tabla CURSO ya tiene registros almacenados en ella cuando se crea el índice. El sistema explorará esta tabla para obtener los valores CNOMBRE y las posiciones de los registros correspondientes para construir el índice. En la práctica, es mejor asignar la sentencia CREATE INDEX antes de que se inserte ningún registro en la tabla. Sin embargo, como ilustra el ejemplo, puede crearse un índice para una tabla que ya contenga registros.

Una vez que se ha creado el índice, el sistema lo mantendrá automáticamente. Por ejemplo, si asignamos una sentencia INSERT para colocar un nuevo registro en la tabla CURSO, el sistema determinará automáticamente el valor CNOMBRE y la posición del nuevo registro, y colocará una nueva entrada en el índice XCNOMBRE. En otras palabras, una vez que se crea el índice, el sistema hace el resto.

Al igual que la sentencia CREATE TABLA, CREATE INDEX producirá actualización del catálogo del sistema. En este caso, el sistema registrará la presencia de un nuevo índice (XCNOMBRE) y su dependencia de la tabla CURSO.

No se produce ningún cambio en la codificación de una sentencia de selección de registros (SELECT) de una tabla, debido a la presencia o ausencia de un índice. La sintaxis de todas las sentencias SELECT permanece igual. Por lo tanto, la mayoría de los usuarios del sistema puede que no perciban la existencia de índices de base de datos. Siempre que una sentencia SELECT sea introducida por el usuario el sistema usa un módulo interno especial llamado optimizador, el cual analiza la sentencia para determinar si el uso de un índice sería beneficioso. Si este es el caso, el sistema consultará el cátalogo para determinar si se encuentra disponible un índice. Si lo está, el sistema utilizará el índice. En caso contrario, el sistema realizará una operación de exploración en busca de los registros deseados.

El uso de un *optimizador* para determinar el camino de acceso más eficaz significa que el sistema posee la importante característica de la independencia física de los datos. El administrador de la base de datos puede modificar las estructuras internas de datos sin producir cambios en los programas de aplicaciones escritos anteriormente. En concreto, los índices pueden ser creados y asignados sin que afecten a la validez de otras sentencias ya existentes o de futuras sentencias SELECT.

© Alfaomega - RC Libros

SYNAPS PREMIUM

¿Cuáles son las ventajas de los índices?

El sistema puede usar un índice para aumentar la eficacia en diversas circunstancias. Algunas de estas circunstancias se describen a continuación:

Acceso directo a un registro especificado: suponiendo que un índice se defina sobre la columna apropiada, el sistema puede evitar la exploración para localizar los registros identificados por una cláusula WHERE.

Ordenación: observar que la Figura 2.3 muestra valores CNOMBRE en orden alfabético (realmente EBCDIC). El sistema puede usar este índice para recuperar registros de la tabla CURSO en secuencia CNOMBRE. En ausencia de índice, el sistema debe iniciar una rutina de ordenación interna que puede ser costosa, especialmente para un gran número de registros. Las versiones actuales de SQL considerarán el uso de un índice, siempre que la sentencia SELECT contenga cualquiera de las siguientes palabras clave o cláusulas: ORDER BY, DISTINCT, GROUP BY o UNION.

Operación join: la operación *join* se presentará en un capítulo posterior. En este momento, simplemente, afirmamos que la operación *join* puede ser difícil de ejecutar y que los índices pueden reducir el coste de este proceso.

No hay límite en el número de índices que se pueden crear para una tabla dada. Sin embargo, existe un coste asociado con cada índice que puede contrarrestar sus ventajas.

¿Cuáles son las desventajas de los índices?

Existen dos factores de coste asociados con los índices de base de datos que prohíben su uso ilimitado. Estos son:

Espacio de disco usado por el índice: un índice, aunque sea más pequeño que la tabla, puede ocupar una cantidad de espacio de almacenamiento considerable en el disco. Una tabla con muchos registros significa que el índice tendrá muchas entradas. Si la tabla tuviera muchos índices, el espacio total del disco usado por todos los índices podría exceder el tamaño de la tabla en sí misma.

Costes de actualización: mientras los índices facilitan el proceso de obtención de registros, dificultan el proceso de actualización. Siempre que un nuevo registro se inserte o se borre de una tabla, el sistema debe hacer el correspondiente cambio de los índices. Si, por ejemplo, una tabla tiene cinco índices, entonces una operación de

© Alfaomega - RC Libros

SYNAPS PREMIUM

inserción o de borrado obliga a una actualización de los cinco índices. Esto podría afectar de una forma importante al tiempo de respuesta.

¿Usará el sistema siempre el índice?

El *optimizador* puede decidir no usar un índice, incluso si se ha definido en la columna apropiada. Esto puede ocurrir en varios casos. Algunos de estos son:

La tabla es muy pequeña: debido a que se emplea mucho tiempo en leer el índice, el *optimizador* puede decidir que es más rápido, simplemente, explorar toda la tabla.

El índice no es lo suficientemente selectivo: el *optimizador* podría estimar que van a ser seleccionados tantos registros y que es más eficaz realizar una operación de exploración para encontrar los registros deseados.

No hay forma de indicarle al sistema que use explícitamente un índice. La decisión de usar un índice para acceder a los datos la toma el *optimizador*. (Sin embargo, es posible modificar parte de la información del catálogo para influir en el *optimizador* para que especifique un camino de acceso que use un índice. Esto podría realizarlo el administrador de la base de datos o un programador de aplicaciones que esté usando el SQL embebido en un programa que se ejecutará muchas veces).

Índices compuestos, índices únicos e índices agrupados

La sentencia CREATE INDEX ofrece otras ventajas, además de las especificadas anteriormente. Incluyendo palabras clave esenciales en la sentencia (que se describirán en el capítulo correspondiente) puede crearse cualquiera de los tipos especiales de índices siguientes:

Índice compuesto: un índice puede crearse sobre varios nombres de columna. Un ejemplo simple de tal índice es un índice nombre-dirección (por ejemplo, FNOMBRE-FADDR) para una tabla que contenga una columna para nombre del miembro del personal docente y otra para la dirección. Observemos que un índice compuesto es, únicamente, un índice simple en el que la clave se compone de datos que se encuentran en varias columnas. Destaquemos también que el orden de los componentes del índice es significativo. Por ejemplo, un índice nombre-dirección (FNOMBRE-FADDR) es diferente de un índice dirección-nombre (FADDR-FNOMBRE). Las actuales versiones SQL permiten la especificación de hasta 16 columnas para un índice compuesto.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Índice único: un índice único es un índice que no puede contener valores duplicados. La presencia de un índice único hace que el sistema rechace cualquier actualización de la tabla que produzca valores duplicados en la(s) columna(s) especificada(s) por el índice. Un índice único debe ser creado para la(s) columna(s) de clave primaria. Aunque solo se pueda especificar una clave primaria para una tabla, sí es posible especificar un número indeterminado de índices únicos para una tabla. Por ejemplo, si deseáramos que el sistema reforzara la unicidad de los valores CNOMBRE, podríamos incluir la palabra clave "UNIQUE" en la sentencia CREATE INDEX relativa a la Figura 2.3.

Índice agrupado: puede diseñarse un índice por tabla como índice agrupado. Cuando se ejecuta la utilidad de carga o de reorganización, el sistema almacenará los registros de la tabla en el orden físico, de acuerdo con la clave del índice agrupado. Para ciertas consultas, el *optimizador* puede capitalizar el conocimiento de los registros almacenados en una determinada secuencia. No existe aún una secuencia en la que el sistema confíe. (Aunque las inserciones en la tabla pueden hacer que los registros ignoren la secuencia, el *optimizador* puede, aun en este caso, encontrar el camino de acceso para satisfacer cualquier consulta).

Estos tres tipos de índices no se excluyen mutuamente, siendo posible que un único índice tenga todas las características anteriores.

Un ejemplo de diseño de una base de datos relacional

A modo de ejemplo se va a presentar el diseño de una base de datos en un contexto educativo cuyas tablas se refieren a los cursos que van a ser impartidos en distintas clases de un centro educativo por varios profesores pertenecientes a diversos departamentos y en los que se matricularán distintos estudiantes. En el centro educativo también existe personal no docente.

Es lógico considerar como tablas de la base de datos CURSO, DEPARTAMENTO, CLASE, ESTUDIANTE, MATRÍCULA, CLAUSTRO y PERSONAL. En los párrafos siguientes se van a definir las características de estas tablas y el diseño de la base de datos.

En primer lugar consideramos la tabla de nombre CURSO, que va a definir las características de los cursos que van a ser impartidos. Los nombres de los campos y los tipos de datos se especifican a continuación:

© Alfaomega - RC Libros

SYNAPS PREMIUM

Nombre	*Tipo de datos*	*Longitud*	*Contenido*
CNO	*Carácter*	3 (*fijo*)	*Número de curso* (*único*)
CNOMBRE	*Carácter*	22 (*variable*)	*Nombre de curso* (*único*)
CDESCP	*Carácter*	25 (*variable*)	*Descripción del curso* (*única*)
CRED	*Entero*	5	*Créditos que vale el curso*
CTARIFA	*Decimal*	5,2	*Valor tarifa laboratorio*
CDEPT	*Carácter*	4 (*fijo*)	*Nombre id. departamento que da el curso*

Los datos de la tabla van a ser los siguientes:

CNO	*CNOMBRE*	*CDESCP*	*CRED*	*CTARIFA*	*CDEPT*
C11	*INTROD. A LAS CC.*	*PARA NOVATOS*	3	100.00	*CIS*
C22	*ESTRUCT. DE DATOS*	*MUY UTIL*	3	50.00	*CIS*
C33	*MATEMÁTICAS DISCRETAS*	*ABSOLUTAMENTE NECESARIO*	3	.00	*CIS*
C44	*CIRCUITOS DIGITALES*	*AH HA!*	3	.00	*CIS*
C55	*ARQITECT. COMPUTADORES*	*MAQ. VON NEUMANN*	3	100.00	*CIS*
C66	*BASES DE DATOS RELACIONALES*	*IMPRESCINDIBLE*	3	500.00	*CIS*
P11	*EMPIRISMO*	*VERLO PARA CREERLO*	3	100.00	*PHIL*
P22	*RACIONALISMO*	*PARA USARLOS CIS*	3	50.00	*PHIL*
P33	*EXISTENCIALISMO*	*PARA USARLOS CIS*	3	200.00	*PHIL*
P44	*SOLIPSISMO*	*PARA MI MISMO*	6	0.00	*PHIL*
T11	*ESCOLASTICISMO*	*PARA BEATOS*	3	150.00	*THEO*
T12	*FUNDAMENTALISMO*	*PARA DESCUIDADOS*	3	90.00	*THEO*
T33	*HEDONISMO*	*PARA SANOS*	3	.00	*THEO*
T44	*COMUNISMO*	*PARA AVAROS*	6	200.00	*THEO*

En esta tabla CURSO puede tomarse CNO como la clave primaria dado que CNO toma valores distintos para todas las filas y no tiene valores faltantes.

Directamente relacionada con la tabla CURSO, vamos a considerar la tabla DEPARTAMENTO, que tiene un registro por cada departamento académico al que pertenecen los profesores que imparten cursos. Sus campos van a ser el nombre identificativo de cada departamento, el edificio al que pertenecen, el número de despacho en que están ubicados y el número de identificación en el cuerpo docente del director del departamento. Los nombres de los campos, los tipos de datos y el contenido de la tabla se especifican a continuación:

Nombre	*Tipo de datos*	*Longitud*	*Contenido*
DEPT	*Carácter*	4 (*fijo*)	*Nombre id. departamento*
DEDIF	*Carácter*	2 (*fijo*)	*Edificio*
DDESPACHO	*Entero*	3	*Despacho*
DCHFNO	*Carácter*	3 (*fijo*)	*Director*

© Alfaomega - RC Libros

SYNAPS PREMIUM

DEPT	*DEDIF*	*DDESPACHO*	*DCHFNO*
THEO	*HU*	200	10
CIS	*SC*	300	80
D.G.	*SC*	100	–
PHIL	*HU*	100	60

Vemos que el campo DEPT toma valores distintos para todas las filas y no tiene valores faltantes. Además, según la política del centro educativo, cada curso debe estar organizado por un departamento académico, lo que nos indica que debemos tomar el campo DEPT como clave primaria de la tabla DEPARTAMENTO.

Para aumentar la integridad de la base de datos, especificamos el campo CDEPT de la tabla CURSO como clave secundaria que referencie la ***tabla padre*** DEPARTAMENTO (se restringen los valores CDEPT de la tabla CURSO a aquellos que se encuentran en la columna DEPT de la tabla DEPARTAMENTO). La tabla CURSO es una **tabla dependiente** o **tabla hijo** de la tabla padre DEPARTAMENTO.

Adicionalmente consideramos la tabla CLASE, que tiene un registro para cada clase que ofrezca un curso. Los campos que contiene son: número de curso, número de sección, número de identificación en el grupo de docentes del profesor, día y hora, edificio y sala en la que se encuentra la clase. Los nombres de los campos, los tipos de datos y el contenido de la tabla se especifican a continuación:

Nombre	*Tipo de datos*	*Longitud*	*Contenido*
CNO	*Carácter*	3 (*fijo*)	*Número de curso*
SEC	*Carácter*	2 (*fijo*)	*Número de sección*
CINSTRFNO	*Carácter*	2 (*fijo*)	*Número de profesor*
CDIA	*Carácter*	2 (*fijo*)	*Día de la semana*
CHORA	*Carácter*	14 (*fijo*)	*Hora de la clase*
CEDIF	*Carácter*	2 (*fijo*)	*Edificio de la clase*
CDESPACHO	*Entero*	3	*Sala de la clase*

CNO	*SEC*	*CINSTRFNO*	*CDIA*	*CHORA*	*CEDIF*	*CDEPACHO*
*C*11	01	08	*LU*	08:00–09:00 *AM*	*SC*	305
*C*11	02	08	*MA*	08:00–09:00 *AM*	*SC*	306
*C*33	01	80	*MI*	09:00–10:00 *AM*	*SC*	305
*C*55	01	85	*JU*	11:00–12:00 *AM*	*HU*	306
*P*11	01	06	*JU*	09:00–10:00 *AM*	*HU*	102
*P*33	01	06	*VI*	11:00–12:00 *AM*	*HU*	201
*T*11	01	10	*LU*	10:00–11:00 *AM*	*HU*	101
*T*11	02	65	*LU*	10:00–11:00 *AM*	*HU*	102
*T*33	01	65	*MI*	11:00–12:00 *AM*	*HU*	101

© Alfaomega - RC Libros

SYNAPS PREMIUM

En esta tabla consideramos el par de campos (CNO, SEC) como clave primaria. Además, sabemos que debido a la política del centro educativo, a cada clase le corresponde algún curso, por lo que cada valor del campo CNO de la tabla CLASE debe coincidir con algún valor CNO existente en la tabla CURSO. De esta forma, CNO en la tabla CLASE es clave secundaria que referencia CURSO, con lo que CURSO es **tabla padre** de CLASE, además de **depender (ser tabla hijo)** de DEPARTAMENTO. Puesto que CLASE es dependiente de CURSO, que por su parte es dependiente de DEPARTAMENTO, decimos que CLASE es **tabla descendiente** de DEPARTAMENTO. Las relaciones existentes entre las tres tablas que llevamos definidas en nuestra base de datos suelen representarse como se indica en la Figura 2-4.

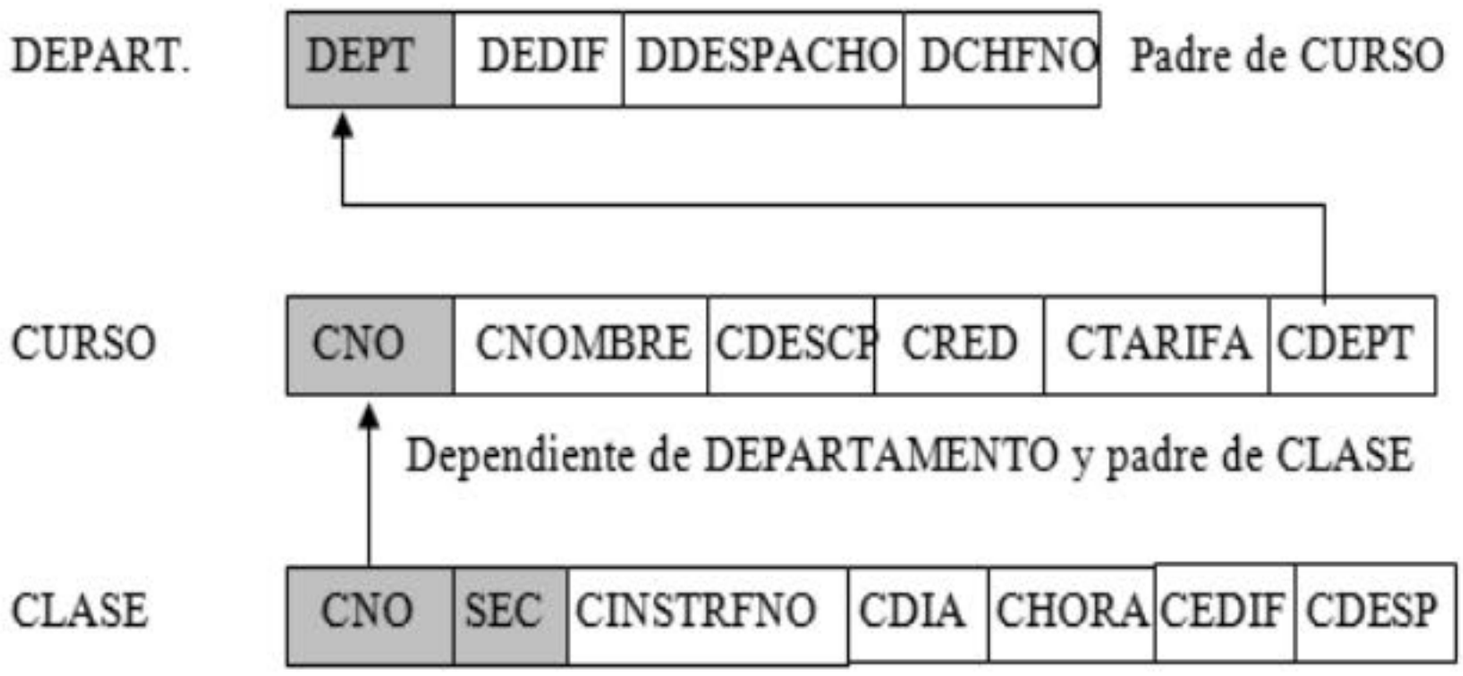


Figura 2-4

A continuación, consideramos las tablas MATRÍCULA (relativa a los datos recogidos cuando un estudiante se matricula en un curso) y ESTUDIANTE (tiene un registro para cada estudiante matriculado en el centro educativo). Los campos que contiene la tabla MATRÍCULA son: número de curso, número de sección, número de estudiante, fecha y hora de matriculación del estudiante.

Los campos que contiene la tabla ESTUDIANTE son: número de estudiante, nombre, dirección, teléfono, fecha de nacimiento, número de identificación de la escuela o facultad (centro educativo) y departamento.

Los nombres de los campos, los tipos de datos y el contenido de la tabla MATRÍCULA se especifican a continuación:

Nombre	*Tipo de datos*	*Longitud*	*Contenido*
CNO	*Carácter*	3 (*fijo*)	*Número de curso*
SEC	*Carácter*	2 (*fijo*)	*Número de sección*
SNO	*Carácter*	3 (*fijo*)	*Número de estudiante*
FEC _ MAT	*Date*		*Fecha de matriculación*
HORA _ MAT	*Time*		*Hora de matriculación*

© Alfaomega - RC Libros

SYNAPS PREMIUM

CNO	*SEC*	*SNO*	*FECHA_MAT*	*HORA_MAT*
*C*11	01	325	1987–01–04	09.41.30
*C*11	01	800	1987–12–15	11.49.00
*C*11	02	100	1987–12–17	09.32.00
*C*11	02	150	1987–12–17	09.32.30
*P*33	01	100	1987–12–23	11.30.00
*P*33	01	800	1987–12–23	11.23.00
*T*11	01	100	1987–12–23	11.21.00
*T*11	01	150	1987–12–15	11.35.30
*T*11	01	800	1987–12–15	14.00.00

Los nombres de los campos, los tipos de datos y el contenido de la tabla ESTUDIANTE se especifican seguidamente:

Nombre	*Tipo de datos*	*Longitud*	*Contenido*
SNO	*Carácter*	3 (*fijo*)	*Número de estudiante*
SNOMBRE	*Carácter*	30 (*variable*)	*Nombre de estudiante*
SDOMI	*Carácter*	15 (*variable*)	*Domicilio*
STLFNO	*Carácter*	12 *(fijo)*	*Teléfono*
SFNACIM	*Carácter*	6 (*fijo*)	*Fecha nacimiento*
SIQ	*Smallint*		*Número id. centro*
SADVFNO	*Carácter*	3(*fijo*)	*Número id. departamen to*
SESP	*Carácter*	4(*fijo*)	*Nombre id. departamen to*

SNO	*SNOMBRE*	*SDOMI*	*STLFNO*	*SFNACIM*	*SIQ*	*SADVFNO*	*SESP*
325	*CURLEY DUBAY*	*CONNECTICUT*	203–123–4567	780517	122	10	*THEO*
150	*LARRY DUBAY*	*CONNECTICUT*	203–123–4567	780517	121	80	*CIS*
100	*MOE DUBAY*	*CONNECTICUT*	203–123–4567	780517	120	10	*THEO*
800	*ROCKY BALBOA*	*PENNSYLVANIA*	112–112–1122	461004	99	60	*PHIL*

La clave primaria de la tabla MATRÍCULA es una composición de tres columnas (CNO, SEC, SNO). Para asegurar la integridad referencial, especificaríamos dos claves secundarias en esta tabla. La clave secundaria compuesta (CNO, SEC) referenciaría la tabla ESTUDIANTE. De esta forma, MATRÍCULA es dependiente de dos tablas: CLASE y ESTUDIANTE. Supongamos, también, que todos los estudiantes deben especializarse en una asignatura correspondiente a algún departamento académico existente. Esto significa que la sentencia CREATE TABLE para ESTUDIANTE, especificaría SESP (especialidad del estudiante) como clave secundaria que referencia la tabla DEPARTAMENTO.

Por lo tanto, DEPARTAMENTO es padre de otras dos tablas, CURSO y ESTUDIANTE. La clave primaria para la tabla ESTUDIANTE es el campo SNO (número de estudiante). La Figura 2-5 refleja esta nueva visión expandida del diseño de nuestra base de datos.

© Alfaomega - RC Libros

SYNAPS PREMIUM

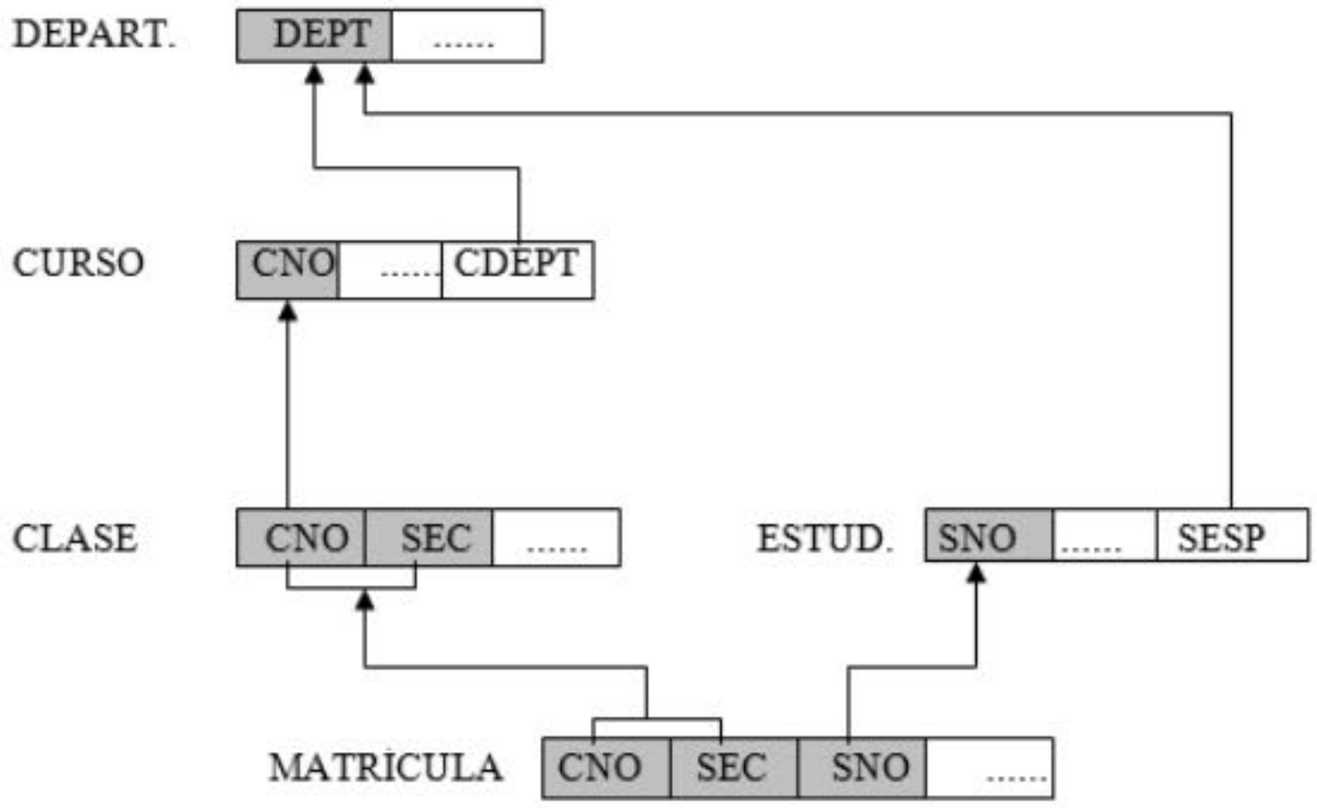


Figura 2-5

Consideremos ahora la tabla CLAUSTRO, que tiene un registro para cada miembro del cuerpo docente del centro educativo (escuela universitaria o facultad) y cuyos campos son: número de identificación de todos los profesores, nombre, dirección, fecha de contratación, número de ayudantes o subordinados que dependen de él, sueldo e identificación del departamento al que se encuentre adscrito. Los nombres de los campos, los tipos de datos y el contenido de la tabla CLAUSTRO se especifican a continuación:

Nombre	*Tipo de datos*	*Longitud*	*Contenido*
FNO	*Carácter*	3 (*fijo*)	*Número de profesor*
FNOMBRE	*Carácter*	15 (*fijo*)	*Nombre de profesor*
FDOMI	*Carácter*	20 (*variable*)	*Domicilio de profesor*
FFCANTI	*Date*		*Fecha contratación*
FNUMDEP	*Smallint*		*Número de ayudantes*
FSUELDO	*Decimal*	(7,2)	*Sueldo*
FDEP	*Caracter*	4 (*fijo*)	*Nombre id. departamento*

FNO	*FNOMBRE*	*FDOMI*	*FFECANTI*	*FNUMDP*	*FSUELDO*	*FDEPT*
06	*KATHY PEPE*	*CALLE DE LAPIEDRA*, 7	1979 – 01 – 15	2	35.000.00	*PHIL*
10	*JESSIE MARTIN*	*DR. DEL ESTE*, 4	1969 – 09 – 01	1	45.000.00	*THEO*
08	*JOSE COHN*	*APTDO. CORREOS* 1138	1979 – 07 – 09	2	35.000.00	*CIS*
85	*AL HARTLEY*	*CALLE DE LA PLATA*	1979 – 09 – 05	7	45.000.00	*CS*
60	*JULIA MARTIN*	*DR. ESTE*, 4	1969 – 09 – 01	1	45.000.00	*PHIL*
65	*LISA BOBAK*	*CAMINO DE LA RISA*, 77	1981 – 09 – 06	–	36.000.00	*THEO*
80	*BARB HLAVATY*	*CALLE DEL SUR*, 489	1982 – 01 – 16	3	35.000.00	*CIS*

© Alfaomega - RC Libros

SYNAPS PREMIUM

El campo FNO es la clave primaria de la tabla CLAUSTRO. Las claves secundarias pueden ser definidas de forma que se establezca una relación cíclica. Supongamos que es norma de la escuela que a cada miembro del personal docente se le asigne algún departamento académico. Entonces, la columna FDEPT, en la tabla CLAUSTRO, se especificaría como clave secundaria, referenciando DEPARTAMENTO (DEPARTAMENTO es la tabla padre y CLAUSTRO la tabla dependiente). Supongamos, también, que el director de cada departamento es miembro del personal docente. Entonces, el valor DCHFNO, en la tabla DEPARTAMENTO, se especifica como clave secundaria que referencia CLAUSTRO. (Aquí, CLAUSTRO es la tabla padre y DEPARTAMENTO es la tabla dependiente). La Figura 2.6 muestra esta **relación cíclica** o **ciclo**.

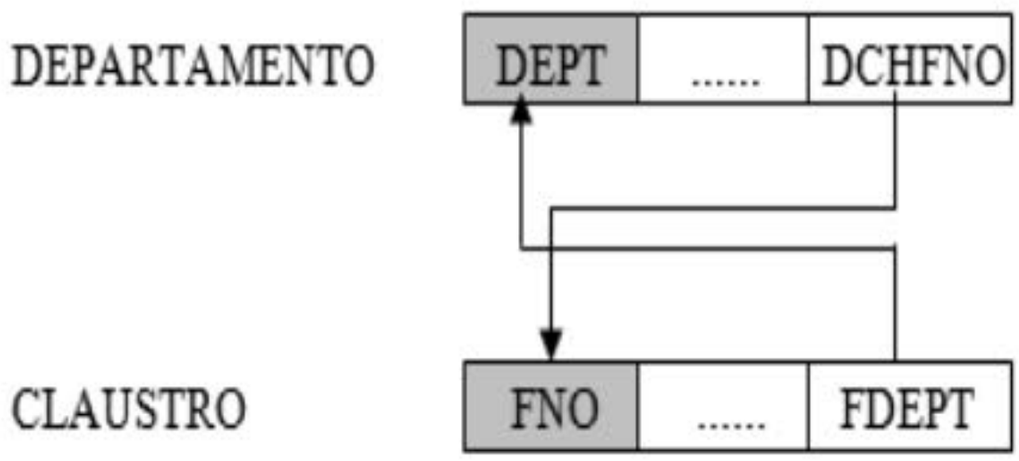


Figura 2-6

A veces, un ciclo incluye más de dos tablas. La TABLA1 podría ser padre de la TABLA2, que es padre de la TABLA3, la cual es, a su vez, padre de la TABLA4, siendo esta padre de la TABLA1. En general, se forma un ciclo si una tabla dependiente o descendente de cualquier tabla se convierte en padre de esa tabla.

Es posible también que una tabla sea padre y dependiente de sí misma, dando lugar a un **ciclo recursivo**. En otras palabras, la tabla contiene una clave secundaria que se referencia a sí misma. Por ejemplo, supongamos que algunos cursos exigen, al menos haber realizado otro curso, que es requisito indispensable (es decir, los alumnos están obligados a realizarlo) para matricularse en una clase del curso. Podríamos indicar el curso que se exige como requisito para la matriculación en la tabla CURSO, modificando esta tabla para incluir una nueva columna, PCNO, que contenga el número de curso del curso que se pidió como prerrequisito a la matriculación. En ese caso, la orden CREATE TABLE para el curso contendría una cláusula adicional de clave secundaria, especificando PCNO como clave secundaria que referencia CURSO. Este ciclo de autorreferencia se llama, a veces, **relación recursiva**.

El tipo de datos y la extensión de cada componente de una clave secundaria deben ser la misma que la clave primaria. Pero una clave secundaria, a diferencia de la primaria, puede contener valores nulos (es decir, no es necesario especificar NOT NULL para una

SYNAPS PREMIUM

columna que sea clave secundaria). Por ejemplo, existen muchos cursos introductorios que no necesitan un curso como requisito previo. Los registros de tales cursos podrían tener un valor nulo en la columna PCNO. Otro ejemplo es la columna SESP en la tabla ESTUDIANTE. Si suponemos que algunos estudiantes no tienen que declarar la especialidad, entonces los valores correspondientes de SESP podrían ser nulos. Observemos, además, que si cualquier parte de una clave secundaria compuesta es nula, entonces toda la clave secundaria se considera como nula.

En la base de datos existe una última tabla de nombre PERSONAL que tiene un registro para cada empleado que no pertenezca al cuerpo docente de la escuela universitaria o facultad. Sus campos son: nombre, título, sueldo e identificación del departamento del que dependa. Estos campos no tienen relación con ninguna otra tabla. Los nombres de los campos, los tipos de datos y el contenido de la tabla PERSONAL se especifican a continuación (ENOMBRE es la clave primaria):

Nombre	*Tipo de datos*	*Longitud*	*Contenido*
ENOMBRE	*Carácter*	15 (*fijo*)	*Nombre*
CARGO	*Carácter*	10 (*fijo*)	*Título*
ESUELDO	*Entero*	5	*Sueldo*
DEPT	*Carácter*	4 (*fijo*)	*Departamento*

ENOMBRE	*CARGO*	*ESUELDO*	*DEPT*
LUCAS	*EVANG* 1	53	*THEO*
MARCOS	*EVANG* 2	52	*THEO*
MATEO	*EVANG* 3	51	*THEO*
DICK NIX	*LADRÓN*	25001	*PHIL*
HANK KISS	*BUFÓN*	25000	*PHIL*
JUAN	*EVANG* 4	54	*THEO*
EUCLIDES	*AYTE. LAB.*	1000	*MATH*
ARQUIMEDES	*AYTE. LAB.*	200	*ENG*
DA VINCI	*AYTE. LAB.*	500	–

Una vez definidas todas las relaciones entre las tablas de la base de datos podemos realizar ya un diagrama completo que refleje su diseño final (Figura 2-7).

© Alfaomega - RC Libros

SYNAPS PREMIUM

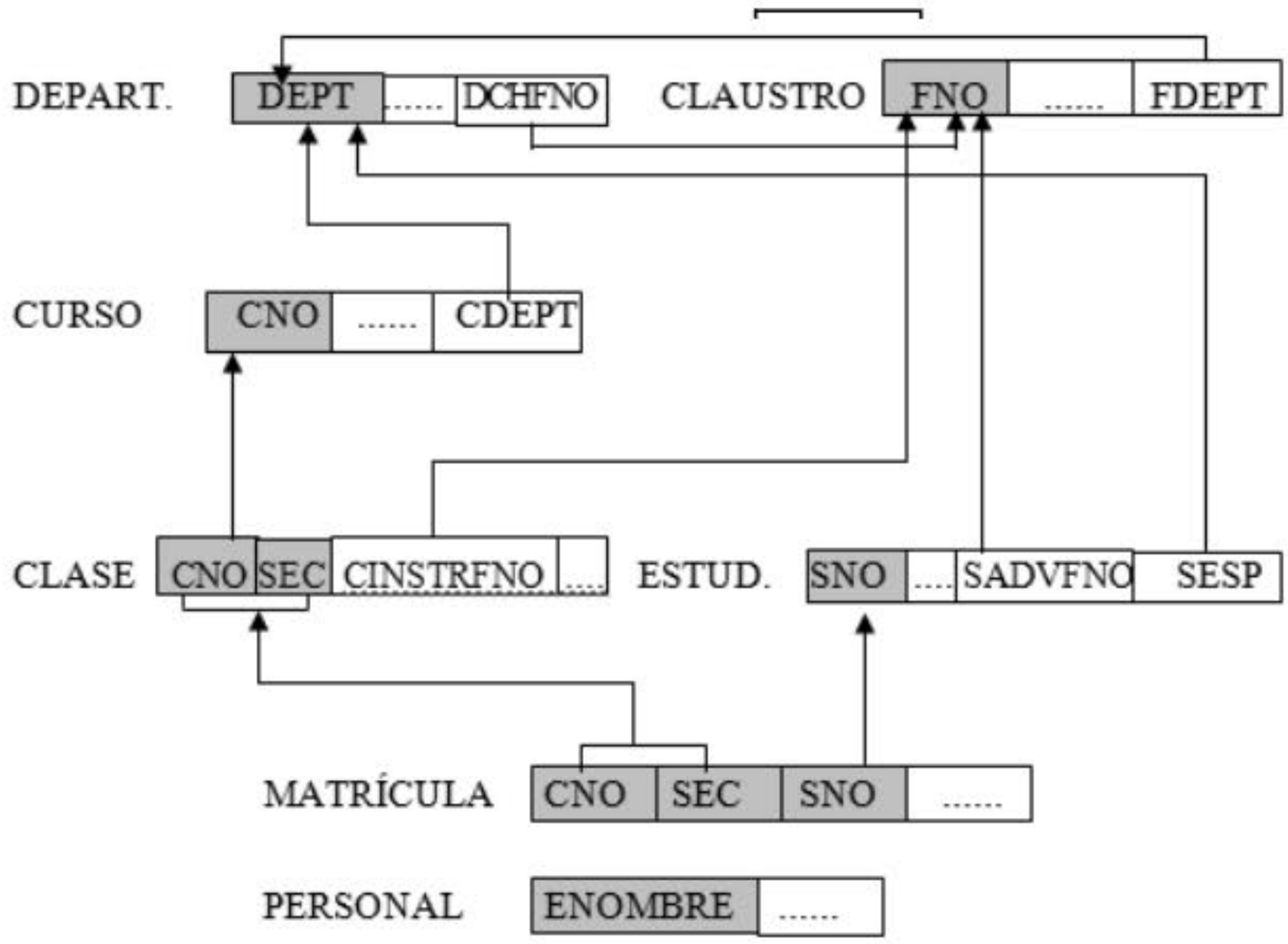


Figura 2-7

Hasta ahora, solo hemos considerado el aspecto estructural de la integridad referencial. Destacaremos, ahora, algunas de las reglas de procesamiento que especifican la respuesta del sistema a la sentencia de manipulación de datos del SQL, la cual referencia valores de clave primaria y/o secundaria. Estas sentencias (INSERT, UPDATE y DELETE) se describirán en capítulos posteriores. Adicionalmente tendremos en cuenta para la integridad referencial lo siguiente:

Inserción de la clave primaria: el sistema verificará que el valor de la clave primaria es único. Si no lo es, la inserción será rechazada.

Inserción de la clave secundaria: el sistema verificará que cualquier clave secundaria tenga el mismo valor que alguna clave primaria existente en la tabla referenciada. Si esto no se cumple, la inserción será rechazada.

Actualización de la clave primaria: el sistema rechazará la actualización, si existe un registro dependiente que referencie la clave primaria.

Actualización de la clave secundaria: el sistema verificará que el nuevo valor de la clave secundaria coincida con alguna clave primaria existente en la tabla referenciada. Si esto no ocurre, la actualización será rechazada.

Borrado de la clave primaria: se puede dirigir la respuesta del sistema a la eliminación de un registro que tenga una clave primaria igual a alguna clave

© Alfaomega - RC Libros

SYNAPS PREMIUM

secundaria correspondiente. Las opciones disponibles son: rechazar cualquier operación de borrado (RESTRICT), sustituir automáticamente los valores nulos por los valores correspondientes de la clave secundaria (SET NULL) y borrar automáticamente cualquier registro dependiente que tenga una clave secundaria igual a la primaria (CASCADE).

Borrado de una clave secundaria: no existen restricciones (a menos que la clave secundaria sea parte de la primaria en la que se apliquen otras restricciones).

El diseñador de la base de datos debe prestar una cuidadosa atención a estas reglas de procesamiento. Existen restricciones e implicaciones adicionales para las tablas descendentes, especialmente aquellas afectadas por relaciones cíclicas. El diseño completo de las relaciones existentes entre las tablas (Figura 2.7) y las restricciones de integridad son el corazón de una base de datos.

CREAR TABLAS

Dentro del lenguaje SQL (Structured Query Language) que Oracle utiliza para tratar la información, se encuentra la categoría de sentencias (o comandos) denominada lenguaje de modificación de datos DML (Data Modification Language). Las sentencias del lenguaje de modificación de datos se usan en la interrogación y manipulación de datos en esquemas de bases de datos ya existentes. Entre estas sentencias destacan las que se utilizan para realizar actualizaciones en las bases de datos y cuyos exponentes más importantes son las sentencias INSERT, UPDATE, DELETE y MERGE (nueva en Oracle 12c).

La creación y administración de tablas de una base de datos se realiza básicamente a partir de las sentencias de Oracle SQL CREATE TABLE, ALTER TABLE y DROP TABLE. La estructura básica de la sentencia CREATE TABLE se presenta en el ejemplo siguiente:

```
SQL> CREATE TABLE oficinas (
  2  Oficina  integer not null,
  3  Ciudad char(15) not null,
  4  Region char(10) not null,
  5  Dir  integer,
  6  Objetivo number,
  7  Ventas number not null
  8  );
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Agregar datos a tablas. La sentencia INSERT

A lo largo de este capítulo se estudian las sentencias que describen las técnicas para manipular datos en las tablas de una base de datos relacional. Puede agregar nuevas filas de datos, modificar los datos de filas existentes y eliminar filas.

En el caso de la agregación de nuevas filas de datos a las tablas ya existentes, la sentencia básica es INSERT.

La sentencia INSERT agrega una o más filas nuevas a una tabla. Tratada de forma simplificada, INSERT tiene la siguiente forma:

```
INSERT [INTO] tabla_o_vista [(lista_de columnas)] valores_de_datos
```

La sentencia hace que *valores_de_datos* se inserte como una o más filas en la *tabla_o_vista* que se nombra. El argumento *lista_de_columnas* es una lista separada por comas de los nombres de tablas que se pueden utilizar para especificar las columnas para las que se suministran datos. Si no se especifica *lista_de_columnas*, todas las columnas de la tabla o vista recibirán datos.

Cuando una *lista_de_columnas* no enumera todas las columnas de la tabla o vista, se inserta un valor de NULL (o el valor predeterminado si se ha definido alguno para la columna) en aquellas columnas que no se hayan enumerado en la lista. Todas las columnas no especificadas en la lista de columnas deben permitir valores NULL o tener un valor predeterminado asignado.

Los *valores_de_datos* suministrados deben corresponderse con la lista de columnas. El número de valores de datos debe ser el mismo que el número de columnas y el tipo de datos, precisión y escala de cada valor de datos debe coincidir con los de la columna correspondiente. Hay dos maneras de especificar los valores de datos:

- Utilizar una cláusula VALUES para especificar los valores de datos para una fila:

```
INSERT [INTO] tabla_o_vista VALUES (lista de valores_de_datos)
```

- Utilizar una subconsulta SELECT para especificar los valores de datos para una o más filas:

```
INSERT INTO tabla_o_vista
        SELECT columnas
        FROM tablas
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

Insertar una fila mediante INSERT INTO...VALUES

La palabra clave VALUES especifica los valores de una fila de una tabla. Los valores se especifican como una lista separada por comas de expresiones escalares cuyos tipos de datos, precisión y escala deben ser iguales, o implícitamente convertibles, a la columna correspondiente de la lista de columnas. Si no se especifica una lista de columnas, los valores deberán especificarse en la misma secuencia que las columnas de la tabla o vista.

Como ejemplo, la sentencia siguiente inserta una fila en la tabla DEPT de la base de datos que crea por defecto Oracle en la instalación:

```
SQL> INSERT INTO dept
  2     VALUES (50, 'PRODUCTION', 'SAN FRANCISCO');

1 fila creada.

SQL>
```

La siguiente sentencia inserta una fila con seis columnas en la tabla EMP table. Una de estas columnas se asigna como nula (NULL) y otra se asigna como numérica con notación científica:

```
SQL> INSERT INTO emp (empno, ename, job, sal, comm, deptno)
  2     VALUES (7890, 'JINKS', 'CLERK', 1.2E3, NULL, 40);

1 fila creada.

SQL>
```

La sentencia siguiente es equivalente a la anterior:

```
INSERT INTO (SELECT empno, ename, job, sal, comm, deptno FROM emp)
  VALUES (7890, 'JINKS', 'CLERK', 1.2E3, NULL, 40);
```

El propósito de la lista de columnas en la sentencia INSERT es hacer corresponder los valores de datos en la cláusula VALUES con las columnas que van a recibirlos. La lista de valores y la lista de columnas deben contener el mismo número de elementos y el tipo de dato de cada valor debe ser compatible con el tipo de datos de la columna correspondiente, o en caso contrario se producirá un error. El estándar ANSI/ISO exige nombres de columna sin cualificar en la lista de columnas, pero muchas implementaciones permiten nombres cualificados. Evidentemente no puede haber ambigüedad alguna en los nombres de columna, ya que deben referenciar todos a columnas de la tabla destino.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Insertar valores NULL

Cuando SQL inserta una nueva fila de datos en una tabla, automáticamente asigna un valor NULL a cualquier columna cuyo nombre falte de la lista de columnas de la sentencia INSERT.

En la sentencia siguiente se inserta el nuevo vendedor Henry Jacobsen a la tabla REPVENTAS omitiéndose las columnas CUOTA y DIRECTOR, que por lo tanto recibirán valores NULL para esta fila:

```
SQL> INSERT INTO REPVENTAS (NAME, EDAD, NUM_EMPL, VENTAS, TITULO, CONTRATO, OFICINA_REP)
  2  VALUES ('Henry Jacobsen', 36, 111, 0.00, 'Dir Ventas', '25-jul-90',13);
```

Como resultado, la fila recién añadida tiene un valor NULL en las columnas CUOTA y DIRECTOR tal y como se muestra en la consulta siguiente:

```
SQL> SELECT *
  2  FROM repventas;

 NUM_EMPL NAME                 EDAD OFICINA_REP TITULO     CONTRATO   DIRECTOR      CUOTA     VENTAS
--------- -------------------- ---------- ---------- ---------- -------- ---------- ---------- --------
       101 Dan Roberts              45         12 Rep Ventas 20/10/86        104     300000   305673
       102 Sue Smith                48         21 Rep Ventas 12/12/86        108     350000   474050
       103 Paul Cruz                29         12 Rep Ventas 01/03/87        104     275000   286775
       104 Bob Smith                33         12 Dir Ventas 19/05/87        106     200000   142594
       105 Bill Adams               37         13 Rep Ventas 12/02/88        104     350000   367911
       106 Sam Clark                52         11 VP Ventas  14/06/88                275000   299912
       107 Nancy Angelli            49         22 Rep Ventas 14/11/88        108     300000   186042
       108 Larry Fitch              62         21 Dir Ventas 12/10/89        106     350000   361865
       109 Mary Jones               31         11 Rep Ventas 12/10/99        106     300000   392725
       110 Tom Snyder               41            Rep ventas 13/01/90        101              75985
       111 Henry Jacobsen           36         13 Dir Ventas 25/07/90                         0
```

La asignación de valores NULL puede ser más puntual, incluyendo las columnas en la lista y especificando la palabra clave NULL en los propios valores, tal y como se muestra en la sentencia siguiente:

```
SQL> INSERT INTO REPVENTAS (NAME,EDAD,NUM_EMPL,VENTAS,CUOTA,TITULO,DIRECTOR,CONTRATO,OFICINA_REP)
  2  VALUES ('Matias Clark',35,11,0.02,NULL,'Dir Ventas',NULL,'26-jul-90',15);
```

Insertar todas las columnas

Por motivos prácticos SQL permite omitir la lista de columnas de la sentencia INSERT, en cuyo caso se genera automáticamente una lista formada por todas las columnas de la tabla en secuencia de izquierda a derecha. La sentencia anterior podría entonces escribirse como sigue:

```
INSERT INTO REPVENTAS
VALUES ('Matias Clark',35,11,0.02,NULL,'Dir Ventas',NULL,'26-jul-90',15);
```


SYNAPS PREMIUM

Lógicamente, cuando se omite la lista de columnas, la palabra clave NULL debe ser utilizada en la lista de valores para asignar explícitamente valores NULL a las columnas. Además, la secuencia de valores de datos debe corresponder exactamente con la secuencia de columnas de la tabla.

Insertar varias filas mediante INSERT INTO...SELECT

La subconsulta SELECT en la instrucción INSERT se puede utilizar para agregar valores de una o más tablas o vistas. Utilizar una subconsulta SELECT permite además que se inserte más de una fila al mismo tiempo.

La lista de selección de la subconsulta debe coincidir con la lista de columnas de la instrucción INSERT. Si no se especifica ninguna lista de columnas, la lista de selección deberá coincidir con las columnas de la tabla o vista en la que se inserta.

En el ejemplo siguiente se inserta en la tabla BONUS el nombre, trabajo, salario y comisión de todos los empleados cuya comisión excede del 25% de su salario o cuyo trabajo sea presidente o manager:

```
SQL> INSERT INTO bonus
  2     SELECT ename, job, sal, comm
  3     FROM emp
  4     WHERE comm > 0.25 * sal
  5     OR job IN ('PRESIDENT', 'MANAGER');
```

Otro de los usos de la instrucción INSERT INTO...SELECT es insertar datos de un origen externo a ORACLE SQL. Por ejemplo, la consulta siguiente inserta una fila en la tabla ACCOUNTS del usuario SCOTT en la base de datos accesible por el enlace de base de datos SALES:

```
INSERT INTO scott.accounts@sales (acc_no, acc_name)
  VALUES (5001, 'BOWER');
```

En general la sentencia INSERT suele tener la siguiente sintaxis:

INSERT INTO [usuario.] tabla[@enlace_bd] [(col1 [, col2]...)]
VALUES (valor1 [,valor2]...)|consulta...);

El usuario opcional *usuario* debe ser un usuario sobre cuya tabla se nos ha concedido permiso de inserción, *tabla* es la tabla en la que se insertan las filas, cada valor de la lista de valores debe corresponderse con cada columna de la lista de columnas y *enlace_bd* es el nombre de una base de datos remota donde se encuentra la tabla.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 2.1. Consideremos una base de datos que contiene información correspondiente a una sencilla aplicación de procesamiento de pedidos para una pequeña empresa de distribución. Consta de cinco tablas:

• CLIENTES, que contiene una fila por cada uno de los clientes de la empresa. Sus campos son NUM_CLIE (número de cliente), EMPRESA, REP_CLIENTE (número de empleado del representante que atiende al cliente) y LIM_CREDITO (límite de crédito).

• REPVENTAS, que contiene una fila por cada uno de los diez vendedores de la empresa. Sus campos son NUM_EMPL (número de empleado), NOMBRE, EDAD, OFICINA_REP (número de oficina de representación del vendedor), TÍTULO, CONTRATO, DIRECTOR, CUOTA (ventas previstas) y VENTAS (ventas realizadas).

• OFICINAS, que contiene una fila por cada una de las cinco oficinas en las que trabajan los vendedores. Sus campos son OFICINA (número de oficina), CIUDAD, REGIÓN, DIR (número de empleado del director), OBJETIVO (ventas anuales previstas) y VENTAS (ventas anuales realizadas).

• PRODUCTOS, que contiene una fila por cada tipo de producto disponible para la venta. Sus campos son ID_FAB (identificador de fabricante), ID_PRODUCTO (identificador de producto), DESCRIPCIÓN, PRECIO y EXISTENCIA.

• PEDIDOS, que contiene una fila por cada pedido ordenado por un cliente. Por simplicidad, se supone que cada pedido se refiere a un solo producto. Sus campos son NUM_PEDIDO (número de pedido), FECHA_PEDIDO (fecha del pedido), CLIE (número de cliente), REP (número de empleado del representante), FAB (identificador de fabricante), PRODUCTO (identificador de producto), CANT (cantidad) e IMPORTE.

Realizar un diseño apropiado para las cinco tablas y las relaciones padre/hijo entre las columnas que contienen. Introducir los datos de las cinco tablas de esta base de datos que se presentan después del gráfico de diseño que soluciona el problema.

En una base de datos relacional bien diseñada cada tabla tiene una columna o combinación de columnas cuyos valores identifican unívocamente cada fila en la tabla. Esta columna (o columnas) se denomina *clave primaria* de la tabla. Miremos en primer lugar la tabla OFICINAS. A primera vista, tanto la columna OFICINA como la columna CIUDAD podrían servir como clave primaria para la tabla, pero si la empresa se amplía y abre dos oficinas de ventas en la misma ciudad, la columna CIUDAD ya no

© Alfaomega - RC Libros

SYNAPS PREMIUM

podría servir como clave primaria. En la práctica, "números de ID", tales como el número de oficina (OFICINA en la tabla OFICINAS), el número de empleado (NUM_EMP en la tabla REPVENTAS) y los números de clientes (NUM_CLIE en la tabla CLIENTES) se eligen con frecuencia como claves primarias. En el caso de la tabla PEDIDOS no hay elección, ya que lo único que identifica unívocamente un pedido es el número de pedido (NUM_PEDIDO).

La tabla PRODUCTOS es un ejemplo de una tabla donde la clave primaria debe ser una *combinación* de columnas. La columna ID_FAB identifica al fabricante de cada producto en la tabla y la columna ID_PRODUCT especifica el número de producto del fabricante. La columna ID_PRODUCT podría ser una buena clave primaria, pero no hay nada que impida que dos fabricantes diferentes utilicen el mismo número para sus productos. Por tanto, debe utilizarse una combinación de las columnas ID_FAB e ID_PRODUCT como clave primaria de la tabla PRODUCTOS. Cada producto de la tabla se garantiza que tiene una combinación única de valores en estas dos columnas.

En cuanto a las claves secundarias, tendremos en cuenta que una columna de una tabla cuyo valor coincide con la clave primaria de alguna otra tabla se denomina *clave secundaria, externa o foránea.* En nuestra base de datos, la columna OFICINA_REP es una clave foránea para la tabla OFICINAS. Aunque es una columna en la tabla REPVENTAS, los valores que esta columna contiene son números de oficina. Coinciden con valores en la columna OFICINA, que es la clave primaria para la tabla OFICINAS. Juntas, una clave primaria y una clave foránea crean una relación padre/hijo entre las tablas que las contienen, del mismo modo que las relaciones padre/hijo de una base de datos jerárquica. Por la misma razón, la columna DIR de la tabla OFICINAS es una clave foránea para la tabla REPVENTAS, ya que los valores que esta columna contiene son números de empleado que coinciden con valores de la columna NUM_EMPL que es clave primaria en la tabla REPVENTAS. En idéntica situación está la columna REP_CLIE de la tabla CLIENTES, que también es una clave foránea para la tabla REPVENTAS, ya que los valores que esta columna contiene son números de empleado que coinciden con valores de la columna NUM_EMPL que es clave primaria en la tabla REPVENTAS. Incluso la columna DIRECTOR de la tabla REPVENTAS puede considerarse como una clave foránea de la propia tabla REPVENTAS, ya que los valores que esta columna contiene son números de empleado que coinciden con valores de la columna NUM_EMPL que es clave primaria en la tabla REPVENTAS.

Lo mismo que una combinación de columnas puede servir como clave primaria de una tabla, una clave foránea puede ser también una combinación de columnas. De hecho, la clave foránea será *siempre* una clave compuesta (multicolumna) cuando

© Alfaomega - RC Libros

SYNAPS PREMIUM

referencia a una tabla con una clave primaria compuesta. Obviamente el número de columnas y los tipos de datos de las columnas en la clave foránea y en la clave primaria deben ser idénticos unos a otros.

Una tabla puede contener más de una clave foránea si está relacionada con más de una tabla adicional. En nuestra base de datos podemos definir tres claves foráneas en la tabla PEDIDOS. Serían las siguientes:

- La columna CLIE es una clave foránea para la tabla CLIENTES, que relaciona cada pedido con el cliente que lo remitió.

- La columna REP es una clave foránea para la tabla REPVENTAS que relaciona cada pedido con el vendedor que lo tomó.

- Las columnas FAB y PRODUCTO juntas son una clave foránea compuesta para la tabla PRODUCTOS, que relaciona cada pedido con el producto solicitado.

Una vez discutido el diseño de la base de datos, podemos representarlo como se indica en la Figura 2.8. Por último, se presentarán los datos de cada tabla:

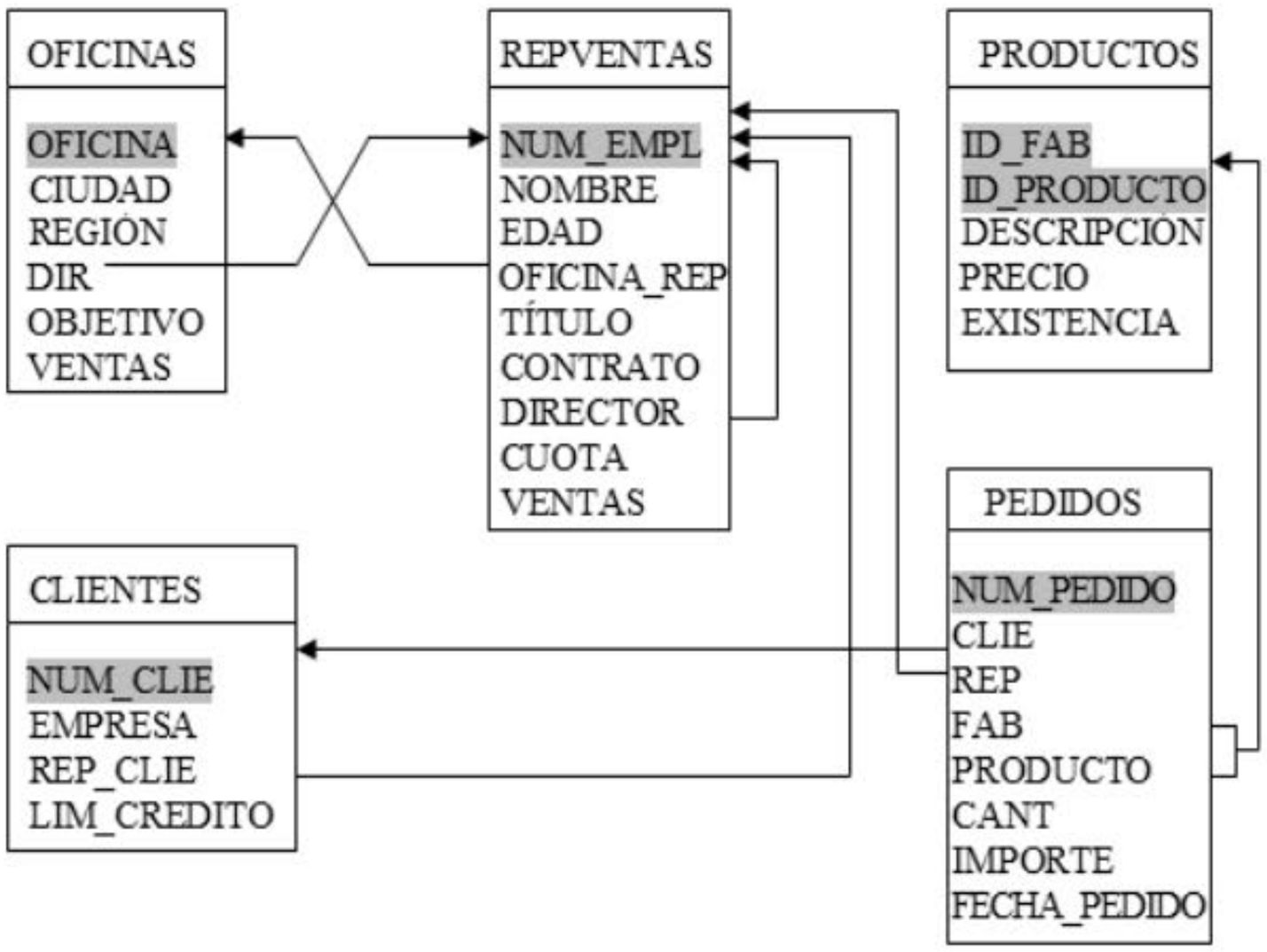


Figura 2.8

© Alfaomega - RC Libros

SYNAPS PREMIUM

PRODUCTOS

ID_FAB	ID_PRODUCTO	DESCRIPCION	PRECIO	EXISTENCIAS
REI	2A45C	V Stago Trinquete	79,00	210
ACI	4100Y	Extractor	2.750,00	25
QSA	XK 47	Reductor	355,00	38
BIC	41672	Plate	180,00	0
IMM	779C	Riostra 2 – Tm	1.875,00	9
ACI	41003	Articulo Tipo3	107,00	207
ACI	41004	Articulo Tipo4	117,00	139
BIC	41003	Manivela	652,00	3
IMM	887P	Perno Riostra	250,00	24
QSA	XK 48	Reductor	134,00	203
REI	2A44L	Bisagra Izqda.	4.500,00	12
FEA	112	Cubierta	148,00	115
IMM	887H	Soporte Riostra	54,00	223
BIC	41089	Retn	225,00	78
ACI	41001	Articulo Tipo1	55,00	277
IMM	775C	Riostra 1 – Tm	1.425,00	5
ACI	4100Z	Montador	2.500,00	28
QSA	XK 48A	Reductor	117,00	37
ACI	41002	Articulo Tipo2	76,00	167
REI	2A44R	Bisagra Dcha.	4.500,00	12
IMM	773C	Riostra 1/2 – Tm	975,00	28
ACI	4100X	Ajustador	25,00	37
FEA	114	Bancada Motor	243,00	15
IMM	887X	Retenedor Riostra	475,00	32
REI	2A44G	Pasador Bisagra	350,00	14

CLIENTES

NUM_CLIE	EMPRESA	REP_CLIE	LIMITE_CREDITO
2111	JCP Inc.	103	50.000,00
2102	Firts Corp.	101	65.000,00
2103	Acme Mfg.	105	50.000,00
2123	Carter & Sons	102	40.000,00
2107	Ace Internatio nal	110	35.000,00
2115	Smithson Corp.	101	20.000,00
2101	Jones Mfg.	106	65.000,00
2112	Zetacorp	108	50.000,00
2121	QMA Assoc.	103	45.000,00
2114	Orion Corp	102	20.000,00
2124	Peter Brothers	107	40.000,00
2108	Holm & Landis	109	55.000,00
2117	J.P. Sinclair	106	35.000,00
2122	Three – Way Lines	105	30.000,00
2120	Rico Enterprise s	102	50.000,00
2106	Fred Lewis Corp.	102	65.000,00
2119	Solomon Inc.	109	25.000,00
2118	Midwest Systems	108	60.000,00
2113	Ian & Schmidt	104	20.000,00
2109	Chen Associates	103	25.000,00
2105	AAA Investment s	101	45.000,00

© Alfaomega - RC Libros

SYNAPS PREMIUM

PEDIDOS

NUM_PEDIDO	FECHA_PEDIDO	CLIE	REP	FAB	PRODUCTO	CANT	IMPORTE
112961	17_DIC_89	2117	106	REI	2A44L	7	31.500,00
113012	11-ENE-90	2111	105	ACI	41003	35	3.745,00
112989	03-ENE-90	2101	106	FEA	114	6	1.458,00
113051	10-FEB-90	2118	108	QSA	K47	4	1.420,00
112968	12-0CT-89	2102	101	ACI	41004	34	3.978,00
110036	30-ENE-90	2107	110	ACI	4100Z	9	22.500,00
113045	02-FEB-90	2112	108	REI	2A44R	10	45.000,00
112963	17-DIC-89	2103	105	ACI	41004	28	3.276,00
113013	14-ENE-90	2118	108	BIC	41003	1	652,00
113058	23-FEB-90	2108	109	FEA	112	10	1.480,00
112997	08-ENE-90	2124	107	BIC	41003	1	652,00
112983	27-DIC-89	2103	105	ACI	41004	6	702,00
113024	20-ENE-90	2114	108	QSA	XK47	20	7.100,00
113062	24-FEB-90	2124	107	FEA	114	10	2.430,00
112979	12-OCT-89	2114	102	ACI	4100Z	6	15.000,00
113027	22-ENE-90	2103	105	ACI	41002	54	4.104,00
113007	08-ENE-90	2112	108	IMM	773C	3	2.925,00
113069	02-MAR-90	2109	107	IMM	775C	22	31.350,00
113034	29-ENE-90	2107	110	REI	2A45C	8	632,00
112992	04-NOV-89	2118	108	ACI	41002	10	760,00
112975	12-0CT-89	2111	103	REI	2A44G	6	2.100,00
113055	15-FEB-90	2108	101	ACI	4100X	6	150,00
113048	10-FEB-90	2120	102	IMM	779C	2	3.750,00
112993	04-ENE-89	2106	102	REI	2A45C	24	1.896,00
113065	27-FEB-90	2106	102	QSA	XK47	6	2.130,00
113003	25-ENE-90	2108	109	IMM	779C	3	5.625,00
113049	10-FEB-90	2118	108	QSA	XK47	2	776,00
112987	31-DIC-89	2103	105	ACI	4100Y	11	27.500,00
113057	18-FEB-90	2111	103	ACI	4100X	24	600,00
113042	02-FEB-90	2113	101	REI	2A44R	5	22.500,00

REPVENTAS

NUM_EMPL	NOMBRE	EDAD	OFICINA_REP	TÍTULO	CONTRATO	DIRECTOR	CUOTA	VENTAS
105	Bill Adams	37	13	Rep Ventas	12-FEB-88	104	350.000,00	367.911,00
109	Mary Jones	31	11	Rep entas	12-OCT-99	101	300.00,00	392.725,00
102	Sue Smith	48	21	Rep Ventas	10-DIC-86	108	350.000,00	474.050,00
106	Sam Clark	52	11	VP Ventas	14-JUN-88	NULL	275.000,00	299.912,00
104	Bob Smith	33	12	Dir Ventas	19-MAY-87	106	200.000,00	142.594,00
101	Dan Roberts	45	12	Rep Ventas	20-OCT-86	104	300.000,00	305673,00
110	Tom Snyder	41	NULL	Rep Ventas	13-ENE-90	101	NULL	75.985,00
108	Larry Fitch	62	21	Dir Ventas	12-OCT-89	106	350.000,00	361.865,00
103	Paul Cruz	29	12	Rep Ventas	01-MR-87	104	275.000,00	286.775,00
107	Nancy Angelli	49	22	Rep Ventas	14-NOV-88	108	300.000,00	186.042,0

OFICINAS

OFICINA	CIUDAD	REGIÓN	DIR	OBJETIVO	VENTAS
22	Denver	Oeste	108	300.000,00	186.042,00
11	New York	Este	106	575.000,00	692.637,00
12	Chicago	Este	104	800.000,00	735.042,00
13	Atlanta	Este	105	350.000,00	367.911,00
21	Los Angeles	Oeste	108	725.000,00	835.915,00

© Alfaomega - RC Libros

SYNAPS PREMIUM

Una vez diseñada la base de datos, se crean sus tablas y se introducen sus campos y registros mediante la sintaxis SQL que se especifica a continuación:

```
CREATE TABLE clientes (
Num_clie            integer not null PRIMARY KEY,
Empresa             char(20) not null,
Repclie             integer,
Limite_credito      numeric
);

INSERT INTO clientes VALUES (2101, 'Jones Mfg.', 106, 65000);
INSERT INTO clientes VALUES (2102, 'First. Corp.', 101, 65000);
INSERT INTO clientes VALUES (2103, 'Acme Mfg.', 105,       50000);
INSERT INTO clientes VALUES (2105, 'AAA Investments', 101, 45000);
INSERT INTO clientes VALUES (2106, 'Fred Lewis Corp.', 102, 65000);
INSERT INTO clientes VALUES (2107, 'Ace International', 110, 35000);
INSERT INTO clientes VALUES (2108, 'Holm Landis', 109, 55000);
INSERT INTO clientes VALUES (2109, 'Chen Associates', 103, 25000);
INSERT INTO clientes VALUES (2111, 'JCP Inc.', 103, 50000);
INSERT INTO clientes VALUES (2112, 'Zetacorp', 108, 50000);
INSERT INTO clientes VALUES (2113, 'Ian Schmidt', 104, 20000);
INSERT INTO clientes VALUES (2114, 'Orion Corp.', 102, 20000);
INSERT INTO clientes VALUES (2115, 'Smithson Corp.', 101, 20000);
INSERT INTO clientes VALUES (2117, 'J.P. Sinclair', 106, 35000);
INSERT INTO clientes VALUES (2118, 'Midwest Systems', 108, 60000);
INSERT INTO clientes VALUES (2119, 'Solomon Inc.', 109, 25000);
INSERT INTO clientes VALUES (2120, 'Rico Enterprises', 102, 50000);
INSERT INTO clientes VALUES (2121, 'QMA Assoc.', 103, 45000);
INSERT INTO clientes VALUES (2122, 'Three-Way Lines', 105, 30000);
INSERT INTO clientes VALUES (2123, 'Carter Sons', 102, 40000);
INSERT INTO clientes VALUES (2124, 'Peter Brothers',       107, 40000);

CREATE TABLE repventas (
Num_empl     integer not null PRIMARY KEY,
Nombre       char(15) not null,
Edad         integer,
Oficina_rep integer,
Titulo       char(10),
Contrato     date not null,
Director     integer,
Cuota        numeric,
Ventas       numeric not null
);

INSERT INTO repventas VALUES  (101, 'Dan Roberts', 45, 12, 'Rep Ventas', '20-
10-1986', 104, 300000, 305673);
INSERT INTO repventas VALUES  (102, 'Sue Smith', 48, 21, 'Rep Ventas', '12-12-
1986', 108, 350000, 474050);
INSERT INTO repventas VALUES (103, 'Paul Cruz', 29, 12, 'Rep Ventas', '01-
04.1987', 104, 275000, 286775);
INSERT INTO repventas VALUES (104, 'Bob Smith', 33, 12, 'Dir Ventas', '19-05-
1987', 106, 200000, 142594);
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

```
INSERT INTO repventas VALUES (105, 'Bill Adams', 37, 13, 'Rep Ventas', '12-02-
1988', 104,  350000, 367911);
INSERT INTO repventas VALUES (106, 'Sam Clark', 52, 11, 'VP Ventas', '14.06-
1988', null, 275000, 299912);
INSERT INTO repventas VALUES (107, 'Nancy Angelli', 49, 22, 'Rep Ventas',
'14.11-1988', 108, 300000, 186042);
INSERT INTO repventas VALUES (108, 'Larry Fitch', 62, 21, 'Dir Ventas', '12-10-
1989', 106, 350000, 361865);
INSERT INTO repventas VALUES (109, 'Mary Jones', 31, 11, 'Rep Ventas', '12-10-
1999', 106, 300000, 392725);
INSERT INTO repventas VALUES (110, 'Tom Snyder', 41, null, 'Rep ventas',
'14.01-1990', 101, null, 75985);
```

```
CREATE TABLE oficinas (
Oficina     integer not null PRIMARY KEY,
Ciudad      char(15) not null,
Region      char(10) not null,
Dir         integer REFERENCES repventas (Num_empl),
Objetivo    number,
Ventas      number not null
);
```

```
INSERT INTO oficinas VALUES (11, 'New York','Este', 106, 575000, 692637);
INSERT INTO oficinas VALUES (12, 'Chicago', 'Este', 104, 800000, 735042);
INSERT INTO oficinas VALUES (13, 'Atlanta', 'Este', 105, 350000, 367911);
INSERT INTO oficinas VALUES (21, 'Los Ángeles', 'Oeste', 108, 725000, 835915);
INSERT INTO oficinas VALUES (22, 'Denver', 'Oeste', 108, 300000, 186042);
```

```
CREATE TABLE productos (
Id_fab              char(3) not null,
Id_producto         char(5) not null,
Descripcion         char(20) not null,
Precio              number not null,
Existencias         integer not null,
CONSTRAINT productos_Id_fab_Id_producto
   PRIMARY KEY (Id_fab, Id_producto)
);
```

```
INSERT INTO productos VALUES ('ACI', '41001', 'Artículo Tipo 1', 55,  277);
INSERT INTO productos VALUES ('ACI', '41002', 'Artículo Tipo 2', 76,  167);
INSERT INTO productos VALUES ('ACI', '41003', 'Artículo Tipo 3', 107,  207);
INSERT INTO productos VALUES ('ACI', '41004', 'Artículo Tipo 4', 117,  139);
INSERT INTO productos VALUES ('ACI', '4100X', 'Ajustador', 25, 37);
INSERT INTO productos VALUES ('ACI', '4100Y','Extractor', 2.750, 25);
INSERT INTO productos VALUES ('ACI', '4100Z','Montador', 2.500,  28);
INSERT INTO productos VALUES ('BIC', '41003','Manivela', 652, 3);
INSERT INTO productos VALUES ('BIC', '41089','Retn', 225, 78);
INSERT INTO productos VALUES ('BIC', '41672','Plate', 180, 0);
INSERT INTO productos VALUES ('FEA', '112  ','Cubierta', 148, 115);
INSERT INTO productos VALUES ('FEA', '114  ','Bancada Motor', 243, 15);
INSERT INTO productos VALUES ('IMM', '773C ', 'Riostra 1/2 Tm', 975, 28);
INSERT INTO productos VALUES ('IMM', '775C ', 'Riostra 1-Tm', 1.425, 5);
INSERT INTO productos VALUES ('IMM', '779C ', 'Riostra 2-Tm', 1.875, 9);
INSERT INTO productos VALUES ('IMM', '878H ','Soporte Riostra', 54, 223);
INSERT INTO productos VALUES ('IMM', '887P ','Perno Riostra', 250, 24);
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
INSERT INTO productos VALUES ('IMM', '887X ','Retenedor Riostra', 475, 32);
INSERT INTO productos VALUES ('QSA', 'XK47 ','Reductor', 355, 38);
INSERT INTO productos VALUES ('QSA', 'XK48 ','Reductor', 134, 203);
INSERT INTO productos VALUES ('QSA', 'XK48A','Reductor', 117, 37);
INSERT INTO productos VALUES ('REI', '2A44G', 'Pasador Bisagra', 350, 14);
INSERT INTO productos VALUES ('REI', '2A44L', 'Bisagra Izqda.', 4.500, 12);
INSERT INTO productos VALUES ('REI', '2A44R', 'Bisagra Dcha.', 4.500, 12);
INSERT INTO productos VALUES ('REI', '2A45C','V Stago Trinquete',  79, 210);
```

```
CREATE TABLE pedidos (
Num_pedido          integer not null PRIMARY KEY,
Fecha_pedido        date not null,
Clie                integer not null REFERENCES clientes (Num_clie),
Rep                 integer REFERENCES repventas (Num_empl),
Fab                 char(3) not null,
Producto            char(5) not null,
Cant                integer not null,
Importe             number not null
);
```

```
INSERT INTO pedidos VALUES (110036, '30-01-1990', 2107, 110, 'ACI', '4100Z',   9, 22500);
INSERT INTO pedidos VALUES (112961, '17-12-1989', 2117, 106, 'REI', '2A44L', 7, 31500);
INSERT INTO pedidos VALUES (112963, '17-12-1989', 2103, 105, 'ACI', '41004', 28, 3276);
INSERT INTO pedidos VALUES (112968, '12-10-1989', 2102, 101, 'ACI', '41004', 34, 3978);
INSERT INTO pedidos VALUES (112975, '12-10-1989', 2111, 103, 'REI', '2A44G', 6, 2100);
INSERT INTO pedidos VALUES (112979, '12-10-1989', 2114, 102, 'ACI', '4100Z', 6, 15000);
INSERT INTO pedidos VALUES (112983, '27-12-1989', 2103, 105, 'ACI', '41004',   6, 702);
INSERT INTO pedidos VALUES (112987, '31-12-1989', 2103, 105, 'ACI', '4100Y', 11, 27500);
INSERT INTO pedidos VALUES (112989, '04.01-1990', 2101, 106, 'FEA', '114', 6, 1458);
INSERT INTO pedidos VALUES (112992, '04.11-1989', 2118, 108, 'ACI', '41002', 10, 760);
INSERT INTO pedidos VALUES (112993, '04.01-1989', 2106, 102, 'REI', '2A45C', 24, 1896);
INSERT INTO pedidos VALUES (112997, '08-01-1990', 2124, 107, 'BIC', '41003', 1, 652);
INSERT INTO pedidos VALUES (113003, '25-01-1990', 2108, 109, 'IMM', '779C', 3, 5625);
INSERT INTO pedidos VALUES (113007, '08-01-1990', 2112, 108, 'IMM', '773C', 3, 2925);
INSERT INTO pedidos VALUES (113012, '11-01-1990', 2111, 105, 'ACI', '41003', 35, 3745);
INSERT INTO pedidos VALUES (113013, '14.01-1990', 2118, 108, 'BIC', '41003', 1, 652);
INSERT INTO pedidos VALUES (113024, '20-01-1990', 2114, 108, 'QSA', 'XK47', 20, 7100);
INSERT INTO pedidos VALUES (113027, '22-01-1990', 2103, 105, 'ACI', '41002',   54, 4104);
INSERT INTO pedidos VALUES (113034, '29-01-1990', 2107, 110, 'REI', '2A45C', 8, 632);
INSERT INTO pedidos VALUES (113042, '02-02-1990', 2113, 101, 'REI', '2A44R', 5, 22500);
INSERT INTO pedidos VALUES (113045, '02-02-1990', 2112, 108, 'REI', '2A44R', 10, 45000);
INSERT INTO pedidos VALUES (113048, '10-02-1990', 2120, 102, 'IMM', '779C', 2, 3750);
INSERT INTO pedidos VALUES (113049, '10-02-1990', 2118, 108, 'QSA', 'XK47', 6, 2130);
INSERT INTO pedidos VALUES (113051, '10-02-1990', 2118, 108, 'QSA', 'XK47', 4, 1420);
INSERT INTO pedidos VALUES (113055, '15-02-1990', 2108, 101, 'ACI', '4100X', 6, 150);
INSERT INTO pedidos VALUES (113057, '18-02-1990', 2111, 103, 'ACI', '4100X', 24, 600);
INSERT INTO pedidos VALUES (113058, '24.02-1990', 2108, 109, 'FEA', '112', 10, 1480);
INSERT INTO pedidos VALUES (113062, '24.02-1990', 2124, 107, 'FEA', '114', 10, 2430);
INSERT INTO pedidos VALUES (113065, '27-02-1990', 2106, 102, 'QSA', 'XK47', 6, 2130);
INSERT INTO pedidos VALUES (113069, '02-04.1990', 2109, 107, 'IMM', '775C', 22, 31350);
```

Ejercicio 2.2. Construir la base de datos en un contexto educativo estudiada como ejemplo en este capítulo a partir de la página 98. Crear las tablas según las condiciones ya especificadas y rellenar sus registro según los datos especificados.

La sintaxis SQL para crear las tablas y añadir sus registros es la siguiente:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
CREATE TABLE departamento (
Dept                char(4) not null PRIMARY KEY,
Dedif               char(2) not null,
Ddespacho           integer not null,
Dchfno              char(3)
);
```

```
INSERT INTO departamento VALUES ('THEO', 'HU', 200, '10');
INSERT INTO departamento VALUES ('CIS', 'SC', 300, '80');
INSERT INTO departamento VALUES ('D.G.', 'SC', 100, ' ');
INSERT INTO departamento VALUES ('PHIL', 'HU', 100, '60');
```

```
CREATE TABLE curso (
Cno                 char(3) not null PRIMARY KEY,
Cnombre             char(22) not null,
Cdescp              char(25) not null,
Cred                integer not null,
Ctarifa             number not null,
Cdept               char(4) not null REFERENCES departamento (Dept)
);
```

```
INSERT INTO curso VALUES ('C11', 'INTROD. A LAS CC.', 'PARA NOVATOS', 3, 100, 'CIS');
INSERT INTO curso VALUES ('C22', 'ESTRUCT. DE DATOS', 'MUY UTIL', 3, 50, 'CIS');
INSERT  INTO curso VALUES ('C33', 'MATEMÁTICAS DISCRETAS', 'ABSOLUTAMENTE NECESARIO', 3, 0, 'CIS');
INSERT INTO curso VALUES ('C44', 'CIRCUITOS DIGITALES', 'AH HA!', 3, 0, 'CIS');
INSERT INTO curso VALUES ('C55', 'ARQUITECT. COMPUTADOR', 'MAQ. VON NEUMANN', 3, 100, 'CIS');
INSERT INTO curso VALUES ('C66', 'BASES DE DATOS RELAC.', 'IMPRESCINDIBLE', 3, 500, 'CIS');
INSERT INTO curso VALUES ('P11', 'EMPIRISMO', 'VERLO PARA CREERLO', 3, 100, 'PHIL');
INSERT INTO curso VALUES ('P22', 'RACIONALISMO', 'PARA USARLOS CIS', 3, 50, 'PHIL');
INSERT INTO curso VALUES ('P33', 'EXISTENCIALISMO', 'PARA USARLOS CIS', 3, 200, 'PHIL');
INSERT INTO curso VALUES ('P44', 'SOLIPSISMO', 'PARA MI MISMO', 6, 0, 'PHIL');
INSERT INTO curso VALUES ('T11', 'ESCOLASTICISMO', 'PARA BEATOS', 3, 150, 'THEO');
INSERT INTO curso VALUES ('T22', 'FUNDAMENTALISMO', 'PARA DESCUIDADOS', 3, 90, 'THEO');
INSERT INTO curso VALUES ('T33', 'HEDONISMO', 'PARA SANOS', 3, 0, 'THEO');
INSERT INTO curso VALUES ('T44', 'COMUNISMO', 'PARA AVAROS', 6, 200, 'THEO');
```

```
CREATE TABLE claustro (
Fno                 char(3) not null PRIMARY KEY,
Fnombre             char(15) not null,
Fdomi               char(20) not null,
Ffcanti             date not null,
Fnumdep             integer,
Fsueldo             number not null,
Fdept               char(4) not null REFERENCES departamento (Dept)
);
```

```
INSERT INTO claustro VALUES ('06', 'KATHY PEPE', 'CALLE LA PIEDRA, 7', '15-01-1979', 2, 35000, 'PHIL');
INSERT INTO claustro VALUES ('10', 'JESSIE MARTIN', 'Dr. DEL ESTE, 4', '01-09-1969', 2, 45000, 'THEO');
INSERT INTO claustro VALUES ('08', 'JOSE COHN', 'AP. CORREOS 1138', '09-07-1979', 2, 35000, 'CIS');
INSERT INTO claustro VALUES ('85', 'AL HARTLEY', 'CALLE DE LA PLATA', '05-09-1979', 7, 45000, 'CIS');
INSERT INTO claustro VALUES ('60', 'JULIA MARTIN', 'Dr. DEL ESTE, 4', '01-09-1969', 1, 45000, 'PHIL');
INSERT INTO claustro VALUES ('65', 'LISA BOBAK', 'CAMINO RISA, 77', '06-09-1981', 0 , 36000, 'THEO');
INSERT INTO claustro VALUES ('80', 'BARB HLAVATY', 'CALLE DEL SUR, 489', '16-01-1982', 3, 35000, 'CIS');
```

```
CREATE TABLE clase (
Cno                 char(3) not null REFERENCES curso (Cno),
Sec                 char(2) not null,
```


SYNAPS PREMIUM

```
Cinstrfno           char(2) not null,
Cdia                char(2) not null,
Chora               char(14) not null,
Cedif               char(2) not null,
Cdespacho           integer not null,
CONSTRAINT clase_Cno_Sec PRIMARY KEY (Cno, Sec)
);
```

```
INSERT INTO clase VALUES ('C11', '01', '08', 'LU', '08:00-09:00 AM', 'SC', 305);
INSERT INTO clase VALUES ('C11', '02', '08', 'MA', '08:00-09:00 AM', 'SC', 306);
INSERT INTO clase VALUES ('C33', '01', '80', 'MI', '09:00-10:00 AM', 'SC', 305);
INSERT INTO clase VALUES ('C55', '01', '85', 'JU', '11:00-12:00 AM', 'HU', 306);
INSERT INTO clase VALUES ('P11', '01', '06', 'JU', '09:00-10:00 AM', 'HU', 102);
INSERT INTO clase VALUES ('P33', '01', '06', 'VI', '11:00-12:00 AM', 'HU', 201);
INSERT INTO clase VALUES ('T11', '01', '10', 'LU', '10:00-11:00 AM', 'HU', 101);
INSERT INTO clase VALUES ('T11', '02', '65', 'LU', '10:00-11:00 AM', 'HU', 102);
INSERT INTO clase VALUES ('T33', '01', '65', 'MI', '11:00-12:00 AM', 'HU', 101);
```

```
CREATE TABLE estudiante (
Sno                 char(3) not null PRIMARY KEY,
Snombre             char(30) not null,
Sdomi               char(15) not null,
Stlfno              char(12) not null,
Sfnacim             char(6) not null,
Siq                 integer not null,
Sadvfno             char(3) not null,
Sesp                char(4) not null REFERENCES departamento (Dept)
);
```

```
INSERT INTO estudiante VALUES ('325', 'CURLEY DUBAY', 'CONNECTICUT',
'204.124.4567', '780517', 122, '10', 'THEO');
INSERT INTO estudiante VALUES ('150', 'LARRY DUBAY', 'CONNECTICUT',
'204.124.4567', '780517', 121, '80', 'CIS');
INSERT INTO estudiante VALUES ('100', 'MOE DUBAY', 'CONNECTICUT',
'204.124.4567', '780517', 120, '10', 'THEO');
INSERT INTO estudiante VALUES ('800', 'ROCKY BALBOA', 'PENSYLVANIA', '112-112-
1122', '461004', 99, '60', 'PHIL');
```

```
CREATE TABLE matricula (
Cno                 char(3) not null,
Sec                 char(2) not null,
Sno                 char(3) not null REFERENCES estudiante (Sno),
Fecha_mat           date not null,
Hora_mat            char(10) not null,
CONSTRAINT matricula_Cno_Sec_Sno PRIMARY KEY (Cno, Sec, Sno)
);
```

```
INSERT INTO matricula VALUES ('C11', '01', '325', '04.01-1987', '09.41.30');
INSERT INTO matricula VALUES ('C11', '01', '800', '15-12-1987', '11.49.00');
INSERT INTO matricula VALUES ('C11', '02', '100', '17-12-1987', '09.32.00');
INSERT INTO matricula VALUES ('C11', '02', '150', '17-12-1987', '09.32.30');
INSERT INTO matricula VALUES ('P33', '01', '100', '24.12-1987', '11.30.00');
INSERT INTO matricula VALUES ('P33', '01', '800', '24.12-1987', '11.23.00');
INSERT INTO matricula VALUES ('T11', '01', '100', '24.12-1987', '11.21.00');
INSERT INTO matricula VALUES ('T11', '01', '150', '15-12-1987', '11.35.30');
INSERT INTO matricula VALUES ('T11', '01', '800', '15-12-1987', '14.00.00');
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

```
CREATE TABLE personal (
Enombre              char(15) not null PRIMARY KEY,
Cargo                char(10) not null,
Esueldo              integer not null,
Dept                 char(4)
);

INSERT INTO personal VALUES ('LUCAS', 'EVANG1', 53, 'THEO');
INSERT INTO personal VALUES ('MARCOS', 'EVANG2', 52, 'THEO');
INSERT INTO personal VALUES ('MATEO', 'EVANG3', 51, 'THEO');
INSERT INTO personal VALUES ('DICK NIX', 'LADRÓN', 25001, 'PHIL');
INSERT INTO personal VALUES ('HANK KISS', 'BUFÓN', 25000, 'PHIL');
INSERT INTO personal VALUES ('JUAN', 'EVANG4', 54, 'THEO');
INSERT INTO personal VALUES ('EUCLIDES', 'AYTE. LAB.', 1000, 'MATH');
INSERT INTO personal VALUES ('ARQUIMEDES', 'AYTE. LAB.', 200, 'ENG');
INSERT INTO personal VALUES ('DAVINCI', 'AYTE. LAB.', 500, ' ');
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

CAPÍTULO 3

ELEMENTOS DE ORACLE SQL. FORMATOS, TIPOS DE DATOS, OPERADORES Y FUNCIONES

UTILIZANDO ORACLE SQL

Oracle SQL es un superconjunto de SQL-92 estándar definido por la American National Standards Institute (ANSI) y la International Standards Organization (ISO). Oracle utiliza el SQL (Structured Query Language), lenguaje de base de datos para almacenar y recuperar información. Incluye las siguientes categorías de sentencias:

- DDL (*Data Definition Language* o *Lenguaje de Definición de Datos*). Utilizadas para crear, alterar, o borrar objetos de base de datos, tales como esquemas, tablas, columnas, vistas y secuencias. Como ejemplos característicos tenemos las instrucciones o comandos CREATE, ALTER y DROP.

- DML (*Data Modification Language* o *Lenguaje de Modificación de Datos*). Utilizadas en la interrogación y manipulación de datos en esquemas ya existentes. Como ejemplos característicos tenemos los comandos SELECT, INSERT, UPDATE y DELETE.

SYNAPS PREMIUM

- TCL (*Transaction Control Language* o *Lenguaje de Control de Transacciones*). Utilizadas para confirmar o restaurar transacciones de base de datos, que son unidades de trabajo que realizan una o más sentencias SQL relacionadas entre sí. Como ejemplos característicos tenemos los comandos COMMIT y ROLLBACK.

- DCL (*Data Control Language* o *Lenguaje de Control de Datos*). Utilizadas en el control de acceso a datos en la base de datos. Como ejemplos característicos tenemos los comandos GRANT y REVOKE.

ELEMENTOS BÁSICOS DE ORACLE SQL

En Oracle SQL se trabaja con distintos tipos de elementos, que podrían clasificarse básicamente como sigue:

- **Literales:** en Oracle los términos *literal* y *valor constante* son sinónimos y se refieren a valores fijos. Puede haber literales carácter (se encierran entre comillas simples) y literales numéricos. Por ejemplo, 50001 es un literal numérico y '101' es un literal carácter.

- **Texto:** literal tipo carácter que debe ser introducido entre comillas simples. Oracle define el texto generalmente como tipo CHAR, siendo la longitud máxima admisible para un texto 4.000 bytes. Ejemplos de texto son: *'Hello', 'Oracle.dbs', 'Jackie''s raincoat'* y *'09-MAR-92'*.

- **Números:** Oracle permite números positivos (precedidos del signo + o de nada), negativos (precedidos del signo -) y en notación científica (incluyendo las letras *e* o E justamente antes de las cifras del exponente) hasta un máximo de precisión de 38 dígitos, utilizando los dígitos entre 0 y 9. El rango permisible para el exponente (potencia de 10) está entre -130 y 125. Como ejemplos de números tenemos: 25, +6.34, 0.5, 25e-03 y -1. Oracle define los números generalmente como tipo NUMBER, con la sintaxis NUMBER(p,s), donde *p* es la precisión (número máximo de dígitos a la izquierda del punto decimal) y *s* es la escala (número de dígitos a la derecha del punto decimal), siendo NUMBER(p,0) = NUMBER(p). Oracle también permite números en punto flotante tipo FLOAT con precisión decimal 38 y precisión binaria 126. El tipo FLOAT(b) especifica precisión binaria de valor *b*, y para pasar a precisión decimal se multiplica *b* por log(2)=0,30103. Para pasar de precisión decimal a binaria se multiplica por 3,32193.

© Alfaomega - RC Libros

SYNAPS PREMIUM

- **Enteros:** Oracle permite números enteros hasta un máximo de precisión de 38 dígitos.

- **Intervalos:** Oracle permite intervalos literales que especifican períodos de tiempo como diferencias en términos de años y meses (YEAR TO MONTH) o en términos de días, horas, minutos y segundos (DAY TO SECOND, DAY TO MINUTE y DAY TO HOUR). Como ejemplos consideramos INTERVAL '123-2' YEAR TO MONTH que indica un intervalo de 123 días y 2 meses: INTERVAL '4 5:12:10.222' DAY TO SECOND que indica un intervalo de 4 días 5 horas 12 minutos 10 segundos y 222 milésimas de segundo, INTERVAL '4 5:12' DAY TO MINUTE que indica 4 días, 5 horas y 12 minutos, e INTERVAL '400 5' DAY(3) TO HOUR que indica 400 días y 5 horas.

- **Tipos de datos:** cada literal utilizado en Oracle lleva asociado un tipo de dato. En un párrafo posterior se expondrán los tipos de datos utilizados en Oracle.

- **Nulos:** un nulo es un elemento que no se especifica en una tabla. No se puede confundir un valor nulo con el valor cero.

- **Comentarios:** ilustran la sintaxis Oracle, pero no afectan a su ejecución. Comienzan con /* y terminan con */ pudiendo ocupar múltiples líneas. También es posible que comiencen con dos guiones --, pero en este caso no pueden pasar a la línea siguiente, considerándose comentario lo incluido entre – y el final de línea.

- **Pseudocolumnas:** se comportan como las columnas de una tabla, pero no se almacenan en las tablas. Se pueden seleccionar pseudocolumnas, pero no se pueden actualizar, insertar o borrar sus valores. Son pseudocolumnas en Oracle las siguientes: CURRVAL (devuelve el valor corriente de una secuencia), NEXTVAL (incrementa la secuencia y devuelve el valor siguiente), LEVEL (vale 1 para un nodo raíz en una fila devuelta por una *query* jerárquica, vale 2 para los nodos hijos, y así sucesivamente), ROWID (identificador de fila) y ROWNUM (número que indica el orden en el cual Oracle selecciona la fila de una tabla o de un conjunto de tablas unidas).

- **Objetos de base de datos:** existen *objetos asociados a esquemas* (clústeres, restricciones, enlaces, disparadores o *triggers*, dimensiones, *snapshots*, librerías de procedimientos externos, clases Java, índices, paquetes, secuencias, funciones almacenadas, procedimientos almacenados, restricciones de integridad, operadores, sinónimos, tablas y vistas) y *objetos no asociados a esquemas*

© Alfaomega - RC Libros

SYNAPS PREMIUM

(contextos, directorios, perfiles, papeles o roles, segmentos rollback, espacios de tablas y usuarios).

- **Esquemas:** Oracle organiza lógicamente (no físicamente) objetos relacionados dentro de un esquema de bases de datos. Por ejemplo, es típico organizar en el mismo esquema todos los elementos de una aplicación (tablas, objetos, etc.). Para más sencillez, en Oracle el concepto de esquema de base de datos está ligado al concepto de usuario de base de datos, correspondiéndose unívocamente cada esquema con una cuenta de usuario, de forma que el usuario y el esquema asociado tienen el mismo nombre.

- **Formatos:** existen distintos formatos numéricos y de fechas que se pueden utilizar en el lenguaje SQL de Oracle. Los distintos formatos pueden ser utilizados como argumentos de funciones SQL como TO_DATE, TO_NUMBER, TO_CHAR y TRUNC.

FORMATOS EN ORACLE SQL

Existen distintos formatos numéricos y de fechas que se pueden utilizar en el lenguaje SQL de Oracle. Los distintos formatos pueden ser utilizados como argumentos de distintas funciones SQL. Por ejemplo, *SELECT TO_CHAR(número, 'formato') FROM DUAL* permite expresar el *número* en el *formato* especificado. Los formatos más importantes se describirán en los párrafos siguientes.

Formatos de elementos numéricos

Elemento	*Ejemplo*	*Descripción*
9	9999	El número de nueves significa el número de dígitos significativos devueltos. Si el número es positivo, el encabezamiento será un espacio, y si es negativo, el encabezamiento será un signo menos.
0	0999 ó 9990	Devuelve ceros al principio o al final.
$	$9999	Prefijo del valor con un símbolo dólar (magnitudes dinerarias).
B	B9999	Devuelve valores cero como blancos (devuelve blancos para la parte entera de números en punto fijo cuando dicha parte entera es 0).
MI	9999MI	Devuelve "-" después de valores negativos y blanco después de valores positivos.
S	S9999	Devuelve para valores positivos precedidos de "+" y valores negativos precedidos de "-".
PR	9999PR	Devuelve los valores negativos entre corchetes angulares < > y los positivos precedidos de un blanco.

© Alfaomega - RC Libros

SYNAPS PREMIUM

D	99D99	Devuelve el carácter decimal en la posición especificada, separando la parte entera y la parte fraccionaria de un número.
G	9G999	Devuelve el grupo separador en la posición especificada.
C	C999	Devuelve el símbolo ISO para la moneda.
L	L999	Devuelve el símbolo local para la moneda en la posición dada.
, (comma)	9,999	Devuelve una coma en la posición especificada.
. (period)	99.99	Devuelve un período, separando las partes integral y fraccional de un número en la posición especificada.
FM	FM90.9	Devuelve un valor sin blancos al principio ni al final.
RN	RN	Devuelve un valor como numerales romanos en mayúsculas.
rn	rn	Devuelve un valor como numerales romanos en minúsculas.
TM	TM	Devuelve el menor número posible de caracteres.
U	U9999	Devuelve en la posición especificada el símbolo de Euro u otro símbolo de moneda especificado como el símbolo corriente.
V	999V99	Devuelve un valor multiplicado por 10^n siendo *n* el número de nueves después de la V.
X	XXXX	Devuelve el valor hexadecimal del número de dígitos especificado.
EEEE	9.999EEEE	Devuelve un valor en notación científica.

A continuación, se presentan varios ejemplos en los que la función TO_CHART con la sintaxis *SELECT TO_CHAR(número, 'formato') FROM DUAL* permite expresar el *número* dado en el *formato* numérico especificado.

número	*'formato'*	*Resultado*
-1234567890	9999999999S	'1234567890-'
0	99.99	' .00'
+0.1	99.99	' .10'
-0.2	99.99	' -.20'
0	90.99	' 0.00'
+0.1	90.99	' 0.10'
-0.2	90.99	' -0.20'
0	9999	' 0'
1	9999	' 1'
0	B9999	' '
1	B9999	' 1'
0	B90.99	' '
+123.456	999.999	' 123.456'
-123.456	999.999	'-123.456'
+123.456	FM999.009	'123.456'
+123.456	9.9EEEE	' 1.2E+02'
+1E+123	9.9EEEE	' 1.0E+123'

© Alfaomega - RC Libros

SYNAPS PREMIUM

+123.456	FM9.9EEEE	'1.2E+02'
+123.45	FM999.009	'123.45'
+123.0	FM999.009	'123.00'
+123.45	L999.99	' $123.45'
+123.45	FML999.99	'$123.45'
+1234567890	9999999999S	'1234567890+'

En la ventana de SQL, las transformaciones tendrían el siguiente aspecto:

```
SQL> select to_char(+123.456, '9.9EEEE') from dual;

TO_CHAR(+
---------
  1.2E+02

SQL> select to_char(-0.2, '99.99') from dual;

TO_CHA
------
  -.20

SQL>
```

Formatos de fechas

Elemento	*Descripción*
SCC ó CC	Siglo; "S" prefija BC fechas con "-".
YYYY ó SYYYY	Años de cuatro dígitos; "S" prefija BC fechas con "-".
IYYY	Años de cuatro dígitos basados en la ISO estándar.
YYY ó YY ó Y	Últimos 3, 2 o 1 dígito(s) para el año.
IYY ó IY ó I	Últimos 3, 2 o 1 dígito(s) para el año en notación ISO estándar.
Y,YYY	Año con coma.
Q	Trimestre (cuarto de año (1, 2, 3, 4; JAN-MAR = 1)).
MM	Mes (01-12; JAN = 01).
MONTH	Nombre del mes; relleno con blancos hasta una longitud de 9 caracteres.
MON	Nombre abreviado del mes.
WW	Semanas del año (1-53) donde la semana 1 comienza en el primer día del año y continúa hasta el séptimo día.
IW	Semanas del año (1-52 o 1-53) basada en la ISO estándar.
W	Semanas del mes (1-5) donde la semana 1 comienza en el primer día del año y continúa hasta el séptimo día.
DDD	Día del año (1-366).

© Alfaomega - RC Libros

SYNAPS PREMIUM

DD	Día del mes (1-31).
D	Día de la semana (1-7).
DAY	Nombre del día, completado con blancos hasta una longitud de 9 caracteres.
DY	Nombre abreviado del día.
AM o PM	Indicador de meridiano.
A.M. o P.M.	Indicador de meridiano con períodos.
HH o HH12	Hora del día (1-12).
HH24	Hora del día (0-23).
MI	Minuto (0-59).
RR	Últimos dos dígitos del año; para años en otros estados.
RRRR	Año redondeado.
RM	Mes numeral romano (I-XII,JAN=1).
SS	Segundos (0-59).
SSSSS	Segundos después de medianoche (0-86399).
- / . ; : "texto"	La puntuación y el texto entre comillas se reproducen en el resultado.

La función TO_DATE transforma un valor carácter en un valor fecha y la función TO_CHAR transforma un valor fecha en un valor carácter. La sintaxis operativa de estas funciones de transformación podría ser la siguiente:

SELECT TO_DATE(caracter, 'formato') FROM DUAL;
SELECT TO_CHAR(fecha, 'formato') FROM DUAL;

A continuación, se presentan algunos ejemplos de transformaciones de formatos de fechas en la ventana SQL:

```
SQL> SELECT TO_CHAR(TO_DATE('27-OCT-98', 'DD-MON-RR') ,'YYYY') "Year"
  2    FROM DUAL;

Year
----
1998

SQL> SELECT TO_CHAR(TO_DATE('27-OCT-17', 'DD-MON-RR') ,'YYYY') "Year"
  2       FROM DUAL;

Year
----
2017

SQL>
```


SYNAPS PREMIUM

TIPOS DE DATOS EN ORACLE SQL

Cada valor manipulado por Oracle tiene asociado un tipo de dato que aporta un conjunto de propiedades a dicho valor. Cuando se crea una tabla (comando CREATE TABLE) se debe especificar el tipo de dato para cada una de sus columnas. Cuando se crea un procedimiento o función será necesario especificar un tipo de dato para cada uno de sus argumentos. Estos tipos de datos definen un dominio de valores que cada columna puede contener o cada argumento puede tener. La tabla siguiente presenta distintas categorías de tipos de datos de Oracle SQL.

Código	*Tipo de dato incorporado*	*Descripción*
1	VARCHAR2 (tamaño)	Variable carácter de longitud máxima en bytes la especificada en *tamaño*. El máximo tamaño es 4.000 y el mínimo es 1. Se debe especificar *tamaño* para VARCHAR2.
1	NVARCHAR2 (tamaño)	Variable carácter de longitud máxima en caracteres o bytes la especificada en *tamaño*, dependiendo de la elección de un conjunto de caracteres nacionales. El máximo tamaño está especificado por el número de caracteres o bytes requerido para almacenar cada carácter con un límite superior de 4.000 bytes. Se debe especificar *tamaño* NVARCHAR2.
2	NUMBER (p,s)	Número con precisión *p* y escala *s* (número de dígitos a la derecha del punto decimal). El rango de p está entre 1 y 38. La escala *s* puede variar desde -84 a 127.
8	LONG	Carácter de longitud variable de hasta 2 gigabytes o 2^{31} - 1 bytes.
12	DATE	Fecha con rango válido desde January 1, 4712 BC hasta December 31, 9999 AD.
180	TIMESTAMP (frac_seg_prec)	Año, Mes y Día, valores de fecha, así como hora. Minuto y segundo, valores de hora, donde frac_seg_prec es el número de dígitos en la parte fraccionaria del campo fecha-hora SECOND.
182	INTERVAL YEAR (p) TO MONTH	Período de tiempo en años y meses con precisión p (p puede variar de 0 a 9 y por defecto vale 2).
183	INTERVAL DAY (p) TO MONTH	Período de tiempo en días, horas, minutos y segundos con precisión p (p puede variar de 0 a 9 y por defecto vale 2).
23	RAW (tamaño)	Datos binarios brutos de longitud en bytes la especificada en tamaño. El máximo tamaño es 2.000 bytes.
24	LONG RAW	Datos binarios brutos de longitud en bytes hasta 2 gigabytes.

© Alfaomega - RC Libros

SYNAPS PREMIUM

69	ROWID	Cadena hexadecimal representando la única dirección de una fila en su tabla.
208	UROWID [(tamaño)]	Cadena hexadecimal representando la dirección lógica de una fila en una tabla organizada con índices. El máximo tamaño por defecto es 4.000 bytes.
96	CHAR (tamaño)	Dato carácter de longitud fija en bytes la especificada en *tamaño*. El máximo tamaño es 2.000 bytes. El mínimo tamaño y tamaño por defecto es 1 byte.
96	NCHAR (tamaño)	Dato carácter de longitud fija en caracteres o bytes la especificada en *tamaño*, dependiendo de si se elige un conjunto de caracteres nacionales. El máximo tamaño está determinado por el número de bytes requeridos para almacenar cada carácter con un límite superior de 2.000 bytes. El mínimo tamaño y tamaño por defecto es 1 carácter o 1 byte, dependiendo del conjunto de caracteres.
112	CLOB	Un objeto carácter largo conteniendo caracteres simples en bytes. Conjuntos de caracteres de ancho fijo y ancho variables son permisibles, ambos usando el conjunto de caracteres de base de datos CHAR. El tamaño máximo es 4 gigabytes.
112	NCLOB	Un objeto carácter largo conteniendo caracteres múltiples en bytes. Conjuntos de caracteres de ancho fijo y ancho variables son permisibles, ambos usando el conjunto de caracteres de base de datos NCHAR. El tamaño máximo es 4 gigabytes. Almacena conjuntos de datos de caracteres nacionales.
113	BLOB	Un objeto binario largo. El tamaño máximo es 4 gigabytes.
114	BFILE	Contiene un localizador de un fichero binario largo almacenado fuera de la base de datos. Posibilita *byte stream I/O access* a LOBs externos residiendo en la base de datos servidora. El máximo tamaño es 4 gigabytes.

Los códigos especificados para los distintos tipos de datos son utilizados internamente por ORACLE. El código de tipo de datos de una columna u objeto atributo es devuelto cuando se utiliza la función DUMP.

Como ejemplos se presentan los siguientes:

- `CREATE TABLE tab1 (col1 NCHAR(30));`

© Alfaomega - RC Libros

SYNAPS PREMIUM

Crea una tabla con una columna carácter de tipo NCHAR que almacena cadenas de hasta 30 caracteres de longitud:

- `CREATE TABLE tab1 (col1 NVARCHAR2(2000));`

Crea una tabla con una columna NVARCHAR2 de 2.000 caracteres de longitud (almacenada como 4.000 bytes, porque cada carácter ocupa dos bytes).

Tipos de datos ANSI, DB2 y SQL/DS

Las instrucciones SQL que crean tablas y clusteres pueden también utilizar tipos de datos ANSI y tipos de datos de productos IBM como SQL/DS y DB2. Oracle reconoce estos tipos de datos y los registra basándose en las reglas de conversión especificadas en la tabla siguiente:

Tipos de datos ANSI SQL	*Tipos de datos Oracle*
CHARACTER(n) CHAR(n)	CHAR(n)
CHARACTER VARYING(n) CHAR VARYING(n)	VARCHAR(n)
NATIONAL CHARACTER(n) NATIONAL CHAR(n) NCHAR(n)	NCHAR(n)
NATIONAL CHARACTER VARYING(n) NATIONAL CHAR VARYING(n) NCHAR VARYING(n)	NVARCHAR2(n)
NUMERIC(p,s) DECIMAL(p,s)	NUMBER(p,s)
INTEGER INT SMALLINT	NUMBER(38)
FLOAT(b) DOUBLE PRECISION REAL	NUMBER

Los tipos de datos NUMERIC y DECIMAL especifican solo números de punto fijo con s=0 por defecto. El tipo de datos FLOAT es un número en punto flotante con una precisión binaria b. La precisión por defecto es 126 binaria o 38 decimal. El tipo de dato DOUBLE PRECISION es un número en punto flotante con precisión binaria 126. El tipo de dato REAL es un número en punto flotante con precisión binaria 63 o 18 de precisión decimal.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Tipos de datos SQL/DS or DB2	*Tipos de datos Oracle*
CHARACTER(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
LONG VARCHAR(n)	LONG
DECIMAL(p,s)	NUMBER(p,s)
INTEGER SMALLINT	NUMBER(38)
FLOAT(b)[b]	NUMBER

Caracteres ASCII y EBCDIC en ORACLE SQL

En Oracle también se puede trabajar con caracteres ASCII y EBCDIC, de acuerdo con lo especificado en las tablas siguientes:

Conjunto de caracteres ASCII en Oracle SQL

Símbolo	*Valor Decimal*	*Símbolo*	*Valor Decimal*
blanco	32	;	59
!	33	<	60
"	34	=	61
#	35	>	62
$	36	?	63
%	37	@	64
&	38	A-Z	67-90
'	39	[	91
(	40	\	92
)	41	]	93
*	42	^	94
+	43	_	95
,	44	`	96
-	45	a-z	97-122
.	46	{	123
/	47	|	124
0-9	48-57	}	125
:	58	~	126

© Alfaomega - RC Libros

SYNAPS PREMIUM

Conjunto de caracteres EBCDIC en ORACLE SQL

Símbolo	*Valor Decimal*	*Símbolo*	*Valor Decimal*
blanco	64	%	108
¢	74	_	109
.	75	>	110
<	76	?	111
(	77	:	122
+	78	#	123
|	79	@	124
&	80	'	125
!	90	=	126
$	91	"	127
*	92	a-i	129-137
)	93	j-r	147-153
;	94	s-z	162-169
ÿ	95	A-I	193-201
-	96	J-R	209-217
/	97	S-Z	226.233

Conversión de tipos de datos

Normalmente, una expresión no debe contener valores con diferentes tipos de datos, por ejemplo, no se puede multiplicar 3 por 'Juan'. Sin embargo, en determinados casos, Oracle permite la conversión de tipos de datos, bien sea de forma automática o de forma explícita mediante una función de conversión. La tabla siguiente presenta la matriz de **posibilidades de conversión automática entre los diferentes tipos de datos**.

	CHAR	VARCHAR2	DATE	DATETIME/ INTERVAL	LONG	NUMBER	RAW	ROWID	CLOB	BLOB	NCHAR	NVAR CHAR	NCLOB
CHAR	--	X	X	X	X	X	X		X		X	X	
VARCHAR2	X	--	X	X	X	X	X	X	X		X	X	
DATE	X	X	--								X	X	
DATETIME/INTERVAL	X	X		--	X						X	X	
LONG	X	X		X	--		X		X		X	X	X
NUMBER	X	X				--					X	X	
RAW	X	X			X		--			X	X	X	
ROWID	X	X						--			X	X	
CLOB	X	X			X				--				
BLOB							X			--			
NCHAR	X	X	X	X	X	X	X	X			--	X	X
NVARCHAR2	X	X	X	X	X	X	X	X			X	--	X
NCLOB					X						X	X	--

Existen determinadas reglas que gobiernan la dirección en la que Oracle realiza las conversiones automáticas de tipos de datos. Destacan las siguientes:

© Alfaomega - RC Libros

SYNAPS PREMIUM

- Durante operaciones `INSERT` o `UPDATE`, Oracle convierte el valor al tipo de dato de la columna afectada.
- Durante operaciones `SELECT FROM`, Oracle convierte los datos de la columna al tipo de la variable destino.
- Cuando comparamos un valor carácter con un valor `NUMBER` Oracle convierte el carácter a `NUMBER`.
- Cuando comparamos un valor carácter con un valor `DATE` Oracle convierte el carácter a `DATE`.
- Cuando se realizan asignaciones, Oracle convierte el valor del lado derecho del signo igual (=) al tipo de dato del valor del lado izquierdo.
- Durante operaciones de concatenación, Oracle convierte tipos de datos no carácter a tipos de datos `CHAR` o `NCHAR`.
- Durante operaciones aritméticas y comparaciones entre tipos datos carácter y no carácter, Oracle convierte de cualquier tipo de dato carácter a número, `date`, o `rowid`, como sea apropiado en cada caso. En operaciones aritméticas entre tipos de datos `CHAR/VARCHAR2` y `NCHAR/NVARCHAR2`, Oracle los convierte a número.
- Comparaciones entre tipos `CHAR/VARCHAR2` y `NCHAR/NVARCHAR2` pueden suponer diferentes conjuntos de caracteres.
- La mayoría de la funciones carácter SQL están capacitadas para aceptar tipos de datos `CLOB`s como parámetros, y Oracle ejecuta conversiones automáticas entre los tipos `CLOB` y `CHAR`. Si el tipo `CLOB` es superior a 4.000 bytes, Oracle convertirá solo los primeros 4.000 bytes a `CHAR`.

La tabla siguiente muestra la ***dirección de las conversiones automáticas entre tipos de datos en ORACLE***:

A -- DE	*CHAR*	*VARCHAR2*	*NCHAR*	*NVARCHAR2*
CHAR	--	VARCHAR2	NCHAR	NVARCHAR2
VARCHAR2	VARCHAR2	--	NVARCHAR2	NVARCHAR2
NCHAR	NCHAR	NCHAR	--	NVARCHAR2
NVARCHAR2	NVARCHAR2	NVARCHAR2	NVARCHAR2	--

© Alfaomega - RC Libros

SYNAPS PREMIUM

A continuación, se presentan algunos ejemplos de conversión automática de tipos de datos.

En primer lugar se presenta la conversión de un texto literal a número:

```
SQL> select sal + '10' from emp;

  SAL+'10'
----------
       810
      1610
      1260
      2985
      1260
      2860
      2460
      3010
      5010
      1510
      1110

  SAL+'10'
----------
       960
      3010
      1310

14 filas seleccionadas.

SQL>
```

En segundo lugar se presenta un ejemplo donde en una condición se compara un valor carácter con uno numérico, en el que Oracle implícitamente convierte el valor carácter a numérico:

```
SQL> select ename from emp where hiredate='19/04/87';

ENAME
----------
SCOTT

SQL>
```

En tercer lugar se presenta un ejemplo en el que Oracle transforma automáticamente una fecha en formato `01-MAY-81` a formato `01/5/81`:

```
SQL> select ename from emp where hiredate='01-MAY-81';

ENAME
----------
BLAKE

SQL>
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Conversión entre distintos tipos de datos mediante funciones SQL

En Oracle también es posible especificar explícitamente conversiones de tipos de datos mediante funciones SQL de conversión. La tabla siguiente muestra las funciones SQL que convierten explícitamente un valor de un tipo en otro:

A --> DE	CHAR, VARCHAR2, NCHAR, NVARCHAR2	NUMBER	Datetime/ Interval	RAW	ROWID	LONG, LONG RAW	CLOB, NCLOB, BLOB
CHAR, VARCHAR2, NCHAR, NVARCHAR2	TO_CHAR (character) TO_NCHAR (character)	TO_NUMBER	TO_DATE TO_TIMESTAMP TO_TIMESTAMP_TZ TO_YMINTERVAL TO_DSINTERVAL	HEXTO RAW	CHARTO ROWID		TO_CLOB TO_NCLOB
NUMBER	TO_CHAR (number) TO_NCHAR (number)	--	TO_DATE NUMTOYMINTERVAL NUMTODSINTERVAL				
Datetime/ Interval	TO_CHAR (date) TO_NCHAR (datetime)		--				
RAW	RAWTOHEX RAWTONHEX			--			TO_BLOB
ROWID	ROWIDTOCHAR				--		
LONG / LONG RAW						--	TO_LOB
CLOB, NCLOB, BLOB	TO_CHAR TO_NCHAR						TO_CLOB TO_NCLOB

Creación de las tablas de la base de datos de ejemplo

Como ejemplo de manejo de tipos de datos vamos a considerar la creación de cuatro tablas que son las que utilizaremos en los ejemplos. Se utiliza el comando CREATE TABLE para la definición de las tablas de la siguiente forma:

```
SQL> CREATE TABLE dept          /*departamentos*/
    (deptno    NUMBER(2)   CONSTRAINT pk_dept PRIMARY KEY,
/*número de departamento*/
     dname     VARCHAR2(14),   /*nombre del departamento*/
     loc       VARCHAR2(13) )  /*localidad del departamento*/;

SQL> CREATE TABLE emp           /*empleados*/
    (empno     NUMBER(4)   CONSTRAINT pk_emp PRIMARY KEY,
/*número de empleado*/
     ename     VARCHAR2(10),   /*nombre del empleado*/
     job       VARCHAR2(9),    /*trabajo del empleado*/
     mgr       NUMBER(4),      /*código de oficio*/
     hiredate  DATE,           /*fecha de contrato*/
     sal       NUMBER(7,2),    /*salario*/
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
      comm        NUMBER(7,2),     /*comisión*/
      deptno      NUMBER(2)     CONSTRAINT fk_deptno REFERENCES dept );
SQL> CREATE TABLE bonus             /*bonificaciones a empleados*/
     (ename       VARCHAR2(10),     /*nombre del empleado*/
      job         VARCHAR2(9),      /*trabajo del empleado*/
      sal         NUMBER,           /*salario del empleado*/
      comm        NUMBER );

SQL> CREATE TABLE salgrade          /*nivel salarial de empleados*/
     (grade       NUMBER,           /*nivel de salario*/
      losal       NUMBER,           /*salario máximo*/
      hisal       NUMBER );         /*salario mínimo*/
```

Estas tablas no es necesario crearlas, ya que si en la base de datos por defecto que crea Oracle al realizar la instalación, accedemos a SQL*Plus mediante el usuario SCOTT y la clave TIGER, las citadas tablas están disponibles. Podemos ver su contenido de la siguiente forma:

```
SQL> SELECT * FROM dept;

DEPTNO  DNAME      LOC
------- ---------- ---------
10      ACCOUNTING NEW YORK
20      RESEARCH   DALLAS
30      SALES      CHICAGO
40      OPERATIONS BOSTON
 SQL> SELECT * FROM emp;

EMPNO  ENAME   JOB          MGR  HIREDATE     SAL    COMM  DEPTNO
-----  ------- --------- ------ --------- ------ ------ -------
 7369  SMITH   CLERK       7902 17-DEC-80    800             20
 7499  ALLEN   SALESMAN    7698 20-FEB-81   1600    300      30
 7521  WARD    SALESMAN    7698 22-FEB-81   1250    500      30
 7566  JONES   MANAGER     7839 02-APR-81   2975             20
 7654  MARTIN  SALESMAN    7698 28-SEP-81   1250   1400      30
 7698  BLAKE   MANAGER     7839 01-MAY-81   2850             30
 7782  CLARK   MANAGER     7839 09-JUN-81   2450             10
 7788  SCOTT   ANALYST     7566 19-APR-87   3000             20
 7839  KING    PRESIDENT         17-NOV-81   5000             10
 7844  TURNER  SALESMAN    7698 08-SEP-81   1500             30
 7876  ADAMS   CLERK       7788 23-MAY-87   1100             20
 7900  JAMES   CLERK       7698 03-DEC-81    950             30
 7902  FORD    ANALYST     7566 03-DEC-81   3000             20
 7934  MILLER  CLERK       7782 23-JAN-82   1300             10
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT * FROM salgrade;

GRADE LOSAL HISAL
----- ----- -----
1     700   1200
2     1201  1400
3     1401  2000
4     2001  3000
5     3001  9999
```

OPERADORES

Los operadores forman parte esencial de cualquier lenguaje y se utilizan para manipular datos individuales llamados operandos o argumentos. Si el operador afecta a un solo operando se denomina individual y si afecta a dos operandos, se llama binario. En Oracle 12c SQL existen varios tipos de operadores que se presentan en los párrafos siguientes.

Operadores aritméticos

Operador	*Propósito*	*Ejemplo*
	Como operadores individuales denotan una expresión positiva o negativa.	`SELECT * FROM emp WHERE -sal < 0;`
+ -	Como operadores binarios denotan suma o sustracción.	`SELECT comm FROM emp WHERE SYSDATE - hiredate > 365;`
* /	Son operadores binarios y denotan multiplicación y división.	`UPDATE emp SET sal = sal * 1.1;`

El aspecto del primer ejemplo sobre Oracle SQL*Plus sería el siguiente:

```
SQL> SELECT * FROM emp
  2    WHERE -sal < 0;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL        COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17/12/80        800                    20
      7499 ALLEN      SALESMAN        7698 20/02/81       1600        300         30
      7521 WARD       SALESMAN        7698 22/02/81       1250        500         30
      7566 JONES      MANAGER         7839 02/04/81       2975                    20
      7654 MARTIN     SALESMAN        7698 28/09/81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01/05/81       2850                    30
      7782 CLARK      MANAGER         7839 09/06/81       2450                    10
      7788 SCOTT      ANALYST         7566 19/04/87       3000                    20
      7839 KING       PRESIDENT            17/11/81       5000                    10
      7844 TURNER     SALESMAN        7698 08/09/81       1500          0         30
      7876 ADAMS      CLERK           7788 23/05/87       1100                    20
      7900 JAMES      CLERK           7698 03/12/81        950                    30
      7902 FORD       ANALYST         7566 03/12/81       3000                    20
      7934 MILLER     CLERK           7782 23/01/82       1300                    10
```

En expresiones aritméticas no se pueden usar dos signos menos consecutivos (--), ya que este signo se utiliza para otras finalidades dentro de SQL. Para indicar doble negación o sustracción de un valor negativo se deben separar los dos signos menos

SYNAPS PREMIUM

consecutivos con un espacio o con paréntesis de forma adecuada. Cuando hay varios signos menos consecutivos es conveniente utilizar paréntesis.

Operador concatenación

Este operador manipula cadenas de caracteres y datos tipo CLOB:

Operador	*Propósito*	*Ejemplo*
||	Concatena cadenas de caracteres y datos CLOB.	`SELECT 'Nombre es ' || ename FROM emp;`

```
SQL> SELECT 'Nombre es ' || ename
  2     FROM emp;

'NOMBREES'||ENAME
--------------------
Nombre es SMITH
Nombre es ALLEN
Nombre es WARD
Nombre es JONES
Nombre es MARTIN
Nombre es BLAKE
Nombre es CLARK
Nombre es SCOTT
Nombre es KING
Nombre es TURNER
Nombre es ADAMS
Nombre es JAMES
Nombre es FORD
Nombre es MILLER

14 filas seleccionadas.

SQL>
```

Operadores de comparación

Este tipo de operadores comparan una expresión con otra. Su resultado puede ser TRUE, FALSE, o UNKNOWN (verdadero, falso o desconocido). La tabla siguiente presenta los operadores de comparación, su propósito y un ejemplo:

Operador	*Propósito*	*Ejemplo*
=	Comprobación de igualdad.	`SELECT * FROM emp WHERE sal = 1500;`
!= ^= <> ¬=	Comprobación de desigualdad.	`SELECT * FROM emp WHERE sal != 1500;`
> <	Comprobación de *Mayor que* y *Menor que*.	`SELECT * FROM emp WHERE sal > 1500;` `SELECT * FROM emp WHERE sal < 1500;`
>=		`SELECT * FROM emp WHERE sal >= 1500;`

© Alfaomega - RC Libros

SYNAPS PREMIUM

<=	Comprobación de *Mayor o igual que* y *Menor o igual que*.	`SELECT * FROM emp WHERE sal <= 1500;`
IN	Comprobación de igual a un miembro. Equivalente a "= ANY".	`SELECT * FROM emp WHERE job IN ('CLERK','ANALYST'); SELECT * FROM emp WHERE sal IN (SELECT sal FROM emp WHERE deptno = 30);`
NOT IN	Equivalente a "!=ALL". Es FALSE si un miembro del conjunto es NULL.	`SELECT * FROM emp WHERE sal NOT IN (SELECT sal FROM emp WHERE deptno = 30); SELECT * FROM emp WHERE job NOT IN ('CLERK', ANALYST');`
ANY SOME	Compara un valor a cada valor de una lista. Debe estar precedido por =, !=, >, <, <=, >=. Es FALSE si la consulta no devuelve filas.	`SELECT * FROM emp WHERE sal = ANY (SELECT sal FROM emp WHERE deptno = 30);`
ALL	Compara un valor con todos los valores de una lista. Debe estar precedido por =, !=, >, <, <=, >=. Es TRUE si la consulta no devuelve filas.	`SELECT * FROM emp WHERE sal >= ALL ( 1400, 3000);`
[NOT] BETWEEN x AND y	[No] mayor o igual que x y menor o igual que y.	`SELECT * FROM emp WHERE sal BETWEEN 2000 AND 3000;`
EXISTS	TRUE si la subconsulta devuelve al menos una fila.	`SELECT ename, deptno FROM dept WHERE EXISTS (SELECT * FROM emp WHERE dept.deptno = emp.deptno);`
x [NOT] LIKE y [ESCAPE 'z']	TRUE si *x* [no] se ajusta al patrón *y*.	`See "LIKE Operator"operator.htm - 997970. SELECT * FROM tab1 WHERE col1 LIKE 'A_C/%E%' ESCAPE '/';`
IS [NOT] NULL	Comprobación de nulidad.	`SELECT ename, deptno FROM emp WHERE comm IS NULL;`

Operadores de conjuntos

Estos operadores combinan los resultados de dos consultas SQL en un resultado único. Estos operadores comprenden la unión (UNION), la intersección (INTERSECT), la diferencia (MINUS) y la unión que incluye valores repetidos (UNION ALL). A continuación, se presentan ejemplos:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT deptno FROM emp
  2  UNION
  3  SELECT deptno FROM dept;

    DEPTNO
----------
        10
        20
        30
        40

SQL>
```

```
SQL> SELECT deptno FROM emp
  2  UNION ALL
  3  SELECT deptno FROM dept;

    DEPTNO
----------
        20
        30
        30
        20
        30
        30
        10
        20
        10
        30
        20
        30
        20
        10
        10
        20
        30
        40

18 filas seleccionadas.

SQL>
```

```
SQL> SELECT deptno FROM emp
  2  INTERSECT
  3  SELECT deptno FROM dept;

    DEPTNO
----------
        10
        20
        30

SQL>
```

```
SQL> SELECT deptno FROM dept
  2  MINUS
  3  SELECT deptno FROM emp;

    DEPTNO
----------
        40

SQL>
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Operadores lógicos

Un operador lógico combina los resultados de dos condiciones componentes para producir un resultado simple basado en ellas o para invertir el resultado de una condición simple. El cuadro siguiente presenta los operadores lógicos, su función y ejemplos:

Operador	*Función*	*Ejemplos*

NOT	Devuelve TRUE si la condición que le sigue es FALSE. Devuelve FALSE si la condición es TRUE. Si la condición es UNKNOWN, el resultado es UNKNOWN.	`SELECT *`
		`  FROM emp`
		`  WHERE NOT (job IS NULL);`
		`SELECT *`
		`  FROM emp`
		`  WHERE NOT`
		`  (sal BETWEEN 1000 AND 2000);`

AND	Devuelve TRUE si ambas condiciones componentes son TRUE. Devuelve FALSE si cualquiera de las condiciones es FALSE. En otro caso devuelve UNKNOWN.	`SELECT *`
		`  FROM emp`
		`  WHERE job = 'CLERK'`
		`  AND deptno = 10;`

OR	Devuelve TRUE si cualquiera de las condiciones componentes es TRUE. Devuelve FALSE si ambas condiciones son FALSE. En otro caso devuelve UNKNOWN.	`SELECT *`
		`  FROM emp`
		`  WHERE job = 'CLERK'`
		`  OR deptno = 10;`

Las tablas de verdad para los operadores AND, OR y NOT se presentan a continuación:

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

	TRUE	FALSE	UNKNOWN
NOT	FALSE	TRUE	UNKNOWN

Orden de prioridad de los operadores

La lista siguiente describe la precedencia relativa entre los operadores SQL de Oracle. Los operadores situados en primer lugar son los de mayor nivel de prioridad (serán evaluados primero). Los operadores con igual nivel de prioridad se sitúan sobre la misma línea. Tenemos el siguiente orden de prioridad:

SYNAPS PREMIUM

1. + (individual), - (individual)
2. *, /
3. +, -, ||
4. Todos los operadores de comparación
5. NOT
6. AND
7. OR

Para alterar este orden de precedencia natural es necesario utilizar paréntesis, de modo que las expresiones situadas más en el interior de los paréntesis serán evaluadas primero.

Funciones

La tabla siguiente muestra las funciones más importantes de Oracle 12c SQL agrupadas por categorías:

Categoría	*Descripción*		
Funciones de fila simple	Devuelven un resultado simple para cada fila de la tabla o vista consultada. Pueden aparecer en listas de selección (si el comando SELECT no contiene una cláusula GROUP BY), cláusulas WHERE, cláusulas START WITH y cláusulas CONNECT BY.		
Funciones numéricas	Aceptan argumento numérico y devuelven resultado numérico. La mayoría de estas funciones devuelven valores con 38 dígitos decimales de precisión. Las funciones trascendentes COS, COSH, EXP, LN, LOG, SIN, SINH, SQRT, TAN y TANH tienen una precisión de 36 dígitos decimales. Las funciones transcendentes ACOS, ASIN, ATAN, y ATAN2 tienen una precisión de 36 dígitos decimales.		
	ABS ACOS ASIN ATAN ATAN2 BITAND CEIL	COS EXP FLOOR LN LOG MOD POWER	ROUND (número) SIN SINH SQRT TAN TANH TRUNC (número) WIDTH_BUCKET
Funciones carácter que devuelven valores carácter	Devuelven valores carácter del mismo tipo que sus argumentos. Funciones que devuelven valores CHAR están limitadas a 2.000 bytes, y las que devuelven valores VARCHAR2 están limitadas a 4.000 bytes de longitud. Funciones que devuelven valores CLOB están limitadas a 4 GB. Si la longitud del valor devuelto excede el límite, Oracle realiza truncamiento y no devuelve mensaje de error.		
	CHR CONCAT INITCAP LOWER LPAD LTRIM NLS_INITCAP	NLS_LOWER NLSSORT NLS_UPPER REPLACE RPAD RTRIM SOUNDEX	SUBSTR TRANSLATE TREAT TRIM UPPER TRANSLATE TRIM

© Alfaomega - RC Libros

SYNAPS PREMIUM

Funciones carácter que devuelven valores numéricos	ASCII	INSTR	LENGTH
Funciones fecha	Todas las funciones devuelven un valor DATE, excepto MONTHS_BETWEEN que devuelve número.		
	ADD_MONTHS CURRENT_DATE CURRENT_TIMESTAMP DBTIMEZONE EXTRACT (datetime) FROM_TZ LAST_DAY LOCALTIMESTAMP	MONTHS_BETWEEN NEW_TIME NEXT_DAY NUMTODSINTERVAL NUMTOYMINTERVAL ROUND (date) SESSIONTIMEZONE SYS_EXTRACT_UTC	SYSTIMESTAMP SYSDATE TO_DSINTERVAL TO_TIMESTAMP TO_TIMESTAMP_TZ TO_YMINTERVAL TRUNC (date) TZ_OFFSET
Funciones de conversión	Las funciones de conversión convierten un valor de un tipo de dato a otro tipo de datos. Suele ser de la forma *tipodedatoorigen*TO*tipodedatodestino*.		
	ASCIISTR BIN_TO_NUM CAST CHARTOROWID COMPOSE CONVERT DECOMPOSE HEXTORAW NUMTODSINTERVAL NUMTOYMINTERVAL RAWTOHEX	RAWTONHEX ROWIDTOCHAR ROWIDTONCHAR TO_CHAR (character) TO_CHAR (datetime) TO_CHAR (number) TO_CLOB TO_DATE TO_DSINTERVAL TO_LOB TO_MULTI_BYTE	 TO_NCHAR (character) TO_NCHAR (datetime) TO_NCHAR (number) TO_NCLOB TO_NUMBER TO_SINGLE_BYTE TO_YMINTERVAL TRANSLATE ... USING UNISTR
Funciones de fila simple no clasificadas	Se trata de funciones que no pueden clasificarse en ninguno de los demás grupos.		
	BFILENAME COALESCE DECODE DUMP EMPTY_BLOB, EMPTY_CLOB EXISTSNODE EXTRACT (XML) GREATEST LEAST	NLS_CHARSET_DECL_LEN NLS_CHARSET_ID NLS_CHARSET_NAME NULLIF NVL NVL2 SYS_CONNECT_BY_PATH SYS_CONTEXT	SYS_DBURIGEN SYS_EXTRACT_UTCSYS_ GUID SYS_TYPEID SYS_XMLAGG SYS_XMLGEN UID USER USERENV VSIZE
Funciones agregadas	Devuelven un resultado simple basado en grupos de filas, en vez de en filas simples. Suelen utilizarse en cláusulas ORDER BY y HAVING. También se utilizan con la cláusula GROUP BY en el comando SELECT.		
	AVG CORR COUNT COVAR_POP COVAR_SAMP	GROUPING GROUPING_ID LAST MAX MIN	REGR_ (linear regression) STDDEV STDDEV_POP STDDEV_SAMP SUM

© Alfaomega - RC Libros

SYNAPS PREMIUM

	CUME_DIST DENSE_RANK FIRST GROUP_ID	PERCENTILE_CONT PERCENTILE_DISC PERCENT_RANK RANK	VAR_POP VAR_SAMP VARIANCE
Funciones analíticas	Devuelven un resultado agregado basado en grupos de filas. Se diferencian de las funciones agregadas en que para cada grupo pueden devolver como resultado múltiples filas. Son el último grupo de operaciones ejecutadas en una consulta excepto la cláusula ORDER BY. Se usan solo en cláusulas ORDER BY.		
	AVG * CORR * COVAR_POP * COVAR_SAMP * COUNT * CUME_DIST DENSE_RANK FIRST FIRST_VALUE * LAG	LAST LAST_VALUE * LEAD MAX * MIN * NTILE PERCENT_RANK PERCENTILE_CONT PERCENTILE_DISC RANK	RATIO_TO_REPORT REGR_ (linear regression) *ROW_NUMBER STDDEV * STDDEV_POP * STDDEV_SAMP * SUM * VAR_POP * VAR_SAMP * VARIANCE *
Funciones de referencia a objetos	DEREF MAKE_REF	REF REFTOHEX	VALUE

Para conocer la sintaxis completa de cada función del cuadro anterior sería necesario consultar el manual del programa. No obstante, en los párrafos siguientes se presentan algunos ejemplos de cada uno de los grupos en los que hemos clasificado las funciones de Oracle 12c SQL.

Funciones numéricas

A continuación, se presenta la sintaxis de las funciones numéricas más importantes para el trabajo en Oracle 12c SQL incluyendo ejemplos:

ABS(n)	*Devuelve el valor absoluto de n.*
ACOS (n)	*Devuelve el arco coseno de n en radianes (n entre -1 y 1).*
ASIN(n)	*Devuelve el arco seno de n en radianes (n entre -1 y 1).*
ATAN(n)	*Devuelve el arco tangente de n en radianes.*
ATAN2(n,m)	*Devuelve el arco tangente de n/m en radianes.*
CEIL(n)	*Devuelve el menor entero mayor o igual que n.*
COS(n)	*Devuelve el coseno de n (n expresado en radianes).*
COSH(n)	*Devuelve el coseno hiperbólico de n.*
EXP(n)	*Devuelve el número e = 2,7172818 elevado a n.*
FLOOR(n)	*Devuelve el mayor entero menor o igual que n.*
Ln(n)	*Devuelve el logaritmo neperiano de n (n>0).*
Log(m,n)	*Devuelve el logaritmo en base m de n (n>0).*

© Alfaomega - RC Libros

SYNAPS PREMIUM

MOD(m,n)	*Devuelve el resto de m dividido por n (devuelve m si n=0).*
POWER(m,n)	*Devuelve m elevado a n (si m<0, n ha de ser entero).*
ROUND(n[,m])	*Redondea n a m decimales (si m se omite, m=0).*
SIGN(n)	*Devuelve 1 si n>0, 0 si n=0 y –1 si n<0.*
SIN(n)	*Devuelve el seno de n (n en radianes).*
SINH(n)	*Devuelve el seno hiperbólico de n.*
SQRT(n)	*Devuelve la raíz cuadrada real de n (n>0).*
TAN(n)	*Devuelve la tangente de n (n en radianes).*
TANH(n)	*Devuelve la tangente hiperbólica de n.*
TRUN(n,m)	*Trunca el número n a m decimales.*

Funciones carácter que devuelven valores carácter

A continuación, se presenta la sintaxis de las funciones más representativas tipo carácter que devuelven valores carácter en el trabajo con Oracle 12c SQL incluyendo ejemplos:

CHAR(n)	*Devuelve n en caracteres binarios.*
CONCAT(a,b)	*Concatena las cadenas a y b.*
INITCAP(c)	*Devuelve la cadena c con las primeras letras de cada palabra en mayúsculas y las demás en minúsculas.*
LOWER(c)	*Devuelve la cadena c en minúsculas.*
LPAD(c,n,b)	*Añade la cadena b en la parte izquierda de la cadena c hasta completar n caracteres.*
LTRIM(c,b)	*Elimina de la parte izquierda de la cadena c todos los caracteres que coincidan con alguno de los incluidos en la cadena b, hasta encontrar un carácter de c distinto a los contenidos en b.*
REPLACE(c,a,b)	*Reemplaza a por b en la cadena c .*
RPAD(c,n,b)	*Añade la cadena b en la parte derecha de la cadena c hasta completar n caracteres.*
RTRIM(c,b)	*Elimina de la parte derecha de la cadena c todos los caracteres que coincidan con alguno de los incluidos en la cadena b, hasta encontrar un carácter de c distinto a los contenidos en b.*
SOUNDEX(c)	*Devuelve la representación fonética de la cadena c.*
SUBSTR(c,m,n)	*Devuelve n caracteres de la cadena c empezando en el emésimo.*
SUBSTRB(c,m,n)	*Devuelve n caracteres de la cadena c empezando en el emésimo. Los argumentos m y n se expresan en bytes.*

© Alfaomega - RC Libros

SYNAPS PREMIUM

TRANSLATE (c,m,n)	*Devuelve c con todas las ocurrencias de cada carácter en m reemplazadas por sus correspondientes caracteres en n. Los caracteres de c que no están en m no son reemplazados.*
UPPER(c)	*Devuelve la cadena c en mayúsculas.*

Funciones carácter que devuelven valores numéricos

A continuación, se presenta la sintaxis de las funciones más representativas tipo carácter que devuelven valores numéricos en el trabajo con Oracle 12c SQL incluyendo ejemplos.

ASCII(c)	*Devuelve la representación decimal del carácter ASCII c.*
INSTR(c,d,m,n)	*Busca en la cadena c, comenzando en su emésimo carácter, la enésima ocurrencia de la cadena d y devuelve la posición en c del primer carácter de esa ocurrencia.*
INSTRB(c,d,m,n)	*Busca en la cadena c, comenzando en su emésimo carácter, la enésima ocurrencia de la cadena d y devuelve la posición en c del primer carácter de esa ocurrencia. El valor devuelto y n en bytes.*
LENGTH(c)	*Devuelve la longitud de la cadena c.*
LENGTHB(c)	*Devuelve la longitud de la cadena c en bytes.*

Funciones de fecha

A continuación, se presenta la sintaxis de las funciones fecha más representativas en el trabajo con Oracle 12c SQL incluyendo ejemplos:

ADD_MONTHS(d,n)	*Devuelve la fecha d más n meses.*
LAST_DAY(d)	*Devuelve el último día del mes que contiene la fecha d.*
MONTHS_BETWEEN(d,e)	*Devuelve el número de meses entre las fechas d y e.*
NEXT_DAY(d,c)	*Devuelve la fecha del primer día de la semana expresado en c posterior a la fecha d.*
SYSDATE	*Devuelve la fecha del sistema.*

Funciones de conversión

A continuación, se presenta la sintaxis de las funciones más representativas en el trabajo con Oracle 12c SQL que convierten un tipo de datos en otro tipo distinto, incluyendo ejemplos:

CHARTOROWID(c)	*Convierte a tipo ROWID la cadena c expresada en tipo CHART o VARCHART2.*

© Alfaomega - RC Libros

SYNAPS PREMIUM

CONVERT(c, origen, destino)	*Convierte c del sistema origen al sistema destino.*
HEXTORAW(c)	*Convierte a tipo fila la cadena hexadecimal c.*
RAWTOHEX(c)	*Convierte a tipo hexadecimal la cadena fila c.*
ROWIDTOCHART(c)	*Convierte a tipo CHART o VARCHART2 la cadena c expresada en tipo ROWID.*
TO_CHAR(d, formato)	*Convierte la fecha d al formato fecha especificado.*
TO_CHAR(n, formato)	*Convierte el número n al formato numérico dado.*
TO_DATE(c, formato)	*Convierte la cadena c al formato fecha especificado.*
TO_NUMBER(c, formato)	*Convierte la cadena c al formato numérico dado.*

Funciones agregadas

A continuación, se presenta la sintaxis de las funciones agregadas más importantes para el trabajo en Oracle 12c SQL incluyendo ejemplos:

AVG([DISTINCT|ALL] n)	*Devuelve la media de la columna n. DISTINCT considera solo los valores distintos y ALL todos.*
COUNT(* [DISTINCT|ALL])	*Devuelve el número de filas de la consulta *.*

Ejercicio 3.1. Construir una consulta SQL que devuelva VERDADERO si en la tabla de empleados (emp) el número de departamento (deptno) no es 5 ni 15.

```
SQL> SELECT 'VERDADERO'
  2       FROM emp
  3       WHERE deptno NOT IN (5,15);

'VERDADER
---------
VERDADERO
VERDADERO
VERDADERO
VERDADERO
VERDADERO
VERDADERO
VERDADERO
VERDADERO
VERDADERO
VERDADERO
VERDADERO
VERDADERO
VERDADERO
VERDADERO
```

Ejercicio 3.2. Construir una consulta SQL que devuelva los salarios de los empleados cuyo nombre comience por SM.

SYNAPS PREMIUM

```
SQL> SELECT sal
  2       FROM emp
  3       WHERE ename LIKE 'SM%';

       SAL
----------
       800
```

Ejercicio 3.3. Construir una consulta SQL que devuelva las características de los empleados con salarios entre 2.000 y 3.000. Listar también las características de los empleados que no tienen comisiones.

```
SQL> SELECT * FROM emp
  2    WHERE sal
  3    BETWEEN 2000 AND 3000;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7566 JONES      MANAGER         7839 02/04/81       2975                    20
      7698 BLAKE      MANAGER         7839 01/05/81       2850                    30
      7782 CLARK      MANAGER         7839 09/06/81       2450                    10
      7788 SCOTT      ANALYST         7566 19/04/87       3000                    20
      7902 FORD       ANALYST         7566 03/12/81       3000                    20

SQL>
```

```
SQL> SELECT *
  2    FROM emp
  3    WHERE comm IS NULL;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17/12/80        800                    20
      7566 JONES      MANAGER         7839 02/04/81       2975                    20
      7698 BLAKE      MANAGER         7839 01/05/81       2850                    30
      7782 CLARK      MANAGER         7839 09/06/81       2450                    10
      7788 SCOTT      ANALYST         7566 19/04/87       3000                    20
      7839 KING       PRESIDENT            17/11/81       5000                    10
      7876 ADAMS      CLERK           7788 23/05/87       1100                    20
      7900 JAMES      CLERK           7698 03/12/81        950                    30
      7902 FORD       ANALYST         7566 03/12/81       3000                    20
      7934 MILLER     CLERK           7782 23/01/82       1300                    10
```

Ejercicio 3.4. Construir una consulta SQL que devuelva las características de los empleados contratados antes del 25 de mayo de 1994 que ganan más de 1.000 dólares al mes.

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT * FROM emp
  2  WHERE hiredate < TO_DATE('25/MAY/94')
  3  AND sal > 1000;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL        COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20/02/81       1600        300         30
      7521 WARD       SALESMAN         7698 22/02/81       1250        500         30
      7566 JONES      MANAGER          7839 02/04/81       2975                    20
      7654 MARTIN     SALESMAN         7698 28/09/81       1250       1400         30
      7698 BLAKE      MANAGER          7839 01/05/81       2850                    30
      7782 CLARK      MANAGER          7839 09/06/81       2450                    10
      7788 SCOTT      ANALYST          7566 19/04/87       3000                    20
      7839 KING       PRESIDENT             17/11/81       5000                    10
      7844 TURNER     SALESMAN         7698 08/09/81       1500          0         30
      7876 ADAMS      CLERK            7788 23/05/87       1100                    20
      7902 FORD       ANALYST          7566 03/12/81       3000                    20
      7934 MILLER     CLERK            7782 23/01/82       1300                    10
```

Ejercicio 3.5. Construir una consulta SQL que devuelva el salario medio de los empleados, así como una medida de la calidad de dicha media. Calcular también los salarios máximo y mínimo.

```
SQL> SELECT AVG(sal) "Salario medio", STDDEV(sal) "Desviación típica",
  2  MAX(DISTINCT sal) "Salario máximo", MIN(DISTINCT sal) "Salario mínimo"
  3  FROM emp;

Salario medio Desviación típica Salario máximo Salario mínimo
------------- ----------------- -------------- --------------
   2073,21429        1182,50322           5000            800

SQL>
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

CAPÍTULO 4

LENGUAJE DE CONSULTA DE DATOS. SENTENCIA SELECT

INTRODUCCIÓN

Dentro del lenguaje SQL (*Structured Query Language*) que Oracle utiliza para tratar la información, se distinguen cuatro categorías de sentencias (o comandos). El lenguaje de modificación de datos DML (*Data Modification Language*) constituye la más sencilla de estas categorías y suele ser la primera en abordarse cuando se trata de aprender SQL. Las sentencias del lenguaje de modificación de datos se usan en la interrogación y manipulación de datos en esquemas de bases de datos ya existentes. Como ejemplos característicos tenemos las sentencias SELECT, INSERT, UPDATE, DELETE, TRUNCATE, LOCK TABLE y MERGE (nueva en Oracle 12c).

Otras categorías de sentencias SQL constituyen los denominados *Lenguaje de Definición de Datos DDL (Data Definition Language)*, *Lenguaje de Control de Transacciones TCL (Transaction Control Language)* y *Lenguaje de control de datos DCL (Data Control Languaje)*. Las sentencias del lenguaje de definición de datos se utilizan para crear, alterar, o borrar objetos de base de datos, tales como esquemas, tablas, columnas, vistas y secuencias. Como ejemplos característicos tenemos las sentencias CREATE, ALTER, DROP, ASSOCIATE, TRUNCATE y COMMENT. Las sentencias del lenguaje de control de transacciones se utilizan para confirmar o restaurar

SYNAPS PREMIUM

transacciones de base de datos, que son unidades de trabajo que realizan una o más sentencias SQL relacionadas entre sí. Como ejemplos característicos tenemos los comandos COMMIT, ROLLBACK, SAVEPOINT y SET TRANSACTION. Las sentencias del lenguaje de control de datos se utilizan en el control de acceso a datos en la base de datos. Como ejemplos característicos tenemos los comandos GRANT, RENAME, REVOKE, ANALYZE y AUDIT. Estas tres categorías de sentencias o comandos suelen utilizarse en tareas típicas de administración de la base de datos.

En este capítulo se tratan las sentencias del lenguaje de modificación de datos, y en capítulos posteriores se tratarán las restantes categorías.

LA SENTENCIA SELECT

En el capítulo 3 se introdujo una versión sencilla de la sintaxis de la sentencia SELECT. Básicamente dicha sintaxis puede resumirse como sigue:

SELECT [{ **DISTINCT** | **UNIQUE** } | **ALL**] lista_selección

FROM referencia_tabla [, referencia_tabla]...
[**WHERE** condición]
[{ claúsula_consulta_jerárquica | cláusula_group_by }
  [claúsula_consulta_jerárquica | cláusula_group_by]...]
[**UNION** | **UNION ALL** | **INTERSECT** | **MINUS** } (subconsulta)]
[cláusula order_by]

La sentencia SELECT recupera filas de una o más tablas (o vistas instantáneas), pudiendo utilizarse como un comando o como una subconsulta en otra sentencia SQL. ALL significa que se devolverán todas las filas que satisfagan las condiciones de WHERE (es el valor por defecto), DISTINCT significa que las filas duplicadas solo se devolverán una vez, y UNIQUE es sinónimo de DISTINCT.

La parte de sintaxis `lista_selección` reúne todas las tablas o vistas de las que se seleccionará la información y puede tener la siguiente estructura completa:

{ nombre_consulta
  |*
  | [usuario.] { tabla | vista | vista instantánea } .*
  | expresión [[**AS**] alias]
  }
  [, { nombre_consulta
    | [usuario.] { tabla | vista | vista instantánea } .*

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
    | expr [[AS] alias]
    }
  ]...
}
```

La parte de sintaxis `nombre_consulta` permite utilizar una subconsulta ya existente. Un asterisco (*) produce como resultado la salida de todas las columnas de todas las tablas de la cláusula FROM, la sintaxis `tabla.*` o `vista.*` significa que se visualizarán todas las columnas de la tabla o vista, `expresión` significa cualquier forma de una columna (nombre de columna, literal, cálculo matemático, función o varias columnas combinadas) siendo lo más común un simple nombre de columna, `alias` es un nombre nuevo de columna o expresión que pasará ser la cabecera de columna en la visualización de resultados (este nombre nuevo debe encerrarse entre comillas si tiene más de una palabra o si contiene caracteres que no sean ni números ni letras).

A las tablas se les proporciona normalmente un alias cuando dos tablas con nombres largos se unen en una consulta y muchas de sus columnas tienen el mismo nombre. Los alias se utilizan también en subconsultas correlacionadas y cuando una tabla se combina consigo misma.

La cláusula FROM lista las tablas que contienen los datos a recuperar por la consulta. La parte de sintaxis `referencia_tabla` hace referencia al usuario y a la tabla o vista de las que se extraerán las filas y puede tener la siguiente estructura:

```
{ nombre_consulta
| [usuario .]
  { tabla [ { PARTITION ( partición )
            | SUBPARTITION ( subpartición )
            }
            [cláusula_simple]
          | @ enlacebd
          ]
  | { vista | vista materializada } [@ enlacebd]
  }
| ( subconsulta [cláusula_restricción_subconsulta] )
| expresión_conjunto_tablas
}
```

La parte de sintaxis `nombre_consulta` permite utilizar una subconsulta ya existente relativa a un determinado bloque que ha sido definida en la parte de

SYNAPS PREMIUM

sintaxis `subconsulta [cláusula_restricción_subconsulta]`. Esta parte de sintaxis no es de uso común.

PARTITION o SUBPARTITION especifican la partición o subpartición de tabla de la cual se recuperan los datos.

La parte de sintaxis `cláusula_restricción_subconsulta` puede tener la siguiente estructura:

WITH { READ ONLY | CHECK OPTION [CONSTRAINT restricción] }

WITH READ ONLY indica que la subconsulta no puede ser actualizada. WITH CHECK OPTION indica que, si la subconsulta se utiliza en lugar de una tabla en una sentencia INSERT, UPDATE o DELETE, Oracle prohíbe cualquier cambio en la tabla que produzca filas que no estén incluidas en la subconsulta.

La parte de sintaxis `enlacebd` hace referencia al nombre de una base de datos remota en la que se sitúa la tabla o vista a la que se hace referencia.

La `condición` de la cláusula WHERE acota las filas de datos en los resultados de la consulta pudiendo ser cualquier expresión válida que se evalúe como verdadero o falso y pudiendo contener funciones, columnas y literales.

La parte de sintaxis `claúsula_consulta_jerárquica` tiene la siguiente estructura:

[**START WITH** condición] **CONNECT BY** connect_by_condición

START WITH especifica la condición que identifica las filas a utilizar como raíces de la consulta jerárquica. CONNECT BY especifica la condición que identifica la relación entre filas padre y filas hijas en la jerarquía.

La parte de sintaxis `cláusula_group_by` tiene la siguiente estructura:

GROUP BY
{ cláusula_simple_de_agrupación | conjunto_de_cláusulas }
[, { cláusula_simple_de_agrupación | conjunto_de_cláusulas }]...
[**HAVING** condición]

La cláusula simple de agrupación tiene la siguiente estructura:

© Alfaomega - RC Libros

SYNAPS PREMIUM

{ { expr | (expr [, expr]...) }
| { **CUBE** | **ROLLUP** }
({ expr | (expr [, expr]...) } [, { expr | (expr [, expr]...) }]...)
| ( )
}

ROLLUP suele usarse para producir subtotales con la función SUM en un conjunto de cláusulas de agrupación y CUBE suele utilizarse para producir tablas cruzadas en cláusulas simples de agrupación.

El conjunto de cláusulas de agrupación puede tener la siguiente estructura:

GROUPING SETS (cláusula_simple_de_agrupación [,cláusula_simple_de_agrupación]...)

GRUPING SETS se usa para producir agrupaciones múltiples de datos.

La cláusula **HAVING** `condición` restringe los grupos de filas devueltos a aquellos para los que la condición es cierta.

La parte de sintaxis `cláusula_order_by` tiene la siguiente estructura:

ORDER [**SIBLINGS**] **BY**
{ expr | posición | c_alias } [**ASC** | **DESC**] [**NULLS FIRST** | **NULLS LAST**]
[, { expr | posición | c_alias } [**ASC** | **DESC**] [**NULLS FIRST** | **NULLS LAST**]]...

La `posición` de la cláusula ORDER BY permite identificar las `expresiones` basándose en su posición relativa en la cláusula SELECT, en vez de en sus nombres. Esto es útil para expresiones complicadas y esencialmente en los operadores UNION, INTERSECT y MINUS. Estos operadores combinan en un resultado simple las filas devueltas por dos sentencias SELECT. Las opciones ASC y DES especifican si se usa una secuencia ascendente o descendente para cada expresión de ORDER BY. SIBLINGS se utiliza solo si se especifica CONNECT BY e indica que Oracle conserva las ordenaciones especificadas en `cláusula_consulta_jerárquica`. NULLS FIRST | NULLS LAST especifican si los valores nulos aparecen al principio o al final de la secuencia ordenada.

En ocasiones, la sentencia SELECT puede finalizar con la sintaxis:

FOR UPDATE
[**OF** [[usuario .] { tabla | vista } .] columna
[, [[usuario .] { tabla | vista } .] columna]...]
[**NOWAIT** | **WAIT** entero]

© Alfaomega - RC Libros

SYNAPS PREMIUM

FOR UPDATE OF coloca bloqueos en las filas que se han seleccionado. Una vez que se ha bloqueado una fila otros usuarios no pueden actualizarla hasta que la liberemos con una orden COMMIT o ROLLBACK. WAIT indica que hay que esperar un número entero de segundos para disponer del control sobre una fila y NOWAIT indica que la recuperación del control es inmediata.

Una sintaxis más resumida de la sentencia SELECT podría ser la siguiente:

```
SELECT [ { DISTINCT | UNIQUE } | ALL ]
            [*|[usuario.] tabla.*
            | expresión [alias] [,expresión [alias]...]]
FROM [usuario.] tabla [@enlacebd] [alias]
   [,[usuario.] tabla [@enlacebd] [alias]...]
[WHERE condición]
[CONNECTED BY condición [START WITH condición]]
[GROUP BY expresión [,expresión]...] [HAVING condición]]
[{ UNION | UNION ALL | INTERSECT | MINUS } SELECT...]
[ORDER BY {expresión|posición} [ASC|DESC]
      [,expresión |posición} [ASC|DESC]...]
[FOR UPDATE [OF [[usuario,]{tabla|vista}.]columna
      [,[[usuario,]{tabla|vista}.]]...]
NOWAIT]]
```

El cuadro siguiente resume la notación utilizada en la sintaxis de la sentencia SELECT. Estas reglas de notación se utilizarán durante todo el libro:

Símbolo	*Significado*
[ ]	Los corchetes encierran alternativas opcionales.
{ }	Las llaves encierran opciones tales que solo uno de ellas es requerida, es decir, encierran alternativas excluyentes.
|	La barra vertical separa alternativas en corchetes y llaves.
...	Los puntos suspensivos muestran sintaxis que puede ser repetida.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejemplos relativos a la sentencia SELECT

A continuación, se presentan varios ejemplos relativos a la sintaxis de la sentencia SELECT basados en la base de datos que se instala por defecto en Oracle 12c.

En primer lugar se obtienen las filas de la tabla EMP cuyo número de departamento es 30:

```
SQL> set linesize 132;
SQL> SELECT *
  2      FROM emp
  3      WHERE deptno = 30;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20/02/81       1600        300         30
      7521 WARD       SALESMAN         7698 22/02/81       1250        500         30
      7654 MARTIN     SALESMAN         7698 28/09/81       1250       1400         30
      7698 BLAKE      MANAGER          7839 01/05/81       2850                    30
      7844 TURNER     SALESMAN         7698 08/09/81       1500          0         30
      7900 JAMES      CLERK            7698 03/12/81        950                    30
```

A continuación, se selecciona nombre, trabajo, salario y número de departamento excepto para los vendedores del departamento número 30.

```
SQL> SELECT ename, job, sal, deptno
  2      FROM emp
  3      WHERE NOT (job = 'SALESMAN' AND deptno = 30);

ENAME      JOB                SAL     DEPTNO
---------- --------- ---------- ----------
SMITH      CLERK             800         20
JONES      MANAGER          2975         20
BLAKE      MANAGER          2850         30
CLARK      MANAGER          2450         10
SCOTT      ANALYST          3000         20
KING       PRESIDENT        5000         10
ADAMS      CLERK            1100         20
JAMES      CLERK             950         30
FORD       ANALYST          3000         20
MILLER     CLERK            1300         10
```

A continuación, se calculan los salarios máximo y mínimo por cada departamento en la tabla Empleados:

SYNAPS PREMIUM

```
SQL> SELECT deptno, MIN(sal), MAX (sal)
  2        FROM emp
  3        GROUP BY deptno;

    DEPTNO   MIN(SAL)   MAX(SAL)
---------- ---------- ----------
        10       1300       5000
        20        800       3000
        30        950       2850

SQL>
```

A continuación, se calculan los salarios máximo y mínimo para los administrativos en cada departamento:

```
SQL> SELECT deptno, MIN(sal), MAX (sal)
  2        FROM emp
  3        WHERE job = 'CLERK'
  4        GROUP BY deptno;

    DEPTNO   MIN(SAL)   MAX(SAL)
---------- ---------- ----------
        10       1300       1300
        20        800       1100
        30        950        950

SQL>
```

La siguiente consulta devuelve el número de empleados y su salario medio anual a través de todas las posibles combinaciones de departamento y categoría de trabajo:

```
SQL> set pagesize 80;
SQL> SELECT DECODE(GROUPING(dname), 1, 'All Departments',
  2         dname) AS dname,
  3      DECODE(GROUPING(job), 1, 'All Jobs', job) AS job,
  4      COUNT(*) "Total Empl", AVG(sal) * 12 "Average Sal"
  5      FROM emp, dept
  6      WHERE dept.deptno = emp.deptno
  7      GROUP BY CUBE (dname, job);

DNAME          JOB       Total Empl Average Sal
-------------- --------- ---------- -----------
ACCOUNTING     CLERK              1       15600
ACCOUNTING     MANAGER            1       29400
ACCOUNTING     PRESIDENT          1       60000
ACCOUNTING     All Jobs           3       35000
RESEARCH       ANALYST            2       36000
RESEARCH       CLERK              2       11400
RESEARCH       MANAGER            1       35700
RESEARCH       All Jobs           5       26100
SALES          CLERK              1       11400
SALES          MANAGER            1       34200
SALES          SALESMAN           4       16800
SALES          All Jobs           6       18800
All Departments ANALYST           2       36000
All Departments CLERK             4       12450
All Departments MANAGER           3       33100
All Departments PRESIDENT         1       60000
All Departments SALESMAN          4       16800
All Departments All Jobs         14  24878,5714
```

Más tarde, se calculan los salarios máximo y mínimo para los administrativos en cada departamento siendo los salarios mínimos inferiores a mil:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT deptno, MIN(sal), MAX (sal)
  2       FROM emp
  3       WHERE job = 'CLERK'
  4       GROUP BY deptno
  5       HAVING MIN(sal) < 1000;

    DEPTNO   MIN(SAL)   MAX(SAL)
---------- ---------- ----------
        20        800       1100
        30        950        950

SQL>
```

A continuación, se obtienen los vendedores con todas sus características ordenados en orden descendente de comisión:

```
SQL> SELECT *
  2      FROM emp
  3      WHERE job = 'SALESMAN'
  4      ORDER BY comm DESC;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7654 MARTIN     SALESMAN         7698 28/09/81       1250       1400         30
      7521 WARD       SALESMAN         7698 22/02/81       1250        500         30
      7499 ALLEN      SALESMAN         7698 20/02/81       1600        300         30
      7844 TURNER     SALESMAN         7698 08/09/81       1500          0         30

SQL>
```

Luego se obtienen los vendedores ordenados primero por orden ascendente de número de departamento y después por orden descendente de salario:

```
SQL> SELECT *
  2  FROM emp
  3  ORDER BY deptno ASC, sal DESC;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7839 KING       PRESIDENT            17/11/81       5000                    10
      7782 CLARK      MANAGER          7839 09/06/81       2450                    10
      7934 MILLER     CLERK            7782 23/01/82       1300                    10
      7788 SCOTT      ANALYST          7566 19/04/87       3000                    20
      7902 FORD       ANALYST          7566 03/12/81       3000                    20
      7566 JONES      MANAGER          7839 02/04/81       2975                    20
      7876 ADAMS      CLERK            7788 23/05/87       1100                    20
      7369 SMITH      CLERK            7902 17/12/80        800                    20
      7698 BLAKE      MANAGER          7839 01/05/81       2850                    30
      7499 ALLEN      SALESMAN         7698 20/02/81       1600        300         30
      7844 TURNER     SALESMAN         7698 08/09/81       1500          0         30
      7521 WARD       SALESMAN         7698 22/02/81       1250        500         30
      7654 MARTIN     SALESMAN         7698 28/09/81       1250       1400         30
      7900 JAMES      CLERK            7698 03/12/81        950                    30
```

La siguiente consulta determina, utilizando una subconsulta, quién trabaja en el departamento de TAYLOR:

SYNAPS PREMIUM

```
SQL> SELECT ename, deptno
  2       FROM emp
  3       WHERE deptno =
  4           (SELECT deptno
  5               FROM emp
  6               WHERE ename = 'JAMES');

ENAME           DEPTNO
---------- ----------
ALLEN               30
WARD                30
MARTIN              30
BLAKE               30
TURNER              30
JAMES               30
```

Selección de todas las tablas de una instancia de base de datos y de todas las columnas de una tabla

A veces puede ser interesante visualizar, tanto todas las tablas de una base de datos, como el contenido de todas las columnas de cada tabla, sobre todo cuando se va a utilizar por primera vez una base de datos y se quiere tener una idea global de su estructura y de los datos que contienen sus tablas. SQL permite utilizar la sentencia SELECT con la sintaxis que se indica a continuación para ver todas las tablas de la base de datos actual.

```
SQL> select table_name from user_tables;

TABLE_NAME
------------------------------
BONUS
DEPT
EMP
SALGRADE

SQL>
```

También SQL permite utilizar el símbolo * como comodín que abrevia todas las columnas de la tabla especificada en la cláusula FROM, lo que permite ver todas las columnas de la tabla. A continuación, se presenta un ejemplo:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT *
  2  FROM emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17/12/80        800                    20
      7499 ALLEN      SALESMAN        7698 20/02/81       1600        300         30
      7521 WARD       SALESMAN        7698 22/02/81       1250        500         30
      7566 JONES      MANAGER         7839 02/04/81       2975                    20
      7654 MARTIN     SALESMAN        7698 28/09/81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01/05/81       2850                    30
      7782 CLARK      MANAGER         7839 09/06/81       2450                    10
      7788 SCOTT      ANALYST         7566 19/04/87       3000                    20
      7839 KING       PRESIDENT            17/11/81       5000                    10
      7844 TURNER     SALESMAN        7698 08/09/81       1500          0         30
      7876 ADAMS      CLERK           7788 23/05/87       1100                    20
      7900 JAMES      CLERK           7698 03/12/81        950                    30
      7902 FORD       ANALYST         7566 03/12/81       3000                    20
      7934 MILLER     CLERK           7782 23/01/82       1300                    10
```

Selección de filas

Las consultas que presentan todas las filas de determinadas tablas son muy útiles en la elaboración de informes y en la inspección de la información, pero a veces es necesario realizar consultas cuyos resultados afecten solamente a parte de las filas de una tabla. Esta funcionalidad la cubre la cláusula WHERE.

En los ejemplos que siguen se utiliza la base de datos en un contexto educativo estudiada como ejemplo en el capítulo 3. También se utiliza la base de datos relativa a una pequeña empresa de distribución creada en el mismo capítulo.

La consulta siguiente muestra toda la información sobre los cursos cuya tarifa es menor que 150:

```
SQL> SELECT *
  2  FROM curso
  3  WHERE ctarifa<150;

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ----------------------- ---------- ---------- ----
C11 INTROD. A LAS CC.      PARA NOVATOS                    3        100 CIS
C22 ESTRUCT. DE DATOS      MUY UTIL                        3         50 CIS
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO         3          0 CIS
C44 CIRCUITOS DIGITALES    AH HA!                          3          0 CIS
C55 ARQUITECT. COMPUTADOR  MAQ. VON NEUMANN                3        100 CIS
P11 EMPIRISMO              VERLO PARA CREERLO              3        100 PHIL
P22 RACIONALISMO           PARA USARLOS CIS                3         50 PHIL
P44 SOLIPSISMO             PARA MI MISMO                   6          0 PHIL
T22 FUNDAMENTALISMO        PARA DESCUIDADOS                3         90 THEO
T33 HEDONISMO              PARA SANOS                      3          0 THEO
```

La consulta siguiente obtiene todas las filas en las que el nombre del curso es superior o igual alfabéticamente a RACIONALISMO:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cnombre >= 'RACIONALISMO';

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
P22 RACIONALISMO           PARA USARLOS CIS                   3         50 PHIL
P44 SOLIPSISMO             PARA MI MISMO                      6          0 PHIL

SQL>
```

La consulta siguiente muestra todas las oficinas en donde las ventas exceden al objetivo:

```
SQL> SELECT ciudad, ventas, objetivo
  2  FROM oficinas
  3  WHERE ventas > objetivo;

CIUDAD           VENTAS   OBJETIVO
---------------- ---------- ----------
New York         692637     575000
Atlanta          367911     350000
Los Ángeles      835915     725000

SQL>
```

La consulta siguiente permite obtener el nombre, ventas y cuota del empleado número 105:

```
SQL> SELECT name, ventas, cuota
  2  FROM repventas
  3  WHERE num_empl=105;

NAME                VENTAS      CUOTA
--------------- ---------- ----------
Bill Adams          367911     350000

SQL>
```

Filas duplicadas y la cláusula DISTINCT

Si una consulta incluye la clave primaria de una tabla en la selección, cada fila en el resultado será única, pero si no se incluye la clave primaria en la selección, en el resultado podrán aparecer filas repetidas. Por ejemplo, si consideramos la consulta que lista los números de empleados de todos los directores de oficinas de ventas, se obtiene la fila 108 repetida.

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT dir
  2  FROM oficinas;

       DIR
----------
       106
       104
       105
       108
       108

SQL>
```

Se pueden eliminar las filas duplicadas del resultado de la consulta anterior utilizando la cláusula DISTINCT en la sentencia SELECT.

```
SQL> SELECT DISTINCT dir
  2  FROM oficinas;

       DIR
----------
       104
       105
       106
       108

SQL>
```

La consulta siguiente obtiene sin valores repetidos el conjunto de todas las tarifas de los cursos:

```
SQL> SELECT DISTINCT ctarifa
  2  FROM curso;

   CTARIFA
----------
         0
        50
        90
       100
       150
       200
       500
```

Ordenación de los resultados de una consulta

SQL permite ordenar los resultados de una consulta incluyendo la cláusula ORDER BY en la sentencia SELECT. En la consulta siguiente se obtiene la tabla personal ordenada por el nombre de los empleados:

SYNAPS PREMIUM

```
SQL> SELECT *
  2  FROM personal
  3  ORDER BY enombre;

ENOMBRE         CARGO          ESUELDO DEPT
--------------- ---------- ---------- ----
ARQUIMEDES      AYTE. LAB.         200 ENG
DAVINCI         AYTE. LAB.         500
DICK NIX        LADRÓN           25001 PHIL
EUCLIDES        AYTE. LAB.        1000 MATH
HANK KISS       BUFÓN            25000 PHIL
JUAN            EVANG4              54 THEO
LUCAS           EVANG1              53 THEO
MARCOS          EVANG2              52 THEO
MATEO           EVANG3              51 THEO
```

En el ejemplo siguiente se obtiene la identificación del departamento, título de empleado y salario para todos los miembros de la plantilla fijando como campo principal de ordenación el departamento en secuencia ascendente y como campo secundario de ordenación el salario en secuencia descendente:

```
SQL> SELECT dept, cargo, esueldo
  2  FROM personal
  3  ORDER BY dept, esueldo DESC;

DEPT CARGO          ESUELDO
---- ---------- ----------
     AYTE. LAB.         500
ENG  AYTE. LAB.         200
MATH AYTE. LAB.        1000
PHIL LADRÓN           25001
PHIL BUFÓN            25000
THEO EVANG4              54
THEO EVANG1              53
THEO EVANG2              52
THEO EVANG3              51
```

En el ejemplo siguiente se muestran las ventas de cada oficina ordenadas en orden alfabético por región, y dentro de cada región, por ciudad:

```
SQL> SELECT ciudad, region, ventas
  2  FROM oficinas
  3  ORDER BY region, ciudad;

CIUDAD          REGION         VENTAS
--------------- ---------- ----------
Atlanta         Este           367911
Chicago         Este           735042
New York        Este           692637
Denver          Oeste          186042
Los Ángeles     Oeste          835915

SQL>
```

Por omisión, SQL ordena los datos en secuencia ascendente, pero puede solicitarse una ordenación descendente mediante la cláusula DES en la especificación de la ordenación. En el ejemplo siguiente se listan las oficinas clasificadas en orden descendente de ventas, de modo que las oficinas con mayores ventas aparezcan en primer lugar:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT ciudad, region, ventas
  2  FROM oficinas
  3  ORDER BY ventas DESC;

CIUDAD           REGION         VENTAS
---------------- ---------- ----------
Los Ángeles      Oeste          835915
Chicago          Este           735042
New York         Este           692637
Atlanta          Este           367911
Denver           Oeste          186042

SQL>
```

Es posible utilizar el número que ocupa la columna de ordenación en la sentencia SELECT en vez de su nombre en la cláusula ORDER BY. Por ejemplo, en la consulta siguiente se listan las oficinas clasificadas en orden descendente de rendimiento de ventas, de modo que las oficinas con mayor rendimiento aparezcan primero:

```
SQL> SELECT ciudad, region, (ventas-objetivo)
  2  FROM oficinas
  3  ORDER BY 3 DESC;

CIUDAD           REGION     (VENTAS-OBJETIVO)
---------------- ---------- -----------------
New York         Este                117637
Los Ángeles      Oeste               110915
Atlanta          Este                 17911
Chicago          Este                -64958
Denver           Oeste              -113958

SQL>
```

En el ejemplo siguiente se listan las oficinas clasificadas en orden alfabético por región, y dentro de cada región en orden descendente de rendimiento de ventas:

```
SQL> SELECT ciudad, region, (ventas-objetivo)
  2  FROM oficinas
  3  ORDER BY region ASC, 3 DESC;

CIUDAD           REGION     (VENTAS-OBJETIVO)
---------------- ---------- -----------------
New York         Este                117637
Atlanta          Este                 17911
Chicago          Este                -64958
Los Ángeles      Oeste               110915
Denver           Oeste              -113958

SQL>
```


SYNAPS PREMIUM

Columnas calculadas

SQL puede utilizar columnas que son el resultado de operaciones entre otras columnas. Por ejemplo, la consulta siguiente obtiene el nombre y el nuevo salario de todos los miembros del personal si se les concede un incremento de sueldo de 100 unidades monetarias:

```
SQL> SELECT enombre, esueldo+100
  2  FROM personal;

ENOMBRE          ESUELDO+100
---------------  -----------
LUCAS                    153
MARCOS                   152
MATEO                    151
DICK NIX                25101
HANK KISS               25100
JUAN                     154
EUCLIDES                1100
ARQUIMEDES               300
DAVINCI                  600
```

La consulta siguiente obtiene el nombre y el nuevo salario de todos los miembros del personal cuyo nuevo salario sea mayor que 25.000 suponiendo que todos los salarios iniciales se han reducido en 100:

```
SQL> SELECT enombre, esueldo-100
  2  FROM personal
  3  WHERE esueldo-100 < 25000;

ENOMBRE          ESUELDO-100
---------------  -----------
LUCAS                    -47
MARCOS                   -48
MATEO                    -49
DICK NIX                24901
HANK KISS               24900
JUAN                     -46
EUCLIDES                 900
ARQUIMEDES               100
DAVINCI                  400
```

La consulta siguiente lista la ciudad, región e importe por encima o por debajo del objetivo para cada oficina:

```
SQL>  SELECT ciudad, region, (ventas-objetivo)
  2   FROM oficinas;

CIUDAD           REGION     (VENTAS-OBJETIVO)
---------------  ---------  -----------------
New York         Este                117637
Chicago          Este                -64958
Atlanta          Este                 17911
Los Ángeles      Oeste               110915
Denver           Oeste              -113958

SQL>
```

La consulta siguiente muestra el valor del inventario para cada producto:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT id_fab, id_producto, descripcion, (existencias*precio)
  2  FROM productos;

ID_ ID_PR DESCRIPCION          (EXISTENCIAS*PRECIO)
--- ----- -------------------- --------------------
ACI 41001 Artículo Tipo 1                     15235
ACI 41002 Artículo Tipo 2                     12692
ACI 41003 Artículo Tipo 3                     22149
ACI 41004 Artículo Tipo 4                     16263
ACI 4100X Ajustador                             925
ACI 4100Y Extractor                           68,75
ACI 41002 Montador                               70
BIC 41003 Manivela                             1956
BIC 41089 Retn                                17550
BIC 41672 Plate                                   0
FEA 112   Cubierta                             17020
FEA 114   Bancada Motor                        3645
IMM 773C  Riostra 1/2 Tm                      27300
IMM 775C  Riostra 1-Tm                        7,125
IMM 779C  Riostra 2-Tm                       16,875
IMM 878H  Soporte Riostra                     12042
IMM 887P  Perno Riostra                        6000
IMM 887X  Retenedor Riostra                   15200
QSA XK47  Reductor                            13490
QSA XK48  Reductor                            27202
QSA XK48A Reductor                             4329
REI 2A44G Pasador Bisagra                      4900
REI 2A44L Bisagra Izqda.                         54
REI 2A44R Bisagra Dcha.                          54
REI 2A45C V Stago Trinquete                   16590
```

La consulta siguiente muestra la nueva cuota consistente en elevar la cuota de cada vendedor un 3% de sus ventas anuales hasta la fecha correspondiente:

```
SQL> SELECT name, cuota, (cuota+(0.03*ventas))
  2  FROM repventas;

NAME                 CUOTA (CUOTA+(0.03*VENTAS))
-------------- ---------- --------------------
Dan Roberts          300000               309170,19
Sue Smith            350000                364221,5
Paul Cruz            275000               283603,25
Bob Smith            200000               204277,82
Bill Adams           350000               361037,33
Sam Clark            275000               283997,36
Nancy Angelli        300000               305581,26
Larry Fitch          350000               360855,95
Mary Jones           300000               311781,75
Tom Snyder
```

Renombrar columnas: cláusula AS

SQL puede utilizar en la salida nombres de columnas distintos a los reales. La cláusula AS se puede usar para cambiar el nombre de una columna del conjunto de resultados o para asignar un nombre a una columna derivada. Cuando se define una columna del conjunto de resultados mediante una referencia a la columna de una tabla o vista, el nombre de la columna del conjunto de resultados es el mismo que el nombre de la columna a la que se hace referencia, pero la cláusula AS se puede usar para asignar un nombre distinto, o alias, a la columna del conjunto de resultados. Esto se puede hacer para mejorar la comprensión. Por ejemplo, la consulta siguiente lista la ciudad, región e importe por encima o por debajo del objetivo para cada oficina denominando la columna relativa al importe como SUPERHABIT:

SYNAPS PREMIUM

```
SQL> SELECT ciudad, region, ventas-objetivo AS "SUPERHABIT"
  2  FROM oficinas;

CIUDAD          REGION     SUPERHABIT
--------------- ---------- ----------
New York        Este           117637
Chicago         Este           -64958
Atlanta         Este            17911
Los Ángeles     Oeste          110915
Denver          Oeste         -113958

SQL>
```

Realmente, la cláusula AS puede omitirse en este caso tal y como se muestra en los ejemplos siguientes:

```
SQL> SELECT ciudad, region, ventas-objetivo "SUPERHABIT"
  2  FROM oficinas;

CIUDAD          REGION     SUPERHABIT
--------------- ---------- ----------
New York        Este           117637
Chicago         Este           -64958
Atlanta         Este            17911
Los Ángeles     Oeste          110915
Denver          Oeste         -113958

SQL>
```

```
SQL> SELECT name "NOMBRE"
  2  FROM repventas
  3  WHERE contrato < '01-ENE-88';

NOMBRE
---------------
Dan Roberts
Sue Smith
Paul Cruz
Bob Smith

SQL>
```

CONSULTAS CON CONDICIONES DE BÚSQUEDA

SQL permite realizar consultas que contienen *condiciones de búsqueda* (o *predicados*) de modo que los resultados son precisamente los que cumplen las citadas condiciones. La comprobación del cumplimiento de las condiciones es como un test o contraste. De hecho, las condiciones de búsqueda suelen agruparse en *contrastes de comparación* (comparan el valor de una expresión con el valor de otra), *contrastes de rango* (examinan si el valor de una expresión cae dentro de un rango dado), *contrastes de pertenencia a un conjunto* (comprueban si el valor de una expresión pertenece a un conjunto dado), *contrastes de correspondencia con patrón* (comprueban si los datos cumplen un patrón dado) y *contrastes de valor nulo* (testean si existen valores nulos en las columnas).

© Alfaomega - RC Libros

SYNAPS PREMIUM

Contrastes de comparación

Mediante los contrastes de comparación, SQL calcula y compara los valores de dos expresiones SQL para cada fila de datos utilizando los operadores de comparación de SQL (=, < >, <, <=, >, >=).

Por ejemplo, la consulta siguiente devuelve los vendedores contratados antes de 1998:

```
Conectado a:
Oracle9i Enterprise Edition Release 9.0.1.1.1 - Production
With the Partitioning option
JServer Release 9.0.1.1.1 - Production

SQL> SELECT name
  2  FROM repventas
  3  WHERE contrato < '01-ENE-88';

NAME
--------------
Dan Roberts
Sue Smith
Paul Cruz
Bob Smith

SQL>
```

En los dos ejemplos siguientes se obtienen, respectivamente, las oficinas cuyas ventas están por debajo del 80% del objetivo y las oficinas no dirigidas por el empleado número 108:

```
SQL> SELECT *
  2  FROM oficinas
  3  WHERE ventas < (0.8*objetivo);

   OFICINA CIUDAD           REGION              DIR   OBJETIVO     VENTAS
---------- ---------------- ---------- ---------- ---------- ----------
        22 Denver           Oeste              108     300000     186042

SQL>
```

```
SQL> SELECT *
  2  FROM oficinas
  3  WHERE dir <> 108;

   OFICINA CIUDAD           REGION              DIR   OBJETIVO     VENTAS
---------- ---------------- ---------- ---------- ---------- ----------
        11 New York         Este               106     575000     692637
        12 Chicago          Este               104     800000     735042
        13 Atlanta          Este               105     350000     367911

SQL>
```

La consulta siguiente devuelve el nombre y el límite de crédito del cliente número 2107:

SYNAPS PREMIUM

```
SQL> SELECT empresa, limite_credito
  2  FROM clientes
  3  WHERE num_clie = 2107;

EMPRESA              LIMITE_CREDITO
-------------------- --------------
Ace International              35000

SQL>
```

Contraste de rango: cláusula BETWEEN...AND

El contraste del rango comprueba si un valor se encuentra entre dos valores especificados.

Como primer ejemplo se obtiene el nombre del curso y la descripción de todos los cursos cuya descripción comience por la palabra ES.

```
SQL> SELECT cnombre, cno, cdescp
  2  FROM curso
  3  WHERE cnombre BETWEEN 'ES' AND 'ESZ';

CNOMBRE                   CNO CDESCP
------------------------- --- ------------------------
ESTRUCT. DE DATOS         C22 MUY UTIL
ESCOLASTICISMO            T11 PARA BEATOS

SQL>
```

Como segundo ejemplo obtenemos todas las características de los pedidos que se han realizado entre el 1 de octubre y el 31 de diciembre de 1989.

```
SQL> SELECT *
  2  FROM pedidos
  3  WHERE fecha_pedido BETWEEN '01-OCT-89' AND '31-DIC-89';

NUM_PEDIDO FECHA_PE      CLIE       REP FAB PRODU       CANT    IMPORTE
---------- -------- ---------- ---------- --- ----- ---------- ----------
    112961 17/12/89       2117        106 REI 2A44L          7      31500
    112963 17/12/89       2103        105 ACI 41004         28       3276
    112968 12/10/89       2102        101 ACI 41004         34       3978
    112975 12/10/89       2111        103 REI 2A44G          6       2100
    112979 12/10/89       2114        102 ACI 41002          6      15000
    112983 27/12/89       2103        105 ACI 41004          6        702
    112987 31/12/89       2103        105 ACI 4100Y         11      27500
    112992 04/11/89       2118        108 ACI 41002         10        760
```

Como tercer ejemplo obtenemos el número de curso y tarifa de todos los cursos de tarifa menor que 50 o mayor que 400.

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT cno, ctarifa
  2  FROM curso
  3  WHERE ctarifa NOT BETWEEN 50 AND 400;

CNO     CTARIFA
--- ----------
C33            0
C44            0
C66          500
P44            0
T33            0

SQL>
```

Como cuarto ejemplo listamos los vendedores cuyas ventas no están entre el 80 y el 120% de su cuota.

```
SQL> set pagesize 80;
SQL> set linesize 132;
SQL> SELECT *
  2  FROM repventas
  3  WHERE ventas NOT BETWEEN (0.8*cuota) AND (1.2*cuota);

  NUM_EMPL NAME                 EDAD OFICINA_REP TITULO     CONTRATO   DIRECTOR      CUOTA     VENTAS
---------- -------------- ---------- ---------- ---------- -------- ---------- ---------- --------
       102 Sue Smith              48         21 Rep Ventas 12/12/86        108     350000     474050
       104 Bob Smith              33         12 Dir Ventas 19/05/87        106     200000     142594
       107 Nancy Angelli          49         22 Rep Ventas 14/11/88        108     300000     186042
       109 Mary Jones             31         11 Rep Ventas 12/10/99        106     300000     392725

SQL>
```

Contraste de pertenencia a un conjunto: cláusula IN

Este tipo de consultas comprueba si un valor dado coincide con uno de una lista de valores especificada. Por ejemplo, la consulta siguiente obtiene toda la información disponible sobre cualquier curso que tenga una tarifa coincidente con algún número del conjunto {12, 12, 50, 75, 90, 100, 500}:

```
SQL> SELECT *
  2  FROM curso
  3  WHERE ctarifa IN (12.12, 50, 75, 90, 100, 500);

CNO CNOMBRE                  CDESCP
--- -----------------------  -------------------------
C11 INTROD. A LAS CC.        PARA NOVATOS
C22 ESTRUCT. DE DATOS        MUY UTIL
C55 ARQUITECT. COMPUTADOR    MAQ. VON NEUMANN
C66 BASES DE DATOS RELAC.    IMPRESCINDIBLE
P11 EMPIRISMO                VERLO PARA CREERLO
P22 RACIONALISMO             PARA USARLOS CIS
T22 FUNDAMENTALISMO          PARA DESCUIDADOS
```

En la consulta siguiente se obtiene toda la información disponible sobre cualquier curso que no tenga una tarifa coincidente con algún número del conjunto {12,12, 50, 75, 90, 100, 500}:

SYNAPS PREMIUM

```
SQL> SELECT *
  2  FROM curso
  3  WHERE ctarifa NOT IN (12.12, 50, 75, 90, 100, 500);

CNO CNOMBRE                CDESCP                        ----
--- ---------------------- ----------------------------
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO
C44 CIRCUITOS DIGITALES    AH HA!
P33 EXISTENCIALISMO        PARA USARLOS CIS
P44 SOLIPSISMO             PARA MI MISMO
T11 ESCOLASTICISMO         PARA BEATOS
T33 HEDONISMO              PARA SANOS
T44 COMUNISMO              PARA AVAROS
```

Si queremos saber los vendedores que trabajan en las oficinas de representación 11, 13 y 22, podremos utilizar la consulta que se presenta a continuación:

```
SQL> set pagesize 80;
SQL> set linesize 132;
SQL> SELECT *
  2  FROM repventas
  3  WHERE oficina_rep in (11, 13, 22);

 NUM_EMPL NAME                 EDAD OFICINA_REP TITULO     CONTRATO   DIRECTOR      CUOTA     VENTAS
--------- --------------- ---------- ---------- ---------- -------- ---------- ---------- --------
      105 Bill Adams              37         13 Rep Ventas 12/02/88        104     350000     367911
      106 Sam Clark               52         11 VP Ventas  14/06/88                275000     299912
      107 Nancy Angelli           49         22 Rep Ventas 14/11/88        108     300000     186042
      109 Mary Jones              31         11 Rep Ventas 12/10/99        106     300000     392725

SQL>
```

La consulta siguiente devuelve número, fecha e importe de todos los pedidos que no se hayan remitido en un jueves de enero de 1990:

```
SQL> SELECT num_pedido, fecha_pedido, importe
  2  FROM pedidos
  3  WHERE fecha_pedido NOT IN ('04-ENE-90', '11-ENE-90', '18-ENE-90', '25-ENE-90');

NUM_PEDIDO FECHA_PE    IMPORTE
---------- -------- ----------
    110036 30/01/90      22500
    112961 17/12/89      31500
    112963 17/12/89       3276
    112968 12/10/89       3978
    112975 12/10/89       2100
    112979 12/10/89      15000
    112983 27/12/89        702
    112987 31/12/89      27500
    112989 03/01/90       1458
    112992 04/11/89        760
    112993 04/01/89       1896
    112997 08/01/90        652
    113007 08/01/90       2925
    113013 14/01/90        652
    113024 20/01/90       7100
    113027 22/01/90       4104
    113034 29/01/90        632
    113042 02/02/90      22500
    113045 02/02/90      45000
    113048 10/02/90       3750
    113049 10/02/90       2130
    113051 10/02/90       1420
    113055 15/02/90        150
    113057 18/02/90        600
    113058 23/02/90       1480
    113062 24/02/90       2430
    113065 27/02/90       2130
    113069 02/03/90      31350
```

Contraste de correspondencia con patrón: cláusula LIKE

Las consultas de correspondencia con patrón recuperan filas para las que el contenido de una columna de texto se corresponde con un texto dado, es decir, comprueban si el valor de una columna se ajusta a un patrón especificado. En este tipo

© Alfaomega - RC Libros

SYNAPS PREMIUM

de consultas, **el signo de porcentaje % se utiliza como comodín**. Por ejemplo, la consulta siguiente obtiene toda la información sobre cualquier curso que tenga una descripción que comience con la cadena 'PARA'.

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cdescp LIKE 'PARA %';

CNO CNOMBRE                CDESCP
--- ---------------------- ------------------------ ----
C11 INTROD. A LAS CC.      PARA NOVATOS
P22 RACIONALISMO           PARA USARLOS CIS
P33 EXISTENCIALISMO        PARA USARLOS CIS
P44 SOLIPSISMO             PARA MI MISMO
T11 ESCOLASTICISMO         PARA BEATOS
T22 FUNDAMENTALISMO        PARA DESCUIDADOS
T33 HEDONISMO              PARA SANOS
T44 COMUNISMO              PARA AVAROS
```

La consulta que se presenta a continuación devuelve el límite de crédito de la empresa Smithson Corp:

```
SQL> SELECT empresa, limite_credito
  2  FROM clientes
  3  WHERE empresa LIKE 'Smith%';

EMPRESA              LIMITE_CREDITO
-------------------- --------------
Smithson Corp.                20000

SQL>
```

La consulta siguiente devuelve la información relativa a todas las empresas cuyo límite de crédito no es 20.000:

```
SQL> SELECT *
  2  FROM clientes
  3  WHERE limite_credito NOT LIKE '20000';

 NUM_CLIE EMPRESA                REPCLIE LIMITE_CREDITO
--------- -------------------- ---------- --------------
     2101 Jones Mfg.                  106          65000
     2102 First. Corp.                101          65000
     2103 Acme Mfg.                   105          50000
     2105 AAA Investments             101          45000
     2106 Fred Lewis Corp.            102          65000
     2107 Ace International           110          35000
     2108 Holm Landis                 109          55000
     2109 Chen Associates             103          25000
     2111 JCP Inc.                    103          50000
     2112 Zetacorp                    108          50000
     2117 J.P. Sinclair               106          35000
     2118 Midwest Systems             108          60000
     2119 Solomon Inc.                109          25000
     2120 Rico Enterprises            102          50000
     2121 QMA Assoc.                  103          45000
     2122 Three-Way Lines             105          30000
     2123 Carter Sons                 102          40000
     2124 Peter Brothers              107          40000
```

Dentro de la cláusula LIKE juega un papel importante el **símbolo subrayado bajo (_)**, que representa exactamente una posición de carácter. La diferencia entre el *símbolo %* y el *símbolo subrayado bajo* es significativa. En primer lugar, el *símbolo %* permite cualquier número de caracteres, mientras que el subrayado bajo permite solo uno. En segundo lugar,

SYNAPS PREMIUM

el símbolo subrayado siempre requiere que aparezca un carácter, mientras que en el *símbolo %* se considera como una coincidencia la simple ausencia de carácter.

La consulta siguiente devuelve la información relativa a todas las empresas que tienen como segunda letra de su nombre la letra e:

```
SQL> SELECT *
  2  FROM clientes
  3  WHERE empresa LIKE '_e%';

  NUM_CLIE EMPRESA                   REPCLIE LIMITE_CREDITO
---------- -------------------- ---------- --------------
      2112 Zetacorp                    108          50000
      2124 Peter Brothers              107          40000

SQL>
```

Esta consulta no sería resoluble solo con el *símbolo %*.

```
SQL> SELECT *
  2  FROM clientes
  3  WHERE empresa LIKE '%e%';

  NUM_CLIE EMPRESA                   REPCLIE LIMITE_CREDITO
---------- -------------------- ---------- --------------
      2101 Jones Mfg.                  106          65000
      2103 Acme Mfg.                   105          50000
      2105 AAA Investments             101          45000
      2106 Fred Lewis Corp.            102          65000
      2107 Ace International           110          35000
      2109 Chen Associates             103          25000
      2112 Zetacorp                    108          50000
      2118 Midwest Systems             108          60000
      2120 Rico Enterprises            102          50000
      2122 Three-Way Lines             105          30000
      2123 Carter Sons                 102          40000
      2124 Peter Brothers              107          40000
```

La consulta siguiente devuelve la información relativa a todas las empresas cuyos nombres empiezan por la letra A:

```
SQL> SELECT *
  2  FROM clientes
  3  WHERE empresa LIKE 'A_%';

  NUM_CLIE EMPRESA                   REPCLIE LIMITE_CREDITO
---------- -------------------- ---------- --------------
      2103 Acme Mfg.                   105          50000
      2105 AAA Investments             101          45000
      2107 Ace International           110          35000

SQL>
```

Contraste de valor nulo: cláusula IS NULL

Suele ser muy útil comprobar los valores NULL existentes en una tabla y manejarlos a nuestro antojo. Por ejemplo, la consulta siguiente visualiza los nombres, número de ayudantes y número de departamento para todos los miembros de la facultad para los que se sabe si tienen o no ayudantes:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT fnombre, fnumdep, fdept
  2  FROM claustro
  3  WHERE fnumdep IS NOT NULL;

FNOMBRE            FNUMDEP FDEP
--------------- ---------- ----
KATHY PEPE               2 PHIL
JESSIE MARTIN            2 THEO
JOSE COHN                2 CIS
AL HARTLEY               7 CIS
JULIA MARTIN             1 PHIL
BARB HLAVATY             3 CIS
LISA BOBAK               0 THEO
```

En la consulta siguiente tratamos de buscar los vendedores que aún no tienen asignada una oficina:

```
SQL> SELECT name
  2  FROM repventas
  3  WHERE oficina_rep IS NULL;

NAME
---------------
Tom Snyder

SQL>
```

La consulta siguiente presenta los vendedores a los que se ha asignado una oficina:

```
SQL> SELECT name
  2  FROM repventas
  3  WHERE oficina_rep IS NOT NULL;

NAME
---------------
Dan Roberts
Sue Smith
Paul Cruz
Bob Smith
Bill Adams
Sam Clark
Nancy Angelli
Larry Fitch
Mary Jones
```

Contrastes compuestos: cláusulas AND, OR y NOT

Muy a menudo suelen utilizarse condiciones de búsqueda compuestas que envuelven los operadores AND, OR y NOT. Utilizando las reglas de la lógica se pueden combinar condiciones de búsqueda SQL simples para formar condiciones compuestas únicas.

La consulta siguiente selecciona toda la información sobre los cursos de tres créditos ofrecidos por el departamento de Filosofía:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cred=3 AND cdept='PHIL';

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------ ---------- ---------- ----
P11 EMPIRISMO              VERLO PARA CREERLO                3        100 PHIL
P22 RACIONALISMO           PARA USARLOS CIS                  3         50 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS                  3        200 PHIL

SQL>
```

La consulta siguiente obtiene toda la información sobre cualquier curso que tenga una tarifa entre 100 y 500, ambos inclusive:

```
SQL> SELECT *
  2  FROM curso
  3  WHERE ctarifa >= 100 AND ctarifa <= 500;

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------ ---------- ---------- ----
C11 INTROD. A LAS CC.      PARA NOVATOS                      3        100 CIS
C55 ARQUITECT. COMPUTADOR  MAQ. VON NEUMANN                  3        100 CIS
C66 BASES DE DATOS RELAC.  IMPRESCINDIBLE                    3        500 CIS
P11 EMPIRISMO              VERLO PARA CREERLO                3        100 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS                  3        200 PHIL
T11 ESCOLASTICISMO         PARA BEATOS                       3        150 THEO
T44 COMUNISMO              PARA AVAROS                       6        200 THEO
```

La consulta siguiente obtiene los vendedores que están por debajo de la cuota y tienen ventas inferiores a 300.000:

```
SQL>  SELECT name, cuota, ventas
  2   FROM repventas
  3   WHERE ventas < cuota AND ventas<300000;

NAME                  CUOTA      VENTAS
--------------- ---------- ----------
Bob Smith             200000     142594
Nancy Angelli         300000     186042

SQL>
```

La consulta siguiente devuelve todos los vendedores que están por debajo de la cuota, pero cuyas ventas no son inferiores a 150.000:

```
SQL> SELECT name, cuota, ventas
  2  FROM repventas
  3  WHERE ventas < cuota AND NOT ventas < 150000;

NAME                  CUOTA      VENTAS
--------------- ---------- ----------
Nancy Angelli         300000     186042

SQL>
```

La consulta siguiente devuelve los vendedores que están por debajo de la cuota o con ventas inferiores a 300.000:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT name, cuota, ventas
  2  FROM repventas
  3  WHERE ventas < cuota OR ventas < 300000;

NAME                 CUOTA      VENTAS
---------------- ---------- ----------
Paul Cruz            275000     286775
Bob Smith            200000     142594
Sam Clark            275000     299912
Nancy Angelli        300000     186042
Tom Snyder                       75985

SQL>
```

Combinación de consultas: cláusula UNION

La cláusula UNION permite combinar los resultados de dos o más consultas en una única tabla. Por ejemplo, la consulta siguiente visualiza los salarios de todo el personal y del claustro de la facultad en una única tabla:

```
SQL> SELECT esueldo FROM personal
  2  UNION
  3  SELECT fsueldo FROM claustro;

   ESUELDO
----------
        51
        52
        53
        54
       200
       500
      1000
     25000
     25001
     35000
     36000

   ESUELDO
----------
     45000
```

La consulta siguiente visualiza los salarios de todo el personal y del claustro de la facultad en una única tabla sin eliminar valores duplicados en el resultado:

```
SQL> SELECT esueldo FROM personal
  2  UNION ALL
  3  SELECT fsueldo FROM claustro;

   ESUELDO
----------
        53
        52
        51
     25001
     25000
        54
      1000
       200
       500
     35000
     45000
     35000
     45000
     45000
     35000
     36000
```

La consulta siguiente lista todos los productos cuyo precio unitario excede de 2.000 o con importe de pedido superior a 30.000:

SYNAPS PREMIUM

```
SQL> SELECT id_fab, id_producto FROM productos WHERE precio > 2000
  2  UNION
  3  SELECT DISTINCT fab, producto FROM pedidos WHERE importe > 30000;

ID_ ID_PR
--- -----
IMM 775C
REI 2A44L
REI 2A44R

SQL>
```

Como la operación UNION combina las filas de dos conjuntos de resultados, podrían aparecer filas duplicadas; sin embargo, la cláusula UNION elimina las filas duplicadas tal y como se observa en la consulta siguiente, que lista todos los productos cuyo precio unitario excede de 2.000 o con importe de pedido superior a 3.000:

```
SQL> SELECT id_fab, id_producto FROM productos WHERE precio > 2000
  2  UNION
  3  SELECT  fab, producto FROM pedidos WHERE importe > 3000;

ID_ ID_PR
--- -----
ACI 41002
ACI 41003
ACI 41004
ACI 4100Y
ACI 4100Z
IMM 775C
IMM 779C
QSA XK47
REI 2A44L
REI 2A44R
```

No obstante, es posible obtener las filas duplicadas a partir de la cláusula UNION ALL, tal y como se especifica en la consulta siguiente:

```
SQL> SELECT id_fab, id_producto FROM productos WHERE precio > 2000
  2  UNION ALL
  3  SELECT  fab, producto FROM pedidos WHERE importe > 3000;

ID_ ID_PR
--- -----
ACI 41002
REI 2A44L
ACI 41004
ACI 41004
ACI 41002
ACI 4100Y
IMM 779C
ACI 41003
QSA XK47
ACI 41002
REI 2A44R
REI 2A44R
IMM 779C
IMM 775C
```

También es posible evitar las filas repetidas en consultas con UNION ALL utilizando adecuadamente la cláusula DISTINCT, tal y como se presenta en el ejemplo siguiente:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL>  SELECT id_fab, id_producto FROM productos WHERE precio > 2000
  2   UNION ALL
  3   SELECT DISTINCT fab, producto FROM pedidos WHERE importe > 3000;

ID_ ID_PR
--- -----
ACI 41002
ACI 41003
ACI 41004
ACI 4100Y
ACI 4100Z
IMM 775C
IMM 779C
QSA XK47
REI 2A44L
REI 2A44R
```

En cuanto a la relación entre UNION y la ordenación hay que tener presente que la cláusula ORDER BY no puede aparecer en ninguna de las dos sentencias SELECT combinadas por una operación UNION; sin embargo, sí puede ser ordenado el conjunto combinado de los resultados situando una cláusula ORDER BY después de la segunda sentencia SELECT. Se tendrá presente que las columnas fruto de la operación UNION no tienen nombre, lo que obliga a utilizar números en la cláusula ORDER BY. A continuación, se obtienen todos los productos cuyo precio unitario excede de 2.000 o con importe de pedido superior a 3.000, ordenando por fabricante y número de pedido:

```
  2  UNION ALL
  3  SELECT  fab, producto FROM pedidos WHERE importe > 3000
  4  ORDER BY 1,2;

ID_ ID_PR
--- -----
ACI 4100Y
ACI 4100Z
ACI 4100Z
ACI 41002
ACI 41003
ACI 41004
ACI 41004
IMM 775C
IMM 779C
IMM 779C
QSA XK47
REI 2A44L
REI 2A44R
REI 2A44R
```

Cuando se utilizan operadores UNION, hay que tener en cuenta las siguientes directrices:

- Todas las listas de selección de las instrucciones que se combinan con UNION deben tener el mismo número de expresiones (nombres de columna, expresiones aritméticas, funciones de agregado, etc.).

- Las columnas correspondientes del conjunto de resultados combinado con UNION, o cualquier subconjunto de columnas que se use en consultas individuales, deben ser del mismo tipo de datos, tener la posibilidad de convertir los tipos de datos de forma implícita o suministrar una conversión explícita. Por ejemplo, no es posible utilizar UNION entre una columna de un tipo de datos *datetime* y una de tipo de datos *binary*, a menos que se suministre una conversión explícita, mientras sí que se puede aplicar UNION

SYNAPS PREMIUM

entre una columna de tipo de datos *money* y una del tipo *int* porque pueden convertirse implícitamente.

- Las columnas del conjunto de resultados de las instrucciones individuales que se combinan con UNION deben estar en el mismo orden, porque UNION compara las columnas de una en una en el orden que se indica en las consultas individuales.

- Cuando se combinan distintos tipos de datos en una operación UNION se convierten con las reglas de precedencia de los tipos de datos.

- Los nombres de columna de la tabla que resulta de UNION se toman de la primera consulta individual de la instrucción UNION. Para hacer referencia a una columna del conjunto de resultados con un nuevo nombre (por ejemplo, en una cláusula ORDER BY), debe hacerse referencia a ella mediante AS.

En Oracle, junto con UNION, suelen utilizarse *INTERSECT* y *MINUS*. Hay ocasiones en las que se necesita combinar informaciones de tipo similar que se encuentran en varias tablas. Un ejemplo clásico puede ser el de dos o más listas de envío que se unen antes de empezar una campaña por correo. Dependiendo del propósito particular de la campaña se pueden enviar cartas de una de las siguientes formas:

A todas las personas de ambas listas (evitando enviar cartas a alguien que se encuentre en las dos listas).

- Solamente a aquellas personas que estén en ambas listas.

- Solo a aquellas personas que estén en una de las listas.

La primera forma se corresponde con UNION, la segunda con INTERSECT y la tercera con MINUS.

Ejercicio 4.1. Considerando la base de datos en un contexto educativo estudiada como ejemplo en el capítulo 3, obtener toda la tabla CURSO, toda la información sobre los cursos gratuitos (tarifa nula) y toda la información que sobre cualquier curso ofrezca el departamento de Filosofía.

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT *
  2  FROM curso;

CNO CNOMBRE                CDESCP                       CRED    CTARIFA CDEP
--- ---------------------- ------------------------ ---------- ---------- ----
C11 INTROD. A LAS CC.      PARA NOVATOS                    3        100 CIS
C22 ESTRUCT. DE DATOS      MUY UTIL                        3         50 CIS
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO         3          0 CIS
C44 CIRCUITOS DIGITALES    AH HA!                          3          0 CIS
C55 ARQUITECT. COMPUTADOR  MAQ. VON NEUMANN                3        100 CIS
C66 BASES DE DATOS RELAC.  IMPRESCINDIBLE                  3        500 CIS
P11 EMPIRISMO              VERLO PARA CREERLO              3        100 PHIL
P22 RACIONALISMO           PARA USARLOS CIS                3         50 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS                3        200 PHIL
P44 SOLIPSISMO             PARA MI MISMO                   6          0 PHIL
T11 ESCOLASTICISMO         PARA BEATOS                     3        150 THEO
T22 FUNDAMENTALISMO        PARA DESCUIDADOS                3         90 THEO
T33 HEDONISMO              PARA SANOS                      3          0 THEO
T44 COMUNISMO              PARA AVAROS                     6        200 THEO
```

```
SQL> SELECT *
  2  FROM curso
  3  WHERE CTARIFA = 0;

CNO CNOMBRE                CDESCP                       CRED    CTARIFA CDEP
--- ---------------------- ------------------------ ---------- ---------- ----
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO         3          0 CIS
C44 CIRCUITOS DIGITALES    AH HA!                          3          0 CIS
P44 SOLIPSISMO             PARA MI MISMO                   6          0 PHIL
T33 HEDONISMO              PARA SANOS                      3          0 THEO

SQL>
```

```
SQL> SELECT *
  2  FROM curso
  3  WHERE CDEPT='PHIL';

CNO CNOMBRE                CDESCP                       CRED    CTARIFA CDEP
--- ---------------------- ------------------------ ---------- ---------- ----
P11 EMPIRISMO              VERLO PARA CREERLO              3        100 PHIL
P22 RACIONALISMO           PARA USARLOS CIS                3         50 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS                3        200 PHIL
P44 SOLIPSISMO             PARA MI MISMO                   6          0 PHIL

SQL>
```

Ejercicio 4.2. Visualizar todos los departamentos académicos que ofrezcan cursos, de modo que no aparezcan valores duplicados en la salida de datos. Además, para cada curso con tarifa menor que 100, visualizar sin filas repetidas el departamento académico que ofrece ese curso y el número de créditos concedidos. Por último, visualizar el número de curso y el nombre de todos los cursos que tengan una tarifa superior a 100, incluyendo en el resultado una tercera columna cuyo contenido sea CURSO CARO para cada fila.

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT DISTINCT cdept
  2  FROM curso;

CDEP
----
CIS
PHIL
THEO

SQL>
```

```
SQL> SELECT DISTINCT cdept, cred
  2  FROM curso
  3  WHERE ctarifa < 100;

CDEP       CRED
---- ----------
CIS           3
PHIL          3
PHIL          6
THEO          3

SQL>
```

```
SQL> SELECT cno, cnombre, 'CURSO CARO'
  2  FROM curso
  3  WHERE ctarifa > 100;

CNO CNOMBRE                   'CURSOCARO
--- ----------------------- ----------
C66 BASES DE DATOS RELAC.   CURSO CARO
P33 EXISTENCIALISMO         CURSO CARO
T11 ESCOLASTICISMO          CURSO CARO
T44 COMUNISMO               CURSO CARO

SQL>
```

Ejercicio 4.3. Visualizar el número de curso, nombre y créditos de cualquier curso que ofrezca el departamento de Informática y Ciencias de la Información ordenando el resultado en secuencia descendente por número de curso. Visualizar también la identificación de los departamentos y el nombre del curso ordenado por código de departamento y dentro de cada departamento por nombre de curso.

```
SQL> SELECT cno, cnombre, cred
  2  FROM curso
  3  WHERE cdept='CIS'
  4  ORDER BY cno DESC;

CNO CNOMBRE                       CRED
--- ----------------------- ----------
C66 BASES DE DATOS RELAC.            3
C55 ARQUITECT. COMPUTADOR            3
C44 CIRCUITOS DIGITALES              3
C33 MATEMÁTICAS DISCRETAS            3
C22 ESTRUCT. DE DATOS                3
C11 INTROD. A LAS CC.                3
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> set pagesize 80
SQL> set linesize 132
SQL> SELECT cdept, cnombre
  2  FROM curso
  3  ORDER BY cdept, cnombre;

CDEP CNOMBRE
---- --------------------
CIS  ARQUITECT. COMPUTADOR
CIS  BASES DE DATOS RELAC.
CIS  CIRCUITOS DIGITALES
CIS  ESTRUCT. DE DATOS
CIS  INTROD. A LAS CC.
CIS  MATEMÁTICAS DISCRETAS
PHIL EMPIRISMO
PHIL EXISTENCIALISMO
PHIL RACIONALISMO
PHIL SOLIPSISMO
THEO COMUNISMO
THEO ESCOLASTICISMO
THEO FUNDAMENTALISMO
THEO HEDONISMO
```

Ejercicio 4.4. Obtener toda la información sobre los cursos que ofrezca el departamento de Informática y Ciencias de la Información con tarifa nula. Asímismo, obtener toda la información sobre cualquier curso que tenga una tarifa que se encuentre entre cero y cien.

```
SQL> SELECT *
  2  FROM curso
  3  WHERE ctarifa = 0
  4  AND CDEPT = 'CIS';

CNO CNOMBRE                  CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO            3          0 CIS
C44 CIRCUITOS DIGITALES    AH HA!                             3          0 CIS

SQL>
```

```
SQL> SELECT *
  2  FROM curso
  3  WHERE ctarifa > 0
  4  AND ctarifa < 100;

CNO CNOMBRE                  CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C22 ESTRUCT. DE DATOS      MUY UTIL                           3         50 CIS
P22 RACIONALISMO           PARA USARLOS CIS                   3         50 PHIL
T22 FUNDAMENTALISMO        PARA DESCUIDADOS                   3         90 THEO

SQL>
```

Ejercicio 4.5. Obtener toda la información sobre los cursos de Filosofía que ofrezcan 3 créditos y que tengan una tarifa que se encuentre entre cero y cien. Asímismo, obtener toda la información existente sobre cualquier curso ofrecido por los departamentos de Informática y Ciencias de la Información o Filosofía.

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT *
  2  FROM curso
  3  WHERE ctarifa > 0
  4  AND ctarifa < 100
  5  AND CDEPT = 'PHIL'
  6  AND cred = 3;

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
P22 RACIONALISMO           PARA USARLOS CIS                3         50 PHIL

SQL>
```

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cdept = 'CIS'
  4  OR cdept = 'PHIL';

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C11 INTROD. A LAS CC.      PARA NOVATOS                    3        100 CIS
C22 ESTRUCT. DE DATOS      MUY UTIL                        3         50 CIS
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO         3          0 CIS
C44 CIRCUITOS DIGITALES    AH HA!                          3          0 CIS
C55 ARQUITECT. COMPUTADOR  MAQ. VON NEUMANN                3        100 CIS
C66 BASES DE DATOS RELAC.  IMPRESCINDIBLE                  3        500 CIS
P11 EMPIRISMO              VERLO PARA CREERLO              3        100 PHIL
P22 RACIONALISMO           PARA USARLOS CIS                3         50 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS                3        200 PHIL
P44 SOLIPSISMO             PARA MI MISMO                   6          0 PHIL
```

Ejercicio 4.6. Obtener toda la información sobre cualquier curso de Informática y Ciencias de la Información o cualquier curso de tarifa nula. Asímismo, obtener toda la información existente sobre cualquier curso que tenga tarifa igual a 50, 100, 150 o 200.

```
SQL> SELECT *
  2  FROM curso
  3  WHERE ctarifa = 0
  4  OR cdept='CIS';

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C11 INTROD. A LAS CC.      PARA NOVATOS                    3        100 CIS
C22 ESTRUCT. DE DATOS      MUY UTIL                        3         50 CIS
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO         3          0 CIS
C44 CIRCUITOS DIGITALES    AH HA!                          3          0 CIS
C55 ARQUITECT. COMPUTADOR  MAQ. VON NEUMANN                3        100 CIS
C66 BASES DE DATOS RELAC.  IMPRESCINDIBLE                  3        500 CIS
P44 SOLIPSISMO             PARA MI MISMO                   6          0 PHIL
T33 HEDONISMO              PARA SANOS                      3          0 THEO
```

```
SQL> SELECT *
  2  FROM curso
  3  WHERE ctarifa=50
  4  OR ctarifa=100
  5  OR ctarifa=150
  6  OR ctarifa=200;

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C11 INTROD. A LAS CC.      PARA NOVATOS                    3        100 CIS
C22 ESTRUCT. DE DATOS      MUY UTIL                        3         50 CIS
C55 ARQUITECT. COMPUTADOR  MAQ. VON NEUMANN                3        100 CIS
P11 EMPIRISMO              VERLO PARA CREERLO              3        100 PHIL
P22 RACIONALISMO           PARA USARLOS CIS                3         50 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS                3        200 PHIL
T11 ESCOLASTICISMO         PARA BEATOS                     3        150 THEO
T44 COMUNISMO              PARA AVAROS                     6        200 THEO
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 4.7. Obtener toda la información existente sobre el nombre de los cursos y la identificación del departamento de todos los cursos que no sean ofrecidos por el departamento de Informática y Ciencias de la Información. Obtener también el nombre y la identificación del departamento de todos los cursos, con la excepción de aquellos ofrecidos por los departamentos de Informática y Ciencias de la Información y Filosofía.

```
SQL> SELECT cnombre, cdept
  2  FROM curso
  3  WHERE NOT cdept='CIS';

CNOMBRE                   CDEP
------------------------- ----
EMPIRISMO                 PHIL
RACIONALISMO              PHIL
EXISTENCIALISMO           PHIL
SOLIPSISMO                PHIL
ESCOLASTICISMO            THEO
FUNDAMENTALISMO           THEO
HEDONISMO                 THEO
COMUNISMO                 THEO
```

```
SQL> SELECT cnombre, cdept
  2  FROM curso
  3  WHERE NOT cdept='CIS'
  4  AND NOT cdept='PHIL';

CNOMBRE                   CDEP
------------------------- ----
ESCOLASTICISMO            THEO
FUNDAMENTALISMO           THEO
HEDONISMO                 THEO
COMUNISMO                 THEO

SQL>
```

Ejercicio 4.8. Obtener toda la información existente sobre los cursos de Teología con tarifa nula o de cualquier curso (independientemente de su departamento y tarifa) que tenga un valor de 6 créditos. Obtener también toda la información existente sobre los cursos de Teología con tarifa nula o que valgan 6 créditos.

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cdept='THEO' AND ctarifa=0
  4  OR cred=6;

CNO CNOMBRE                   CDESCP                          CRED    CTARIFA CDEP
--- ------------------------- ------------------------- ---------- ---------- ----
P44 SOLIPSISMO                PARA MI MISMO                      6          0 PHIL
T33 HEDONISMO                 PARA SANOS                         3          0 THEO
T44 COMUNISMO                 PARA AVAROS                        6        200 THEO

SQL>
```


SYNAPS PREMIUM

```
SQL> SELECT *
  2  FROM curso
  3  WHERE NOT (cdept='CIS' AND ctarifa=0);

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C11 INTROD. A LAS CC.      PARA NOVATOS                       3        100 CIS
C22 ESTRUCT. DE DATOS      MUY UTIL                           3         50 CIS
C55 ARQUITECT. COMPUTADOR  MAQ. VON NEUMANN                   3        100 CIS
C66 BASES DE DATOS RELAC.  IMPRESCINDIBLE                     3        500 CIS
P11 EMPIRISMO              VERLO PARA CREERLO                 3        100 PHIL
P22 RACIONALISMO           PARA USARLOS CIS                   3         50 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS                   3        200 PHIL
P44 SOLIPSISMO             PARA MI MISMO                      6          0 PHIL
T11 ESCOLASTICISMO         PARA BEATOS                        3        150 THEO
T22 FUNDAMENTALISMO        PARA DESCUIDADOS                   3         90 THEO
T33 HEDONISMO              PARA SANOS                         3          0 THEO

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
T44 COMUNISMO              PARA AVAROS                        6        200 THEO
```

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cdept='THEO'
  4  AND (ctarifa=0 OR cred=6);

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
T33 HEDONISMO              PARA SANOS                         3          0 THEO
T44 COMUNISMO              PARA AVAROS                        6        200 THEO

SQL>
```

Ejercicio 4.9. Obtener toda la información existente sobre los cursos que no pertenecen al departamento de Informática y Ciencias de la Información o sobre cualquier curso (independientemente de su departamento) que tenga tarifa nula y valga 3 créditos. Obtener también toda la información disponible sobre todas las filas de la tabla CURSO, exceptuando los cursos de Informática y Ciencias de la información que tengan tarifa nula.

```
SQL> SELECT *
  2  FROM curso
  3  WHERE NOT cdept='CIS'
  4  OR ctarifa=0
  5  AND cred=3;

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO            3          0 CIS
C44 CIRCUITOS DIGITALES    AH HA!                             3          0 CIS
P11 EMPIRISMO              VERLO PARA CREERLO                 3        100 PHIL
P22 RACIONALISMO           PARA USARLOS CIS                   3         50 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS                   3        200 PHIL
P44 SOLIPSISMO             PARA MI MISMO                      6          0 PHIL
T11 ESCOLASTICISMO         PARA BEATOS                        3        150 THEO
T22 FUNDAMENTALISMO        PARA DESCUIDADOS                   3         90 THEO
T33 HEDONISMO              PARA SANOS                         3          0 THEO
T44 COMUNISMO              PARA AVAROS                        6        200 THEO
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 4.10. Obtener toda la información existente sobre los cursos que valgan 2, 6 ó 9 créditos. Obtener también toda la información de todos los cursos que no pertenecen a los departamentos de Teología o Informática y Ciencias de la Información.

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cred IN (2, 6, 9);

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
P44 SOLIPSISMO             PARA MI MISMO                      6          0 PHIL
T44 COMUNISMO              PARA AVAROS                        6        200 THEO

SQL>
```

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cdept NOT in ('THEO', 'CIS');

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
P11 EMPIRISMO              VERLO PARA CREERLO                 3        100 PHIL
P22 RACIONALISMO           PARA USARLOS CIS                   3         50 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS                   3        200 PHIL
P44 SOLIPSISMO             PARA MI MISMO                      6          0 PHIL

SQL>
```

```
SQL> SELECT *
  2  FROM curso
  3  WHERE ctarifa BETWEEN 100 AND 200;

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C11 INTROD. A LAS CC.      PARA NOVATOS                       3        100 CIS
C55 ARQUITECT. COMPUTADOR  MAQ. VON NEUMANN                   3        100 CIS
P11 EMPIRISMO              VERLO PARA CREERLO                 3        100 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS                   3        200 PHIL
T11 ESCOLASTICISMO         PARA BEATOS                        3        150 THEO
T44 COMUNISMO              PARA AVAROS                        6        200 THEO
```

Ejercicio 4.11. Obtener toda la información existente sobre los cursos cuya tarifa se encuentre entre 100 y 200, incluyendo ambos valores. Obtener también toda la información de todos los cursos con tarifa menor que 100 o mayor que 200.

```
SQL> SELECT *
  2  FROM curso
  3  WHERE ctarifa NOT BETWEEN 100 AND 200;

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C22 ESTRUCT. DE DATOS      MUY UTIL                           3         50 CIS
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO            3          0 CIS
C44 CIRCUITOS DIGITALES    AH HA!                             3          0 CIS
C66 BASES DE DATOS RELAC.  IMPRESCINDIBLE                     3        500 CIS
P22 RACIONALISMO           PARA USARLOS CIS                   3         50 PHIL
P44 SOLIPSISMO             PARA MI MISMO                      6          0 PHIL
T22 FUNDAMENTALISMO        PARA DESCUIDADOS                   3         90 THEO
T33 HEDONISMO              PARA SANOS                         3          0 THEO
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 4.12. Obtener toda la información existente sobre los cursos cuyo nombre comience con la letra C. Obtener también toda la información de todos los cursos de los departamentos de Informática y Ciencias de la Información, Teología o Matemáticas que valgan 3 créditos y con tarifa entre 50 y 300 presentando el resultado ordenado por curso dentro de cada departamento.

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cnombre BETWEEN 'C' AND 'CZZ';

CNO CNOMBRE                CDESCP                      CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C44 CIRCUITOS DIGITALES    AH HA!                             3          0 CIS
T44 COMUNISMO              PARA AVAROS                        6        200 THEO

SQL>
```

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cdept IN ('CIS', 'THEO', 'DIR')
  4  AND ctarifa BETWEEN 50 AND 300
  5  AND cred=3
  6  ORDER BY cdept, cnombre;

CNO CNOMBRE                CDESCP                      CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C55 ARQUITECT. COMPUTADOR   MAQ. VON NEUMANN                   3        100 CIS
C22 ESTRUCT. DE DATOS       MUY UTIL                           3         50 CIS
C11 INTROD. A LAS CC.       PARA NOVATOS                       3        100 CIS
T11 ESCOLASTICISMO          PARA BEATOS                        3        150 THEO
T22 FUNDAMENTALISMO         PARA DESCUIDADOS                   3         90 THEO

SQL>
```

Ejercicio 4.13. Obtener la información existente sobre todos los cursos introductorios. Obtener también la información de todos los cursos cuyo nombre finaliza con 'ISMO'.

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cnombre LIKE 'INTROD%';

CNO CNOMBRE                CDESCP                      CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C11 INTROD. A LAS CC.       PARA NOVATOS                       3        100 CIS

SQL>
```

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cnombre LIKE '%ISM%';

CNO CNOMBRE                CDESCP                      CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
P11 EMPIRISMO              VERLO PARA CREERLO                 3        100 PHIL
P22 RACIONALISMO           PARA USARLOS CIS                   3         50 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS                   3        200 PHIL
P44 SOLIPSISMO             PARA MI MISMO                      6          0 PHIL
T11 ESCOLASTICISMO         PARA BEATOS                        3        150 THEO
T22 FUNDAMENTALISMO        PARA DESCUIDADOS                   3         90 THEO
T33 HEDONISMO              PARA SANOS                         3          0 THEO
T44 COMUNISMO              PARA AVAROS                        6        200 THEO
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 4.14. Obtener la información existente sobre todos los cursos que contienen la secuencia 'SC' en alguna posición de su nombre. Obtener también el departamento de todos los cursos cuyo nombre finaliza con 'S' despreciando los valores duplicados.

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cnombre LIKE '%SC%';

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO            3          0 CIS
T11 ESCOLASTICISMO         PARA BEATOS                       3        150 THEO

SQL>
```

```
SQL> SELECT DISTINCT cdept
  2  FROM curso
  3  WHERE cdept LIKE '%S'
  4  OR CDEPT LIKE '%S '
  5  OR cdept LIKE '%S  '
  6  OR cdept LIKE 'S    ';

CDEP
----
CIS

SQL>
```

Ejercicio 4.15. Obtener la información existente sobre todos los cursos que contienen la letra 'H' en la segunda posición de su número de identificación de departamento. Obtener también la información existente sobre todos los cursos que tengan una vocal como segunda letra de su nombre.

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cdept LIKE '_H__';

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
P11 EMPIRISMO              VERLO PARA CREERLO                3        100 PHIL
P22 RACIONALISMO           PARA USARLOS CIS                  3         50 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS                  3        200 PHIL
P44 SOLIPSISMO             PARA MI MISMO                     6          0 PHIL
T11 ESCOLASTICISMO         PARA BEATOS                       3        150 THEO
T22 FUNDAMENTALISMO        PARA DESCUIDADOS                  3         90 THEO
T33 HEDONISMO              PARA SANOS                        3          0 THEO
T44 COMUNISMO              PARA AVAROS                       6        200 THEO
```

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cnombre LIKE '_A%'
  4  OR cnombre LIKE '_E%'
  5  OR cnombre LIKE '_I%'
  6  OR cnombre LIKE '_O%'
  7  OR cnombre LIKE '_U%';

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- ---------------------- ------------------------- ---------- ---------- ----
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO            3          0 CIS
C44 CIRCUITOS DIGITALES    AH HA!                             3          0 CIS
C66 BASES DE DATOS RELAC.  IMPRESCINDIBLE                     3        500 CIS
P22 RACIONALISMO           PARA USARLOS CIS                   3         50 PHIL
P44 SOLIPSISMO             PARA MI MISMO                      6          0 PHIL
T22 FUNDAMENTALISMO        PARA DESCUIDADOS                   3         90 THEO
T33 HEDONISMO              PARA SANOS                         3          0 THEO
T44 COMUNISMO              PARA AVAROS                        6        200 THEO
```


SYNAPS PREMIUM

Ejercicio 4.16. Obtener la información existente sobre todos los cursos que no tengan una vocal como segunda letra de su nombre.

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cnombre NOT LIKE '_A%'
  4  OR cnombre NOT LIKE '_E%'
  5  OR cnombre NOT LIKE '_I%'
  6  OR cnombre NOT LIKE '_O%'
  7  OR cnombre NOT LIKE '_U%';

CNO CNOMBRE                CDESCP                        CRED    CTARIFA CDEP
--- -------------------- ------------------------- ---------- ---------- ----
C11 INTROD. A LAS CC.      PARA NOVATOS                   3        100 CIS
C22 ESTRUCT. DE DATOS      MUY UTIL                       3         50 CIS
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO        3          0 CIS
C44 CIRCUITOS DIGITALES    AH HA!                         3          0 CIS
C55 ARQUITECT. COMPUTADOR  MAQ. VON NEUMANN               3        100 CIS
C66 BASES DE DATOS RELAC.  IMPRESCINDIBLE                 3        500 CIS
P11 EMPIRISMO              VERLO PARA CREERLO             3        100 PHIL
P22 RACIONALISMO           PARA USARLOS CIS               3         50 PHIL
P33 EXISTENCIALISMO        PARA USARLOS CIS               3        200 PHIL
P44 SOLIPSISMO             PARA MI MISMO                  6          0 PHIL
T11 ESCOLASTICISMO         PARA BEATOS                    3        150 THEO
T22 FUNDAMENTALISMO        PARA DESCUIDADOS               3         90 THEO
T33 HEDONISMO              PARA SANOS                     3          0 THEO
T44 COMUNISMO              PARA AVAROS                    6        200 THEO
```

Ejercicio 4.17. Obtener la información relativa a nombre y tarifa de todos los cursos del departamento de Informática y Ciencias de la Información suponiendo que la tarifa actual para cada curso se ve aumentada en 25. Resolver la consulta anterior suponiendo que se produce una duplicación de la tarifa de los cursos del departamento de Informática y Ciencias de la Información.

```
SQL> SELECT cnombre, ctarifa, ctarifa+25 "NUEVA TARIFA"
  2  FROM curso
  3  WHERE cdept='CIS';

CNOMBRE                    CTARIFA NUEVA TARIFA
-------------------- ---------- ------------
INTROD. A LAS CC.             100          125
ESTRUCT. DE DATOS              50           75
MATEMÁTICAS DISCRETAS           0           25
CIRCUITOS DIGITALES             0           25
ARQUITECT. COMPUTADOR         100          125
BASES DE DATOS RELAC.         500          525
```

```
  1  SELECT cnombre, ctarifa, ctarifa*2 "NUEVA TARIFA"
  2  FROM curso
  3* WHERE cdept='CIS'

CNOMBRE                    CTARIFA NUEVA TARIFA
-------------------- ---------- ------------
INTROD. A LAS CC.             100          200
ESTRUCT. DE DATOS              50          100
MATEMÁTICAS DISCRETAS           0            0
CIRCUITOS DIGITALES             0            0
ARQUITECT. COMPUTADOR         100          200
BASES DE DATOS RELAC.         500         1000
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 4.18. Obtener la información relativa a nombre y tarifa de todos los cursos del departamento de Informática y Ciencias de la Información suponiendo que la tarifa actual se multiplica por 2.375. ¿Y si la tarifa se reduce a un tercio de su actual valor? ¿Cuál es la relación entre la tarifa y el número de créditos para los cursos de este departamento? ¿Cuál sería la tarifa media por hora de crédito para los cursos de este departamento si la tarifa fuese incrementada en 25?

```
SQL> SELECT cnombre, ctarifa, ctarifa/3 "NUEVA TARIFA"
  2  FROM curso
  3  WHERE cdept='CIS';

CNOMBRE                    CTARIFA NUEVA TARIFA
-------------------- ---------- ------------
INTROD. A LAS CC.             100   33,3333333
ESTRUCT. DE DATOS              50   16,6666667
MATEMÁTICAS DISCRETAS           0            0
CIRCUITOS DIGITALES             0            0
ARQUITECT. COMPUTADOR         100   33,3333333
BASES DE DATOS RELAC.         500   166,666667
BASES DE DATOS RELAC.         500       1187,5
```

```
SQL> SELECT cnombre, ctarifa, ctarifa/cred "TARIFA POR CRÉDITO"
  2  FROM curso
  3  WHERE cdept='CIS';

CNOMBRE                    CTARIFA TARIFA POR CRÉDITO
-------------------- ---------- ------------------
INTROD. A LAS CC.             100         33,3333333
ESTRUCT. DE DATOS              50         16,6666667
MATEMÁTICAS DISCRETAS           0                  0
CIRCUITOS DIGITALES             0                  0
ARQUITECT. COMPUTADOR         100         33,3333333
BASES DE DATOS RELAC.         500         166,666667
```

```
SQL> SELECT cnombre, ctarifa, ctarifa/cred "TARIFA POR CRÉDITO",
  2  (ctarifa+25)/cred "TARIFA MEDIA"
  3  FROM curso
  4  WHERE cdept='CIS';

CNOMBRE                    CTARIFA TARIFA POR CRÉDITO TARIFA MEDIA
-------------------- ---------- ------------------ ------------
INTROD. A LAS CC.             100         33,3333333   41,6666667
ESTRUCT. DE DATOS              50         16,6666667           25
MATEMÁTICAS DISCRETAS           0                  0   8,33333333
CIRCUITOS DIGITALES             0                  0   8,33333333
ARQUITECT. COMPUTADOR         100         33,3333333   41,6666667
BASES DE DATOS RELAC.         500         166,666667          175
```

Se observa que cuando las expresiones contienen valores enteros y decimales, el resultado es decimal y el sistema lo presenta con la mayor exactitud posible. Además, cuando las divisiones presentan un número infinito de decimales, el sistema presenta los resultados con la mayor precisión posible, que por defecto es 10 dígitos.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 4.19. Obtener la información relativa a nombre y tarifa de todos los cursos del departamento de Informática y Ciencias de la Información que tienen una tarifa por hora de crédito mayor que 30. Obtener también la información relativa a nombre y tarifa de todos los cursos del departamento de Informática y Ciencias de la Información suponiendo que la tarifa actual para cada curso se ve aumentada en 25 ordenando el resultado por el nuevo valor de la tarifa.

```
SQL> SELECT cnombre, ctarifa, ctarifa/cred "TARIFA POR CRÉDITO"
  2  FROM curso
  3  WHERE cdept='CIS'
  4  AND ctarifa/cred>30;

CNOMBRE                   CTARIFA TARIFA POR CRÉDITO
--------------------- ---------- ------------------
INTROD. A LAS CC.            100         33,3333333
ARQUITECT. COMPUTADOR        100         33,3333333
BASES DE DATOS RELAC.        500         166,666667

SQL>
```

```
SQL> SELECT cnombre, ctarifa, ctarifa+25 "NUEVA TARIFA"
  2  FROM curso
  3  WHERE cdept='CIS'
  4  ORDER BY 3;

CNOMBRE                   CTARIFA NUEVA TARIFA
--------------------- ---------- ------------
MATEMÁTICAS DISCRETAS           0           25
CIRCUITOS DIGITALES             0           25
ESTRUCT. DE DATOS              50           75
INTROD. A LAS CC.             100          125
ARQUITECT. COMPUTADOR         100          125
BASES DE DATOS RELAC.         500          525
```

Se observa que para ordenar por una columna calculada, es necesario hacer referencia a ella mediante el número que ocupa en la lista de selección de la cláusula SELECT.

© Alfaomega - RC Libros

SYNAPS PREMIUM

CAPÍTULO 5

CONSULTAS MULTITABLA (JOINS) Y CONSULTAS RESUMEN

INTRODUCCIÓN

Las consultas multitabla o JOINS, también denominadas combinaciones o composiciones, permiten recuperar datos de dos tablas o más según las relaciones lógicas entre ellas. Las combinaciones indican cómo debería utilizar Oracle los datos de una tabla para seleccionar las filas de otra tabla.

Una condición de combinación (o composición) define la forma en la que dos tablas se relacionan en una consulta al:

- Especificar la columna de cada tabla que debe usarse para la combinación. Una condición de combinación típica especifica una clave externa de una tabla y su clave asociada en otra tabla.

- Especificar un operador lógico (=, <>, etc.) para usarlo en los valores de comparación de las columnas.

Por ejemplo, la consulta siguiente combina las tablas pedidos y clientes (que se separan por comas en la cláusula FROM) mostrando el número de pedido y su importe y el número de cliente y su límite de crédito, utilizando para la combinación

SYNAPS PREMIUM

la columna CLIE de la tabla PEDIDOS y la columna NUM_CLIE de la tabla CLIENTES mediante el operador lógico =:

```
SELECT num_pedido, importe, empresa, limite_credito
FROM pedidos, clientes
WHERE clie=num_clie;
```

Cuando en una consulta simple se hace referencia a múltiples tablas, ninguna de las referencias a las columnas debe ser ambigua. Cualquier nombre de columna que esté duplicado en varias tablas a las que se hace referencia en la consulta debe ser precedido del nombre de la tabla y de un punto. Por ejemplo, el campo EMPRESA de la tabla CLIENTES se referenciará como CLIENTES.EMPRESA.

La consulta anterior podría escribirse como sigue:

```
SELECT pedidos.num_pedido, pedidos.importe,
    clientes.empresa, clientes.limite_credito
FROM pedidos, clientes
WHERE clie=num_clie;
```

Cuando el nombre de una columna no está duplicado en varias de las tablas usadas en la consulta, las referencias al mismo no tienen que calificarse con el nombre de la tabla. Algunas veces, una instrucción SELECT es difícil de entender porque no hay nada que indique qué tabla ha suministrado cada columna. Se puede mejorar la comprensión de la consulta si todas las columnas se califican con sus nombres de tabla. La comprensión puede aumentarse más si se usan alias de tablas, especialmente cuando los propios nombres de las tablas se deben calificar con los nombres de las bases de datos y de los propietarios.

JOIN DE DOS TABLAS

Un JOIN de dos tablas es una composición entre las mismas basada en la coincidencia exacta (u otro tipo de comparación) de dos columnas, una de cada tabla. El JOIN forma parejas de filas haciendo coincidir los contenidos de las columnas relacionadas. La consulta siguiente visualiza la información de las tablas curso y departamento para la que los valores de CDEPT en curso coinciden con los valores de DEPT en departamento:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> set pagesize 80
SQL> set linesize 200
SQL> SELECT *
  2  FROM curso, departamento
  3  WHERE cdept=dept;

CNO CNOMBRE                CDESCP                      CRED    CTARIFA CDEP DEPT DE  DDESPACHO DCH
--- ---------------------- ------------------------- ------- ---------- ---- ---- -- ---------- -
C11 INTROD. A LAS CC.      PARA NOVATOS                    3        100 CIS  CIS  SC         300 80
C22 ESTRUCT. DE DATOS      MUY UTIL                        3         50 CIS  CIS  SC         300 80
C33 MATEMÁTICAS DISCRETAS  ABSOLUTAMENTE NECESARIO         3          0 CIS  CIS  SC         300 80
C44 CIRCUITOS DIGITALES    AH HA!                          3          0 CIS  CIS  SC         300 80
C55 ARQUITECT. COMPUTADOR  MAQ. VON NEUMANN                3        100 CIS  CIS  SC         300 80
C66 BASES DE DATOS RELAC.  IMPRESCINDIBLE                  3        500 CIS  CIS  SC         300 80
P11 EMPIRISMO              VERLO PARA CREERLO              3        100 PHIL PHIL HU         100 60
P22 RACIONALISMO           PARA USARLOS CIS                3         50 PHIL PHIL HU         100 60
P33 EXISTENCIALISMO        PARA USARLOS CIS                3        200 PHIL PHIL HU         100 60
P44 SOLIPSISMO             PARA MI MISMO                   6          0 PHIL PHIL HU         100 60
T11 ESCOLASTICISMO         PARA BEATOS                     3        150 THEO THEO HU         200 10
T22 FUNDAMENTALISMO        PARA DESCUIDADOS                3         90 THEO THEO HU         200 10
T33 HEDONISMO              PARA SANOS                      3          0 THEO THEO HU         200 10
T44 COMUNISMO              PARA AVAROS                     6        200 THEO THEO HU         200 10
```

En la consulta siguiente, para cada curso con una tarifa mayor que 100 se visualiza el nombre del curso y su tarifa junto con el número facultativo del jefe de departamento que ofrece el curso:

```
SQL> SELECT cnombre, ctarifa, dchfno
  2  FROM curso, departamento
  3  WHERE cdept=dept AND ctarifa > 100;

CNOMBRE                     CTARIFA DCH
--------------------- ---------- ---
BASES DE DATOS RELAC.          500 80
EXISTENCIALISMO                200 60
ESCOLASTICISMO                 150 10
COMUNISMO                      200 10

SQL>
```

La consulta siguiente combina las tablas pedidos y clientes mostrando en el resultado el número de pedido y su importe y el nombre de cliente y su límite de crédito, utilizando para la combinación la columna CLIE de la tabla PEDIDOS y la columna NUM_CLIE de la tabla CLIENTES mediante el operador lógico igualdad:

```
SQL> SELECT num_pedido, importe, empresa, limite_credito
  2  FROM pedidos, clientes
  3  WHERE clie=num_clie;

NUM_PEDIDO    IMPORTE EMPRESA              LIMITE_CREDITO
---------- ---------- -------------------- --------------
    110036      22500 Ace International            35000
    112961      31500 J.P. Sinclair                35000
    112963       3276 Acme Mfg.                    50000
    112968       3978 First Corp.                  65000
    112975       2100 JCP Inc.                     50000
    112979      15000 Orion Corp.                  20000
    112983        702 Acme Mfg.                    50000
    112987      27500 Acme Mfg.                    50000
    112989       1458 Jones Mfg.                   65000
    112992        760 Midwest Systems              60000
    112993       1896 Fred Lewis Corp.             65000
    112997        652 Peter Brothers               40000
    113003       5625 Holm Landis                  55000
    113007       2925 Zetacorp                     50000
    113012       3745 JCP Inc.                     50000
    113013        652 Midwest Systems              60000
    113024       7100 Orion Corp.                  20000
    113027       4104 Acme Mfg.                    50000
    113034        632 Ace International            35000
    113042      22500 Ian Schmidt                  20000
    113045      45000 Zetacorp                     50000
    113048       3750 Rico Enterprises             50000
    113049       2130 Midwest Systems              60000
    113051       1420 Midwest Systems              60000
    113055        150 Holm Landis                  55000
    113057        600 JCP Inc.                     50000
    113058       1480 Holm Landis                  55000
    113062       2430 Peter Brothers               40000
    113065       2130 Fred Lewis Corp.             65000
    113069      31350 Chen Associates              25000
```


SYNAPS PREMIUM

La consulta de un JOIN de dos tablas tiene el mismo aspecto que cualquier consulta realizada con SELECT, pero presenta dos características novedosas. En primer lugar, la cláusula FROM presenta dos tablas separadas por una coma (`FROM pedidos, clientes`). En segundo lugar la condición de búsqueda (`WHERE clie=num_clie`) compara columnas de dos tablas diferentes. Además, como toda condición de búsqueda, restringe las filas que aparecen en el resultado de la consulta. Dado que se trata de una consulta que afecta a dos tablas, la condición de búsqueda restringe las parejas de filas que generan los resultados.

Podríamos decir que la consulta anterior genera resultados solo para los pares de filas en los que el número de cliente (CLIE) en la tabla PEDIDOS coincide con el número de cliente (NUM_CLIE) en la tabla CLIENTES.

Las consultas multitabla más comunes implican a dos tablas que tienen una relación natural padre/hijo. En la consulta anterior, cada pedido (hijo) tiene un cliente asociado (padre), y cada cliente (padre) puede tener muchos pedidos asociados (hijos). Los pares de filas que generan los resultados de la consulta son combinaciones de fila padre/hijo. En términos de clave primaria/clave foránea, la tabla que contiene la clave foránea es el hijo en la relación y la tabla que contiene la clave primaria es el padre. En la consulta debe especificarse una condición de búsqueda que compare la clave foránea y la clave primaria.

En el ejemplo siguiente se listan los vendedores (tabla REPVENTAS), la ciudad (tabla OFICINAS) y la región en que trabajan (tabla OFICINAS):

```
SQL> SELECT name, ciudad, region
  2  FROM repventas, oficinas
  3  WHERE oficina_rep=oficina;

NAME            CIUDAD          REGION
--------------- --------------- ----------
Dan Roberts     Chicago         Este
Sue Smith       Los Ángeles     Oeste
Paul Cruz       Chicago         Este
Bob Smith       Chicago         Este
Bill Adams      Atlanta         Este
Sam Clark       New York        Este
Nancy Angelli   Denver          Oeste
Larry Fitch     Los Ángeles     Oeste
Mary Jones      New York        Este
```

En este caso se ha elegido la condición de búsqueda del JOIN (WHERE oficina_rep=oficina) porque esos campos relacionan las dos tablas debido a que OFICINA_REP es el campo de la tabla REPVENTAS que es clave foránea para el campo OFICINA de la tabla OFICINAS (obsérvese el diseño de las tablas en el Ejercicio 3.1).

© Alfaomega - RC Libros

SYNAPS PREMIUM

En el ejemplo siguiente se obtienen las oficinas (tabla OFICINAS) y los nombres y títulos de sus directores (tabla REPVENTAS ambos):

```
SQL> SELECT ciudad, name, titulo
  2  FROM oficinas, repventas
  3  WHERE dir=num_empl;

CIUDAD          NAME            TITULO
--------------- --------------- ----------
New York        Sam Clark       VP Ventas
Chicago         Bob Smith       Dir Ventas
Atlanta         Bill Adams      Rep Ventas
Los Ángeles     Larry Fitch     Dir Ventas
Denver          Larry Fitch     Dir Ventas

SQL>
```

En este caso se ha elegido la condición de búsqueda del JOIN (`WHERE dir=num_empl`) porque esos campos relacionan las dos tablas debido a que DIR es un campo de la tabla OFICINAS que es clave foránea para el campo NUM_EMPL de la tabla REPVENTAS.

JOINS con criterios de selección de filas

La condición de búsqueda de un JOIN no tiene por qué ser la única condición de búsqueda presente en una consulta multitabla, ya que puede combinarse con otras condiciones de búsqueda que restrinjan aún más las filas devueltas en el resultado. Por ejemplo, la consulta siguiente lista las oficinas y los nombres y títulos de sus directores mostrando únicamente las oficinas con un objetivo de ventas superior a 600.000:

```
SQL> SELECT ciudad, name, titulo
  2  FROM oficinas, repventas
  3  WHERE dir=num_empl
  4  AND objetivo < 600000;

CIUDAD          NAME            TITULO
--------------- --------------- ----------
New York        Sam Clark       VP Ventas
Atlanta         Bill Adams      Rep Ventas
Denver          Larry Fitch     Dir Ventas

SQL>
```

La consulta siguiente visualiza el nombre del empleado y su sueldo junto con la tarifa del curso y la diferencia entre la tarifa y el salario para los miembros del personal tutores de curso cuya tarifa excede del sueldo al menos es 52:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT enombre, esueldo, ctarifa, ctarifa-esueldo
  2  FROM personal, curso
  3  WHERE dept=cdept AND (ctarifa-esueldo) >= 52;

ENOMBRE              ESUELDO    CTARIFA CTARIFA-ESUELDO
--------------- ---------- ---------- ---------------
LUCAS                    53        150              97
JUAN                     54        150              96
MARCOS                   52        150              98
MATEO                    51        150              99
LUCAS                    53        200             147
JUAN                     54        200             146
MARCOS                   52        200             148
MATEO                    51        200             149
```

JOINS con múltiples columnas de emparejamiento

Cuando dos tablas están relacionadas por varias relaciones clave foránea/clave primaria, pueden especificarse ambos pares de columnas de emparejamiento en la condición de búsqueda del JOIN. Por ejemplo, las columnas FAB y PRODUCTO de la tabla PEDIDOS son juntas una clave foránea para la tabla PRODUCTOS emparejadas con las columnas ID_FAB e ID_PRODUCTO (ambas forman la clave primaria de la tabla PRODUCTOS). De esta forma, la consulta siguiente lista todos los pedidos mostrando los importes y las descripciones del producto:

```
  1  SELECT num_pedido, importe, descripcion
  2  FROM pedidos, productos
  3  WHERE fab=id_fab
  4* AND producto=id_producto

NUM_PEDIDO    IMPORTE DESCRIPCION
---------- ---------- --------------------
    110036      22500 Montador
    112961      31500 Bisagra Izqda.
    112963       3276 Artículo Tipo 4
    112968       3978 Artículo Tipo 4
    112975       2100 Pasador Bisagra
    112979      15000 Montador
    112983        702 Artículo Tipo 4
    112987      27500 Extractor
    112989       1458 Bancada Motor
    112992        760 Artículo Tipo 2
    112993       1896 V Stago Trinquete
    112997        652 Manivela
    113003       5625 Riostra 2-Tm
    113007       2925 Riostra 1/2 Tm
    113012       3745 Artículo Tipo 3
    113013        652 Manivela
    113024       7100 Reductor
    113027       4104 Artículo Tipo 2
    113034        632 V Stago Trinquete
    113042      22500 Bisagra Dcha.
    113045      45000 Bisagra Dcha.
    113048       3750 Riostra 2-Tm
    113049       2130 Reductor
    113051       1420 Reductor
    113055        150 Ajustador
    113057        600 Ajustador
    113058       1480 Cubierta
    113062       2430 Bancada Motor
    113065       2130 Reductor
    113069      31350 Riostra 1-Tm
```

En este tipo de consulta, la condición de búsqueda exige que los pares de filas relacionados en las tablas PEDIDOS y PRODUCTOS son aquellas en los que ambos pares de columnas coincidentes (condición de búsqueda del JOIN) tienen los mismos valores.

© Alfaomega - RC Libros

SYNAPS PREMIUM

JOINS DE TRES O MÁS TABLAS

Un JOIN de tres o más tablas es una composición entre las mismas basada en la coincidencia exacta (u otro tipo de comparación) de tres o más columnas, una de cada tabla. El JOIN forma filas haciendo coincidir los contenidos de las columnas relacionadas. Como ejemplo, en la consulta siguiente relacionaremos las tablas PEDIDOS CLIENTES y REPVENTAS para obtener la lista de los pedidos superiores a 25.000, incluyendo el nombre del vendedor que tomó el pedido y el nombre del cliente que lo solicitó.

```
SQL> SELECT num_pedido, importe, empresa, name
  2  FROM pedidos, clientes, repventas
  3  WHERE clie=num_clie
  4  AND rep=num_empl
  5  AND importe>25000;

NUM_PEDIDO    IMPORTE EMPRESA                   NAME
---------- ---------- ---------------------- ---------------
    112961      31500 J.P. Sinclair          Sam Clark
    112987      27500 Acme Mfg.              Bill Adams
    113045      45000 Zetacorp               Larry Fitch
    113069      31350 Chen Associates        Nancy Angelli

SQL>
```

La consulta se ha basado en que la columna CLIE es clave foránea para la tabla CLIENTES que enlaza cada pedido con el cliente que lo remitió. Por otra parte, la columna REP es una clave foránea para la tabla REPVENTAS que liga cada pedido con el vendedor que lo aceptó.

La consulta siguiente produce una lista de clase para la primera sección del curso de existencialismo visualizando el número de curso, número de sección y nombre del curso seguido por el número de estudiante de cada estudiante matriculado en la clase:

```
SQL> SELECT CLASE.cno, CLASE.sec, cnombre,  sno
  2  FROM MATRICULA, CURSO, CLASE
  3  WHERE CURSO.cno=CLASE.cno
  4  AND CLASE.cno=MATRICULA.cno
  5  AND MATRICULA.sec=CLASE.sec
  6  AND MATRICULA.SEC='01';

CNO SE CNOMBRE                 SNO
--- -- ---------------------- ---
C11 01 INTROD. A LAS CC.      325
C11 01 INTROD. A LAS CC.      800
P33 01 EXISTENCIALISMO        100
P33 01 EXISTENCIALISMO        800
T11 01 ESCOLASTICISMO         100
T11 01 ESCOLASTICISMO         150
T11 01 ESCOLASTICISMO         800
```

La consulta siguiente produce una lista de clase para la segunda sección del curso de existencialismo visualizando el número de curso, número de sección, nombre del curso y número facultativo del instructor seguido por el número de estudiante de cada estudiante matriculado en la clase:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT CLASE.cno, CLASE.sec, cnombre, fno, sno
  2  FROM MATRICULA, CURSO, CLASE, CLAUSTRO
  3  WHERE CURSO.cno=CLASE.cno
  4  AND CLASE.cno=MATRICULA.cno
  5  AND MATRICULA.sec=CLASE.sec
  6  AND MATRICULA.SEC='02';

CNO SE CNOMBRE                   FNO SNO
--- -- ---------------------- --- ---
C11 02 INTROD. A LAS CC.       06  100
C11 02 INTROD. A LAS CC.       06  150
C11 02 INTROD. A LAS CC.       10  100
C11 02 INTROD. A LAS CC.       10  150
C11 02 INTROD. A LAS CC.       08  100
C11 02 INTROD. A LAS CC.       08  150
C11 02 INTROD. A LAS CC.       85  100
C11 02 INTROD. A LAS CC.       85  150
C11 02 INTROD. A LAS CC.       60  100
C11 02 INTROD. A LAS CC.       60  150
C11 02 INTROD. A LAS CC.       80  100
C11 02 INTROD. A LAS CC.       80  150
C11 02 INTROD. A LAS CC.       65  100
C11 02 INTROD. A LAS CC.       65  150
```

En la consulta siguiente relacionaremos las tablas PEDIDOS CLIENTES y REPVENTAS para obtener la lista de pedidos superiores a 25.000 mostrando el nombre del cliente que lo remitió y el nombre del vendedor asignado a ese cliente:

```
SQL> SELECT num_pedido, importe, empresa, name
  2  FROM pedidos, clientes, repventas
  3  WHERE clie=num_clie
  4  AND repclie=num_empl
  5  AND importe>25000;

NUM_PEDIDO    IMPORTE EMPRESA              NAME
---------- ---------- -------------------- --------------
    112961      31500 J.P. Sinclair        Sam Clark
    112987      27500 Acme Mfg.            Bill Adams
    113045      45000 Zetacorp             Larry Fitch
    113069      31350 Chen Associates      Paul Cruz

SQL>
```

Esta última consulta relaciona cada pedido con su cliente (la columna CLIE de la tabla PEDIDOS es clave foránea para la tabla CLIENTES) y cada cliente con su vendedor (la columna REPCLIE de la tabla CLIENTES es clave foránea para la tabla REPVENTAS).

A continuación, se presenta una consulta que involucra a cuatro tablas (JOIN cuádruple) para obtener los pedidos superiores a 25.000 mostrando el nombre del cliente que los ordenó, el vendedor asociado al cliente y la oficina donde el vendedor trabaja:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL>  SELECT num_pedido, importe, empresa, name, ciudad
  2  FROM pedidos, clientes, repventas, oficinas
  3  WHERE clie=num_clie
  4  AND repclie=num_empl
  5  AND oficina_rep=oficina
  6  AND importe>25000;

NUM_PEDIDO    IMPORTE EMPRESA              NAME           CIUDAD
---------- ---------- -------------------- -------------- --------------
    112961      31500 J.P. Sinclair        Sam Clark      New York
    112987      27500 Acme Mfg.            Bill Adams     Atlanta
    113045      45000 Zetacorp             Larry Fitch    Los Ángeles
    113069      31350 Chen Associates      Paul Cruz      Chicago

SQL>
```

La condición de búsqueda del JOIN ha utilizado el campo CLIE de la tabla PEDIDOS que es clave foránea para el campo NUM_CLIE de la tabla CLIENTES. Utiliza también el campo REPCLIE de la tabla CLIENTES que es clave foránea para el campo NUM_EMPL de la tabla REPVENTAS. Por último, utiliza el campo OFICINA_REP de la tabla REPVENTTAS que es clave foránea para el campo OFICINA de la tabla OFICINAS.

Hasta aquí hemos visto ejemplos de consultas con JOINS puros; sin embargo, SQL no exige que las columnas de emparejamiento tengan relación clave primaria/clave foránea. De hecho, dos columnas cualesquiera de dos tablas pueden utilizarse como columnas de emparejamiento siempre y cuando subtipos de datos sean comparables. Por ejemplo, la consulta siguiente devuelve todos los pedidos recibidos en los días en que un nuevo vendedor fue contratado:

```
SQL> SELECT num_pedido, importe, fecha_pedido, name
  2  FROM pedidos, repventas
  3  WHERE fecha_pedido=contrato;

NUM_PEDIDO    IMPORTE FECHA_PE NAME
---------- ---------- -------- --------------
    112968       3978 12/10/89 Larry Fitch
    112975       2100 12/10/89 Larry Fitch
    112979      15000 12/10/89 Larry Fitch

SQL>
```

El resultado de la consulta anterior se compone sencillamente de los pares de filas de las tablas PEDIDOS y REPVENTAS en los que el valor de la columna FECHA_PEDIDO coincide con el valor en la columna CONTRATO para el vendedor.

JOIN CON CONDICIONES DE BÚSQUEDA SIN IGUALDAD

Sabemos que un JOIN de dos tablas es una composición entre las mismas basada en la coincidencia exacta (u otro tipo de comparación) de dos columnas (una de cada tabla) reflejada en la condición de búsqueda del JOIN.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Hasta aquí solo hemos utilizado JOINS en los que en la condición de búsqueda únicamente aparece la igualdad; sin embargo, en la condición de búsqueda del JOIN puede especificarse un operador lógico cualquiera (=, <>, etc.) para usarlo en los valores de comparación de las columnas. El JOIN forma parejas de filas haciendo cumplirse las condiciones del JOIN en los contenidos de las columnas relacionadas.

Como ejemplo listamos todas las combinaciones de vendedores y oficinas en donde la cuota del vendedor es superior al objetivo de la oficina:

```
SQL> SELECT name, cuota, ciudad, objetivo
  2  FROM repventas, oficinas
  3  WHERE cuota>objetivo;

NAME                 CUOTA CIUDAD           OBJETIVO
-------------- ---------- --------------- ----------
Sue Smith          350000 Denver              300000
Bill Adams         350000 Denver              300000
Larry Fitch        350000 Denver              300000

SQL>
```

La condición de búsqueda del JOIN anterior también puede usarse como sigue:

```
SQL> SELECT name, cuota, ciudad, objetivo
  2  FROM repventas, oficinas
  3  WHERE cuota-objetivo>0;

NAME                 CUOTA CIUDAD           OBJETIVO
-------------- ---------- --------------- ----------
Sue Smith          350000 Denver              300000
Bill Adams         350000 Denver              300000
Larry Fitch        350000 Denver              300000

SQL>
```

JOINS INTERNOS Y EXTERNOS

El concepto que conocemos hasta ahora de JOIN corresponde a lo que se conoce como JOIN INTERNO u operación de composición que combina información procedente de dos tablas mediante la formación de pares de filas relacionadas en las dos tablas según la condición de búsqueda del JOIN. Los pares de filas que forman la tabla compuesta son los de las columnas correspondientes en cada una de las dos tablas que tienen el mismo valor. Por ejemplo, si queremos listar los vendedores y las ciudades en que trabajan podemos utilizar la consulta que se especifica a continuación:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT name, ciudad
  2  FROM repventas, oficinas
  3  WHERE oficina_rep=oficina;

NAME            CIUDAD
--------------- ---------------
Dan Roberts     Chicago
Sue Smith       Los Ángeles
Paul Cruz       Chicago
Bob Smith       Chicago
Bill Adams      Atlanta
Sam Clark       New York
Nancy Angelli   Denver
Larry Fitch     Los Ángeles
Mary Jones      New York
```

En la consulta anterior, la cláusula FROM genera todas las combinaciones posibles de filas de las tablas REPVENTAS y OFICINAS produciendo la tabla producto de las dos, cuyo número de filas es el producto de las filas de las dos tablas. La cláusula WHERE introduce una restricción seleccionando únicamente las filas de la tabla producto para las que el número de oficina de representación del vendedor de la tabla REPVENTAS coincide con el número de oficina de la tabla OFICINAS. Pero al comprobar esta última coincidencia (o emparejamiento) el JOIN INTERNO es muy estricto y elimina del resultado las filas que provoquen el más mínimo problema de emparejamiento (por ejemplo, los valores NULL de cualquiera de los campos). En nuestro ejemplo, OFICINA_REP tiene un valor nulo para el registro *Tom Snyder* que no se corresponde con ninguno de los números de oficina de la tabla OFICINAS, por lo que este registro no aparecerá en el resultado del JOIN al no existir un emparejamiento estricto.

Si queremos que este tipo de resultados, para los que no hay un emparejamiento estricto entre las filas de la condición de búsqueda del JOIN, aparezca en el resultado del JOIN, tenemos que acudir al concepto de JOIN EXTERNO o composición externa. Por lo tanto, la composición externa relaja las condiciones de la composición interna y aporta más información al resultado.

JOIN EXTERNO completo

Un JOIN EXTERNO completo es una ampliación de un JOIN INTERNO (o simplemente JOIN) que contiene la siguiente información:

- El JOIN INTERNO (o JOIN) de las dos tablas.

- Por cada fila de la primera tabla que no haya correspondido a ninguna fila de la segunda tabla, se añade una fila a los resultados, utilizando los valores de

SYNAPS PREMIUM

las columnas de la primera tabla y suponiendo un valor NULL para todas las columnas de la segunda tabla correspondientes en el emparejamiento.

- Por cada fila de la segunda tabla que no haya correspondido a ninguna fila de la primera tabla, se añade una fila a los resultados, utilizando los valores de las columnas de la segunda tabla y suponiendo un valor NULL para todas las columnas de la primera tabla correspondientes en el emparejamiento.

En cuanto a notación, el JOIN EXTERNO completo utiliza la sintaxis OUTER JOIN en la cláusula FROM de la consulta y la sintaxis ON en lugar de la cláusula WHERE que precede la condición de búsqueda del JOIN. Este tipo de sintaxis también puede utilizarse para el JOIN normal (o JOIN INTERNO) cambiando OUTER JOIN por INNER JOIN.

JOIN EXTERNO izquierdo y derecho

En el párrafo anterior hemos visto el JOIN EXTERNO completo, pero también puede considerarse el JOIN EXTERNO IZQUIERDO, que se produce entre dos tablas ofreciendo la siguiente información:

- El JOIN INTERNO (o JOIN) de las dos tablas.
- Por cada fila de la primera tabla que no haya correspondido a ninguna fila de la segunda tabla, se añade una fila a los resultados, utilizando los valores de las columnas de la primera tabla y suponiendo un valor NULL para todas las columnas de la segunda tabla correspondientes en el emparejamiento.

En cuanto a notación, el JOIN EXTERNO IZQUIERDO utiliza la sintaxis LEFT JOIN en la cláusula FROM de la consulta y la sintaxis ON en lugar de la cláusula WHERE que precede la condición de búsqueda del JOIN.

Simétricamente, el JOIN EXTERNO DERECHO se produce entre dos tablas ofreciendo la siguiente información:

- El JOIN INTERNO (o JOIN) de las dos tablas.
- Por cada fila de la segunda tabla que no haya correspondido a ninguna fila de la primera tabla, se añade una fila a los resultados, utilizando los valores de las columnas de la segunda tabla y suponiendo un valor NULL para todas las columnas de la primera tabla correspondientes en el emparejamiento.

© Alfaomega - RC Libros

SYNAPS PREMIUM

La consulta siguiente presenta un JOIN INTERNO (o simplemente JOIN) de las tablas REPVENTAS y OFICINAS para obtener los vendedores, sus oficinas de representación y las ciudades en que trabajan:

```
  1  SELECT name, ciudad, oficina_rep
  2  FROM repventas, oficinas
  3* WHERE oficina_rep=oficina

NAME            CIUDAD          OFICINA_REP
--------------- --------------- -----------
Dan Roberts     Chicago                  12
Sue Smith       Los Ángeles              21
Paul Cruz       Chicago                  12
Bob Smith       Chicago                  12
Bill Adams      Atlanta                  13
Sam Clark       New York                 11
Nancy Angelli   Denver                   22
Larry Fitch     Los Ángeles              21
Mary Jones      New York                 11
```

Se observa que el registro relativo a *Tom Snyder* no aparecerá en el resultado del JOIN porque no se corresponde con ninguno de los números de oficina de la tabla OFICINAS al tener un valor NULL en la tabla REP_VENTAS. Pero este problema lo solventaremos a través del JOIN EXTERNO IZQUIERDO que se presenta a continuación:

```
SQL> SELECT name, ciudad, oficina_rep
  2  FROM repventas LEFT JOIN oficinas
  3  ON oficina_rep = oficina;

NAME            CIUDAD          OFICINA_REP
--------------- --------------- -----------
Mary Jones      New York                 11
Sam Clark       New York                 11
Bob Smith       Chicago                  12
Paul Cruz       Chicago                  12
Dan Roberts     Chicago                  12
Bill Adams      Atlanta                  13
Larry Fitch     Los Ángeles              21
Sue Smith       Los Ángeles              21
Nancy Angelli   Denver                   22
Tom Snyder
```

El JOIN interno (o JOIN) también podría realizarse mediante la sintaxis que se presenta a continuación:

```
SQL> SELECT name, ciudad, oficina_rep
  2  FROM repventas INNER JOIN oficinas
  3  ON oficina_rep = oficina;

NAME            CIUDAD          OFICINA_REP
--------------- --------------- -----------
Dan Roberts     Chicago                  12
Sue Smith       Los Ángeles              21
Paul Cruz       Chicago                  12
Bob Smith       Chicago                  12
Bill Adams      Atlanta                  13
Sam Clark       New York                 11
Nancy Angelli   Denver                   22
Larry Fitch     Los Ángeles              21
Mary Jones      New York                 11
```

En los párrafos siguientes se formaliza la sintaxis de los JOINS internos y externos.

INNER JOIN

Combina registros de dos tablas siempre que existan valores coincidentes en un campo común. Estamos ante el JOIN normal utilizado hasta ahora que admite la sintaxis alternativa que se indica a continuación en la cláusula FROM.

SYNAPS PREMIUM

Sintaxis

FROM *tabla1* INNER JOIN *tabla2* ON *tabla1.campo1 operadorComparación tabla2.campo2*

La operación INNER JOIN consta de estos apartados:

Apartado	Descripción
tabla1, tabla2	El nombre de las tablas cuyos registros se van a combinar.
campo1, campo2	Nombres de los campos que se combinan. Si no son numéricos, los campos deberán ser del mismo tipo de datos y contener la misma clase de datos, pero no es necesario que tengan el mismo nombre.
operadorComparación	Cualquier operador de comparación relacional: "=," "<," ">," "<=," ">=," o "<>."

Observaciones

Puede utilizar una operación INNER JOIN en cualquier cláusula FROM. Es el tipo de combinación más común. Las combinaciones internas combinan los registros de dos tablas siempre que tengan valores coincidentes en un campo común a ambas tablas. Puede utilizar INNER JOIN con las tablas REPVENTAS y OFICINAS para seleccionar todos los vendedores de cada oficina. En cambio, para seleccionar todas las oficinas (aunque alguna no tenga empleados asignados) o todos los vendedores (aunque alguno no esté asignado a ninguna oficina), puede utilizar una operación LEFT JOIN o RIGHT JOIN para crear una combinación externa. Puede combinar dos campos numéricos cualesquiera de tipos similares.

El siguiente ejemplo muestra cómo podría combinar las tablas REPVENTAS y OFICINAS por los campos OFICINA_REP y OFICINA para mostrar el nombre y el título del vendedor y el número de oficina, sus ventas objetivo y su ciudad. No olvidemos que el campo OFICINA_REP de la tabla REPVENTAS es clave externa para el campo OFICINA de la tabla OFICINAS.

```
SQL> SELECT name, titulo,  objetivo, oficina, ciudad
  2  FROM repventas INNER JOIN oficinas
  3  ON oficina_rep=oficina;

NAME            TITULO       OBJETIVO    OFICINA CIUDAD
--------------- ---------- ---------- ---------- ---------------
Dan Roberts     Rep Ventas     800000         12 Chicago
Sue Smith       Rep Ventas     725000         21 Los Ángeles
Paul Cruz       Rep Ventas     800000         12 Chicago
Bob Smith       Dir Ventas     800000         12 Chicago
Bill Adams      Rep Ventas     350000         13 Atlanta
Sam Clark       VP Ventas      575000         11 New York
Nancy Angelli   Rep Ventas     300000         22 Denver
Larry Fitch     Dir Ventas     725000         21 Los Ángeles
Mary Jones      Rep Ventas     575000         11 New York

9 filas seleccionadas.

SQL>
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

En el ejemplo anterior, OFICINA_REP es un campo de la condición del JOIN, pero no se incluye en la salida de la consulta porque no está incluido en la instrucción SELECT. Para incluir los campos de la condición del JOIN en la salida, incluya el nombre de campo en la instrucción SELECT.

También puede vincular varias cláusulas ON en una instrucción JOIN, mediante la sintaxis siguiente:

SELECT *campos*
FROM *tabla1* INNER JOIN *tabla2*
ON *tabla1.campo1 operadorComparación tabla2.campo1* AND
ON *tabla1.campo2 operadorComparación tabla2.campo2*) OR
ON *tabla1.campo3 operadorComparación tabla2.campo3*)];

También puede anidar instrucciones JOIN con la sintaxis siguiente:

SELECT *campos*
FROM *tabla1* INNER JOIN
(*tabla2* INNER JOIN [( ]*tabla3*
[INNER JOIN [( ]*tablax* [INNER JOIN ...)]
ON *tabla3.campo3 operadorComparación tablax.campox*)]
ON *tabla2.campo2 operadorComparación tabla3.campo3*)
ON *tabla1.campo1 operadorComparación tabla2.campo2*;

Una operación LEFT JOIN o RIGHT JOIN se puede anidar dentro de una operación INNER JOIN, pero una operación INNER JOIN no se puede anidar dentro de LEFT JOIN o RIGHT JOIN.

LEFT JOIN, RIGHT JOIN

Combinan registros de la tabla de origen cuando se utiliza la definición del JOIN en cualquier cláusula FROM. Admiten la sintaxis que se indica a continuación en la cláusula FROM.

Sintaxis

FROM *tabla1* [LEFT | RIGHT] JOIN *tabla2*
ON *tabla1.campo1 operadorComparación tabla2.campo2*

Las operaciones LEFT JOIN y RIGHT JOIN constan de estos apartados:

© Alfaomega - RC Libros

SYNAPS PREMIUM

Apartado	Descripción
tabla1, tabla2	El nombre de las tablas cuyos registros se van a combinar.
campo1, campo2	Nombres de los campos que se combinan. Los campos deben ser del mismo tipo de datos y contener la misma clase de datos, pero no es necesario que tengan el mismo nombre.
operadorComparación	Cualquier operador de comparación relacional: "=," "<," ">," "<=," ">=," o "<>."

Observaciones

Utilice una operación LEFT JOIN para crear una combinación (JOIN) externa izquierda. Las combinaciones externas izquierda pueden incluir todos los registros de la primera (parte izquierda) de dos tablas, aunque no haya valores coincidentes para los registros de la segunda tabla (parte derecha). Utilice una operación RIGHT JOIN para crear una combinación externa derecha. Las combinaciones externas izquierda pueden incluir todos los registros de la segunda (parte derecha) de dos tablas, aunque no haya valores coincidentes para los registros de la primera tabla (parte izquierda).

Por ejemplo, podría utilizar LEFT JOIN con las tablas OFICINAS (parte izquierda) y VENDEDORES (parte derecha) para seleccionar todas las oficinas, incluidas aquellas que no tengan vendedores asignados a ellas. Para seleccionar todos los vendedores, incluidos aquellos que no estén asignados a ninguna oficina, podría usar RIGHT JOIN.

El siguiente ejemplo muestra cómo podría combinar las tablas REPVENTAS y OFICINAS por los campos OFICINA_REP y OFICINA para mostrar el nombre y el título del vendedor y el número de oficina, sus ventas objetivo y su ciudad:

```
SQL>  SELECT name, titulo,  objetivo, oficina, ciudad
  2   FROM repventas LEFT JOIN oficinas
  3   ON oficina_rep=oficina;

NAME            TITULO       OBJETIVO    OFICINA CIUDAD
--------------- ---------- ---------- ---------- --------------
Mary Jones      Rep Ventas     575000         11 New York
Sam Clark       VP Ventas      575000         11 New York
Bob Smith       Dir Ventas     800000         12 Chicago
Paul Cruz       Rep Ventas     800000         12 Chicago
Dan Roberts     Rep Ventas     800000         12 Chicago
Bill Adams      Rep Ventas     350000         13 Atlanta
Larry Fitch     Dir Ventas     725000         21 Los Ángeles
Sue Smith       Rep Ventas     725000         21 Los Ángeles
Nancy Angelli   Rep Ventas     300000         22 Denver
Tom Snyder      Rep ventas
```

Para crear una consulta que incluya solo los registros en los cuales los datos de los campos combinados son los mismos, utilice una operación INNER JOIN.

© Alfaomega - RC Libros

SYNAPS PREMIUM

El ejemplo siguiente es similar al anterior, pero utilizando una operación RIGHT JOIN:

```
SQL> SELECT name, titulo,  objetivo, oficina, ciudad
  2  FROM repventas RIGHT JOIN oficinas
  3  ON oficina_rep=oficina;

NAME            TITULO       OBJETIVO    OFICINA CIUDAD
--------------- ---------- ---------- ---------- ---------------
Dan Roberts     Rep Ventas     800000         12 Chicago
Sue Smith       Rep Ventas     725000         21 Los Ángeles
Paul Cruz       Rep Ventas     800000         12 Chicago
Bob Smith       Dir Ventas     800000         12 Chicago
Bill Adams      Rep Ventas     350000         13 Atlanta
Sam Clark       VP Ventas      575000         11 New York
Nancy Angelli   Rep Ventas     300000         22 Denver
Larry Fitch     Dir Ventas     725000         21 Los Ángeles
Mary Jones      Rep Ventas     575000         11 New York
```

Se tendrán en cuenta las siguientes reglas:

- Una operación LEFT JOIN o RIGHT JOIN se puede anidar en una operación INNER JOIN, pero INNER JOIN no se puede anidar en LEFT JOIN o RIGHT JOIN. Vea la explicación del anidamiento en el apartado dedicado a INNER JOIN para ver cómo se anidan combinaciones dentro de otras combinaciones.
- Puede vincular múltiples cláusulas ON. Vea la explicación de la vinculación de cláusulas en el apartado dedicado a INNER JOIN para ver cómo se hace.

A continuación, se presenta un ejemplo de consulta que lista el nombre de la empresa cliente y las características de los pedidos para el número de cliente 2103. Se presentan solo las filas para las que hay una correspondencia entre el número de la empresa cliente (NUM_CLIE) y el cliente asignado al pedido (CLIENTE).

```
SQL> SELECT empresa, num_pedido, importe
  2  FROM clientes INNER JOIN pedidos
  3  ON num_clie=clie AND num_clie=2103;

EMPRESA              NUM_PEDIDO    IMPORTE
-------------------- ---------- ----------
Acme Mfg.                112963       3276
Acme Mfg.                112983        702
Acme Mfg.                112987      27500
Acme Mfg.                113027       4104

SQL>
```

Si en la consulta anterior queremos que aparezcan todas las empresas cliente, sin exigir una correspondencia entre el número de la empresa cliente (NUM_CLIE) y el cliente asignado al pedido (CLIENTE), podemos utilizar el siguiente JOIN IZQUIERDO externo:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL>  SELECT empresa, num_pedido, importe
  2   FROM clientes LEFT JOIN pedidos
  3   ON num_clie=clie AND num_clie=2103;

EMPRESA               NUM_PEDIDO    IMPORTE
-------------------- ---------- ----------
Jones Mfg.
First. Corp.
Acme Mfg.                 112963       3276
Acme Mfg.                 112983        702
Acme Mfg.                 112987      27500
Acme Mfg.                 113027       4104
AAA Investments
Fred Lewis Corp.
Ace International
Holm Landis
Chen Associates
JCP Inc.
Zetacorp
Ian Schmidt
Orion Corp.
Smithson Corp.
J.P. Sinclair
Midwest Systems
Solomon Inc.
Rico Enterprises
QMA Assoc.
Three-Way Lines
Carter Sons
Peter Brothers
```

Si en la consulta anterior queremos que aparezcan todos los números de pedido e importes, sin exigir una correspondencia entre el número de la empresa cliente (NUM_CLIE) y el cliente asignado al pedido (CLIENTE), podremos utilizar el siguiente JOIN DERECHO externo:

```
SQL> SELECT empresa, num_pedido, importe
  2  FROM clientes RIGHT JOIN pedidos
  3  ON num_clie=clie AND num_clie=2103;

EMPRESA               NUM_PEDIDO    IMPORTE
-------------------- ---------- ----------
Acme Mfg.                 113027       4104
Acme Mfg.                 112987      27500
Acme Mfg.                 112983        702
Acme Mfg.                 112963       3276
                          113042      22500
                          113048       3750
                          112968       3978
                          113058       1480
                          113055        150
                          113003       5625
                          112961      31500
                          113069      31350
                          113062       2430
                          112997        652
                          113024       7100
                          112979      15000
                          113057        600
                          113012       3745
                          112975       2100
                          113065       2130
                          112993       1896
                          113051       1420
                          113049       2130
                          113013        652
                          112992        760
                          113045      45000
                          113007       2925
                          113034        632
                          110036      22500
                          112989       1458
```

Casos especiales en consultas multitabla

En el trabajo con consultas multitabla pueden aparecer problemas de **nombres de campos (columnas) duplicados en las distintas tablas**, que pueden resolverse utilizando un calificador de tabla consistente en anteponer al nombre del campo el

© Alfaomega - RC Libros

SYNAPS PREMIUM

de la tabla a que pertenece separados por un punto. En el ejemplo siguiente se muestran el nombre y las ventas de los vendedores, así como las ventas y la ciudad de cada oficina de representación de los vendedores:

```
SQL> SELECT name, repventas.ventas, oficinas.ventas, ciudad
  2  FROM repventas, oficinas
  3  WHERE oficina_rep=oficina;

NAME                 VENTAS     VENTAS CIUDAD
-------------- ---------- ---------- ---------------
Dan Roberts          305673     735042 Chicago
Sue Smith            474050     835915 Los Ángeles
Paul Cruz            286775     735042 Chicago
Bob Smith            142594     735042 Chicago
Bill Adams           367911     367911 Atlanta
Sam Clark            299912     692637 New York
Nancy Angelli        186042     186042 Denver
Larry Fitch          361865     835915 Los Ángeles
Mary Jones           392725     692637 New York
```

También es posible en consultas multitabla realizar la **selección simultánea de todos los campos de las tablas involucradas utilizando el símbolo *.** En el ejemplo siguiente se listan todos los vendedores y todas las oficinas en que trabajan:

```
SQL> set linesize 180
SQL> SELECT *
  2  FROM repventas, oficinas
  3  WHERE oficina_rep=oficina;

  NUM_EMPL NAME                 EDAD OFICINA_REP TITULO     CONTRATO   DIRECTOR      CUOTA     VENTAS    OFICINA CIUD
---------- -------------- ---------- ----------- ---------- -------- ---------- ---------- --------
       101 Dan Roberts            45          12 Rep Ventas 20/10/86        104     300000     305673         12 Chi
       102 Sue Smith              48          21 Rep Ventas 12/12/86        108     350000     474050         21 Los
       103 Paul Cruz              29          12 Rep Ventas 01/03/87        104     275000     286775         12 Chic
       104 Bob Smith              33          12 Dir Ventas 19/05/87        106     200000     142594         12 Chic
       105 Bill Adams             37          13 Rep Ventas 12/02/88        104     350000     367911         13 Atla
       106 Sam Clark              52          11 VP Ventas  14/06/88                275000     299912         11 New
       107 Nancy Angelli          49          22 Rep Ventas 14/11/88        108     300000     186042         22 De
       108 Larry Fitch            62          21 Dir Ventas 12/10/89        106     350000     361865         21 Los
       109 Mary Jones             31          11 Rep Ventas 12/10/99        106     300000     392725         11 New
```

También pueden elegirse todos los campos solo de una de las tablas implicadas en la consulta. En el ejemplo siguiente se informa de todos los vendedores y los lugares en que trabajan:

```
SQL> SELECT repventas.*, ciudad, region
  2  FROM repventas, oficinas
  3  WHERE oficina_rep=oficina;

  NUM_EMPL NAME                 EDAD OFICINA_REP TITULO     CONTRATO   DIRECTOR      CUOTA     VENTAS CIUDAD
---------- -------------- ---------- ----------- ---------- -------- ---------- ---------- --------
       101 Dan Roberts            45          12 Rep Ventas 20/10/86        104     300000     305673 Chicago
       102 Sue Smith              48          21 Rep Ventas 12/12/86        108     350000     474050 Los Ángeles
       103 Paul Cruz              29          12 Rep Ventas 01/03/87        104     275000     286775 Chicago
       104 Bob Smith              33          12 Dir Ventas 19/05/87        106     200000     142594 Chicago
       105 Bill Adams             37          13 Rep Ventas 12/02/88        104     350000     367911 Atlanta
       106 Sam Clark              52          11 VP Ventas  14/06/88                275000     299912 New York
       107 Nancy Angelli          49          22 Rep Ventas 14/11/88        108     300000     186042 Denver
       108 Larry Fitch            62          21 Dir Ventas 12/10/89        106     350000     361865 Los Ángeles
       109 Mary Jones             31          11 Rep Ventas 12/10/99        106     300000     392725 New York
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

También es posible que una consulta multitabla se base en una relación de una tabla consigo misma (**autocomposición**). Para componer una tabla consigo misma, SQL utiliza un alias temporal para evitar confusiones en la consulta. Por ejemplo, para listar los nombres de los vendedores y sus directores se puede utilizar el alias EMPS para la tabla REPVENTAS y posteriormente el alias DIRS para la misma tabla, consiguiéndose la consulta que se especifica a continuación:

```
SQL>  SELECT emps.name, dirs.name
  2   FROM repventas emps, repventas dirs
  3   WHERE emps.director=dirs.num_empl;

NAME            NAME
--------------- ---------------
Dan Roberts     Bob Smith
Sue Smith       Larry Fitch
Paul Cruz       Bob Smith
Bob Smith       Sam Clark
Bill Adams      Bob Smith
Nancy Angelli   Larry Fitch
Larry Fitch     Sam Clark
Mary Jones      Sam Clark
Tom Snyder      Dan Roberts
```

Como es natural no es necesario utilizar alias las dos veces que se usa la misma tabla.

Para diferenciar bastaría utilizar el alias solo una vez tal y como se ve a continuación:

```
SQL> SELECT repventas.name, dirs.name
  2  FROM repventas, repventas dirs
  3  WHERE repventas.director=dirs.num_empl;

NAME            NAME
--------------- ---------------
Dan Roberts     Bob Smith
Sue Smith       Larry Fitch
Paul Cruz       Bob Smith
Bob Smith       Sam Clark
Bill Adams      Bob Smith
Nancy Angelli   Larry Fitch
Larry Fitch     Sam Clark
Mary Jones      Sam Clark
Tom Snyder      Dan Roberts
```

Como ejemplo adicional consideramos la consulta que ofrece los vendedores con una cuota superior a la de su director:

```
SQL> SELECT repventas.name, repventas.cuota, dirs.cuota
  2  FROM repventas, repventas dirs
  3  WHERE repventas.director=dirs.num_empl
  4  AND repventas.cuota>dirs.cuota;

NAME                 CUOTA      CUOTA
--------------- ---------- ----------
Dan Roberts          300000     200000
Paul Cruz            275000     200000
Bill Adams           350000     200000
Larry Fitch          350000     275000
Mary Jones           300000     275000

SQL>
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

En la consulta siguiente se comparan los sueldos de cada pareja de miembros del personal y siempre que el sueldo del primero sea mayor que el del segundo en más de 1.000, se visualiza el nombre y sueldos de ambos seguido de la diferencia de los sueldos:

```
SQL> SELECT S1.enombre, S1.esueldo, S2.enombre, S2.esueldo, S1.esueldo-S2.esueldo
  2  FROM personal S1, personal S2
  3  WHERE (S1.ESUELDO-S2.ESUELDO) > 100;

ENOMBRE           ESUELDO ENOMBRE           ESUELDO S1.ESUELDO-S2.ESUELDO
--------------- ---------- --------------- ---------- ---------------------
DICK NIX             25001 LUCAS                  53                 24948
HANK KISS            25000 LUCAS                  53                 24947
EUCLIDES              1000 LUCAS                  53                   947
ARQUIMEDES             200 LUCAS                  53                   147
DAVINCI                500 LUCAS                  53                   447
DICK NIX             25001 MARCOS                 52                 24949
HANK KISS            25000 MARCOS                 52                 24948
EUCLIDES              1000 MARCOS                 52                   948
ARQUIMEDES             200 MARCOS                 52                   148
DAVINCI                500 MARCOS                 52                   448
DICK NIX             25001 MATEO                  51                 24950
HANK KISS            25000 MATEO                  51                 24949
EUCLIDES              1000 MATEO                  51                   949
ARQUIMEDES             200 MATEO                  51                   149
DAVINCI                500 MATEO                  51                   449
DICK NIX             25001 JUAN                   54                 24947
HANK KISS            25000 JUAN                   54                 24946
EUCLIDES              1000 JUAN                   54                   946
ARQUIMEDES             200 JUAN                   54                   146
DAVINCI                500 JUAN                   54                   446
DICK NIX             25001 EUCLIDES             1000                 24001
HANK KISS            25000 EUCLIDES             1000                 24000
DICK NIX             25001 ARQUIMEDES            200                 24801
HANK KISS            25000 ARQUIMEDES            200                 24800
EUCLIDES              1000 ARQUIMEDES            200                   800
DAVINCI                500 ARQUIMEDES            200                   300
DICK NIX             25001 DAVINCI               500                 24501
HANK KISS            25000 DAVINCI               500                 24500
EUCLIDES              1000 DAVINCI               500                   500
```

CONSULTAS RESUMEN

SQL permite realizar consultas que permiten resumir los datos a través de funciones de columna y mediante cláusulas GROUP BY y HAVING de la sentencia SELECT. Las funciones de columna posibles en Oracle SQL ya fueron estudiadas en un capítulo anterior. Estas funciones aceptan una columna completa de datos como argumentos y producen como salida un único dato que resume la columna. De ahí que las consultas que contemplan funciones de este tipo se denominen consultas resumen o consultas sumarias.

Funciones de agregado en la lista de selección

Las funciones de agregado (SUM, AVG, COUNT, COUNT(*), MAX, MIN, etc.) generan valores de resumen en los conjuntos de resultados de las consultas. Una función de agregado procesa todos los valores seleccionados en una única columna para generar un único resultado. Las funciones de agregado se pueden aplicar a todas las filas de una tabla, a un subconjunto de la tabla especificado por una cláusula WHERE o a uno o varios grupos de filas de la tabla. Cuando se aplica una función de agregado, se genera un valor individual por cada conjunto de filas.

© Alfaomega - RC Libros

SYNAPS PREMIUM

La palabra clave opcional DISTINCT se puede usar con SUM, AVG y COUNT para eliminar los valores duplicados antes de que se establezca la función de agregado (el valor predeterminado es ALL). SUM y AVG se pueden usar solo con columnas numéricas, como, por ejemplo, las de los tipos de datos int, smallint, decimal, numeric, float, real y money. MIN y MAX no se pueden usar con tipos de datos bit. Con estas excepciones, las funciones de agregado se pueden usar con cualquier tipo de columna.

En el ejemplo siguiente se obtienen los valores medios, máximos y mínimos para todos los cursos de Informática y Ciencias de la Información cuya tarifa es no nula:

```
SQL> SELECT AVG(ctarifa), MAX(ctarifa), MIN(ctarifa)
  2  FROM curso
  3  WHERE cdept='CIS' AND ctarifa <> 0;

AVG(CTARIFA) MAX(CTARIFA) MIN(CTARIFA)
------------ ------------ ------------
       187,5          500           50

SQL>
```

En el siguiente ejemplo se obtiene el nombre del primer curso que aparece en la secuencia alfabética:

```
SQL> SELECT MIN(cnombre)
  2  FROM curso;

MIN(CNOMBRE)
----------------------
ARQUITECT. COMPUTADOR

SQL>
```

En el ejemplo siguiente se obtienen la cuota promedio, las ventas promedio y el rendimiento de cuota promedio de los vendedores:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT AVG(cuota) "CUOTA PROMEDIO", AVG(ventas) "VENTAS PROMEDIO", AVG(100*(ventas/cuota))
  2  FROM repventas;

CUOTA PROMEDIO VENTAS PROMEDIO AVG(100*(VENTAS/CUOTA))
-------------- --------------- ----------------------
        300000        289353,2                102,60015

SQL>
```

En el ejemplo siguiente se calcula el mejor rendimiento de ventas de todos los vendedores:

```
SQL> SELECT MAX(100*(ventas/cuota)) "RENDIMIENTO ÓPTIMO DE VENTAS"
  2  FROM repventas;

RENDIMIENTO ÓPTIMO DE VENTAS
----------------------------
                  135,442857

SQL>
```

En el ejemplo siguiente se calculan las cuotas y ventas totales para todos los vendedores, así como sus cuotas máxima y mínima:

```
SQL> SELECT SUM (cuota) "CUOTA TOTAL", SUM(ventas) "VENTAS TOTALES", MIN(cuota), MAX(cuota)
  2  FROM repventas;

CUOTA TOTAL VENTAS TOTALES MIN(CUOTA) MAX(CUOTA)
----------- -------------- ---------- ----------
    2700000        2893532     200000     350000

SQL>
```

En el ejemplo siguiente se obtienen los importes total y medio de los pedidos aceptados por Bill Adams:

```
SQL> SELECT SUM(importe) "IMPORTE TOTAL", AVG(importe) "IMPORTE MEDIO"
  2  FROM pedidos, repventas
  3  WHERE name='Bill Adams' AND rep=num_empl;

IMPORTE TOTAL IMPORTE MEDIO
------------- -------------
        39327        7865,4

SQL>
```

El tipo de resultado que devuelve una función de agregado puede tener una precisión mayor que las entradas, de modo que es lo suficientemente grande como

SYNAPS PREMIUM

para contener el valor del resultado de agregado. Por ejemplo, las funciones SUM o AVG devuelven un valor *int* cuando el tipo de datos de entrada es *smallint*.

Las funciones de agregado no se pueden usar en una cláusula WHERE. Sin embargo, una instrucción SELECT con funciones de agregado en su lista de selección incluye, a menudo, una cláusula WHERE que restringe las filas a las que se aplica la función de agregado.

Si una instrucción SELECT incluye una cláusula WHERE (aunque no una cláusula GROUP BY), una función de agregado producirá un valor individual para el subconjunto de filas especificado por la cláusula WHERE. Esto es cierto tanto si está operando en todas las filas de una tabla como si opera solo en un subconjunto de filas definido por una cláusula WHERE. Una función de este tipo recibe el nombre de agregado escalar.

Por ejemplo, la consulta siguiente devuelve el precio promedio de los productos del fabricante ACI:

```
SQL> SELECT AVG(precio)
  2  FROM productos
  3  WHERE id_fab='ACI';

AVG(PRECIO)
-----------
 55,0357143

SQL>
```

Función count(*)

La función COUNT( ) devuelve el número de elementos de un grupo, por ejemplo el número de valores de datos de una columna. Su sintaxis es la siguiente:

COUNT ({ [ALL | DISTINCT] expresión] | * })

El argumento ALL aplica la función de agregado a todos los valores. ALL es el valor predeterminado.

El argumento DISTINCT especifica que COUNT devuelva el número de valores únicos (distintos) no NULL de la columna.

© Alfaomega - RC Libros

SYNAPS PREMIUM

El argumento EXPRESIÓN es una expresión de cualquier tipo excepto funciones de agregado y subconsultas.

El argumento * especifica que se tienen que contar todas las filas para devolver el número total de filas de una tabla. COUNT(*) no recibe parámetros y no se puede utilizar con DISTINCT. COUNT(*) no requiere el parámetro *expresión* porque, por definición, no utiliza información acerca de ninguna columna concreta. COUNT(*) devuelve el número de filas de una tabla especificada sin eliminar las duplicadas. Cuenta todas las filas, incluidas las que contienen valores NULL.

No es posible utilizar agregados diferentes, por ejemplo AVG(DISTINCT columna), COUNT(DISTINCT columna), MAX(DISTINCT columna), MIN(DISTINCT columna) ni SUM(DISTINCT columna), cuando se utiliza CUBE o ROLLUP. De hacerlo, SQL devuelve un mensaje de error y cancela la consulta.

COUNT(*) devuelve el número de elementos de un grupo incluidos los valores NULL y duplicados.

COUNT(ALL *expresión*) evalúa *expresión* en todas las filas del grupo y devuelve el número de valores no NULL.

COUNT(DISTINCT *expresión*) evalúa *expresión* en todas las filas del grupo y devuelve el número de valores únicos no NULL.

Como ejemplos se presentan las consultas que devuelven el número de vendedores que superan su cuota y el número de pedidos con importe superior a 25.000.

```
SQL> SELECT COUNT(*)
  2  FROM repventas
  3  WHERE ventas>cuota;

  COUNT(*)
----------
         7

SQL>
```

```
SQL> SELECT COUNT(*)
  2  FROM pedidos
  3  WHERE importe>25000;

  COUNT(*)
----------
         4

SQL>
```

En la consulta siguiente se visualizan los cursos que ofrece el departamento de Filosofía y la tarifa total para esos cursos:

SYNAPS PREMIUM

```
SQL> SELECT COUNT(*), SUM(ctarifa)
  2  FROM curso
  3  WHERE cdept='PHIL';

  COUNT(*) SUM(CTARIFA)
---------- ------------
         4          350

SQL>
```

Valores null en las funciones de agregado

Para trabajar con valores NULL en funciones de agregado sobre columnas es necesario tener presente que si alguno de los valores de la columna es NULL, se ignora para calcular el valor de la función de agregado relativa a la columna.

Además, si todos los valores de la columna son NULL, las funciones de columna SUM, AVG, MIN y MAX devuelven un valor NULL y la función COUNT devuelve un valor de cero. Si la columna no tiene datos, las funciones de agregado devolverán el valor cero.

En el ejemplo siguiente se observa cómo la función COUNT(*) ignora los valores NULL de una columna:

```
SQL> SELECT count(*), count(ventas), count(cuota)
  2  FROM repventas;

  COUNT(*) COUNT(VENTAS) COUNT(CUOTA)
---------- ------------- ------------
        10            10            9

SQL>
```

En el ejemplo siguiente se visualiza para todos los departamentos el nombre del departamento y el número de todos los miembros de personal en el departamento:

```
SQL> SELECT dept, COUNT(*)
  2  FROM personal
  3  GROUP BY dept;

DEPT   COUNT(*)
---- ----------
                1
ENG             1
MATH            1
PHIL            2
THEO            4

SQL>
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

En el ejemplo siguiente se visualiza el nombre y departamento de cualquier miembro de personal que esté asignado a un departamento conocido:

```
SQL> SELECT enombre, dept
  2  FROM personal
  3  WHERE dept IS NOT NULL;

ENOMBRE          DEPT
---------------- ----
LUCAS            THEO
MARCOS           THEO
MATEO            THEO
DICK NIX         PHIL
HANK KISS        PHIL
JUAN             THEO
EUCLIDES         MATH
ARQUIMEDES       ENG
DAVINCI
```

Tratamiento de filas duplicadas con DISTINCT

La palabra clave DISTINCT es opcional con SUM, AVG y COUNT. Cuando se usa DISTINCT, los valores duplicados se eliminan antes de que se calcule la suma, el promedio o el recuento. Si se usa DISTINCT, la expresión solo deberá constar del nombre de una columna y no puede incluir una expresión aritmética.

DISTINCT solo puede ser especificada una vez en una consulta. Si aparece en el argumento de una función de columna, no podrá aparecer en ninguna otra. Si se especifica delante de la lista de selección, no podrá aparecer en ninguna función de columna. DISTINCT solo puede ser especificada por segunda vez dentro de una subconsulta.

En el ejemplo siguiente se calculan los títulos diferentes existentes para los vendedores:

```
SQL> SELECT count(DISTINCT titulo)
  2  FROM repventas;

COUNT(DISTINCTTITULO)
---------------------
                    4

SQL>
```

De no haber incluido la palabra clave DISTINCT, el resultado de la consulta sería el que se presenta a continuación:

SYNAPS PREMIUM

```
SQL> SELECT count( titulo)
  2  FROM repventas;

COUNT(TITULO)
-------------
           10

SQL>
```

En la consulta siguiente se obtiene el número de vendedores que superan sus cuotas de venta:

```
SQL> SELECT COUNT(DISTINCT oficina_rep)
  2  FROM repventas
  3  WHERE ventas>cuota;

COUNT(DISTINCTOFICINA_REP)
--------------------------
                         4

SQL>
```

La consulta siguiente devuelve el nombre y el límite de crédito del cliente número 2107:

```
SQL> SELECT empresa, limite_credito
  2  FROM clientes
  3  WHERE nun_clie = 2107;

EMPRESA              LIMITE_CREDITO
-------------------- --------------
Ace International             35000

SQL>
```

Agrupar filas con GROUP BY

La cláusula GROUP BY se usa para producir valores de agregado para cada fila del conjunto de resultados. Cuando se usan sin una cláusula GROUP BY, las funciones de agregado solo devuelven un valor de agregado para la instrucción SELECT.

En la consulta siguiente, para cada departamento que ofrece cursos se obtiene la identificación del departamento seguida del número total de créditos ofrecidos por el departamento:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT cdept, SUM(cred)
  2  FROM curso
  3  GROUP BY cdept;

CDEP  SUM(CRED)
---- ----------
CIS           18
PHIL          15
THEO          15

SQL>
```

En la consulta siguiente, para cada departamento que ofrece cursos se obtiene la identificación del departamento y el total de tarifas para los cursos ofrecidos por el departamento no considerando los cursos de seis créditos. En el ejemplo siguiente se obtiene el tamaño medio de pedido por cada vendedor:

```
SQL> SELECT cdept, SUM(ctarifa)
  2  FROM curso
  3  WHERE cred <>6
  4  GROUP BY cdept;

CDEP SUM(CTARIFA)
---- ------------
CIS            750
PHIL           350
THEO           240

SQL>
```

```
SQL> SELECT rep, AVG(importe)
  2  FROM pedidos
  3  GROUP BY rep;

       REP AVG(IMPORTE)
---------- ------------
       101         8876
       102         5694
       103         1350
       105       7865,4
       106        16479
       107   11477,3333
       108   8569,57143
       109       3552,5
       110        11566
```

En el ejemplo siguiente se muestra el número de vendedores asignados a cada oficina:

```
SQL> SELECT oficina_rep, COUNT(*)
  2  FROM repventas
  3  GROUP BY oficina_rep;

OFICINA_REP   COUNT(*)
----------- ----------
         11          2
         12          3
         13          1
         21          2
         22          1
                     1
```

A continuación de las palabras clave GROUP BY se coloca una lista de columnas, conocidas como las columnas de agrupamiento. La cláusula GROUP BY restringe las filas del conjunto de resultados; solo hay una fila por cada valor distinto de la columna o columnas de agrupamiento. Cada fila del conjunto de resultados contiene los datos de resumen relacionados con el valor específico de sus columnas de agrupamiento.

Hay restricciones en los elementos que se pueden especificar en la lista de selección cuando una instrucción SELECT contiene una cláusula GROUP BY. Los elementos permitidos en la lista de selección son:

SYNAPS PREMIUM

- Las columnas de agrupamiento.
- Expresiones que devuelven un solo valor por cada valor de las columnas de agrupamiento, como, por ejemplo, las funciones de agregado que tienen un nombre de columna como uno de sus parámetros. Estas se conocen como agregados vectoriales.

SQL permite realizar consultas agrupando resultados en base a **múltiples columnas de agrupamiento**. Por ejemplo, la consulta siguiente presenta los pedidos totales agrupados por cada cliente y por cada vendedor:

```
SQL> SELECT rep, clie, sum(importe)
  2  FROM pedidos
  3  GROUP BY rep, clie;

       REP        CLIE SUM(IMPORTE)
---------- ---------- -------------
       101        2102          3978
       101        2108           150
       101        2113         22500
       102        2106          4026
       102        2114         15000
       102        2120          3750
       103        2111          2700
       105        2103         35582
       105        2111          3745
       106        2101          1458
       106        2117         31500
       107        2109         31350
       107        2124          3082
       108        2112         47925
       108        2114          7100
       108        2118          4962
       109        2108          7105
       110        2107         23132
```

Si además de agrupar los datos, se necesita ordenarlos, es posible utilizar la cláusula ORDER BY y la cláusula GROUP BY, simultáneamente. Por ejemplo, en la consulta siguiente se obtienen los pedidos totales por cada cliente de cada vendedor ordenados por cliente y, dentro de cada cliente, ordenados por vendedor:

```
SQL> SELECT clie, rep, SUM(importe)
  2  FROM pedidos
  3  GROUP BY clie, rep
  4  ORDER BY clie, rep;

      CLIE        REP SUM(IMPORTE)
---------- ---------- ------------
      2101        106         1458
      2102        101         3978
      2103        105        35582
      2106        102         4026
      2107        110        23132
      2108        101          150
      2108        109         7105
      2109        107        31350
      2111        103         2700
      2111        105         3745
      2112        108        47925
      2113        101        22500
      2114        102        15000
      2114        108         7100
      2117        106        31500
      2118        108         4962
      2120        102         3750
      2124        107         3082
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

La cláusula GROUP BY contiene los siguientes componentes:

- Una o más expresiones sin agregados. Normalmente, suelen ser referencias a las columnas de agrupación.

- Opcionalmente, la palabra clave ALL, que especifica que se devuelvan todos los grupos generados por la cláusula GROUP BY, incluso si alguno de los grupos no tiene filas que cumplan las condiciones de búsqueda.

- CUBE o ROLLUP.

- Normalmente, la cláusula HAVING se usa junto con la cláusula GROUP BY, aunque también se puede especificar sin ella.

Puede agrupar por una expresión siempre que no incluya funciones de agregado. En una cláusula GROUP BY, debe especificar el nombre de una columna de tabla o vista, no el nombre de una columna del conjunto de resultados asignada con una cláusula AS.

Puede **usar una cláusula WHERE en una consulta que contenga una cláusula GROUP BY**. Las filas que no cumplan las condiciones de la cláusula WHERE son eliminadas antes de que se realice el agrupamiento. Por ejemplo, en la consulta siguiente se calculan los pedidos totales por cada vendedor:

```
SQL> SELECT num_empl, name, SUM(importe)
  2  FROM pedidos, repventas
  3  WHERE rep=num_empl
  4  GROUP BY num_empl, name;

  NUM_EMPL NAME            SUM(IMPORTE)
---------- --------------- ------------
       101 Dan Roberts            26628
       102 Sue Smith              22776
       103 Paul Cruz               2700
       105 Bill Adams             39327
       106 Sam Clark              32958
       107 Nancy Angelli          34432
       108 Larry Fitch            59987
       109 Mary Jones              7105
       110 Tom Snyder             23132
```

Si **la columna de agrupamiento contiene un valor NULL**, esa fila se convierte en un grupo en los resultados. Si la columna de agrupamiento contiene varios valores NULL, estos se colocarán en un grupo individual.

Si se usa ALL, los resultados de la consulta incluirán todos los grupos generados por la cláusula GROUP BY, incluso si alguno de los grupos no tiene filas que cumplan

SYNAPS PREMIUM

las condiciones. Sin ALL, una instrucción SELECT que incluye GROUP BY no muestra los grupos cuyas filas no cumplen los criterios de selección.

La cláusula **HAVING establece las condiciones de la cláusula GROUP BY** de la misma forma que WHERE interactúa con SELECT. Mientras que las condiciones de búsqueda de WHERE se aplican antes de que se produzca la operación de agrupamiento, las condiciones de búsqueda de HAVING se aplican después. La sintaxis de la cláusula HAVING es similar a la de la cláusula WHERE, con la diferencia de que HAVING puede contener funciones de agregado. Las cláusulas HAVING pueden hacer referencia a cualquiera de los elementos que aparecen en la lista de selección. Por ejemplo, en la consulta siguiente se calcula el tamaño de pedido promedio para cada vendedor cuyos pedidos totalizan más de 30.000:

```
SQL> SELECT rep, avg(importe)
  2  FROM pedidos
  3  GROUP BY rep
  4  HAVING SUM(importe) > 30000;

       REP AVG(IMPORTE)
---------- ------------
       105       7865,4
       106        16479
       107   11477,3333
       108   8569,57143

SQL> |
```

La comprensión de la secuencia correcta en la que se aplican las cláusulas WHERE, GROUP BY y HAVING ayuda a codificar consultas que sean eficientes:

- La cláusula WHERE se utiliza para filtrar las filas que resultan de las operaciones especificadas en la cláusula FROM.
- La cláusula GROUP BY se usa para agrupar el resultado de la cláusula WHERE.
- La cláusula HAVING se usa para filtrar las filas del resultado agrupado.

En la consulta siguiente se calcula, para cada oficina con dos o más personas, la cuota total y las ventas totales para todos los vendedores que trabajan en la oficina:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT ciudad, SUM(cuota), SUM(repventas.ventas)
  2  FROM oficinas, repventas
  3  WHERE oficina=oficina_rep
  4  GROUP BY ciudad
  5  HAVING COUNT(*) >= 2;

CIUDAD           SUM(CUOTA) SUM(REPVENTAS.VENTAS)
---------------- ---------- --------------------
Chicago              775000                735042
Los Ángeles          700000                835915
New York             575000                692637

SQL>
```

Es más eficiente especificar en la cláusula WHERE las condiciones de búsqueda que se pueden establecer antes o después de la operación de agrupamiento. Esto reduce el número de filas que tienen que agruparse. Como ejemplo tenemos:

```
SQL> SELECT REP, AVG(IMPORTE) "IMPORTE MEDIO"
  2  FROM pedidos
  3  WHERE importe>20000
  4  GROUP BY rep
  5  HAVING SUM(importe)>30000;

       REP IMPORTE MEDIO
---------- -------------
       106         31500
       107         31350
       108         45000

SQL>
```

Las únicas condiciones de búsqueda que se deben especificar en la cláusula HAVING son aquellas que se deben aplicar una vez que se hayan realizado las operaciones de agrupamiento. Cuando en HAVING se incluyen varias condiciones, se combinan mediante AND, OR o NOT. Como ejemplo tenemos la consulta:

```
SQL> SELECT REP, AVG(IMPORTE) "IMPORTE MEDIO"
  2  FROM pedidos
  3  GROUP BY rep
  4  HAVING SUM(importe)>30000 OR AVG(IMPORTE)>6000;

       REP IMPORTE MEDIO
---------- -------------
       101          8876
       105        7865,4
       106         16479
       107   11477,3333
       108   8569,57143
       110         11566
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

RESUMIR DATOS

Los sistemas de ayuda a la toma de decisiones suelen basarse en la generación de informes mediante resúmenes de datos. SQL proporciona dos componentes flexibles y eficaces para generar análisis de la información. En primer lugar los denominados *Servicios de transformación de datos* (DTS), que permiten extraer datos de transacciones y transformarlos en agregados de resumen en un almacén o un puesto de datos. En segundo lugar los *servicios de análisis*, que organizan los datos de un almacén de datos en cubos multidimensionales con una información de resumen calculada previamente para proporcionar respuestas rápidas a las consultas analíticas complejas.

Las aplicaciones que generan informes de resumen sencillos pueden utilizar los operadores CUBE o ROLLUP. Ambos son parte de la cláusula GROUP BY de la instrucción SELECT.

Estos operadores generan conjuntos de resultados que contienen tanto filas de detalle de cada elemento del conjunto de resultados como filas de resumen de cada grupo que muestra los totales agregados para ese grupo. La cláusula GROUP BY puede utilizarse para generar resultados que contengan agregados para cada grupo, aunque no filas de detalle.

Cuando GROUP BY se utiliza sin estos operadores, devuelve un conjunto individual de resultados con una fila por grupo que contiene los subtotales agregados del grupo. En el conjunto de resultados no hay filas de detalle.

Resumir datos con CUBE

El operador CUBE genera un cubo multidimensional como conjunto de resultados. Un cubo multidimensional es una expansión de datos que registran sucesos individuales. La expansión se basa en columnas que el usuario desea analizar. Estas columnas se llaman dimensiones. El cubo es un conjunto de resultados que contiene una tabla cruzada de todas las combinaciones posibles de dimensiones.

El operador CUBE se especifica en la cláusula GROUP BY de una instrucción SELECT. La lista de selección contiene las columnas de dimensión y las expresiones de funciones de agregado. GROUP BY especifica las columnas de dimensión entre paréntesis. El conjunto de resultados contiene todas las combinaciones posibles de los valores de las

© Alfaomega - RC Libros

SYNAPS PREMIUM

columnas de dimensiones, junto con los valores de agregado de las filas subyacentes que coinciden con esa combinación de valores de dimensión.

A continuación, se presenta un ejemplo de cubo cuyas dimensiones son los vendedores (REP) y los clientes (CLIE), calculándose los pedidos totales por cada cliente y cada vendedor para los vendedores con número superior a 105:

```
SQL> SELECT rep, clie, SUM(importe)
  2  FROM pedidos
  3  WHERE REP>105
  4  GROUP BY CUBE (rep, clie);

       REP       CLIE SUM(IMPORTE)
---------- ---------- ------------
       106       2101         1458
       106       2117        31500
       106                   32958
       107       2109        31350
       107       2124         3082
       107                   34432
       108       2112        47925
       108       2114         7100
       108       2118         4962
       108                   59987
       109       2108         7105
       109                    7105
       110       2107        23132
       110                   23132
                 2101         1458
                 2107        23132
                 2108         7105
                 2109        31350
                 2112        47925
                 2114         7100
                 2117        31500
                 2118         4962
                 2124         3082
                             157614
```

Utilizar GROUPING para distinguir valores NULL

Los valores NULL generados por la operación CUBE presentan un problema: ¿cómo se puede distinguir un valor NULL generado por la operación CUBE de un valor NULL devuelto en los datos reales? Esto se consigue con la función GROUPING. La función GROUPING devuelve 0, si el valor de la columna procede de datos reales, y 1 si el valor de la columna es un NULL generado por la operación CUBE. La instrucción SELECT se puede escribir para que utilice la función GROUPING para sustituir la cadena ALL en lugar de cualquier NULL generado.

A continuación, se presenta una consulta, basada en la instancia por defecto que se instala con Oracle 12c, que devuelve el número de empleados y su salario medio anual a través de todas las posibles combinaciones de departamento y categoría de trabajador:

SYNAPS PREMIUM

```
SQL> SELECT DECODE(GROUPING(dname), 1, 'All Departments',
  2         dname) AS dname,
  3      DECODE(GROUPING(job), 1, 'All Jobs', job) AS job,
  4      COUNT(*) "Total Empl", AVG(sal) * 12 "Average Sal"
  5      FROM emp, dept
  6      WHERE dept.deptno = emp.deptno
  7      GROUP BY CUBE (dname, job);

DNAME          JOB       Total Empl Average Sal
-------------- --------- ---------- -----------
ACCOUNTING     CLERK              1       15600
ACCOUNTING     MANAGER            1       29400
ACCOUNTING     PRESIDENT          1       60000
ACCOUNTING     All Jobs           3       35000
RESEARCH       ANALYST            2       36000
RESEARCH       CLERK              2       11400
RESEARCH       MANAGER            1       35700
RESEARCH       All Jobs           5       26100
SALES          CLERK              1       11400
SALES          MANAGER            1       34200
SALES          SALESMAN           4       16800
SALES          All Jobs           6       18800
All Departments ANALYST           2       36000
All Departments CLERK             4       12450
All Departments MANAGER           3       33100
All Departments PRESIDENT         1       60000
All Departments SALESMAN          4       16800
All Departments All Jobs         14  24878,5714
```

El operador CUBE se puede usar para generar cubos *n*-dimensionales (cubos con cualquier número de dimensiones). Las instrucciones SELECT que contiene un operador CUBE con muchas dimensiones pueden generar conjuntos de resultados grandes, porque estas instrucciones generan filas para todas las combinaciones de los valores de todas las dimensiones. Estos conjuntos grandes de resultados pueden contener demasiados datos como para que resulten fáciles de leer y entender. Una solución a este problema es colocar la instrucción SELECT en una vista A continuación, la vista se puede utilizar para consultar solo los valores de dimensión que resulten de interés.

Resumir datos con ROLLUP

El operador ROLLUP resulta útil para generar informes que contienen subtotales y totales. El operador ROLLUP genera un conjunto de resultados que es similar a los conjuntos de resultados generados por el operador CUBE. Las diferencias entre CUBE y ROLLUP son:

- CUBE genera un conjunto de resultados que muestra agregados para todas las combinaciones de valores de las columnas seleccionadas.
- ROLLUP genera un conjunto de resultados que muestra agregados para una jerarquía de valores de las columnas seleccionadas.

El último ejemplo de consulta presentado con CUBE ofrece el siguiente resultado al utilizarlo con ROLLUP:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL>  SELECT DECODE(GROUPING(dname), 1, 'All Departments',
  2         dname) AS dname,
  3       DECODE(GROUPING(job), 1, 'All Jobs', job) AS job,
  4       COUNT(*) "Total Empl", AVG(sal) * 12 "Average Sal"
  5       FROM emp, dept
  6       WHERE dept.deptno = emp.deptno
  7       GROUP BY ROLLUP (dname, job);

DNAME          JOB       Total Empl Average Sal
-------------- --------- ---------- -----------
ACCOUNTING     CLERK              1       15600
ACCOUNTING     MANAGER            1       29400
ACCOUNTING     PRESIDENT          1       60000
ACCOUNTING     All Jobs           3       35000
RESEARCH       ANALYST            2       36000
RESEARCH       CLERK              2       11400
RESEARCH       MANAGER            1       35700
RESEARCH       All Jobs           5       26100
SALES          CLERK              1       11400
SALES          MANAGER            1       34200
SALES          SALESMAN           4       16800
SALES          All Jobs           6       18800
All Departments All Jobs         14  24878,5714
```

Ejercicio 5.1. De todo el personal asignado a los actuales departamentos, seleccionar toda la información acerca del personal y de sus respectivos departamentos.

```
SQL> SELECT *
  2  FROM personal, departamento
  3  WHERE PERSONAL.dept = DEPARTAMENTO.dept;

ENOMBRE         CARGO         ESUELDO DEPT DEPT DE  DDESPACHO DCH
--------------- --------- ---------- ---- ---- -- ---------- ---
LUCAS           EVANG1            53 THEO THEO HU        200 10
MARCOS          EVANG2            52 THEO THEO HU        200 10
MATEO           EVANG3            51 THEO THEO HU        200 10
DICK NIX        LADRÓN         25001 PHIL PHIL HU        100 60
HANK KISS       BUFÓN          25000 PHIL PHIL HU        100 60
JUAN            EVANG4            54 THEO THEO HU        200 10
```

Ejercicio 5.2. Para cada miembro del personal cuyo salario anual exceda de 1.000, visualizar su nombre, sueldo, código de departamento y edificio de destino.

```
SQL> SELECT enombre, esueldo, PERSONAL.dept, dedif, ddespacho
  2  FROM personal, departamento
  3  WHERE PERSONAL.dept = DEPARTAMENTO.dept
  4  AND esueldo > 100;

ENOMBRE          ESUELDO DEPT DE  DDESPACHO
--------------- ---------- ---- -- ----------
DICK NIX             25001 PHIL HU        100
HANK KISS            25000 PHIL HU        100

SQL>
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 5.3. Para cada curso con una tarifa superior a 175, mostrar el nombre del curso, tarifa y número de facultativo del jefe responsable del curso, visualizando la salida en orden ascendente por nombre de curso.

```
SQL> SELECT cnombre, ctarifa, dchfno
  2  FROM curso, departamento
  3  WHERE cdept = dept
  4  AND ctarifa > 175
  5  ORDER BY cnombre;

CNOMBRE                    CTARIFA DCH
-------------------- ---------- ---
BASES DE DATOS RELAC.         500 80
COMUNISMO                     200 10
EXISTENCIALISMO               200 60

SQL>
```

Ejercicio 5.4. Visualizar el nombre y cargo de cada miembro del personal que trabaje en el edificio de humanidades.

```
SQL> SELECT enombre, cargo
  2  FROM personal, departamento
  3  WHERE PERSONAL.dept = DEPARTAMENTO.dept
  4  AND dedif = 'HU';

ENOMBRE         CARGO
--------------- ----------
LUCAS           EVANG1
MARCOS          EVANG2
MATEO           EVANG3
DICK NIX        LADRÓN
HANK KISS       BUFÓN
JUAN            EVANG4
```

Ejercicio 5.5. Para cada departamento que ofrece servicios de tutoría, visualizar el identificador de departamento junto con la tarifa media de los cursos que este ofrece y el sueldo medio de los miembros del personal que pueden autorizar dichos cursos. Clasificar la salida por identificador de departamento.

```
SQL> SELECT dept, AVG(ctarifa), AVG(esueldo)
  2  FROM personal, curso
  3  WHERE dept=cdept
  4  GROUP BY dept
  5  ORDER BY dept;

DEPT AVG(CTARIFA) AVG(ESUELDO)
---- ------------ ------------
PHIL         87,5      25000,5
THEO          110         52,5

SQL>
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 5.6. Para cualquier curso que tenga un miembro del personal disponible para ser tutor, visualizar el número, nombres y cargos de los miembros del personal que pueden servir de tutores para ese curso, y la ubicación de sus respectivos edificios y despachos. Clasificar la salida por nombre de los miembros del personal y número de curso. Asímismo, para cada departamento al que se hace referencia en la tabla PERSONAL formar un comité compuesto por dos miembros del personal del departamento de modo que, para cada posible pareja de miembros del personal se visualice su código de departamento seguido de los nombres de los miembros del personal. El resultado debe contener una fila por cada posible pareja de miembros del personal.

```
SQL> SELECT cno, enombre, cargo, dedif, ddespacho
  2  FROM curso, personal, departamento
  3  WHERE cdept=PERSONAL.dept
  4  AND cdept=DEPARTAMENTO.dept
  5  ORDER BY cno, enombre;

CNO ENOMBRE          CARGO      DE  DDESPACHO
--- --------------- ---------- -- ----------
P11 DICK NIX         LADRÓN     HU         100
P11 HANK KISS        BUFÓN      HU         100
P22 DICK NIX         LADRÓN     HU         100
P22 HANK KISS        BUFÓN      HU         100
P33 DICK NIX         LADRÓN     HU         100
P33 HANK KISS        BUFÓN      HU         100
P44 DICK NIX         LADRÓN     HU         100
P44 HANK KISS        BUFÓN      HU         100
T11 JUAN             EVANG4     HU         200
T11 LUCAS            EVANG1     HU         200
T11 MARCOS           EVANG2     HU         200
T11 MATEO            EVANG3     HU         200
T22 JUAN             EVANG4     HU         200
T22 LUCAS            EVANG1     HU         200
T22 MARCOS           EVANG2     HU         200
T22 MATEO            EVANG3     HU         200
T33 JUAN             EVANG4     HU         200
T33 LUCAS            EVANG1     HU         200
T33 MARCOS           EVANG2     HU         200
T33 MATEO            EVANG3     HU         200
T44 JUAN             EVANG4     HU         200
T44 LUCAS            EVANG1     HU         200
T44 MARCOS           EVANG2     HU         200
T44 MATEO            EVANG3     HU         200
```

```
SQL> SELECT PER1.dept, PER1.enombre, PER2.enombre
  2  FROM personal PER1, personal PER2
  3  WHERE PER1.dept = PER2.dept;

DEPT ENOMBRE          ENOMBRE
---- --------------- ---------------
     DAVINCI         DAVINCI
ENG  ARQUIMEDES      ARQUIMEDES
MATH EUCLIDES        EUCLIDES
PHIL DICK NIX        DICK NIX
PHIL HANK KISS       DICK NIX
PHIL DICK NIX        HANK KISS
PHIL HANK KISS       HANK KISS
THEO LUCAS           LUCAS
THEO JUAN            LUCAS
THEO MARCOS          LUCAS
THEO MATEO           LUCAS
THEO LUCAS           JUAN
THEO JUAN            JUAN
THEO MARCOS          JUAN
THEO MATEO           JUAN
THEO LUCAS           MARCOS
THEO JUAN            MARCOS
THEO MARCOS          MARCOS
THEO MATEO           MARCOS
THEO LUCAS           MATEO
THEO JUAN            MATEO
THEO MARCOS          MATEO
THEO MATEO           MATEO
```

Ejercicio 5.7. La tabla EMPLEADO contiene todos los empleados de una empresa y sus edades, pero no contiene su especialidad en el trabajo. La tabla OFICIOEMPLEADO contiene únicamente aquellos empleados que tienen alguna especialidad. Producir un informe de todos los empleados con sus edades y especialidad. Producir también un informe con todos los empleados con sus edades y especialidad, independientemente de si tienen alguna especialidad o no (JOIN EXTERNO).

La primera cuestión se resuelve con un JOIN interno de una de las dos formas siguientes:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT A.nombre, edad, oficio
  2  FROM empleado A, oficioempleado B
  3  WHERE A.nombre = B.nombre
  4  ORDER BY A.nombre;

NOMBRE                         EDAD OFICIO
------------------------- ---------- --------------------
ALONSO TELLEZ                    23 EXCELENTE
DÍAZ JIMÉNEZ                     18 EXCELENTE
DONOSO ANTÓN                     18 PRECISO
EGUÍBAR PUENTE                   43 LENTO
EZQUERRO TOMÉ                    43 BUENO
GOMEZ ORDUÑA                     41 EXCELENTE
GONZÁLEZ LÓPEZ                   23 EXCELENTE
GONZÁLEZ ORBEA                   41 PRECISO
JUÁREZ NORIEGA                   25 EXCELENTE
PANADERO JUAN                    21 PRECISO
PAVÓN LÓPEZ                      21 EXCELENTE
PÉREZ LÓPEZ                      21 RÁPIDO
TEJERINA RUIZ                    31 BUENO
VÁZQUEZ LÓPEZ                    67 LENTO
```

```
SQL> SELECT A.nombre, edad, oficio
  2  FROM empleado A INNER JOIN oficioempleado B
  3  ON A.nombre = B.nombre
  4  ORDER BY A.nombre;

NOMBRE                         EDAD OFICIO
------------------------- ---------- --------------------
ALONSO TELLEZ                    23 EXCELENTE
DÍAZ JIMÉNEZ                     18 EXCELENTE
DONOSO ANTÓN                     18 PRECISO
EGUÍBAR PUENTE                   43 LENTO
EZQUERRO TOMÉ                    43 BUENO
GOMEZ ORDUÑA                     41 EXCELENTE
GONZÁLEZ LÓPEZ                   23 EXCELENTE
GONZÁLEZ ORBEA                   41 PRECISO
JUÁREZ NORIEGA                   25 EXCELENTE
PANADERO JUAN                    21 PRECISO
PAVÓN LÓPEZ                      21 EXCELENTE
PÉREZ LÓPEZ                      21 RÁPIDO
TEJERINA RUIZ                    31 BUENO
VÁZQUEZ LÓPEZ                    67 LENTO
```

La segunda cuestión se resuelve con un JOIN izquierdo externo de las dos formas siguientes:

```
SQL> SELECT A.nombre, edad, oficio
  2  FROM empleado A LEFT JOIN oficioempleado B
  3  ON A.nombre = B.nombre
  4  ORDER BY A.nombre;

NOMBRE                         EDAD OFICIO
------------------------- ---------- --------------------
ALONSO TELLEZ                    23 EXCELENTE
ÁLVAREZ RODRÍGUEZ                23
DÍAZ JIMÉNEZ                     18 EXCELENTE
DONOSO ANTÓN                     18 PRECISO
EGUÍBAR PUENTE                   43 LENTO
EZQUERRO TOMÉ                    43 BUENO
GOMEZ ORDUÑA                     41 EXCELENTE
GONZÁLEZ LÓPEZ                   23 EXCELENTE
GONZÁLEZ ORBEA                   41 PRECISO
JUÁREZ NORIEGA                   25 EXCELENTE
PANADERO JUAN                    21 PRECISO
PAVÓN LÓPEZ                      21 EXCELENTE
PÉREZ LÓPEZ                      21 RÁPIDO
TEJERINA RUIZ                    31 BUENO
VÁZQUEZ LÓPEZ                    67 LENTO
```

```
SQL> SELECT A.nombre, edad, oficio
  2  FROM empleado A, oficioempleado B
  3  WHERE A.nombre = B.nombre (+)
  4  ORDER BY A.nombre;

NOMBRE                         EDAD OFICIO
------------------------- ---------- --------------------
ALONSO TELLEZ                    23 EXCELENTE
ÁLVAREZ RODRÍGUEZ                23
DÍAZ JIMÉNEZ                     18 EXCELENTE
DONOSO ANTÓN                     18 PRECISO
EGUÍBAR PUENTE                   43 LENTO
EZQUERRO TOMÉ                    43 BUENO
GOMEZ ORDUÑA                     41 EXCELENTE
GONZÁLEZ LÓPEZ                   23 EXCELENTE
GONZÁLEZ ORBEA                   41 PRECISO
JUÁREZ NORIEGA                   25 EXCELENTE
PANADERO JUAN                    21 PRECISO
PAVÓN LÓPEZ                      21 EXCELENTE
PÉREZ LÓPEZ                      21 RÁPIDO
TEJERINA RUIZ                    31 BUENO
VÁZQUEZ LÓPEZ                    67 LENTO
```

Ejercicio 5.8. Para cada departamento que ofrezca cursos determinar el valor medio de la tarifa de todos los cursos que tengan tres créditos ofrecidos por cada departamento. Visualizar la salida en secuencia ascendente por identificación de departamento.

```
SQL> SELECT cdept, avg(ctarifa)
  2  FROM curso
  3  WHERE cred=3
  4  GROUP BY cdept
  5  ORDER BY cdept;

CDEP AVG(CTARIFA)
---- ------------
CIS            125
PHIL   116,666667
THEO            80

SQL>
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 5.9. Para cada valor distinto de tarifa determinar el número total de créditos para los cursos que tengan ese valor de tarifa. Ordenar el resultado en secuencia descendente por tarifa.

```
SQL> SELECT ctarifa, SUM(cred)
  2  FROM curso
  3  GROUP BY ctarifa
  4  ORDER BY ctarifa DESC;

  CTARIFA  SUM(CRED)
--------- ----------
      500          3
      200          9
      150          3
      100          9
       90          3
       50          6
        0         15
```

Ejercicio 5.10. Para todos los departamentos excepto el de Teología, que tengan una tarifa media de sus cursos mayor que 100, obtener su identificación y su tarifa media.

```
SQL> SELECT cdept, AVG(ctarifa)
  2  FROM curso
  3  WHERE NOT cdept= 'THEO'
  4  GROUP BY cdept
  5  HAVING AVG(ctarifa) > 100;

CDEP AVG(CTARIFA)
---- ------------
CIS            125

SQL>
```

Ejercicio 5.11. Hallar las tarifas media, máxima y mínima por crédito dentro de cada departamento solo para aquellos grupos con tarifa mínima positiva.

```
SQL> SELECT cdept, cred, AVG(ctarifa), MAX(ctarifa), MIN(ctarifa)
  2  FROM curso
  3  GROUP BY cdept, cred
  4  HAVING MIN(ctarifa) > 0;

CDEP       CRED AVG(CTARIFA) MAX(CTARIFA) MIN(CTARIFA)
---- ---------- ------------ ------------ ------------
PHIL          3   116,666667          200           50
THEO          6          200          200          200

SQL>
```


SYNAPS PREMIUM

Ejercicio 5.12. Hallar la tarifa media para aquellos departamentos en los que dicha tarifa media sea mayor que 100 y que ofrezcan menos de seis cursos.

```
SQL> SELECT cdept, AVG(ctarifa)
  2  FROM curso
  3  GROUP BY cdept
  4  HAVING AVG(ctarifa) > 100
  5  AND COUNT(*) < 6;

CDEP AVG(CTARIFA)
---- ------------
THEO          110

SQL>
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

CAPÍTULO 6

SUBCONSULTAS Y USO AVANZADO DE FUNCIONES EN ORACLE SQL

SUBCONSULTAS

Una subconsulta es una consulta SELECT que devuelve un valor único y está anidada en las instrucciones SELECT, INSERT, UPDATE o DELETE, o dentro de otra subconsulta. Una subconsulta se puede utilizar en cualquier parte en la que se permita una expresión.

En el ejemplo siguiente se visualiza el número, nombre y departamento de los cursos con la mínima tarifa:

SYNAPS PREMIUM

```
SQL> SELECT cno, cnombre, cdept
  2  FROM curso
  3  WHERE ctarifa= (SELECT MIN(ctarifa)
  4                  FROM curso);

CNO CNOMBRE                   CDEP
--- ----------------------- ----
C33 MATEMÁTICAS DISCRETAS   CIS
C44 CIRCUITOS DIGITALES     CIS
P44 SOLIPSISMO              PHIL
T33 HEDONISMO               THEO

SQL>
```

En el ejemplo siguiente se listan las oficinas cuyo objetivo de ventas excede a la suma de las cuotas de los vendedores individuales:

```
SQL> SELECT ciudad
  2  FROM oficinas
  3  WHERE objetivo>(SELECT SUM(cuota)
  4                  FROM repventas
  5                  WHERE oficina_rep=oficina);

CIUDAD
---------------
Chicago
Los Ángeles

SQL>
```

En SQL, normalmente no hay diferencias de rendimiento entre una instrucción que incluya una subconsulta y una versión semánticamente equivalente que no lo haga. Sin embargo, en algunos casos en los que se debe comprobar la existencia de algo, una consulta normal produce mejores resultados que una subconsulta. De lo contrario, la consulta anidada debe ser procesada para cada resultado de la consulta externa con el fin de asegurar la eliminación de los duplicados. En tales casos, la utilización de consultas normales produciría los mejores resultados.

Una subconsulta anidada en la instrucción externa SELECT tiene los siguientes componentes:

- Una consulta SELECT normal, que incluye los componentes normales de la lista de selección.
- Una cláusula normal FROM que incluye uno o más nombres de tablas o vistas.
- Una cláusula opcional WHERE.
- Una cláusula opcional GROUP BY.
- Una cláusula opcional HAVING.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Por ejemplo, la consulta siguiente lista los vendedores cuyo tamaño de pedido medio para productos fabricados por ACI es superior al tamaño del pedido medio global:

```
SQL> SELECT name, AVG(importe)
  2  FROM repventas, pedidos
  3  WHERE num_empl=rep
  4  AND fab='ACI'
  5  GROUP BY name
  6  HAVING AVG(importe) > (SELECT AVG(importe)
  7                         FROM pedidos);

NAME            AVG(IMPORTE)
--------------- ------------
Sue Smith              15000
Tom Snyder             22500

SQL>
```

La cláusula SELECT de una subconsulta se incluye siempre entre paréntesis. Una subconsulta puede anidarse dentro de la cláusula WHERE o HAVING de una instrucción externa SELECT, INSERT, UPDATE o DELETE, o dentro de otra subconsulta. Se puede disponer de hasta 32 niveles de anidamiento, aunque el límite varía dependiendo de la memoria disponible y de la complejidad del resto de las expresiones de la consulta. Las consultas individuales puede que no admitan anidamientos por encima de 32 niveles. Una subconsulta puede aparecer en cualquier parte en la que se pueda usar una expresión, si devuelve un valor individual.

Si una tabla aparece solo en una subconsulta y no en la consulta externa, las columnas de esa tabla no se pueden incluir en el resultado (la lista de selección de la consulta externa).

Las instrucciones que incluyen una subconsulta normalmente lo hacen en la cláusula WHERE y tienen uno de estos formatos:

- WHERE *expresión* [NOT] IN **(***subconsulta***)**
- WHERE *expresión operador_comparación* [ANY | ALL] **(***subconsulta***)**
- WHERE [NOT] EXISTS **(***subconsulta***)**

En la consulta siguiente se presenta una consulta que visualiza el nombre, número, departamento y tarifa de los cursos que tienen la segunda tarifa más cara, sabiendo que la tarifa más cara es 500. Adicionalmente se visualizan el nombre y el salario de cualquier empleado miembro del personal cuyo sueldo sea menor o igual que la máxima tarifa del curso.

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT cno, cnombre, cdept, ctarifa
  2  FROM curso
  3  WHERE ctarifa = (SELECT MAX(ctarifa)
  4                   FROM curso
  5                   WHERE ctarifa <> 500);

CNO CNOMBRE                  CDEP    CTARIFA
--- -------------------- ---- ----------
P33 EXISTENCIALISMO          PHIL         200
T44 COMUNISMO                THEO         200

SQL>
```

```
SQL> SELECT enombre, esueldo
  2  FROM personal
  3  WHERE esueldo <= (SELECT MAX(ctarifa)
  4                    FROM curso);

ENOMBRE             ESUELDO
--------------- ----------
LUCAS                   53
MARCOS                  52
MATEO                   51
JUAN                    54
ARQUIMEDES             200
DAVINCI                500
```

En la consulta siguiente se visualiza toda la información acerca de cualquier curso de Informática y Ciencias de la Información con una tarifa menor que el sueldo medio de cualquiera asignado al departamento de Teología:

```
SQL> SELECT *
  2  FROM curso
  3  WHERE cdept='CIS'
  4  AND ctarifa < (SELECT AVG(esueldo)
  5                 FROM personal
  6                 WHERE dept='THEO');

CNO CNOMBRE               CDESCP                        CRED    CTARIFA CDEP
--- -------------------- ---------------------------- ---------- ---------- ----
C22 ESTRUCT. DE DATOS     MUY UTIL                            3         50 CIS
C33 MATEMÁTICAS DISCRETAS ABSOLUTAMENTE NECESARIO             3          0 CIS
C44 CIRCUITOS DIGITALES   AH HA!                              3          0 CIS

SQL>
```

En la consulta siguiente se listan las oficinas en las que el objetivo de ventas de la oficina excede a la suma de las cuotas de los vendedores individuales. Adicionalmente, se listan los vendedores cuyas cuotas son iguales o superiores al objetivo de la oficina de ventas de Atlanta:

```
SQL> SELECT ciudad
  2  FROM oficinas
  3  WHERE objetivo > (SELECT SUM(cuota)
  4                    FROM repventas
  5                    WHERE oficina_rep=oficina);

CIUDAD
---------------
Chicago
Los Ángeles

SQL>
```

```
SQL> SELECT name
  2  FROM repventas
  3  WHERE cuota >= (SELECT objetivo
  4                  FROM oficinas
  5                  WHERE ciudad='Atlanta');

NAME
---------------
Sue Smith
Bill Adams
Larry Fitch

SQL>
```

En algunas instrucciones SQL, la subconsulta se puede evaluar como si fuera una consulta independiente. Conceptualmente, los resultados de la subconsulta se sustituyen en la consulta externa.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Hay tres tipos básicos de subconsultas; son las que:

- Operan en listas incorporadas con IN, o en aquellas que modificó un operador de comparación mediante ANY o ALL.
- Se introducen con un operador de comparación sin modificar y deben devolver un valor individual.
- Son pruebas de existencia que se introducen con EXISTS.

Reglas de las subconsultas

Las subconsultas están sujetas a una serie de limitaciones entre las que destacan las siguientes:

- La lista de selección de una subconsulta presentada con un operador de comparación solo puede incluir una expresión o nombre de columna (excepto EXISTS e IN, que operan en SELECT * o en una lista, respectivamente).
- Si la cláusula WHERE de una consulta externa incluye un nombre de columna, deberá ser compatible en cuanto a combinación con la columna nombrada en la lista de selección de la subconsulta.
- Puesto que deben devolver un valor individual, las subconsultas presentadas por un operador de comparación sin modificar (uno no seguido de la palabra clave ANY o ALL) no pueden incluir cláusulas GROUP BY y HAVING.
- La palabra clave DISTINCT no se puede usar con subconsultas que incluyan GROUP BY.
- No se pueden especificar las cláusulas COMPUTE e INTO.
- Solo se puede especificar ORDER BY si se especifica también TOP.
- Una vista creada con una subconsulta no se puede actualizar.
- La lista de selección de una subconsulta presentada con EXISTS consta, por convención, de un asterisco (*) en lugar de un nombre de columna individual. Las reglas de una subconsulta presentada con EXISTS son idénticas a las de una lista de selección estándar porque una subconsulta presentada con EXISTS constituye una prueba de existencia y devuelve TRUE o FALSE en lugar de datos.
- Los nombres de columna de una instrucción son calificados implícitamente por la tabla a la que se hace referencia en la cláusula FROM del mismo nivel,

© Alfaomega - RC Libros

SYNAPS PREMIUM

pero nunca está de más indicar explícitamente el nombre de la tabla, y siempre se puede anular las presunciones implícitas acerca de los nombres de tabla con calificaciones explícitas.

Subconsultas con IN

El resultado de una subconsulta presentada con IN (o con NOT IN) es una lista de cero o más valores. Una vez que la subconsulta devuelve los resultados, la consulta externa hace uso de ellos.

La consulta siguiente visualiza el nombre y el departamento de cualquier miembro de la facultad que no esté impartiendo clases durante este semestre. Adicionalmente se visualiza el número de curso y nombre en donde se halle registrado el estudiante 800:

```
SQL> SELECT fnombre, fdept
  2  FROM claustro
  3  WHERE fno NOT IN (SELECT cinstrfno
  4                    FROM clase);

FNOMBRE          FDEP
---------------- ----
JULIA MARTIN     PHIL

SQL>
```

```
SQL> SELECT cno, cnombre
  2  FROM curso
  3  WHERE cno IN (SELECT cno
  4                FROM matricula
  5                WHERE SNO = '800');

CNO CNOMBRE
--- ----------------------
C11 INTROD. A LAS CC.
P33 EXISTENCIALISMO
T11 ESCOLASTICISMO

SQL>
```

La consulta siguiente lista los vendedores que trabajan en oficinas que superan sus objetivos. Adicionalmente se listan los vendedores que no trabajan en oficinas dirigidas por el empleado 108:

```
SQL> SELECT name
  2  FROM repventas
  3  WHERE oficina_rep IN (SELECT OFICINA
  4                        FROM oficinas
  5                        WHERE ventas > objetivo);

NAME
---------------
Sue Smith
Bill Adams
Sam Clark
Larry Fitch
Mary Jones

SQL>
```

```
SQL> SELECT name
  2  FROM repventas
  3  WHERE oficina_rep NOT IN(SELECT OFICINA
  4                           FROM oficinas
  5                           WHERE dir=108);

NAME
---------------
Dan Roberts
Paul Cruz
Bob Smith
Bill Adams
Sam Clark
Mary Jones
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Subconsultas con operadores de comparación

Las subconsultas se pueden presentar con uno de los operadores de comparación (=, < >, >, > =, <, ! >, ! < o < =). Una subconsulta presentada con un operador de comparación sin modificar (un operador de comparación no seguido de ANY o ALL) debe devolver un valor individual en lugar de una lista de valores, como las subconsultas presentadas con IN. Si tal subconsulta devuelve más de un valor, SQL mostrará un mensaje de error.

La consulta siguiente lista todos los productos del fabricante ACI para los cuales las existencias superan a las existencias del producto 4004:

```
SQL> SELECT descripcion, existencias
  2  FROM productos
  3  WHERE id_fab='ACI'
  4  AND existencias > (SELECT existencias
  5                     FROM productos
  6                     WHERE id_fab='ACI'
  7                     AND id_producto='41004');

DESCRIPCION          EXISTENCIAS
-------------------- -----------
Artículo Tipo 1              277
Artículo Tipo 2              167
Artículo Tipo 3              207

SQL>
```

Las subconsultas presentadas con operadores de comparación sin modificar incluyen, a menudo, funciones agregadas, puesto que estas devuelven un valor individual. Por ejemplo, la consulta siguiente busca los productos con un precio superior al precio actual mínimo:

```
SQL> SELECT DISTINCT producto
  2  FROM pedidos
  3  WHERE importe > (SELECT MIN(importe)
  4                   FROM pedidos);

PRODU
-----
112
114
2A44G
2A44L
2A44R
2A45C
41002
41003
41004
4100X
4100Y
4100Z
773C
775C
779C
XK47
```


SYNAPS PREMIUM

Puesto que las subconsultas presentadas con operadores de comparación sin modificar deben devolver un valor individual, no pueden incluir cláusulas GROUP BY o HAVING a menos que sepa que la propia cláusula GROUP BY o HAVING devuelve un valor individual. Por ejemplo, la consulta siguiente busca los productos con un precio superior al precio actual mínimo del cliente 2111:

```
SQL> SELECT DISTINCT producto
  2  FROM pedidos
  3  WHERE importe > (SELECT MIN(importe)
  4                   FROM pedidos
  5                   GROUP BY clie
  6                   HAVING clie='2111');

PRODU
-----
112
114
2A44G
2A44L
2A44R
2A45C
41002
41003
41004
4100Y
4100Z
773C
775C
779C
XK47
```

Subconsultas con EXISTS

Cuando se presenta una subconsulta con la palabra clave EXISTS funciona como una prueba de existencia. La cláusula WHERE de la consulta externa comprueba la existencia de las filas devueltas por la subconsulta. La subconsulta, en realidad, no produce ningún dato, devuelve el valor TRUE o FALSE.

Una subconsulta presentada con EXISTS tiene la siguiente sintaxis:

WHERE [NOT] EXISTS (subconsulta)

La consulta siguiente visualiza el nombre y el identificador de departamento de cualquier miembro del claustro asignado a un departamento que ofrezca un curso de 6 créditos. Adicionalmente se visualizan el nombre y el identificador de departamento de cualquier miembro del claustro asignado a un departamento que no ofrezca un curso de 6 créditos:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT fnombre, fdept
  2  FROM claustro
  3  WHERE EXISTS (SELECT *
  4                FROM curso
  5                WHERE cdept=CLAUSTRO.fdept
  6                AND cred=6);

FNOMBRE         FDEP
--------------- ----
KATHY PEPE      PHIL
JESSIE MARTIN   THEO
JULIA MARTIN    PHIL
LISA BOBAK      THEO

SQL>
```

```
SQL> SELECT fnombre, fdept
  2  FROM claustro
  3  WHERE NOT EXISTS (SELECT *
  4                FROM curso
  5                WHERE cdept=CLAUSTRO.fdept
  6                AND cred=6);

FNOMBRE         FDEP
--------------- ----
JOSE COHN       CIS
AL HARTLEY      CIS
BARB HLAVATY    CIS

SQL>
```

La consulta siguiente devuelve los productos para los cuales existe al menos un pedido en la tabla pedidos que se refiere al producto en cuestión y que tiene un importe de al menos 25.000. Adicionalmente se listan los clientes asignados a Sue Smith que no han remitido un pedido superior a 3.000:

```
SQL> SELECT descripcion
  2  FROM productos
  3  WHERE EXISTS (SELECT num_pedido
  4                FROM pedidos
  5                WHERE producto=id_producto
  6                AND fab=id_fab
  7                AND importe > 25000);

DESCRIPCION
--------------------
Extractor
Riostra 1-Tm
Bisagra Izqda.
Bisagra Dcha.

SQL>
```

```
SQL> SELECT empresa
  2  FROM clientes
  3  WHERE repclie = (SELECT num_empl
  4                   FROM repventas
  5                   WHERE name = 'Sue Smith')
  6  AND NOT EXISTS (SELECT *
  7                  FROM pedidos
  8                  WHERE clie = num_clie
  9                  AND importe > 3000);

EMPRESA
--------------------
Fred Lewis Corp.
Carter Sons

SQL>
```

La consulta siguiente lista las oficinas en donde haya un vendedor cuya cuota represente más del 55% del objetivo de la oficina:

```
SQL> SELECT ciudad
  2  FROM oficinas
  3  WHERE EXISTS (SELECT *
  4                FROM repventas
  5                WHERE oficina_rep=oficina
  6                AND cuota > (0.55*objetivo));

CIUDAD
---------------
Atlanta
Denver

SQL>
```

Obsérvese que las subconsultas presentadas con EXISTS son ligeramente distintas de las restantes subconsultas en los aspectos siguientes:

SYNAPS PREMIUM

- La palabra clave EXISTS no se precede de un nombre de columna, constante u otra expresión.

- La lista de selección de una subconsulta presentada por EXISTS casi siempre consta de un asterisco (*). No hay razón para enumerar los nombres de las columnas porque simplemente se está comprobando la existencia de filas que cumplan las condiciones especificadas en la subconsulta.

La palabra clave EXISTS es importante porque a veces no hay una formulación alternativa de la subconsulta. Aunque algunas consultas formuladas con EXISTS no se pueden expresar de ninguna otra forma, todas las consultas que usan IN o un operador de comparación modificado por ANY o ALL se pueden expresar con EXISTS.

NOT EXISTS funciona igual que EXISTS, salvo por el hecho de que la cláusula WHERE en la que se utiliza se cumple, si la subconsulta no devuelve ninguna fila. Las subconsultas presentadas con EXISTS y NOT EXISTS se pueden usar en dos operaciones de la teoría de conjuntos: intersección y diferencia. La intersección de dos conjuntos contiene los elementos que pertenecen a los dos conjuntos originales. La diferencia contiene los elementos que pertenecen solo al primero de los dos conjuntos.

Subconsultas con ANY, SOME o ALL

Los operadores de comparación que presentan una subconsulta se pueden modificar mediante las palabras clave ALL o ANY. SOME es un equivalente del estándar SQL-92 de ANY.

Las subconsultas presentadas con un operador de comparación modificado devuelven una lista de cero o más valores y pueden incluir una cláusula GROUP BY o HAVING. Estas subconsultas se pueden formular con EXISTS.

Si se usa como ejemplo el operador de comparación >, >ALL significa mayor que cualquier valor; es decir, mayor que el valor máximo. Por ejemplo, >ALL (1, 2, 3) significa mayor que 3. >ANY significa mayor que, como mínimo, un valor, es decir, mayor que el mínimo. Entonces, >ANY (1, 2, 3) significa mayor que 1. Para que una fila de una subconsulta con >ALL satisfaga la condición especificada en la consulta exterior, el valor de la columna que presenta la subconsulta debe ser mayor que cada valor de la lista de los valores devueltos por la subconsulta. De forma parecida, >ANY significa que, para que una fila satisfaga la condición especificada en la consulta exterior, el valor de la columna que presenta la subconsulta debe ser mayor que, como mínimo, uno de los valores de la lista devuelta por la subconsulta.

© Alfaomega - RC Libros

SYNAPS PREMIUM

Si la subconsulta no devuelve ningún valor, la consulta completa no podrá devolver ningún valor. Sin embargo, el operador < >ANY difiere de NOT IN ya que se tiene que < >ANY significa no = a o no = b o no = c... y NOT IN significa no = a y no = b y no = c... <>ALL significa lo mismo que NOT IN.

La consulta siguiente visualiza el nombre y el número de ayudantes para aquellos miembros de la facultad que tienen tantos ayudantes como el número de créditos ofrecidos para cualquier curso. Adicionalmente se muestran los vendedores que han aceptado un pedido que supone más del 10% de su cuota:

```
SQL> SELECT fnombre, fnumdep
  2  FROM claustro
  3  WHERE fnumdep = ANY (SELECT cred
  4                       FROM curso);

FNOMBRE             FNUMDEP
--------------- ----------
BARB HLAVATY             3

SQL>
```

```
SQL> SELECT name
  2  FROM repventas
  3  WHERE (0.1*cuota) < ANY (SELECT importe
  4                           FROM pedidos
  5                           WHERE rep = num_empl);

NAME
---------------
Sam Clark
Nancy Angelli
Larry Fitch

SQL>
```

La consulta siguiente devuelve cada uno de los vendedores que para alguna oficina no son directores de oficina. Adicionalmente se listan las oficinas y sus objetivos para las que todos los vendedores tienen ventas que superan al 50% del objetivo de la oficina:

```
SQL> SELECT name, edad
  2  FROM repventas
  3  WHERE NOT (num_empl = ANY (SELECT dir
  4                             FROM oficinas));

NAME                 EDAD
--------------- ----------
Dan Roberts              45
Sue Smith                48
Paul Cruz                29
Nancy Angelli            49
Mary Jones               31
Tom Snyder               41
```

```
SQL> SELECT ciudad, objetivo
  2  FROM oficinas
  3  WHERE (0.50*objetivo < ALL(SELECT ventas
  4                             FROM repventas
  5                             WHERE oficina_rep=oficina));

CIUDAD           OBJETIVO
--------------- ----------
New York            575000
Atlanta             350000
Denver              300000

SQL>
```

Subconsultas usadas en cualquier lugar de una expresión

En SQL, una subconsulta se puede usar allí donde se pueda usar una expresión en las instrucciones SELECT, UPDATE, INSERT y DELETE, excepto en una lista ORDER BY.

SYNAPS PREMIUM

La siguiente consulta busca el importe de los productos del fabricante ACI, el importe promedio de todos los productos y la diferencia entre el importe del producto y el precio promedio de todos los productos:.

```
SQL>  SELECT producto, importe,
  2   (SELECT AVG(importe) FROM pedidos) AS MEDIA,
  3   importe-(SELECT AVG(importe) FROM pedidos) AS DIFERENCIA
  4   FROM pedidos
  5   WHERE fab='ACI';

PRODU    IMPORTE      MEDIA DIFERENCIA
----- ---------- ---------- ----------
41002       22500     8301,5    14198,5
41004        3276     8301,5    -5025,5
41004        3978     8301,5    -4323,5
41002       15000     8301,5     6698,5
41004         702     8301,5    -7599,5
4100Y       27500     8301,5    19198,5
41002         760     8301,5    -7541,5
41003        3745     8301,5    -4556,5
41002        4104     8301,5    -4197,5
4100X         150     8301,5    -8151,5
4100X         600     8301,5    -7701,5
```

Subconsultas con múltiples niveles de anidamiento

Una subconsulta puede incluir en su interior una o varias subconsultas y en una instrucción se puede anidar cualquier número de subconsultas. SQL no especifica un número máximo de niveles de anidación de subconsultas pero, en la práctica, la acumulación de niveles de subconsultas consume más tiempo para la respuesta y provoca dificultades en su lectura y comprensión.

En la consulta siguiente se visualiza el número de estudiantes y la fecha de matriculación de todos los estudiantes que están matriculados en al menos un curso ofrecido por un departamento ubicado en el edificio Ciencias SC. Adicionalmente se muestran los clientes cuyos vendedores están asignados a oficinas cuya región de ventas es el Oeste.

```
SQL> SELECT sno, fecha_mat
  2  FROM matricula
  3  WHERE cno IN (SELECT cno
  4                FROM curso
  5                WHERE cdept IN (SELECT dept
  6                                FROM departamento
  7                                WHERE DEDIF = 'SC'));

SNO FECHA_MA
--- --------
325 04/01/87
800 15/12/87
100 17/12/87
150 17/12/87
```

```
SQL> SELECT empresa
  2  FROM clientes
  3  WHERE repclie IN (SELECT nun_empl
  4                    FROM repventas
  5                    WHERE oficina_rep IN (SELECT oficina
  6                                          FROM oficinas
  7                                          WHERE region='Oeste'));

EMPRESA
--------------------
Fred Lewis Corp.
Zetacorp
Orion Corp.
Midwest Systems
Rico Enterprises
Carter Sons
Peter Brothers
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

SUBCONSULTAS CORRELACIONADAS

Una subconsulta correlacionada es una subconsulta que hace referencia a una columna de la instrucción externa. La consulta interna se ejecuta para cada fila candidata de la instrucción externa.

Se pueden evaluar muchas consultas mediante la ejecución de la subconsulta una vez y la sustitución del valor o valores resultantes en la cláusula WHERE de la consulta externa. En las consultas que incluyen una subconsulta correlacionada (conocida también como una subconsulta repetida), la subconsulta depende de la consulta externa para sus valores. Esto significa que la subconsulta se ejecuta varias veces, una vez por cada fila que pueda ser seleccionada por la consulta externa.

La consulta siguiente devuelve las oficinas cuyas ventas están por debajo del objetivo medio:

```
SQL> SELECT ciudad
  2  FROM oficinas
  3  WHERE ventas < (SELECT AVG(objetivo)
  4                  FROM oficinas);

CIUDAD
---------------
Atlanta
Denver

SQL>
```

Subconsultas correlacionadas con alias

Las subconsultas correlacionadas se pueden utilizar en operaciones como, por ejemplo, seleccionar datos de una tabla a la que se hace referencia en la consulta externa. En este caso debe de utilizar un alias de tabla (llamado también un nombre de correlación) para especificar de forma no ambigua qué referencia de tabla utilizar. La consulta siguiente visualiza el nombre, identificador de cada departamento y salario de aquellos miembros de la facultad cuyo salario es mayor que el salario medio de su departamento:

```
SQL> SELECT fnombre, fdept
  2  FROM claustro FX
  3  WHERE fsueldo > (SELECT AVG(fsueldo)
  4                   FROM claustro
  5                   WHERE fdept = FX.fdept);

FNOMBRE          FDEP
---------------- ----
JESSIE MARTIN    THEO
AL HARTLEY       CIS
JULIA MARTIN     PHIL

SQL>
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

La consulta siguiente visualiza el nombre e identificador de departamento de cualquier catedrático que tenga un número de profesores ayudantes mayor que la media de créditos para cursos ofrecidos por su departamento. Adicionalmente se obtienen los directores mayores de 40 años y que dirigen a un vendedor cuyas ventas superan la cuota y que no trabaja en la misma oficina de ventas que el director:

```
Fichero  Editar  Buscar  Opciones  Ayuda
SQL> SELECT fnombre, fdept
  2  FROM claustro FX
  3  WHERE fnumdep > (SELECT AVG(cred)
  4                   FROM curso
  5                   WHERE cdept=FX.fdept);

FNOMBRE          FDEP
---------------  ----
AL HARTLEY       CIS

SQL>
```

```
SQL> SELECT name
  2  FROM repventas DIRS
  3  WHERE edad > 40
  4  AND DIRS.nun_empl IN (SELECT director
  5                        FROM repventas EMPS
  6                        WHERE EMPS.cuota > EMPS.ventas
  7                        AND EMPS.oficina_rep <> DIRS.oficina_rep);

NAME
--------------
San Clark
Larry Fitch

SQL>
```

Subconsultas correlacionadas con operadores de comparación

En las subconsultas correlacionadas pueden utilizarse distintos operadores de comparación, tal y como ya se ha visto en el ejemplo anterior.

En la consulta siguiente se utilizan el operador > en el WHERE externo de la consulta y el operador = en el WHERE interno con la finalidad de obtener todas las oficinas cuyos objetivos exceden a la suma de las cuotas de los vendedores que trabajan en ellas. A continuación, en una nueva consulta se utilizan el operador > en el WHERE externo de la consulta y también el operador > en el WHERE interno con la finalidad de obtener la lista de directores que tienen más de 40 años y que dirigen a un vendedor que está por encima de la cuota:

```
SQL> SELECT ciudad
  2  FROM oficinas
  3  WHERE objetivo > (SELECT SUM(cuota)
  4                    FROM repventas
  5                    WHERE oficina_rep=oficina);

CIUDAD
---------------
Chicago
Los Ángeles

SQL>
```

```
SQL> SELECT name
  2  FROM repventas
  3  WHERE edad > 40
  4  AND num_empl IN (SELECT director
  5                   FROM repventas
  6                   WHERE ventas > cuota);

NAME
---------------
Sam Clark
Larry Fitch

SQL>
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Subconsultas correlacionadas con una cláusula HAVING

Una subconsulta correlacionada se puede usar también en la cláusula HAVING de una consulta externa. Por ejemplo, la consulta siguiente obtiene los vendedores cuyo tamaño de pedido medio para productos fabricados por ACI es superior al tamaño del pedido medio global:

```
SQL> SELECT name, AVG(importe)
  2  FROM repventas, pedidos
  3  WHERE num_empl=rep
  4  AND fab='ACI'
  5  GROUP BY name
  6  HAVING AVG(importe) > (SELECT AVG(importe)
  7                         FROM pedidos);

NAME            AVG(IMPORTE)
--------------- ------------
Sue Smith              15000
Tom Snyder             22500

SQL>
```

Podemos complicar un poco la consulta anterior obteniendo los vendedores cuyo tamaño de pedido medio para productos fabricados por ACI es al menos tan grande como el tamaño de pedido medio global de ese vendedor:

```
SQL> SELECT name, AVG(importe)
  2  FROM repventas, pedidos
  3  WHERE num_empl=rep
  4  AND fab='ACI'
  5  GROUP BY name, num_empl
  6  HAVING AVG(importe) >= (SELECT AVG(importe)
  7                          FROM pedidos
  8                          WHERE rep=num_empl);

NAME            AVG(IMPORTE)
--------------- ------------
Bill Adams            7865,4
Sue Smith              15000
Tom Snyder             22500

SQL>
```

UNION, INTERSECT y MINUS en subconsultas

Es típico utilizar uniones, intersecciones y diferencias dentro de la sintaxis de las subconsultas. Por ejemplo, la consulta siguiente busca aquellos nombres de empleados que están en la tabla POSTERIOR o en la tabla plantilla. Adicionalmente, se presenta la consulta anterior, pero con el operador OR:

SYNAPS PREMIUM

```
SQL> SELECT nombre
  2  FROM empleado
  3  WHERE nombre IN (SELECT nombre FROM posterior)
  4                   UNION
  5                  (SELECT nombre FROM plantilla);

NOMBRE
-------------------------
ALONSO TELLEZ
DORADO RODRÍGUEZ
DÍAZ JIMÉNEZ
EZQUERRO TOMÉ
GONZÁLEZ LÓPEZ
GONZÁLEZ ORBEA
JUÁREZ NORIEGA
PEDRERO LÁINEZ
PÉREZ LÓPEZ
TEJERINA RUIZ
VÁZQUEZ LÓPEZ
```

```
SQL> SELECT nombre
  2  FROM empleado
  3  WHERE nombre IN (SELECT nombre FROM posterior)
  4                   OR nombre IN
  5                  (SELECT nombre FROM plantilla);

NOMBRE
-------------------------
ALONSO TELLEZ
DÍAZ JIMÉNEZ
EZQUERRO TOMÉ
GONZÁLEZ ORBEA
PÉREZ LÓPEZ
VÁZQUEZ LÓPEZ
GONZÁLEZ LÓPEZ
JUÁREZ NORIEGA
TEJERINA RUIZ
```

La consulta siguiente presenta los nombres de empleados que están tanto en la tabla POSTERIOR como en la tabla PLANTILLA. Luego, se ve la misma consulta con AND:

```
SQL> SELECT nombre
  2  FROM empleado
  3  WHERE nombre IN (SELECT nombre FROM posterior)
  4                   INTERSECT
  5                  (SELECT nombre FROM plantilla);

NOMBRE
-------------------------
ALONSO TELLEZ
EZQUERRO TOMÉ
PÉREZ LÓPEZ
VÁZQUEZ LÓPEZ

SQL>
```

```
SQL> SELECT nombre
  2  FROM empleado
  3  WHERE nombre IN (SELECT nombre FROM posterior)
  4                   AND nombre IN
  5                  (SELECT nombre FROM plantilla);

NOMBRE
-------------------------
ALONSO TELLEZ
EZQUERRO TOMÉ
PÉREZ LÓPEZ
VÁZQUEZ LÓPEZ

SQL>
```

La consulta siguiente presenta los nombres de empleados que están en la tabla POSTERIOR y no están en la tabla POSTERIOR. A continuación, se presenta la misma consulta con AND NOT IN:

```
SQL> SELECT nombre
  2  FROM empleado
  3  WHERE nombre IN (SELECT nombre FROM posterior)
  4                   MINUS
  5                  (SELECT nombre FROM plantilla);

NOMBRE
-------------------------
GONZÁLEZ LÓPEZ
JUÁREZ NORIEGA
TEJERINA RUIZ

SQL>
```

```
SQL> SELECT nombre
  2  FROM empleado
  3  WHERE nombre IN (SELECT nombre FROM posterior)
  4                   AND nombre NOT IN
  5                  (SELECT nombre FROM plantilla);

NOMBRE
-------------------------
GONZÁLEZ LÓPEZ
JUÁREZ NORIEGA
TEJERINA RUIZ

SQL>
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

USO AVANZADO DE FUNCIONES Y VARIABLES

En un capítulo anterior se estudiaron los tipos de funciones en Oracle. No obstante, aquí se profundizará algo más en el uso de las funciones aportando ejemplos.

Funciones de cadena

Entre las funciones de cadena más utilizadas se encuentran las siguientes:

||:	Une o concatena cadenas.
INITCAP (cadena)	Cambia la primera letra de una palabra a mayúsculas.
INSTR (cadena, 'caracter', inicio, n)	Busca la enésima ocurrencia de un carácter en una cadena comenzando a buscar en el lugar *inicio.*
LENGHT(cadena)	Indica la longitud de una cadena.
LOWER(cadena)	Convierte todas las letras de una cadena a minúsculas.
LPAD(cadena, longitud, 'caracteres')	Da a la cadena una determinada longitud añadiendo un conjunto de caracteres a su izquierda.
LTRIM(cadena, 'caracteres')	Suprime un conjunto de caracteres en la parte izquierda de una cadena.
RPAD(cadena, longitud, 'caracteres')	Da a la cadena una determinada longitud añadiendo un conjunto de caracteres a su derecha.
RTRIM(cadena, 'caracteres')	Suprime un conjunto de caracteres de la parte derecha de una cadena.
SOUNDEX('palabra')	Busca palabras que se parezcan a una dada.
SUBSTR(cadena,m,n)	Extrae la subcadena de *cadena* que se encuentra entre los caracteres que ocupan los lugares *m* y *n*.
UPPER(cadena)	Convierte todas las letras de una cadena a mayúsculas.

© Alfaomega - RC Libros

SYNAPS PREMIUM

En la consulta siguiente se presentan en letras mayúsculas los nombres de las ciudades en donde están las oficinas. Adicionalmente se presenta una consulta que obtiene, separadas por una coma, la ciudad y la región donde se sitúan las oficinas:

```
SQL> SELECT ciudad, UPPER(ciudad)
  2  FROM oficinas;

CIUDAD          UPPER(CIUDAD)
--------------- ---------------
New York        NEW YORK
Chicago         CHICAGO
Atlanta         ATLANTA
Los Ángeles     LOS ÁNGELES
Denver          DENVER

SQL>
```

```
SQL> SELECT ciudad||', '||region
  2  FROM oficinas;

CIUDAD||','||REGION
---------------------------
New York        , Este
Chicago         , Este
Atlanta         , Este
Los Ángeles     , Oeste
Denver          , Oeste
```

En la consulta siguiente se presentan, separadas por una coma, la ciudad y la región donde se sitúan las oficinas, de modo que la coma se encuentra inmediatamente después del último carácter del nombre de la ciudad (*RTRIM sin el segundo argumento elimina los espacios en blanco situados a la derecha de la cadena*):

```
SQL> SELECT RTRIM(ciudad)||', '||region
  2  FROM oficinas;

RTRIM(CIUDAD)||','||REGION
--------------------------
New York, Este
Chicago, Este
Atlanta, Este
Los Ángeles, Oeste
Denver, Oeste

SQL>
```

En la consulta siguiente se presentan las ciudades y las regiones separadas por una línea de puntos hasta completar 35 caracteres. Adicionalmente se presenta la misma consulta pero obteniendo en la salida los puntos pegados al final del nombre de las ciudades:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT RPAD(ciudad, 35, '.'), region
  2  FROM oficinas;

RPAD(CIUDAD,35,'.')                REGION
---------------------------------- ----------
New York        ..................... Este
Chicago         ..................... Este
Atlanta         ..................... Este
Los Ángeles     ..................... Oeste
Denver          ..................... Oeste

SQL>
```

```
SQL> SELECT RPAD(RTRIM(ciudad),35,'.'),region
  2  FROM oficinas;

RPAD(RTRIM(CIUDAD),35,'.')         REGION
---------------------------------- ----------
New York........................... Este
Chicago............................ Este
Atlanta............................ Este
Los Ángeles........................ Oeste
Denver............................. Oeste

SQL>
```

La consulta siguiente presenta las ventas con dos ceros decimales. Así mismo, se presentan las ciudades donde se encuentran las oficinas con su nombre inicial, en mayúsculas, en minúsculas, y con la primera letra en mayúsculas:

```
SQL> SELECT ventas||','||'00'
  2  FROM repventas;

VENTAS||','||'00'
-----------------------------------------
305673,00
474050,00
286775,00
142594,00
367911,00
299912,00
186042,00
361865,00
392725,00
75985,00
```

```
SQL> SELECT ciudad, UPPER(ciudad), LOWER(ciudad), INITCAP(LOWER(ciudad))
  2  FROM oficinas;

CIUDAD          UPPER(CIUDAD)   LOWER(CIUDAD)   INITCAP(LOWER(C
--------------- --------------- --------------- ---------------
New York        NEW YORK        new york        New York
Chicago         CHICAGO         chicago         Chicago
Atlanta         ATLANTA         atlanta         Atlanta
Los Ángeles     LOS ÁNGELES     los ángeles     Los Ángeles
Denver          DENVER          denver          Denver

SQL>
```

La consulta siguiente elimina los puntos situados al final de los nombres de las empresas clientes (*El RTRIM interior elimina los espacios situados a la derecha de los nombres de las empresas, ya que no se especifica el segundo argumento*). Asímismo, se obtienen las fechas de contrato de los vendedores, primeramente con día y mes y, después, con mes y año:

```
SQL> SELECT empresa, RTRIM(RTRIM(empresa),'.')
  2  FROM clientes;

EMPRESA              RTRIM(RTRIM(EMPRESA)
-------------------- --------------------
Jones Mfg.           Jones Mfg
First. Corp.         First. Corp
Acme Mfg.            Acme Mfg
AAA Investments      AAA Investments
Fred Lewis Corp.     Fred Lewis Corp
Ace International    Ace International
Holm Landis          Holm Landis
Chen Associates      Chen Associates
JCP Inc.             JCP Inc
Zetacorp             Zetacorp
Ian Schmidt          Ian Schmidt
Orion Corp.          Orion Corp
Smithson Corp.       Smithson Corp
J.P. Sinclair        J.P. Sinclair
Midwest Systems      Midwest Systems
Solomon Inc.         Solomon Inc
Rico Enterprises     Rico Enterprises
QMA Assoc.           QMA Assoc
Three-Way Lines      Three-Way Lines
Carter Sons          Carter Sons
Peter Brothers       Peter Brothers
```

```
SQL> SELECT SUBSTR(contrato,1,5), SUBSTR(contrato,4,8)
  2  FROM repventas;

SUBST SUBST
----- -----
20/10 10/86
12/12 12/86
01/03 03/87
19/05 05/87
12/02 02/88
14/06 06/88
14/11 11/88
12/10 10/89
12/10 10/99
13/01 01/90
```


SYNAPS PREMIUM

En la consulta siguiente se presentan los nombres de las empresas clientes seguidos de la primera ocurrencia de la letra '*a*' dentro de los mismos. Pero si lo que se quiere es obtener la segunda ocurrencia de la letra 'a', la consulta quedaría como se indica en la segunda salida (la sintaxis *INSTR(cadena, 'caracteres', inicio, n)* da la posición de la ocurrencia enésima de los caracteres especificados empezando a contar en el carácter que ocupa el lugar 'inicio'):

```
SQL> SELECT name, INSTR(name,'a')
  2  FROM repventas;

NAME            INSTR(NAME,'A')
--------------- ---------------
Dan Roberts                   2
Sue Smith                     0
Paul Cruz                     2
Bob Smith                     0
Bill Adams                    8
Sam Clark                     2
Nancy Angelli                 2
Larry Fitch                   2
Mary Jones                    2
Tom Snyder                    0
```

```
SQL> SELECT name, INSTR(name,'a',1,2)
  2  FROM repventas;

NAME            INSTR(NAME,'A',1,2)
--------------- -------------------
Dan Roberts                       0
Sue Smith                         0
Paul Cruz                         0
Bob Smith                         0
Bill Adams                        0
Sam Clark                         7
Nancy Angelli                     0
Larry Fitch                       0
Mary Jones                        0
Tom Snyder                        0
```

En la consulta siguiente se utiliza la función SOUNDEX para obtener los fabricantes la cantidad fabricada y su importe para todos los fabricantes cuya descripción suene como ACI:

```
SQL> SELECT fab, cant, importe
  2  FROM pedidos
  3  WHERE SOUNDEX(fab)=SOUNDEX('ACI');

FAB        CANT    IMPORTE
--- ---------- ----------
ACI           9      22500
ACI          28       3276
ACI          34       3978
ACI           6      15000
ACI           6        702
ACI          11      27500
ACI          10        760
ACI          35       3745
ACI          54       4104
ACI           6        150
ACI          24        600
```

Funciones numéricas

A continuación, se presenta la sintaxis de las funciones numéricas más importantes de Oracle SQL ilustrada con ejemplos.

© Alfaomega - RC Libros

SYNAPS PREMIUM

a+b	Suma de *a* y *b*
a-b	Diferencia entre *a* y *b*
a*b	Producto de *a* y *b*
a/b	Cociente entre *a* y *b*
ABS(a)	Valor absoluto de *a*
CEIL(a)	Entero más pequeño mayor o igual que *a*
COS(a)	Coseno de *a* (en radianes)
COSH(a)	Coseno hiperbólico de *a*
EXP(a)	Número *e* elevado al exponente *a*
FLOOR(a)	Mayor entero menor o igual que *a*
LN(a)	Logaritmo neperiano de *a*
LOG(a)	Logaritmo decimal de *a*
MOD(a,b)	Resto de *a* dividido por *b*
NVL(a,b)	Sustituye *a* por *b* siempre y cuando *a* sea NULL
POWER(a,b)	Halla *a* elevado a *b*
ROUND(a,b)	Redondea *a* con precisión *b*
SIGN(a)	Vale 1 si *a* es positivo y -1 si *a* es negativo
SIN(a)	Seno de *a*
SINH(a)	Seno hiperbólico de *a*
SQRT(a)	Raíz cuadrada de *a*
TAN(a)	Tangente de *a*
TANH(a)	Tangente hiperbólica de *a*
TRUN(a,b)	Trunca *a* con precisión *b*
VSIZE(a)	Tamaño de almacenamiento de *a* en Oracle
AVG(columna)	Media de la columna
COUNT(columna)	Número de elementos en la columna
MAX(columna)	Máximo valor de la columna
MIN(columna)	Mínimo valor de la columna
STDDEV(columna)	Desviación típica de la columna
SUM(columna)	Suma los valores de la columna
VARIANCE(columna)	Varianza de la columna
GREATEST(a,b,c,...)	Mayor valor de la lista
LEAST(a,b,c,...)	Menor valor de la lista

La consulta siguiente lista los números de oficina de los representantes de ventas sustituyendo los valores nulos por un cero:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT oficina_rep, NVL(oficina_rep,0)
  2  FROM repventas;

OFICINA_REP NVL(OFICINA_REP,0)
----------- ------------------
         12                 12
         21                 21
         12                 12
         12                 12
         13                 13
         11                 11
         22                 22
         21                 21
         11                 11
                             0
```

La consulta siguiente presenta el ratio ventas/cuota para cada representante de ventas con precisión por defecto, con redondeo a dos decimales y con truncamiento por el segundo decimal:

```
SQL> r
  1  SELECT name, ventas/cuota, ROUND(ventas/cuota,2), TRUNC(ventas/cuota,2
  2* FROM repventas

NAME            VENTAS/CUOTA ROUND(VENTAS/CUOTA,2) TRUNC(VENTAS/CUOTA,2)
--------------- ------------ -------------------- --------------------
Dan Roberts          1,01891                 1,02                 1,01
Sue Smith         1,35442857                 1,35                 1,35
Paul Cruz         1,04281818                 1,04                 1,04
Bob Smith             ,71297                  ,71                  ,71
Bill Adams        1,05117429                 1,05                 1,05
Sam Clark         1,09058909                 1,09                 1,09
Nancy Angelli         ,62014                  ,62                  ,62
Larry Fitch           1,0339                 1,03                 1,03
Mary Jones        1,30908333                 1,31                  1,3
Tom Snyder
```

La consulta siguiente presenta el ratio ventas sobre cuota aproximado por las funciones CEIL y FLOOR:

```
SQL> SELECT ventas/cuota, CEIL(ventas/cuota), FLOOR(ventas/cuota)
  2  FROM repventas;

VENTAS/CUOTA CEIL(VENTAS/CUOTA) FLOOR(VENTAS/CUOTA)
------------ ------------------ -------------------
     1,01891                  2                   1
  1,35442857                  2                   1
  1,04281818                  2                   1
      ,71297                  1                   0
  1,05117429                  2                   1
  1,09058909                  2                   1
      ,62014                  1                   0
      1,0339                  2                   1
  1,30908333                  2                   1
```

La consulta siguiente presenta los logaritmos decimal y neperiano, así como la raíz cuadrada y el cuadrado de las ventas:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT ventas, LOG(10,ventas), POWER(ventas,2), SQRT(ventas), LN(ventas)
  2  FROM repventas;

    VENTAS LOG(10,VENTAS) POWER(VENTAS,2) SQRT(VENTAS) LN(VENTAS)
---------- -------------- --------------- ------------ ----------
    305673     5,48525708      9,3436E+10   552,877021 12,6302712
    474050     5,67582415      2,2472E+11    688,51289 13,0690681
    286775     5,45754129      8,2240E+10   535,513772 12,5664532
    142594     5,15410125      2,0333E+10   377,616207 11,8677567
    367911     5,56574277      1,3536E+11   606,556675 12,8155963
    299912     5,47699384      8,9947E+10   547,642219 12,6112444
    186042       5,269611      3,4612E+10   431,325863 12,1337277
    361865     5,55854658      1,3095E+11   601,552159 12,7990265
    392725     5,59408855      1,5423E+11   626,677748 12,8808649
     75985     4,88072787      5773720225   275,653768 11,2382912
```

Las funciones pueden anidarse a nuestro gusto tal y como se indica en la siguiente consulta:

```
SQL> SELECT ventas, EXP(LN(ventas)), POWER(SQRT(ventas),2)
  2  FROM repventas;

    VENTAS EXP(LN(VENTAS)) POWER(SQRT(VENTAS),2)
---------- --------------- ---------------------
    305673          305673                305673
    474050          474050                474050
    286775          286775                286775
    142594          142594                142594
    367911          367911                367911
    299912          299912                299912
    186042          186042                186042
    361865          361865                361865
    392725          392725                392725
     75985           75985                 75985
```

La consulta siguiente presenta las ventas totales, medias, máximas y mínimas por cada oficina de representación:

```
SQL> SELECT oficina_rep, SUM(ventas), AVG(ventas), MAX(ventas), MIN(vent
  2  FROM repventas
  3  GROUP BY oficina_rep;

OFICINA_REP SUM(VENTAS) AVG(VENTAS) MAX(VENTAS) MIN(VENTAS)
----------- ----------- ----------- ----------- -----------
         11      692637    346318,5      392725      299912
         12      735042      245014      305673      142594
         13      367911      367911      367911      367911
         21      835915    417957,5      474050      361865
         22      186042      186042      186042      186042
                  75985       75985       75985       75985
```

La consulta siguiente calcula la desviación típica y la varianza de las ventas medias por cada oficina de representación:

SYNAPS PREMIUM

```
SQL> SELECT oficina_rep, STDDEV(ventas), VARIANCE(ventas)
  2  FROM repventas
  3  GROUP BY oficina_rep;

OFICINA_REP STDDEV(VENTAS) VARIANCE(VENTAS)
----------- -------------- ----------------
         11     65628,7017       4307126485
         12     89200,2012       7956675901
         13              0                0
         21     79326,7742       6292737113
         22              0                0
                         0                0
```

La consulta siguiente presenta el mayor de los valores entre ventas y cuota (y el menor de los valores entre ventas y cuota) para cada representante de ventas:

```
SQL>  SELECT name, GREATEST(ventas,cuota) "ALTAS" , LEAST(ventas,cuota) "BAJAS"
  2   FROM repventas;

NAME                 ALTAS      BAJAS
-------------- ---------- ----------
Dan Roberts          305673     300000
Sue Smith            474050     350000
Paul Cruz            286775     275000
Bob Smith            200000     142594
Bill Adams           367911     350000
Sam Clark            299912     275000
Nancy Angelli        300000     186042
Larry Fitch          361865     350000
Mary Jones           392725     300000
Tom Snyder
```

La consulta siguiente calcula el número de cuotas y su media, y el número de cuotas distintas y su media:

```
SQL> SELECT COUNT(cuota), COUNT(DISTINCT cuota), AVG(cuota), AVG(DISTINCT cuota)
  2  FROM repventas;

COUNT(CUOTA) COUNT(DISTINCTCUOTA) AVG(CUOTA) AVG(DISTINCTCUOTA)
------------ -------------------- ---------- ------------------
           9                    4     300000             281250

SQL>
```

La consulta siguiente calcula el extremo superior y el extremo inferior de las ventas por oficina de representación:

© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT (MAX(ventas)-MIN(ventas))/2, (MAX(ventas)+MIN(ventas))/2
  2  FROM repventas
  3  GROUP BY oficina_rep;

(MAX(VENTAS)-MIN(VENTAS))/2 (MAX(VENTAS)+MIN(VENTAS))/2
--------------------------- ---------------------------
                   46406,5                    346318,5
                   81539,5                    224133,5
                         0                      367911
                   56092,5                    417957,5
                         0                      186042
                         0                       75985
```

Funciones de fechas

A continuación, se presenta la sintaxis de las funciones para el trabajo con fechas más importantes de Oracle SQL ilustrada con ejemplos:

SYDATE	Devuelve la fecha actual del sistema
ADD_MONTHS(fecha,a)	Añade a meses a la fecha
GREATEST(f1,f2,…)	Devuelve la mayor de las fechas de la lista
LEAST(f1,f2,…)	Devuelve la menor de las fechas de la lista
LAST_DAY(fecha)	Devuelve el último día del mes de la fecha
MONTHS_BETWEEN(f1,f2)	Devuelve los meses entre las fechas f1 y f2.
NEXT_DAY(fecha,'día')	Fecha siguiente al día de la semana dado
NEW_TIME(fecha, 'z1', 'z2')	Cambia la fecha de una zona horaria a otra
ROUND(fecha, 'formato')	Redondea la fecha según el formato dado
TRUNC(fecha, 'formato')	Trunca la fecha según el formato dado
TO_CHAR(fecha, 'formato')	Cambia la fecha de formato
TO_DATE(cadena, 'formato')	Convierte la cadena en el formato de fecha dado

Las **posibles zonas horarias** para utilizar con la función NEW_TIME son las siguientes:

Zona horaria	***Significado***	***Zona horaria***	***Significado***
AST/ADT	*Zona del atlántico*	*BST/BDT*	*Zona de Bering*
CST/CDT	*Zona central*	*EST/EDT*	*Zona este*
GMT	*Zona Greenwich*	*HST/HDT*	*Zona Alaska-Hawaii*
MST/MDT	*Zona Montaña Rocosa*	*NST*	*Zona de Terranova*
PST/PDT	*Zona del pacífico*	*YST/YDT*	*Zona de Yukón*

Los **formatos más comunes de fechas** son los siguientes:

SYNAPS PREMIUM

Formato	*Significado*	*Ejemplo*	*Formato*	*Significado*	*Ejemplo*
MM	*Número de mes*	*12*	*RM*	*Mes en romanos*	*XII*
MON	*Mes con tres letras*	*AGO*	*MONTH*	*Mes con todas las letras*	*AGOSTO*
DDD	*Día del año (Nº)*	*354*	*DD*	*Día del mes (Nº)*	*23*
D	*Día de la semana (Nº)*	*6*	*DY*	*Día de la semana (3 letras)*	*JUE*
DAY	*Día de la semana completo*	*JUEVES*	*YYYY*	*Año con cuatro dígitos*	*1958*
SYYYY	*Año con signo (a. de Cristo)*	*-1000*	*YYY*	*Tres últimos dígitos del año*	*946*
YY	*Dos últimos dígitos del año*	*78*	*Y*	*Último dígito del año*	*9*
RR	*Fecha actual (2 últimos dígitos)*		*YEAR*	*Año en letras*	
Q	*Número de trimestre*	*3*	*WWW*	*Nº de semana en el año*	*45*
IW	*Semaño del año (ISO)*		*W*	*Nº de semana en el mes*	*3*
J	*Día juliano*	*242222*	*HH*	*Hora del día (de 1 a 12)*	*11*
HH12	*Igual que HH*	*11*	*HH24*	*Hora del día (de 1 a 24)*	*23*
MI	*Minuto de la hora*	*51*	*SS*	*Segundo de la hora*	*43*
SSSS	*Segundos desde media noche*	*43000*	*A.M.*	*A.M o P.M según corresponda*	
P.M.	*Igual que A.M.*		*AM*	*Igual que A.M (sin puntos)*	
PM	*Igual que P.M. (sin puntos)*		*BC o B.C.*	*Antes de cristo*	
AD o A.D.	*Después de Cristo*		*fm*	*Prefijo para meses y días de igual ancho*	
TH	*Sufijo de Nº de mes o día*		*SP*	*Sufijo que obliga a escribir con letras*	

Como primera consulta se presenta la fecha actual del sistema. La segunda presenta los días transcurridos desde la fecha del contrato de los vendedores hasta la fecha actual del sistema:

```
SQL> SELECT sysdate
  2  FROM dual;

SYSDATE
--------
19/01/01

SQL>
```

```
SQL> SELECT sysdate-contrato
  2  FROM repventas;

SYSDATE-CONTRATO
----------------
       5205,77192
       5152,77192
       5073,77192
       4994,77192
       4725,77192
       4602,77192
       4449,77192
       4117,77192
       465,771921
       4024,77192
```

La consulta siguiente añade medio año a la fecha de contrato de los vendedores:

```
SQL>  SELECT contrato, ADD_MONTHS(contrato,6) "SEIS MÁS"
  2  FROM repventas;

CONTRATO SEIS MÁS
-------- --------
20/10/86 20/04/87
12/12/86 12/06/87
01/03/87 01/09/87
19/05/87 19/11/87
12/02/88 12/08/88
14/06/88 14/12/88
14/11/88 14/05/89
12/10/89 12/04/90
12/10/99 12/04/00
13/01/90 13/07/90
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

La consulta siguiente selecciona la mayor y la menor de entre dos fechas dadas:

```
SQL> SELECT LEAST('15-ENE-92','15-DIC-92') "MENOR", GREATEST('15-ENE-92','15-DIC-92') "MAYOR"
  2  FROM dual;

MENOR     MAYOR
--------- ---------
15-DIC-92 15-ENE-92

SQL>
```

La consulta siguiente presenta las fechas del lunes siguiente al día de contrato de los vendedores con el fin de empezar a computar semanas completas de trabajo. Además, se presenta una consulta que ofrece la fecha del último día del mes del contrato de los vendedores con el objeto de empezar a pagar comisiones y primas:

```
SQL> SELECT contrato, NEXT_DAY(contrato, 'Lunes')
  2  from repventas;

CONTRATO NEXT_DAY
-------- --------
20/10/86 27/10/86
12/12/86 15/12/86
01/03/87 02/03/87
19/05/87 25/05/87
12/02/88 15/02/88
14/06/88 20/06/88
14/11/88 21/11/88
12/10/89 16/10/89
12/10/99 18/10/99
13/01/90 15/01/90
```

```
SQL> SELECT contrato, LAST_DAY(contrato)
  2  from repventas;

CONTRATO LAST_DAY
-------- --------
20/10/86 31/10/86
12/12/86 31/12/86
01/03/87 31/03/87
19/05/87 31/05/87
12/02/88 29/02/88
14/06/88 30/06/88
14/11/88 30/11/88
12/10/89 31/10/89
12/10/99 31/10/99
13/01/90 31/01/90
```

La consulta siguiente ofrece la fecha del último día del mes del contrato de los vendedores con el objeto de empezar a pagar comisiones y primas. Adicionalmente se muestra una consulta que presenta los meses transcurridos entre la fecha de contrato de los vendedores y la fecha actual del sistema:

```
SQL> SELECT contrato, LAST_DAY(contrato)
  2  from repventas;

CONTRATO LAST_DAY
-------- --------
20/10/86 31/10/86
12/12/86 31/12/86
01/03/87 31/03/87
19/05/87 31/05/87
12/02/88 29/02/88
14/06/88 30/06/88
14/11/88 30/11/88
12/10/89 31/10/89
12/10/99 31/10/99
13/01/90 31/01/90
```


SYNAPS PREMIUM

```
SQL> SELECT sysdate, contrato, MONTHS_BETWEEN(sysdate, contrato)
  2  FROM repventas;

SYSDATE  CONTRATO MONTHS_BETWEEN(SYSDATE,CONTRATO)
-------- -------- --------------------------------
19/01/01 20/10/86                       170,993202
19/01/01 12/12/86                       169,251267
19/01/01 01/03/87                       166,606106
19/01/01 19/05/87                              164
19/01/01 12/02/88                       155,251267
19/01/01 14/06/88                       151,186751
19/01/01 14/11/88                       146,186751
19/01/01 12/10/89                       135,251267
19/01/01 12/10/99                       15,2512668
19/01/01 13/01/90                       132,219009
```

En la consulta siguiente se presenta la fecha de contrato y el primer mes después de haber pasado 6 meses de la fecha del contrato de los vendedores, con el objetivo de hallar la fecha de su subida de sueldo. Así mismo, se presenta una consulta que halla los días transcurridos desde las fechas de los contratos de los vendedores hasta la fecha actual del sistema redondeada y truncada, respectivamente:

```
SQL> SELECT contrato-ROUND(sysdate), contrato-TRUNC(sysdate)
  2  FROM repventas;

CONTRATO-ROUND(SYSDATE) CONTRATO-TRUNC(SYSDATE)
----------------------- -----------------------
                  -5206                   -5205
                  -5153                   -5152
                  -5074                   -5073
                  -4995                   -4994
                  -4726                   -4725
                  -4603                   -4602
                  -4450                   -4449
                  -4118                   -4117
                   -466                    -465
                  -4025                   -4024
```

```
SQL> SELECT contrato, LAST_DAY(ADD_MONTHS(contrato,6))+1 "SUBIDA"
  2  FROM repventas;

CONTRATO SUBIDA
-------- --------
20/10/86 01/05/87
12/12/86 01/07/87
01/03/87 01/10/87
19/05/87 01/12/87
12/02/88 01/09/88
14/06/88 01/01/89
14/11/88 01/06/89
12/10/89 01/05/90
12/10/99 01/05/00
13/01/90 01/08/90
```

Las tres consultas siguientes presentan distintos formatos de las fechas de contrato de los vendedores:

```
SQL> SELECT contrato, TO_CHAR(contrato, 'Month, DD, YYYY')
  2  FROM repventas;

CONTRATO TO_CHAR(CONTRATO,'MO
-------- --------------------
20/10/86 Octubre   , 20, 1986
12/12/86 Diciembre , 12, 1986
01/03/87 Marzo     , 01, 1987
19/05/87 Mayo      , 19, 1987
12/02/88 Febrero   , 12, 1988
14/06/88 Junio     , 14, 1988
14/11/88 Noviembre , 14, 1988
12/10/89 Octubre   , 12, 1989
12/10/99 Octubre   , 12, 1999
13/01/90 Enero     , 13, 1990
```

```
SQL> SELECT contrato, TO_CHAR(contrato, 'fmMonth, DD, YYYY')
  2  FROM repventas;

CONTRATO TO_CHAR(CONTRATO,'FM
-------- --------------------
20/10/86 Octubre, 20, 1986
12/12/86 Diciembre, 12, 1986
01/03/87 Marzo, 1, 1987
19/05/87 Mayo, 19, 1987
12/02/88 Febrero, 12, 1988
14/06/88 Junio, 14, 1988
14/11/88 Noviembre, 14, 1988
12/10/89 Octubre, 12, 1989
12/10/99 Octubre, 12, 1999
13/01/90 Enero, 13, 1990
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

```
SQL> SELECT contrato, TO_CHAR(contrato, '"Comenzó el " DDSP "de" fmMonth "de" YYYY') "FORMATO"
  2  FROM repventas;

CONTRATO FORMATO
-------- ------------------------------------------
20/10/86 Comenzó el  TWENTY de Octubre de 1986
12/12/86 Comenzó el  TWELVE de Diciembre de 1986
01/03/87 Comenzó el  ONE de Marzo de 1987
19/05/87 Comenzó el  NINETEEN de Mayo de 1987
12/02/88 Comenzó el  TWELVE de Febrero de 1988
14/06/88 Comenzó el  FOURTEEN de Junio de 1988
14/11/88 Comenzó el  FOURTEEN de Noviembre de 1988
12/10/89 Comenzó el  TWELVE de Octubre de 1989
12/10/99 Comenzó el  TWELVE de Octubre de 1999
13/01/90 Comenzó el  THIRTEEN de Enero de 1990
```

En la consulta siguiente se cambian de zona horaria las fechas de los contratos de los vendedores:

```
SQL> SELECT contrato, NEW_TIME(contrato, 'EST', 'HST')
  2  FROM repventas;

CONTRATO NEW_TIME
-------- --------
20/10/86 19/10/86
12/12/86 11/12/86
01/03/87 28/02/87
19/05/87 18/05/87
12/02/88 11/02/88
14/06/88 13/06/88
14/11/88 13/11/88
12/10/89 11/10/89
12/10/99 11/10/99
13/01/90 12/01/90
```

FUNCIONES DE CONVERSIÓN Y TRANSFORMACIÓN

En Oracle hay tres funciones básicas cuya finalidad es convertir un tipo de datos en otro: TO_CHAR transforma a tipo carácter datos fecha o numéricos, TO_DATE transforma a tipo fecha datos numéricos o carácter y TO_NUMBER transforma a tipo numérico datos carácter. Si combinamos estas funciones con otras de trabajo con cadenas, como SUBSTR, tendremos la posibilidad de realizar muchas transformaciones. No olvidemos que *SUBSTR(cadena, m, n)* extrae la subcadena de *cadena* que se encuentra entre los caracteres que ocupan los lugares *m* y *n*. También son potentes otras funciones de cadenas como la concatenación (||), INITCAP (pone la primera letra en mayúsculas), LPAD, etc. También hay otras funciones de fechas interesantes como MONTHS-BETWEN (que cuenta meses entre dos fechas) y funciones numéricas muy útiles como FLOOR (devuelve la parte entera) que pueden dar mucho juego en el campo de las transformaciones.

© Alfaomega - RC Libros

SYNAPS PREMIUM

En la consulta siguiente se transforma el número puro 948033515 en una cadena de caracteres con un guion entre su quinta y sexta cifras:

```
SQL> SELECT SUBSTR(TO_CHAR(948033515),1,5)||'_'||
  2  SUBSTR(TO_CHAR(948033515),6) "código"
  3  from DUAL;

código
----------
94803_3515

SQL>
```

Conversiones automáticas

Oracle también realiza conversiones automáticas de tipos de datos. Cualquier tipo NUMBER o DATE será convertido a CHAR si una función lo necesita. De la misma forma, una cadena CHAR será convertida a NUMBER si contiene solo números, puntos decimales o el signo menos a la izquierda, no existiendo espacios entre los números ni otros caracteres. Una cadena CHAR será convertida a fechas solo si está en el formato DD-MM-YY. Los tipos DATE y NUMBER no son convertibles entre sí automáticamente.

En la consulta siguiente, la función FLOOR trata a una cadena como si fuera tipo NUMBER. En la segunda consulta, la función MONTHS_BETWENN convierte dos cadenas literales al tipo DATE:

```
SQL> SELECT FLOOR('-323,78')
  2  FROM dual;

FLOOR('-323,78')
----------------
            -324

SQL>
```

```
SQL> SELECT MONTHS_BETWEEN('20-MAY-92','18-FEB-92')
  2  FROM dual;

MONTHS_BETWEEN('20-MAY-92','18-FEB-92')
---------------------------------------
                             3,06451613

SQL>
```

En la consulta siguiente se lista la fecha del sistema, un día antes y un día después:

```
SQL> SELECT sysdate, sysdate-1, sysdate+1
  2  FROM dual;

SYSDATE  SYSDATE- SYSDATE+
-------- -------- --------
19/01/01 18/01/01 20/01/01

SQL>
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Funciones TRANSLATE y DECODE

Existen dos funciones de transformación en Oracle bastante interesantes. Se trata de TRANSLATE y DECODE cuyas sintaxis son las siguientes:

TRANSLATE(cadena1, parte, cadena2): Mira cada carácter de *cadena1*, busca en ella cada carácter de *parte* y lo sustituye en *cadena2* en la misma posición en que se encontraba en *cadena1*.

DECODE(cadena1, parte1, cadena2, parte2, cadena3, parte3, ...,cadena1): Consiste en una aplicación continuada de la función TRANSLATE.

A continuación, se presentan algunos ejemplos de consultas con TRANSLATE y DECODE:

```
SQL> SELECT TRANSLATE(7671234,234567890,'BCDEFGHIJ')
  2  FROM dual;

TRANSLA
-------
GFG1BCD

SQL>
```

```
SQL> SELECT TRANSLATE('HALL','ABCDEFGHIJKLMNOPQRSTUVWXYZ',
  2                           'BCDEFGHIJKLMNOPQRSTUVWXYZA') "QUIÉN"
  3  FROM dual;

QUIÉ
----
IBMM

SQL>
```

```
SQL> SELECT DECODE(oficina_rep,'13','TRECE', '11','ONCE',oficina_rep),
  2  name,ventas
  3  FROM repventas;

DECODE(OFICINA_REP,'13','TRECE','11','ON NAME                VENTAS
---------------------------------------- --------------- ----------
12                                       Dan Roberts         305673
21                                       Sue Smith           474050
12                                       Paul Cruz           286775
12                                       Bob Smith           142594
TRECE                                    Bill Adams          367911
ONCE                                     Sam Clark           299912
22                                       Nancy Angelli       186042
21                                       Larry Fitch         361865
ONCE                                     Mary Jones          392725
                                         Tom Snyder           75985
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

En la última consulta se han sustituido los números de oficina 12 y 13 por sus valores en letras. En la consulta siguiente se sustituyen todos los números de oficina por sus valores en letras:

```
DECODE(OFICINA_REP,'13','TRECE','11','ON NAME                 VENTAS
-------------------------------------- -------------- ----------
DOCE                                   Dan Roberts        305673
VEINTIUNO                              Sue Smith          474050
DOCE                                   Paul Cruz          286775
DOCE                                   Bob Smith          142594
TRECE                                  Bill Adams         367911
ONCE                                   Sam Clark          299912
VEINTIDOS                              Nancy Angelli      186042
VEINTIUNO                              Larry Fitch        361865
ONCE                                   Mary Jones         392725
                                       Tom Snyder          75985
```

Ejercicio 6.1. Obtener el número, nombre y tarifa de los cursos con tarifa mínima no nula.

```
SQL> SELECT cno, cnombre, ctarifa
  2  FROM curso
  3  WHERE ctarifa = (SELECT MIN(ctarifa)
  4                   FROM curso
  5                   WHERE NOT ctarifa = 0);

CNO CNOMBRE                    CTARIFA
--- ---------------------- ----------
C22 ESTRUCT. DE DATOS              50
P22 RACIONALISMO                   50

SQL>
```

Ejercicio 6.2. Obtener el número, nombre y tarifa de los cursos con tarifa menor que la media.

```
SQL> SELECT cno, cnombre, ctarifa
  2  FROM curso
  3  WHERE ctarifa < (SELECT AVG(ctarifa)
  4                   FROM curso);

CNO CNOMBRE                    CTARIFA
--- ---------------------- ----------
C11 INTROD. A LAS CC.             100
C22 ESTRUCT. DE DATOS              50
C33 MATEMÁTICAS DISCRETAS           0
C44 CIRCUITOS DIGITALES             0
C55 ARQUITECT. COMPUTADOR         100
P11 EMPIRISMO                     100
P22 RACIONALISMO                   50
P44 SOLIPSISMO                      0
T22 FUNDAMENTALISMO                90
T33 HEDONISMO                       0
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 6.3. Obtener el número, nombre y tarifa de los cursos con tarifa mayor o igual que el sueldo de cualquier miembro del personal.

```
SQL> SELECT cno, cnombre, ctarifa
  2  FROM curso
  3  WHERE ctarifa >= (SELECT MIN(esueldo)
  4                    FROM personal);

CNO CNOMBRE                    CTARIFA
--- -------------------- ----------
C11 INTROD. A LAS CC.             100
C55 ARQUITECT. COMPUTADOR         100
C66 BASES DE DATOS RELAC.         500
P11 EMPIRISMO                     100
P33 EXISTENCIALISMO               200
T11 ESCOLASTICISMO                150
T22 FUNDAMENTALISMO                90
T44 COMUNISMO                     200
```

Ejercicio 6.4. Obtener el nombre y cargo de cada miembro del personal que trabaja en el edificio de Humanidades.

```
SQL> SELECT enombre, cargo
  2  FROM personal
  3  WHERE dept IN (SELECT dept
  4                 FROM departamento
  5                 WHERE dedif = 'HU');

ENOMBRE          CARGO
---------------  ----------
LUCAS            EVANG1
MARCOS           EVANG2
MATEO            EVANG3
DICK NIX         LADRÓN
HANK KISS        BUFÓN
JUAN             EVANG4
```

Ejercicio 6.5. Obtener el número y nombre de cualquier miembro de la facultad que sea jefe de cualquier departamento que ofrezca un curso de 6 créditos.

```
SQL> SELECT DISTINCT fno, fnombre
  2  FROM claustro
  3  WHERE fno IN (SELECT dchfno
  4                FROM departamento
  5                WHERE dept IN (SELECT cdept
  6                               FROM curso
  7                               WHERE cred = 6));

FNO FNOMBRE
--- ---------------
10  JESSIE MARTIN
60  JULIA MARTIN

SQL>
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 6.6. Obtener el número, cargo e identificador de cada departamento de cada miembro del personal asignado a un departamento no existente (sin identificador).

```
SQL> SELECT enombre, cargo, dept
  2  FROM personal
  3  WHERE dept NOT IN (SELECT dept
  4                     FROM departamento);

ENOMBRE          CARGO      DEPT
---------------  ---------- ----
EUCLIDES         AYTE. LAB. MATH
ARQUIMEDES       AYTE. LAB. ENG
DAVINCI          AYTE. LAB.

SQL>
```

Ejercicio 6.7. Obtener, para cada departamento que ofrezca cursos, el identificador de departamento y la tarifa media de los cursos ofrecidos por el departamento, siempre y cuando sea mayor que la tarifa media de todas las tarifas de cursos.

```
SQL> SELECT cdept, AVG(ctarifa)
  2  FROM curso
  3  GROUP BY cdept
  4  HAVING AVG(ctarifa) > (SELECT AVG(ctarifa)
  5                         FROM curso);

CDEP AVG(CTARIFA)
---- ------------
CIS            125

SQL>
```

Ejercicio 6.8. Obtener el número de curso, nombre y tarifa de cada curso cuya tarifa exceda del salario de cualquier miembro de personal.

```
SQL> SELECT cno, cnombre, ctarifa
  2  FROM curso
  3  WHERE ctarifa > ANY (SELECT esueldo
  4                       FROM personal);

CNO CNOMBRE                    CTARIFA
--- ----------------------- ----------
C11 INTROD. A LAS CC.              100
C55 ARQUITECT. COMPUTADOR          100
C66 BASES DE DATOS RELAC.          500
P11 EMPIRISMO                      100
P33 EXISTENCIALISMO                200
T11 ESCOLASTICISMO                 150
T22 FUNDAMENTALISMO                 90
T44 COMUNISMO                      200
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 6.9. Obtener el número de curso, nombre y tarifa de cada curso cuya tarifa sea menor que todos los salarios de todos los miembros del personal.

```
SQL> SELECT cno, cnombre, ctarifa
  2  FROM curso
  3  WHERE ctarifa < ALL (SELECT esueldo
  4                       FROM personal);

CNO CNOMBRE                   CTARIFA
--- -------------------- ----------
C22 ESTRUCT. DE DATOS             50
C33 MATEMÁTICAS DISCRETAS          0
C44 CIRCUITOS DIGITALES            0
P22 RACIONALISMO                  50
P44 SOLIPSISMO                     0
T33 HEDONISMO                      0
```

Ejercicio 6.10. Obtener, para cada departamento que ofrezca cursos, el identificador de ese departamento seguido del número, nombre y tarifa del departamento que organiza el curso con mayor tarifa.

```
SQL> SELECT cdept, cno, cnombre, ctarifa
  2  FROM curso CX
  3  WHERE ctarifa = (SELECT MAX(ctarifa)
  4                   FROM curso
  5                   WHERE cdept = CX.cdept);

CDEP CNO CNOMBRE                   CTARIFA
---- --- -------------------- ----------
CIS  C66 BASES DE DATOS RELAC.         500
PHIL P33 EXISTENCIALISMO               200
THEO T44 COMUNISMO                     200

SQL>
```

Ejercicio 6.11. Obtener el nombre y el cargo de cualquier miembro del personal asignado a un departamento existente.

```
SQL> SELECT enombre, cargo
  2  FROM personal
  3  WHERE EXISTS (SELECT *
  4                FROM departamento
  5                WHERE DEPARTAMENTO.dept = PERSONAL.dept);

ENOMBRE         CARGO
--------------- ----------
LUCAS           EVANG1
MARCOS          EVANG2
MATEO           EVANG3
DICK NIX        LADRÓN
HANK KISS       BUFÓN
JUAN            EVANG4
```

© Alfaomega - RC Libros

SYNAPS PREMIUM

Ejercicio 6.12. Obtener el nombre y el cargo de cualquier miembro del personal que no esté asignado a un departamento existente.

```
SQL> SELECT enombre, cargo
  2  FROM personal
  3  WHERE NOT EXISTS (SELECT *
  4                    FROM departamento
  5                    WHERE DEPARTAMENTO.dept = PERSONAL.dept);

ENOMBRE         CARGO
--------------- ----------
EUCLIDES        AYTE. LAB.
ARQUIMEDES      AYTE. LAB.
DAVINCI         AYTE. LAB.

SQL>
```

Ejercicio 6.13. Obtener la longitud real de cada nombre de curso ofrecido por el departamento de Teología.

```
SQL> SELECT LENGTH(cnombre)
  2  FROM curso
  3  WHERE cdept = 'PHIL';

LENGTH(CNOMBRE)
---------------
             22
             22
             22
             22

SQL>
```

Ejercicio 6.14. Obtener la primera letra de todos los nombres de cursos de Informática y Ciencias de la Información. Obtener también los caracteres que aparecen en las posiciones tercera, cuarta, quinta y sexta. Por último, mostrar los valores del número de curso sin el primer carácter.

```
SQL> SELECT SUBSTR(cnombre,1,1),
  2         SUBSTR(cnombre,3,5),
  3         SUBSTR(cno,2)
  4  FROM curso
  5  WHERE cdept = 'CIS';

S SUBST SU
- ----- --
I TROD. 11
E TRUCT 22
M TEMÁT 33
C RCUIT 44
A QUITE 55
B SES D 66
```


© Alfaomega - RC Libros

SYNAPS PREMIUM

ÍNDICE ANALÍTICO

A

Actualizaciones de software 3
Agrupar filas con GROUP BY 192
Almacenes de Datos 20, 22

B

Base de datos como Base de Datos de Contenedor 16
Base de datos con Oracle Enterprise Manager Cloud Control 26
Base de Datos Global 21
Base de datos relacional 45, 46, 47, 51, 52, 54, 55, 56, 57, 59, 67, 76, 81

C

Características de SQL 51
Casos especiales en consultas multitabla 182
Chequeo de validez.....59
Ciclos referenciales 61
Cláusula AS.....139, 140, 195
Cláusula BETWEEN...AND.....142
Cláusula DISTINCT 134, 135, 150
Cláusula HAVING.....196
Cláusula IN 143
Cláusula IS NULL 146
Cláusula LIKE 144, 145
Cláusula UNION 149, 150
Cláusula WHERE en una consulta que contenga una cláusula GROUP BY 195
Cláusulas AND, OR y NOT 147
Clave primaria.... 54, 55, 57, 59, 60, 61, 67, 68, 69, 70, 71, 73, 74, 75, 81, 82, 134, 168, 170, 173
Columnas calculadas.....137
Combinación de consultas.....149
Compartición de datos 48
Conectar a base de datos 41
Configuración de Instalación Avanzada..16
Configuración de Instalación Típica 10
Configurar actualizaciones de seguridad..1
Consultas con condiciones de búsqueda 140
Consultas multitabla.....182
Consultas resumen 185
Contraseñas de Esquemas de Base de Datos 33
Contraste de correspondencia con patrón 144
Contraste de pertenencia a un conjunto 143
Contraste de rango 142
Contraste de valor nulo 146

SYNAPS PREMIUM

Contrastes compuestos 147
Contrastes de comparación 141
Control de acceso 48
Conversión de tipos de datos 104
Conversiones automáticas.....236

D

Datos requeridos 59
Definición de datos.....48, 56
desventajas de los índices 65
Directorio Base de Oracle 11
Directorio de Inventario de Oracle 18
Diseño de una base de datos relacional .54
Disparadores.....62

E

Edición de Base de Datos.....10, 13, 16
Elementos básicos de Oracle 12c SQL 94

F

Filas duplicadas.....134
Formatos de elementos numéricos.....96
Formatos de fechas 98
Formatos en Oracle 12c SQL.....96
Función count(*).....188
Funciones.....114
Funciones agregadas 115, 119
Funciones carácter.....114, 117, 118
Funciones de agregado en la lista de selección.....185
Funciones de cadena 223
Funciones de conversión 115, 118
Funciones de conversión y transformación 235
Funciones de fecha 118
Funciones de fechas 231
Funciones numéricas 114, 116, 226
Funciones TRANSLATE y DECODE 237

G

GROUP BY 192
GROUPING para distinguir valores NULL 199
Grupo de Discos de ASM 29
Grupo OSDBA.....10, 14, 35

I

Identificadores de Base de Datos 21
Índice agrupado.....67
Índice compuesto 66
Índice único 66
Índices.....62
índices agrupados.....66
Índices compuestos, índices únicos e índices agrupados.....66
índices únicos e índices agrupados.....66
Inicio de oracle 12c.....41
INNER JOIN ... 176, 177, 178, 179, 180, 181
Insertar todas las columnas.....79
Insertar una fila mediante INSERT INTO...VALUES.....77
Insertar valores NULL 78
Insertar varias filas mediante INSERT INTO...SELECT 80
Instalación de Base de Datos de Instancia Única 8
Instalación de Base de Datos de Oracle Real Application Clusters.....8
Instalación de Base de Datos Oracle RAC OneNode 8
Instalar Tipo de Base de Datos de Cluster (RAC).....8
Integridad 58
integridad de datos.....58
Integridad de datos 49
Integridad de entidad 59
Integridad referencial 59

J

JOIN con condiciones de búsqueda sin igualdad.....173
JOIN de dos tablas 166
JOIN EXTERNO completo 175
JOIN EXTERNO izquierdo y derecho.....176
JOIN INTERNO.....174

© Alfaomega - RC Libros

SYNAPS PREMIUM

JOINS con criterios de selección de filas 169
JOINS con múltiples columnas de emparejamiento.....170
Joins de tres o más tablas.....170
Joins internos y externos 174
Juego de Caracteres (Clase de Escritorio, Sólo para Instalación Típica).....10, 14

L

LEFT JOIN 176, 178, 179, 180, 181
Lenguaje de control de datos 123
Lenguaje de Control de Transacciones ..93, 123
Lenguaje de Definición de Datos 93, 123
lenguaje de modificación de datos 76, 123, 124
Lenguaje SQL 46

M

Manipulación de datos 48, 56
modelo de datos.....45, 54
Modelo relacional.....45
múltiples columnas de agrupamiento ..194

N

Nombre de la Base de Datos Global 10, 14, 21

O

Objetivos esenciales de SQL 49
opción Desconectar de base de datos....42
Opciones de Almacenamiento de Base de Datos 27
Opciones de instalación.....5
Opciones de Instalación de Grid 8
Opciones de Recuperación 30
Opciones de Redundancia 29
Operador concatenación 110
Operadores.....109
Operadores aritméticos.....109
Operadores de comparación 110
Operadores de conjuntos 111
Operadores lógicos.....113
Orden de prioridad de los operadores .113
Ordenación de los resultados de una consulta 135

P

Proceso de instalación de Oracle 12c 1

R

Recuperación de datos 48
Recuperación Rápida 30, 31, 32
reglas de Codd 55, 57
Reglas de las subconsultas.....211
relación recursiva 73
Renombrar columnas 139
Restricciones de integridad 56
Resumir datos.....197
Resumir datos con CUBE 198
Resumir datos con ROLLUP 200
RIGHT JOIN.....178, 179, 180, 181

S

Selección de filas 133
Selección de todas las tablas de una instancia de base de datos 132
Seleccionar Lista de Nodos 9
Seleccionar Tipo de Instalación 9
Sentencia INSERT.....76
Subconsultas.....207
Subconsultas con ANY, SOME o ALL 216
Subconsultas con EXISTS.....214
Subconsultas con IN.....212
Subconsultas con múltiples niveles de anidamiento 218
Subconsultas con operadores de comparación.....212
Subconsultas correlacionadas 218
Subconsultas correlacionadas con alias219
Subconsultas correlacionadas con operadores de comparación 220
Subconsultas correlacionadas con una cláusula HAVING.....220
Subconsultas usadas en cualquier lugar de una expresión.....217

© Alfaomega - RC Libros

SYNAPS PREMIUM

T

Tipo de Configuración.....20
Tipos de datos.... 67, 68, 69, 70, 71, 72, 74, 77, 82, 95, 100, 101, 102, 104, 105, 106, 107, 151, 186, 236
Tipos de datos ANSI, DB2 y SQL/DS.....102
Tipos de datos en Oracle 12c SQL.....99
TRANSLATE y DECODE 237
Tratamiento de filas duplicadas con DISTINCT.....191

U

Ubicación de Archivo de Base de Datos 10, 12, 27
Ubicación del Software.....10, 12, 18
UNION, INTERSECT y MINUS en subconsultas.....221
Uso avanzado de funciones y variables 222

V

Valores null en las funciones de agregado 190
ventajas de los índices.....64

W

WHERE.....195

© Alfaomega - RC Libros

SYNAPS PREMIUM