Práctica 3. Desarrollo de bases de datos con ORACLE™

3.1 Introducción a Oracle

Comencemos con una introducción a Oracle. Para ello, hablaremos de

- El servidor de Oracle
- Estructuras de la base de datos y Gestión del espacio
- Estructuras de memoria y procesos
- Concurrencia y consistencia
- Las operaciones de iniciar y parar la base de datos
- Seguridad de la base de datos
- Copias de seguridad y Restauración de copias

3.1.1 El servidor de Oracle

Un Servidor Oracle es un Sistema de Gestión de Bases de Datos Relacionales que proporciona un enfoque abierto, comprensible e integrado para la gestión de información. Un Servidor Oracle consta de una base de datos Oracle y de una instancia del Servidor Oracle. Más adelante vamos a ver cuál es la relación entre la base de datos y la instancia. El modo básico de trabajo con ORACLE está basado en SQL. Además de SQL, Oracle cuenta con un lenguaje procedural, denominado PL/SQL, en el que se pueden programar sentencias SQL mediante el uso de estructuras de control de flujo, utilización de variables, definición de procedimientos, y en general, características propias de los lenguajes de programación.

Una base de datos Oracle consta de una estructura lógica y de una estructura física. Como se trata de dos estructuras separadas, el almacenamiento puede gestionarse de forma independiente sin afectar al acceso a las estructuras lógicas de los datos, lo que se conoce como independencia física.

La estructura de la base de datos física (nivel físico) viene determinado por los archivos del sistema operativo que forman la base de datos. Cada base de datos Oracle está formada por tres tipos de archivos: uno o más *archivos de datos*, dos o más *archivos de registro de operaciones* (log), y uno o más *archivos de control*. Estos archivos proporcionan el almacenamiento físico para una base de datos Oracle.

La estructura lógica de una base de datos viene determinada por:

- Uno o más *tablespaces* (áreas lógicas de almacenamiento)
- Los objetos del esquema de la base de datos. Un esquema es una colección de objetos. Los objetos del esquema son las estructuras lógicas que hacen referencia directa a los datos de la base de datos. Ejemplos de objetos del esquema son estructuras como las tablas, vistas, secuencias, procedimientos guardados, sinónimos, índices y clústers.

Son estas estructuras lógicas las que determinan cómo se utiliza en espacio físico de una base de datos. Así pues, estos objetos del esquema y las relaciones entre ellos, son los que forman el diseño de una base de datos relacional.

Cada vez que se inicia una base de datos, se reserva espacio para un área global de sistema (SGA, S*ystem Global Area*) y se inician los procesos background de Oracle. El área global del sistema es un área de memoria utilizada para guardar la información de la base de datos compartida por los usuarios de la base de datos. A la combinación de procesos de background y de búfers de memoria se le denomina *instancia* de Oracle.

Una instancia Oracle tiene dos tipos de procesos: procesos de usuario y procesos de Oracle.

- Un proceso de usuario ejecuta el código de un programa de aplicación (p.e. una consulta).
- Los procesos de Oracle son procesos del servidor que realizan el trabajo necesario para responder a los procesos de usuario, y también son los encargados de realizar las tareas de mantenimiento del Servidor Oracle

En la figura siguiente se ilustra una instancia de Oracle con varios procesos

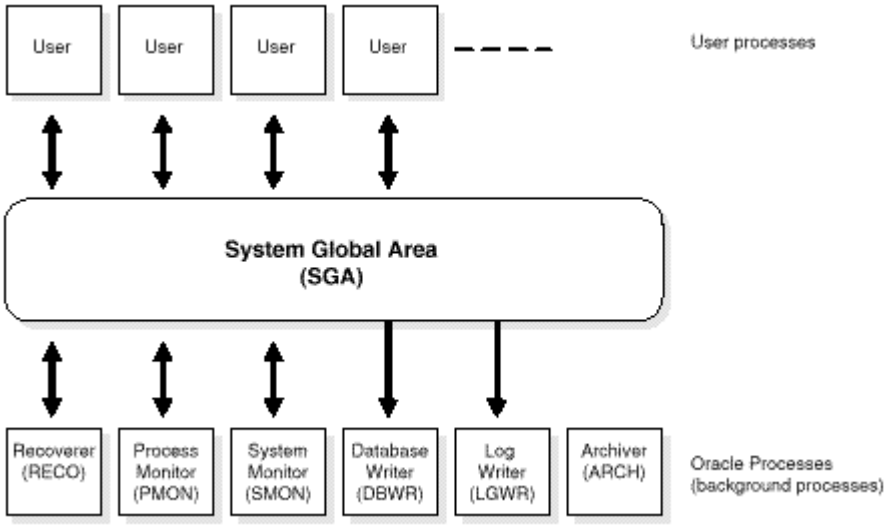

Figura 3.1. Una instancia de Oracle.

Si los procesos de usuario y los procesos del servidor se ejecutan en máquinas distintas dentro de una red, dichos procesos se comunican utilizando Net8. Net8 es una

interfaz para protocolos de comunicaciones estándar que permite la transmisión de datos entre computadoras.

Una base de datos Oracle es una colección de datos que es tratado como una unidad. El propósito general de una base de datos es el de permitir guardar y recuperar información.

Una base de datos puede estar abierta (accesible) o cerrada (no accesible). Lo normal es que la base de datos esté abierta y pueda ser utilizada. En cambio, hay situaciones en las que debido a situaciones de administración concretas, es necesario que los datos de la base de datos no estén disponibles para los usuarios.

3.1.2. Estructura de una base de datos Oracle y gestión del espacio

A continuación veremos la arquitectura de una base de datos Oracle, incluyendo las estructuras físicas y lógicas que forman una base de datos.

Una base de datos Oracle es una colección de datos que es tratada como una unidad. El propósito de una base de datos es permitir guardar y recuperar información. La base de datos tiene estructuras lógicas y físicas.

3.1.2.1. Estructuras lógicas de la base de datos

A continuación explicaremos las estructuras lógicas de la base de datos, como son los tablespaces, los objetos del esquema, los bloques de datos, las extensiones y los segmentos.

Una base de datos está dividida en unidades lógicas de almacenamiento denominadas *tablespaces*. Un tablespace se utiliza para agrupar estructuras lógicas relacionadas. Por ejemplo, podemos tener un tablespace para la agrupación de todos los objetos de una aplicación, de forma que se puedan simplificar las tareas de administración.

A continuación se ilustra la relación entre bases de datos, tablespaces y archivos de datos.

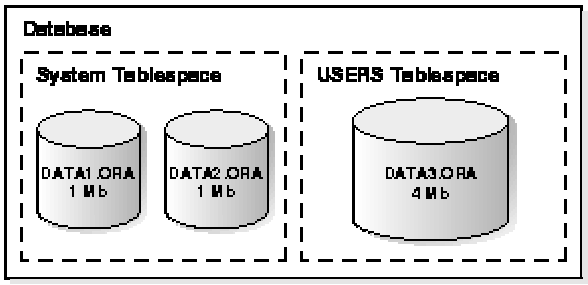

Figura 3.2. Bases de datos, tablespaces y archivos de datos.

En esta figura podemos ver que:

- Cada base de datos está dividida en uno o más tablespaces.
- Para cada tablespace se crea uno o más archivos de datos, de forma que almacenen físicamente los datos de todas las estructuras lógicas de un tablespace.

- El tamaño de un tablespace es la suma de los tamaños de los archivos de datos del tablespace (el tablespace SYSTEM tiene 2 MB, mientras que el tablespace USERS tiene 4 MB).
- El tamaño de la base de datos es la suma de los tamaños de los tablespaces (6 MB).

Un tablespace puede estar accesible o no. Normalmente están accesibles, de forma que los usuarios pueden acceder a la información del tablespace. En cambio, hay situaciones en las que hay que hacer que un tablespace no esté disponible, haciendo que una parte de la base de datos no esté disponible, mientras que se sigue permitiendo el acceso normal al resto de la base de datos. Esto se realiza para facilitar las tareas de administración de la base de datos.

Un esquema es una colección de objetos. Los objetos del esquema son estructuras lógicas que hacen referencia directamente a los datos de la base de datos. Ejemplos de objetos del esquema son estructuras tales como las tablas, vistas, secuencias, procedimientos guardados, sinónimos, índices y clusters. (No hay relación directa entre un tablespace y un esquema. Los objetos de un mismo esquema pueden estar en distintos tablespaces, y un tablespace puede tener objetos de distintos esquemas).

3.1.2.2. Bloques de datos, extensiones y segmentos

Oracle permite la utilización eficiente de las estructuras lógicas de datos mediante distintos niveles de granularidad. Para ello, Oracle utiliza los bloques de datos, las extensiones y los segmentos.

Al nivel más detallado, los datos de la base de datos se guardan es bloques de datos. Un bloque de datos se corresponde con un número de bytes del nivel físico. Al crear la base de datos se utiliza un tamaño de bloque de datos por omisión.

El siguiente nivel de detalle para espacios se denomina extensión. Una extensión es un número de bloques de datos contiguos, y se utilizan para guardar un tipo de información específico.

El siguiente nivel de detalle para el almacenamiento lógico en una base de datos se corresponde con los segmentos. Un segmento es un conjunto de extensiones asignado a una estructura lógica concreta. Podemos distinguir varios tipos de segmentos.

- Segmentos de datos. Todos los datos de las tablas se guardan en las extensiones de su segmento de datos. Cada clúster tiene un segmento de datos.
- Segmentos de índices. Cada índice tiene un segmento de índice que guarda todos sus datos.
- Segmentos de Rollback. El Administrador de la base de datos crea varios segmentos de rollback, de forma que se pueda guardar temporalmente la base de datos. Esto permite deshacer operaciones.

- Segmentos temporales. Estos segmentos son creados por Oracle cuando las sentencias SQL necesitan un espacio temporal para completar su ejecución. Una vez finalizada la ejecución, se devuelven las extensiones de los segmentos temporales para que puedan ser utilizados de nuevo.

3.1.2.3. Estructuras físicas de la base de datos

A continuación explicaremos las estructuras físicas de una base de datos Oracle, como son los archivos de datos, los archivos de registro de operaciones, y los archivos de control.

Cada base de datos tiene uno o más archivos físicos de datos. Los archivos de datos contienen todos los datos de la base de datos. Los datos de las estructuras lógicas de la base de datos, como son las tablas y los índices, se guardan físicamente en los archivos de datos de la base de datos. Estos archivos de datos tienen estas características:

- Un archivo de datos sólo puede estar asociado a una base de datos.
- Los archivos de la base de datos tienen propiedades configuradas de forma que puedan ampliarse cuando la base de datos se quede sin espacio.
- Uno o más archivos de datos forman una unidad de almacenamiento lógica denominada tablespace.

Para mejorar el rendimiento de la base de datos, los datos que se leen de los archivos de datos se guardan en la memoria caché de Oracle. De esta forma cuando un usuario solicita acceder a ciertos datos, si estos no están en la caché, son leídos de los archivos de datos correspondientes y posteriormente guardados en la caché.

En cuanto a la escritura esto no ocurre exactamente igual. Los datos nuevos o modificados no siempre se escriben directamente en los archivos de datos. De esta forma se reduce el número de accesos a disco. Lo que ocurre es que los datos se colocan en memoria y se escriben de una vez mediante un proceso background de escritura de Oracle.

3.1.2.4. Archivos de registro de operaciones

Cada base de datos Oracle tiene dos o más archivos de registro de operaciones. La función principal de estos archivos es el de guardar todos los cambios realizados sobre los datos. De esta forma, si ocurre un error al escribir en los archivos de datos, los cambios se pueden recuperar de los archivos de registro de operaciones.

Los archivos de registro de operaciones son algo crítico en la protección de errores de la base de datos. Esta protección implica también a los propios archivos de registro de operaciones. Para ello, Oracle dispone de varias copias de estos archivos (incluso en discos distintos), de forma que si uno falla, se utilizaría el otro.

La información de un archivo de registro de operaciones sólo se utiliza para recuperar la base de datos que ha sufrido un error de dispositivo o de sistema. Por ejemplo, si ocurre un corte de tensión en un sistema convencional, los datos de la

memoria que aún no hayan sido guardados en los archivos de datos se perderían. Sin embargo, esta situación se puede solucionar una vez que se vuelva a abrir la base de datos. Concretamente, Oracle examina los archivos de registro de operaciones más recientes, y restaura la base de datos al estado en el que ocurrió el error.

3.1.2.5. Archivos de control

Cada base de datos Oracle tiene un archivo de control. Un archivo de control contiene información específica sobre la estructura física de la base de datos. Por ejemplo, contiene el nombre de la base de datos, los nombre y rutas de acceso de los archivos de datos y de los archivos de registro de operaciones, y la fecha de creación de la base de datos.

Sobre estos archivos de control también se utiliza la política de seguridad de los archivos de registro de operaciones.

Cada vez que se inicia una instancia de una base de datos Oracle, se utiliza su archivo de control para identificar a la base de datos y a los archivos de registro de operaciones que se tienen que abrir, de forma que pueda continuar el funcionamiento seguro de la base de datos. Si se realiza una modificación del esquema o de las estructuras de la base de datos, estas modificaciones se guardarán automáticamente en el archivo de control.

3.1.3. Estructuras de memoria y procesos

Ahora hablaremos de las estructuras de memoria y de los procesos utilizados por un servidor de Oracle para la gestión de una base de datos. En concreto, hablaremos de las características de la arquitectura de Oracle que permiten:

- el acceso concurrente de varios usuarios a una base de datos
- ofrecer un alto rendimiento para los sistemas de bases de datos multiusuario y multiaplicación

Un servidor Oracle utiliza estructuras de memoria y procesos para gestionar y acceder a la base de datos. Todas las estructuras de memoria que existen en la base de datos están en la memoria principal de las computadoras que constituyen el sistema de base de datos. Los procesos son los trabajos o tareas que trabajan en la memoria de estas computadoras.

3.1.3.1. Estructuras de memoria

Oracle crea y utiliza estructuras de memoria para realizar varios trabajos. Por ejemplo, se utiliza la memoria para guardar el código de programa que se está ejecutando y para guardar los datos compartidos por varios usuarios. Oracle tiene asociadas varias estructuras básicas de memoria: el área global del sistema (que incluye a los búfers de la base de datos, los búfers de registro de operaciones, y la zona de paginación compartida) y las áreas globales de programa.

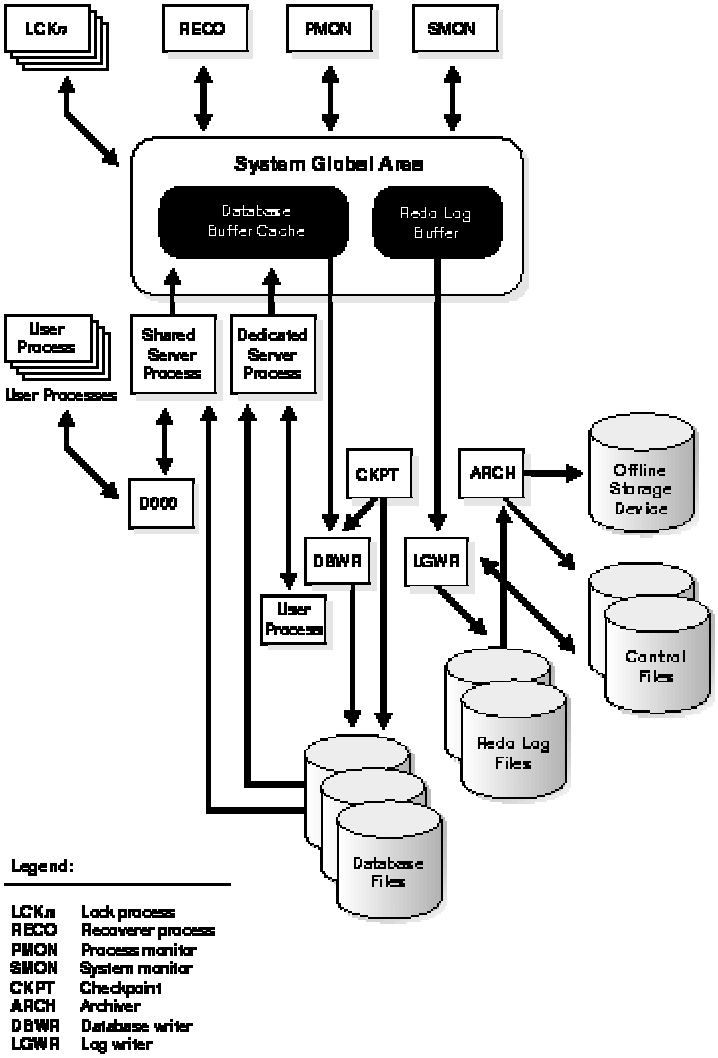

Figura 3.3. Estructuras de memoria y Procesos de Oracle.

3.1.3.2. Área Global del Sistema (SGA)

El área global del sistema (SGA) es una región de memoria compartida que contiene información sobre datos y control de una instancia Oracle. Un SGA y los procesos background de Oracle forman una instancia de Oracle.

Oracle reserva el área global del sistema cuando se inicia una instancia, y la libera cuando se finaliza la instancia. Cada instancia tiene su propio SGA, y los usuarios que estén conectados a un servidor Oracle comparten los datos en el SGA.

La información que se guarda en el SGA se guarda en distintos tipos de estructuras de datos, como son las que vamos a ver a continuación. Estas estructuras de datos se definen al crear la base de datos, y se hace con un tamaño por omisión.

- Búfer caché de la base de datos. Guardan los bloques usados más recientemente (o más frecuentemente). Esto reduce el número de accesos a disco aumentando el rendimiento de la base de datos.
- Búfer de registro de operaciones. Guarda un registro de las modificaciones realizadas sobre la base de datos. Estos búfers se guardan

en un archivo de registro de operaciones, que son los que se utilizan para la recuperación de bases de datos.

- Zona de paginación compartida. Se trata de una parte del SGA que contiene zonas de memoria compartida, como son las zonas SQL compartidas (contienen el árbol de ejecución de una consulta y permiten la reutilización)

3.1.3.3. Cursores

Un cursor es un puntero para la memoria asociada a una sentencia concreta. Estos cursores suelen utilizarse cuando se quiere controlar las fases de ejecución de una sentencia SQL con la intención de aumentar el rendimiento.

3.1.3.4. Área Global de Programa (PGA)

El área global de programa es un búfer de memoria que contiene información de datos y control para un proceso del servidor. Oracle crea un PGA cada vez que se inicia un proceso del servidor.

3.1.3.5. Arquitectura de procesos

Un proceso es un "hilo de control" o un mecanismo de un sistema operativo que puede ejecutar una serie de pasos. Algunos sistemas operativos utilizan los términos de trabajo o tarea. Normalmente, un proceso tiene su propia área de memoria privada sobre la que se ejecuta.

Un servidor Oracle tiene dos tipos de procesos: procesos de usuario y procesos de Oracle.

- Procesos de usuario. Un proceso de usuario es creado para ejecutar el código de un programa de aplicación. El proceso de usuarios también gestiona la comunicación con los procesos del servidor.
- Procesos del servidor. Se trata de procesos creados por Oracle para gestionar las peticiones de los procesos de los usuarios conectados. Un proceso de servidor se encarga de la comunicación con los procesos de usuario y de la interacción con Oracle para llevar a cabo las peticiones de los procesos de usuario. Por ejemplo, si un usuario solicita datos que no se encuentran en los búfers del SGA de la base de datos, el proceso de servidor adecuado lee los bloques de datos apropiados de los archivos de datos y los coloca en el SGA.

3.1.3.6. Procesos background

Oracle crea un conjunto de procesos background para cada instancia. Se trata de procesos internos que Oracle realiza en segundo plano (asíncronamente) para aumentar el rendimiento de la base de datos.

Un SGA y un conjunto de procesos background de Oracle forman una instancia de Oracle. Cada instancia de Oracle puede utilizar varios procesos background. Estos

son los procesos DBWR, LGWR, CKPT, SMON, PMON, ARCH, RECO, Dnnn, y LCKn.

- Database Writer (DBWR. Escritor de la base de datos). Escribe en los archivos de datos los bloques modificados a partir del búfer de la base de datos.
- Log Writer (LGWR. Escritor del registro). Escribe en disco las entradas del registro de operaciones.
- Checkpoint (CKPT). A menudo hay que escribir (DBWR) en los archivos de datos todos los búfers modificados del SGA. Este momento es del "checkpoint". El proceso Checkpoint es el encargado de llamar al DBWR indicándole que actualice los archivos de datos y de control.
- System Monitor (SMON. Monitor del sistema). Realiza la recuperación de instancias durante el inicio de una instancia. SMON también borra segmentos temporales que ya no se utilizan y recupera transacciones detenidas debido a algún tipo de error del sistema.
- Process Monitor (PMON. Monitor de procesos). Recupera procesos cuando falle un proceso de usuario. PMON es el encargado de la limpieza de la caché y de la liberalización de los recursos utilizados por un proceso.
- Archiver (ARCH. Archivador). Copia los búfers de registro en los archivos de registro.
- Recoverer (RECO. Recuperador). Resuelve transacciones distribuidas pendientes debidos a errores de red o de sistema. Actúa de forma temporizada, y conecta las bases de datos remotas para confirmar o deshacer transacciones pendientes.
- Dispatcher (Dnnn). Son procesos background optativos y son los responsables de enrutar las peticiones a los procesos de servidor compartidos y devolver las repuestas a los procesos de usuario correspondientes.
- Lock (LCKn). Bloquean inter-instancias en servidores paralelos.

3.1.3.7. Ejemplo de funcionamiento de Oracle

Este ejemplo ilustra el funcionamiento de Oracle en una situación determinada en la que los procesos de usuario y del servidor están en máquinas distintas (conectadas a través de una red).

1. Se está ejecutando una instancia en el servidor Oracle.
2. Una estación cliente que ejecuta una aplicación en un proceso de usuario. La aplicación cliente intenta conectarse al servidor utilizando el controlador Net8 apropiado.

3. El servidor está ejecutando el controlador Net8 apropiado. El servidor detecta la petición de conexión de la aplicación y crea un proceso de servidor dedicado para atender al proceso de usuario.
4. El usuario ejecuta y valida una sentencia SQL. Por ejemplo, el usuario cambia el nombre de una columna en una tabla.
5. El proceso del servidor recibe la sentencia y comprueba el área de paginación compartida para ver si el área SQL contiene una sentencia SQL idéntica. Si se encuentra alguna, el proceso del servidor comprueba los privilegios del usuario y se utiliza el área SQL compartida para procesar la sentencia. Si no se encuentra una sentencia SQL idéntica se reserva un nuevo área para la sentencia para ser analizada y procesada.
6. El proceso del servidor recupera los datos del archivo de datos actual (tabla) o del SGA.
7. El proceso del servidor modifica los datos en el SGA. El proceso DBWR escribe los bloques modificados en el disco cuando sea conveniente. Como la transacción está validada, el proceso LGWR guarda esta operación en el archivo de registro de transacciones.
8. Si la transacción tiene éxito, el proceso del servidor envía un mensaje a la aplicación a través de la red. Si no tiene éxito se envía un mensaje de error por la red.
9. A lo largo de este procedimiento, también se ejecutan los otros procesos background, vigilando las condiciones que necesitan una intervención. Además, el servidor de la base de datos gestiona transacciones de otros usuarios y previene la contención entre transacciones que trabajen con los mismos datos.

Estos pasos sólo han tratado de describir las operaciones más básicas que realiza Oracle.

3.1.4. Concurrencia y Consistencia

A continuación hablaremos de los mecanismos que utiliza Oracle para conseguir los siguientes objetivos clave de un sistema de gestión de información:

- Los datos tienen que ser leídos y modificados de forma consistente
- Hay que maximizar la concurrencia en un entorno multiusuario
- Se necesita un alto rendimiento para maximizar la productividad de los usuarios de un sistema de base de datos

3.1.4.1. Concurrencia

Una cuestión fundamental en un sistema de gestión de bases de datos multiusuario es cómo se realiza el control de la concurrencia, o lo que es lo mismo, el acceso simultáneo a los mismos datos por parte de distintos usuarios. Sin los controles de concurrencia

adecuados, los datos podrían ser modificados de forma incorrecta, violando la integridad de los datos.

Si varios usuarios están accediendo a los mismos datos, una forma de gestionar la concurrencia de los datos es hacer que cada usuario espere a su turno. El objetivo de un SGBD es reducir esta espera, de forma que no exista o que sea inapreciable para los usuarios. Todas las sentencias en DML deben ejecutarse con la menor interferencia posible, sin sacrificar el rendimiento ni la integridad de los datos.

Oracle resuelve estas cuestiones mediante varios tipos de bloqueo y un modelo de consistencia multiversión, de los que hablaremos más adelante. Estas características están basadas en el concepto de transacción.

3.1.4.2. Consistencia de lectura

La consistencia de lectura de Oracle hace lo siguiente:

- Garantiza que el conjunto de datos vistos por una sentencia sea consistente y con respecto a unidad de tiempo atómica, y no cambia durante la ejecución de la sentencia (*consistencia a nivel de sentencia*).
- Asegura que los lectores de los datos de la base de datos no esperarán a escritores u otros lectores de los mismos datos.
- Asegura que los escritores de los datos de la base de datos no esperarán a lectores de los mismos datos.
- Asegura que los escritores sólo esperarán a otros escritores si intentan modificar las mismas filas en transacciones concurrentes.

La forma más sencilla de pensar en la implementación de Oracle para la consistencia de lectura es imaginar a cada usuario trabajando con su propia copia de la base de datos, de aquí lo del modelo de consistencia multiversión.

Para gestionar el modelo de consistencia multiversión, Oracle tiene que crear un conjunto de datos consistentes de lectura cuando se consulte y se actualice una tabla. Cuando se produce una actualización, se modifican los datos originales y se guardan en los segmentos de rollback. Mientras esta actualización siga siendo parte de una transacción no validada, cualquier usuario que consulte dichos datos, sólo verá los datos originales. Esto quiere decir que Oracle utiliza la información actual del SGA y la información de los segmentos de rollback para construir una vista de una tabla consistente de lectura.

Sólo cuando se valida una transacción, los cambios se hacen permanentes, y por tanto las sentencias siguientes a la validación de una transacción, sólo verán los cambios realizados por la transacción validada.

Por omisión, Oracle garantiza la consistencia de lectura a nivel de sentencia. En cambio, a veces es necesario la consistencia de lectura a nivel de transacción. Es decir, poder ejecutar varias consultas dentro de una transacción, de forma que todas las consultas no vean los efectos de otras transacciones que estén ejecutándose concurrentemente.

3.1.4.3. Mecanismos de bloqueo

Oracle también utiliza bloqueos para controlar el acceso concurrente a los datos. Los bloqueos son mecanismos que tratan de prevenir la interacción destructiva entre usuarios que están accediendo a los datos.

Los bloqueos se utilizan para conseguir dos objetivos importantes de las bases de datos:

- Consistencia. Asegura que los datos que ve un usuario no van a ser modificados por otros usuarios hasta que no termine con ellos.
- Integridad. Asegura que los datos de la base de datos reflejan todos los cambios realizados en la secuencia correcta.

3.1.4.4. Bloqueo automático

El bloqueo de Oracle se realiza de forma automática y no necesita intervención del usuario. El gestor de bloqueo de Oracle bloquea los datos de la tabla a nivel de fila, con lo que se minimiza la contención.

El mecanismo de bloqueo de Oracle permite que se consulten datos bloqueados, pero prohibe la modificación de dichos datos.

Si el usuario desea evitar el bloqueo automático a nivel de fila, Oracle permite el bloqueo manual a nivel de fila y a nivel de tabla. Esta es la situación de consultar registros que posteriormente van a ser actualizados.

3.1.5. Las operaciones de iniciar y parar la base de datos

Las bases de datos de Oracle no están disponibles hasta que no se haya iniciado el servidor de Oracle y hasta que no se haya abierto la base de datos. Estas operaciones tienen que ser realizadas por el administrador de la base de datos.

El proceso de iniciar una base de datos consta de tres pasos:

1. Iniciar una instancia del servidor de Oracle
2. Montar la base de datos
3. Abrir la base de datos

Al iniciarse el servidor de Oracle, se utiliza un archivo de parámetros que contiene parámetros de inicialización. Estos parámetros especifican el nombre de la base de datos, la cantidad de memoria que hay que reservar, los nombre de los archivos de control, y otros parámetros de este tipo.

El proceso de parada de una base de datos a la que estamos conectados consta de tres pasos:

1. Cerrar la base de datos
2. Desmontar la base de datos
3. Parar la instancia del servidor de Oracle

Cuando se cierra una base de datos, Oracle realiza estos tres pasos de forma automática.

3.1.6. Seguridad de la base de datos

Los sistemas de bases de datos multiusuario, como es el caso de Oracle, disponen de mecanismos de seguridad que controlan el acceso y el uso de la base de datos. Por ejemplo, estos mecanismos hacen lo siguiente:

- Previenen accesos no autorizados a la base de datos
- Previenen el acceso no autorizado a objetos del esquema
- Controlan la utilización del disco
- Controlan el uso de los recursos del sistema (como el tiempo de CPU)
- Realizan una auditoría de las acciones de los usuarios.

A cada usuario de la base de datos se le asocia un esquema con el mismo nombre. Un esquema es una colección lógica de objetos (tablas, vistas, índices, funciones, ...). Por omisión, cada usuario de la base de datos crea y tiene acceso a todos los objetos de su esquema.

La seguridad de la base de datos puede ser de dos tipos: la seguridad del sistema y la seguridad de los datos.

La seguridad del sistema incluye mecanismos que controlan el acceso y la utilización de la base de datos a nivel de sistema. Por ejemplo, la seguridad del sistema incluye:

- Combinaciones válidas usuario/contraseña
- Cantidad de espacio de disco disponible para los objetos de un usuario
- Recursos de un usuario

Los mecanismos de seguridad comprueban:

- si un usuario está autorizado para conectarse a la base de datos
- si está activada la auditoría de la base de datos
- las operaciones que puede realizar un usuario

La seguridad de los datos incluye a los mecanismos que controlan el acceso y el uso de la base de datos a nivel de objetos. Por ejemplo, la seguridad de datos comprueba:

- qué usuarios tiene acceso a objetos específicos del sistema y los tipos de acciones que se pueden realizar sobre el objeto (p.e. sólo lectura)
- las acciones que son auditadas por cada objeto del esquema

3.1.6.1. Mecanismos de seguridad

El servidor de Oracle proporciona un control de acceso discrecional, que es una forma de restringir el acceso mediante privilegios. Para que un usuario pueda realizar una operación sobre un objeto, dicho usuario tiene que tener el privilegio correspondiente para dicha operación sobre dicho objeto. La concesión de estos privilegios la realizan cierto tipo de usuarios privilegiados a su "antojo". Por esto se trata de un control de acceso "discrecional"

Oracle proporciona distintas posibilidades para gestionar la seguridad de la base de datos:

- usuarios y esquemas de la base de datos
- privilegios
- roles
- cuotas de almacenamiento
- limitación de recursos
- auditoría

La figura siguiente ilustra la relación entre las distintas posibilidades de seguridad de Oracle.

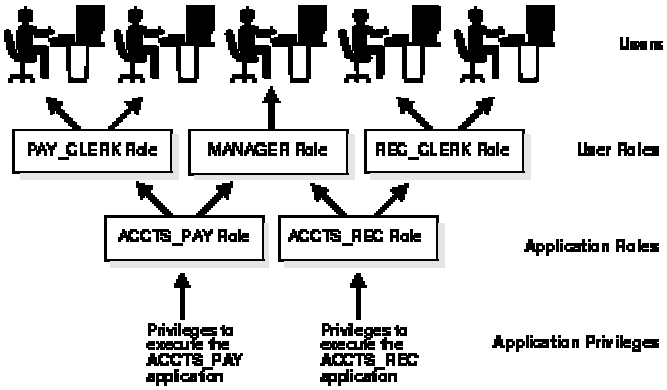

Figura 3.4. Posibilidades de seguridad de Oracle.

A continuación comentaremos cada una de estas posibilidades de seguridad.

3.1.6.2. Usuarios y esquemas de la base de datos

Cada base de datos Oracle tiene una lista de usuarios. Para acceder a una base de datos, un usuario tiene que utilizar una aplicación de la base de datos e intentar realizar una conexión a la base de datos con un nombre de usuario válido para dicha base de datos. Además, cada nombre de usuario tiene una contraseña asociada para prevenir usos no autorizados.

Cada usuario tiene un *dominio de seguridad*, es decir, un conjunto de propiedades que determinan aspectos tales como:

- acciones (privilegios y roles) permitidas al usuario

- cuotas de tablespaces (espacio de disco disponible) para el usuario
- limitación de recursos del sistema (p.e. tiempo de CPU) para el usuario

3.1.6.3. Privilegios

Un privilegio es un derecho a ejecutar un tipo de sentencia SQL. Ejemplos de privilegios son:

- conectarse a la base de datos (crear una sesión)
- crear una tabla en su propio esquema
- seleccionar filas de tablas de otro usuario
- ejecutar algún procedimiento almacenado

Los privilegios de una base de datos Oracle se pueden clasificar en privilegios de sistema y privilegios sobre objetos.

- Privilegios de sistema. Permiten a los usuarios realizar una acción en el sistema o una acción concreta en un tipo particular de objeto. Por ejemplo, los privilegios para crear un tablespace o para eliminar una tabla de la base de datos son privilegios de sistema. Estos privilegios suelen estar asignados sólo a administradores y a programadores, ya que se trata de operaciones muy delicadas.
- Privilegios sobre objetos. Permiten a los usuarios realizar acciones concretas sobre un objeto específico del esquema. Por ejemplo, el privilegio de borrar filas de una tabla es un privilegio sobre objetos. Los privilegios sobre objetos se asignan a los usuarios finales de forma que puedan realizar sus acciones permitidas sobre la base de datos.

Los privilegios son concedidos a los usuarios de forma que puedan acceder y modificar los datos de la base de datos. Estos privilegios se pueden conceder de dos formas distintas:

- Concesión explícita de privilegios. Por ejemplo, el privilegio de insertar registros a la tabla EMP está concedido explícitamente a SCOTT.
- Concesión de privilegios a través de roles (un grupo de privilegios), y estos roles pueden ser concedidos a los usuarios. Por ejemplo, el privilegio de insertar registros en la tabla EMP ha sido concedido al rol CLERK (ORDENANZA), que ha sido concedido a los usuarios SCOTT y BRIAN.

Como los privilegios mediante roles son más sencillos de utilizar, los privilegios se suelen conceder mediante roles, en lugar de mediante la concesión de privilegios a usuarios específicos.

Oracle proporciona los roles como un mecanismo ágil para facilitar la concesión de privilegios. Los roles son grupos de privilegios que se pueden conceder a usuarios o a otros roles. A continuación se muestran propiedades de los roles que facilitan la gestión de privilegios:

- Reducción de la concesión de privilegios. En lugar de conceder explícitamente el mismo conjunto de privilegios a varios usuarios, el administrador de la base de datos puede conceder los privilegios para un grupo de usuarios a través de un rol. Lo que hace el administrador es conceder el rol a cada miembro de dicho grupo.
- Gestión dinámica de privilegios. Cuando hay que modificar los privilegios de un grupo, sólo hay que modificar los privilegios del rol. Los dominios de seguridad de todos los usuarios con dicho rol reflejarán los cambios realizados sobre el rol
- Disponibilidad selectiva de privilegios. Los roles concedidos a un usuario se pueden activar o desactivar de forma selectiva. Esto permite controlar específicamente a un usuario en una situación concreta.
- Conocimiento de la aplicación. Se puede diseñar una aplicación para activar o desactivar los roles de forma automática cuando los usuarios intenten utilizar la aplicación.

3.1.6.4. Cuotas de almacenamiento

Oracle dispone de métodos para controlar y limitar el uso del espacio de disco asignado a cada usuario, incluyendo cuotas para tablespaces por omisión y temporales.

- Tablespace por omisión. Cada usuario tiene asignado un tablespace por omisión. Cuando un usuario crea una tabla, un índice o un clúster, y no especifica un tablespace para dicho objeto, se utiliza el tablespace por omisión. (Por supuesto, el usuario tiene que tener permiso para poder crear dicho objeto).
- Tablespace temporal. Cada usuario tiene un tablespace temporal. Cuando un usuario ejecuta una sentencia SQL que requiere la creación de segmentos temporales (p.e. la creación de un índice), se utiliza el tablespace temporal del usuario.

Oracle puede limitar la cantidad de espacio de disco disponible para los objetos de un esquema. Se pueden definir cuotas (límites de espacio) para cada tablespace del usuario. Estas cuotas permiten controlar la cantidad de espacio de disco consumida por los objetos de un usuario.

3.1.6.5. Perfiles y limitación de recursos

Cada usuario está asignado a un perfil que especifica las limitaciones sobre distintos recursos del sistema disponibles para el usuario incluyendo

- número de sesiones concurrentes
- tiempo de CPU
    - Por sesión
    - Por sentencias SQL

- Cantidad de operaciones E/S lógicas
    - Por sesión
    - Por sentencias SQL
- Cantidad de tiempo inactivo por sesión
- Cantidad de tiempo de conexión
- Restricciones de contraseña
    - Bloqueo de cuenta después de varios intentos fallidos
    - Caducidad de la contraseña
    - Reutilización de la contraseña y restricciones de complejidad

Se pueden crear distintos perfiles y asignarlos a cada uno de los usuarios de la base de datos. Si a un usuario no se le asigna ningún perfil, se le asignará un perfil por omisión.

3.1.6.6. Auditoría

Oracle permite la auditoría (registro) selectiva de las acciones de los usuarios para ayudar a analizar usos extraños de la base de datos. La auditoría se puede realizar a tres niveles: a nivel de sentencia, a nivel de privilegios y a nivel de objetos.

- Auditoría de sentencias. Consiste en registrar ciertas sentencias SQL, y se puede realizar para todos los usuarios o para algunos en concreto. Por ejemplo, registrar las conexiones y desconexiones de la base de datos de SCOTT.
- Auditoría de privilegios. Consiste en registrar el uso de privilegios delicados, y se puede realizar para todos los usuarios o para algunos en concreto.
- Auditoría de objetos. Consiste en registrar los accesos a objetos concretos del esquema.

Para todos los tipos de auditoría, Oracle permite registrar de forma selectiva las sentencias que se han ejecutado con éxito, las que han fallado, o ambas. Esto permite analizar sentencias malintencionadas.

3.1.7. Copias de seguridad y restauración de copias

A continuación estudiaremos las estructuras y los mecanismos utilizados por Oracle para proporcionar:

- la recuperación de bases de datos debido a distintos tipos de errores
- operaciones de recuperación flexibles que se ajusten a cualquier situación
- disponibilidad de los datos durante las operaciones de copia y recuperación de forma que los usuarios del sistema puedan seguir trabajando

3.1.7.1. Importancia de la recuperación

Como siempre existe la posibilidad de que ocurra un fallo en el hardware, estos fallos pueden afectar a la base de datos. Por tanto, deben existir mecanismos que permitan recuperar la base de datos. Así pues, los objetivos tras un error son

- asegurar que las transacciones validadas han tenido efecto y que se reflejan en la base de datos recuperada
- devolver a la base de datos tan rápido como sea posible a un estado correcto aislando a los usuarios de los problemas derivados del error

Entre las circustancias que pueden detener el funcionamiento de una base de datos Oracle tenemos

- Errores de usuario. Permiten recuperar la base de datos a su estado en un instante de tiempo concreto. Por ejemplo, para recuperar una base de datos que ha sufrido la eliminación accidental de una tabla.
- Errores de sentencia. Ocurren cuando se produce un error lógico en la ejecución de una sentencia (p.e. una expresión SQL incorrecta).
- Errores de proceso. Se trata de errores en procesos de usuario (p.e. desconexión anormal o terminación anormal de un proceso). Esto ha de impedir que otros procesos de usuario continúen trabajando. El proceso background PMON detecta de forma automática el proceso erróneo o es informado de ello por SQL*Net. PMON resuelve el problema deshaciendo la transacción no validada del proceso y liberando los recursos utilizados por el proceso
- Errores de instancia. Se deben a errores hardware (p.e. caídas de tensión), o a errores hardware (fallo del sistema operativo). Como no se ha realizado la escritura de los datos de los búfers del SGA, al volver a iniciar la instancia, Oracle recupera los búfers del SGA a partir del archivo de registro de operaciones
- Errores de disco. Se deben a errores de lectura o escritura en un archivo. La recuperación que se realiza corresponde a la copia de seguridad más reciente respecto al instante del error de disco y a los archivos de registro de operaciones.

Oracle utiliza varias estructuras de control para proporcionar una recuperación completa de la base de datos: el archivo de registro de operaciones, los segmentos de rollback, un archivo de control, y las copias de seguridad necesarias.

- El archivo de registro de operaciones. Conjunto de archivos que protegen las modificaciones realizadas en la base de datos pero que aún no se han escrito en disco. Este archivo consta de dos partes: el archivo de registro de operaciones en línea y el archivo de registro de operaciones archivados

  - El archivo de registro de operaciones en línea. Conjunto de dos o más archivos de registro de operaciones en línea que registra todos los cambios validados realizados en la base de datos. Al validar una transacción, el proceso LGWR guarda en un archivo de registro de operaciones en línea las entradas guardadas temporalmente en los búfers del SGA.
  - El archivo de registro de operaciones archivados. Opcionalmente, se pueden guardar los archivos de registro de operaciones en línea. Estos archivos guardados (offline) constituyen los archivos de registro de operaciones archivados.

- Archivos de control. Los archivos de control de la base de datos guardan, entre otras cosas, información sobre la estructura de la base de datos e información del registro de operaciones escritas por el LGWR. La información de este archivo de control es la que se utiliza para asistir al proceso de recuperación.
- Segmentos rollback. Graban la información para poder deshacer. En el proceso de recuperación, una vez que se hayan aplicado todos los cambios guardados en el archivo de registro de operaciones, Oracle utiliza la información del segmento de rollback para deshacer transacciones no validadas.

3.1.7.2. Copia de seguridad de la base de datos

Como un error de disco puede estropear uno o varios archivos, es necesario poder recuperar las copias de seguridad más recientes. Hay varias formas de hacer una copia de seguridad de los archivos de una base de datos.

- Copia de seguridad de toda la base de datos. Copia de seguridad de todos los archivos de datos, archivos de registro de operaciones en línea y del archivo de control de la base de datos. Para hacer este tipo de copia de seguridad la base de datos tiene que estar cerrada.
- Copia de seguridad parcial. Copia de algunos archivos de la base de datos (p.e. archivos de datos de los tablespaces o de un archivo de control). Estas copias de seguridad se han de utilizar en combinación con el archivo de registro de operaciones.

3.1.7.3. Pasos básicos de la recuperación

Debido a la forma en que el DBWR escribe en los archivos de datos los búfers de la base de datos, puede ocurrir que un archivo de datos pueda contener algunos bloques que estuviesen modificándose por transacciones no validadas y puede que no contenga algunos bloques modificados por transacciones validadas. Por tanto aparecen dos situaciones erróneas:

- Bloques con modificaciones validadas y no escritas en los archivos de datos, por lo que los cambios sólo estarán en el archivo de registro de

operaciones. Por tanto, hay que aplicar lo registrado a los archivos de datos.

- Como el archivo de registro de operaciones puede tener datos no validados, hay que borrar de los archivos de datos las modificaciones realizadas por las transacciones no validadas

Para resolver esta situación, Oracle realiza dos pasos: rehacer y deshacer (*rolling forward* y *rolling back*).

- Rolling Forward (Rehacer). Volver a aplicar a los archivos de datos todas las modificaciones guardadas en el archivo de registro de operaciones. Oracle realiza este paso de forma automática al volver a iniciar la base de datos. Después del roll forward, los archivos de datos tendrán todos los cambios validados, así como los cambios no validados que estaban guardados en el archivo de registro de operaciones.
- Rolling Back (Deshacer). Una vez que se ha realizado el roll forward, todos los cambios que no estaban validados están sin hacer. Una vez que se aplican los archivos de registro de operaciones, se utilizan los segmentos de rollback para identificar y deshacer transacciones que no se llegaron a validar, a pesar de estar guardadas en el archivo de registro de operaciones.

Estas operaciones de copia de seguridad y de restauración pueden ser asistidas por el Recovery Manager de Oracle.

Para ello, esta utilidad mantiene un repositorio denominado catálogo de recuperación, que incluye información sobre los archivos de copias de seguridad y de los archivos de registro de operaciones. Este gestor utiliza el catálogo para automatizar las operaciones de restauración.

3.2 SQL Net Easy Configuration

Cuando se trabaja con Oracle, normalmente se hace en un entorno cliente servidor, en el que en la parte servidor se encuentra Oracle (como gestor de base de datos) y la propia base de datos, y en la parte cliente se encuentran las herramientas de desarrollo y explotación de la base de datos. Entre estas herramientas se encuentran el entorno interactivo de SQL para Oracle, SQL*Plus, y las herramientas de desarrollo de Oracle, Developer 2000. Además se encontrarían las de diseño, Designer 2000, pero que nosotros no estudiaremos.

Para utilizar la base de datos desde la parte cliente, previamente el administrador de la base de datos ha debido crear en la parte servidor lo que se conoce como instancia de la base de datos (memoria y procesos de Oracle), que puede ser vista como la base de datos y procesos de gestión de datos. Además, el administrador tendrá que haber creado las cuentas de usuario con sus privilegios correspondientes.

Entonces el cliente realiza la conexión a la instancia y puede acceder de forma remota a la base de datos, desarrollar en la parte cliente y reducir la carga del servidor gracias a la arquitectura cliente servidor. Normalmente esta conexión desde la parte

cliente tiene asociado un nombre al que se conoce como alias de la base de datos. Este alias, no es más que un nombre de instancia, el nombre del servidor de Oracle y el tipo de protocolo de comunicación que se va a establecer entre el cliente y el servidor.

La creación y administración de estos alias se realiza a través de SQL Net Easy Configuration, que es una herramienta de Oracle para la creación y modificación de alias de bases de datos.

3.2.1. Componentes de un alias de base de datos

Una alias de base de datos para conexiones a través de TCP/IP está formado por:

- Un nombre local, que es el que se utiliza desde el cliente para identificar realmente a la base de datos remota
- Un protocolo de comunicación (TCP/IP), que indica el protocolo con el que se comunican cliente y servidor
- Una dirección TCP/IP o nombre de host, que indica la dirección IP o el nombre del servidor
- Un nombre de instancia, que se corresponde con el nombre de la instancia de la base de datos que queremos utilizar

Si se tratase de comunicación a través de SPX, se introduciría el nombre de servicio SPX del servidor en lugar de la dirección IP.

3.2.2. Creación de un alias de base de datos

Para crear un alias de base de datos, ejecutaremos SQL Net Easy Configuration, que se encuentra en Inicio|Programas|Oracle for Windows NT, y en el cuadro de diálogo que aparece seleccionaremos **Add Database Alias**.

A continuación elegimos el nombre con el que accederemos desde el cliente a la base de datos remota (p.e. MiBD). Recuerde que este nombre es de la parte cliente, y por tanto local a su equipo, por lo que si quiere acceder desde otro equipo, tendrá que utilizar uno de los alias creados, o bien crear uno nuevo.

Una vez elegido el nombre con el que accederemos a la base de datos, seleccionaremos el tipo de protocolo de comunicación, que en este caso es TCP/IP, y pasaremos al paso siguiente.

En este paso indicaremos la dirección IP del servidor Oracle (p.e. 193.147.118.224) y el nombre de la instancia que ha creado el administrador de la base de datos para nosotros (ejemplo), con lo que debemos llegar a un cuadro de diálogo como el de la figura siguiente, en el que se nos indica si son correctos los datos. Si lo son aceptaremos para añadir este alias, y si no podremos volver atrás para realizar alguna modificación.

Una vez que hayamos finalizado con la definición del alias seleccionaremos Exit SQL Net Easy Configuration y pulsaremos **OK**.

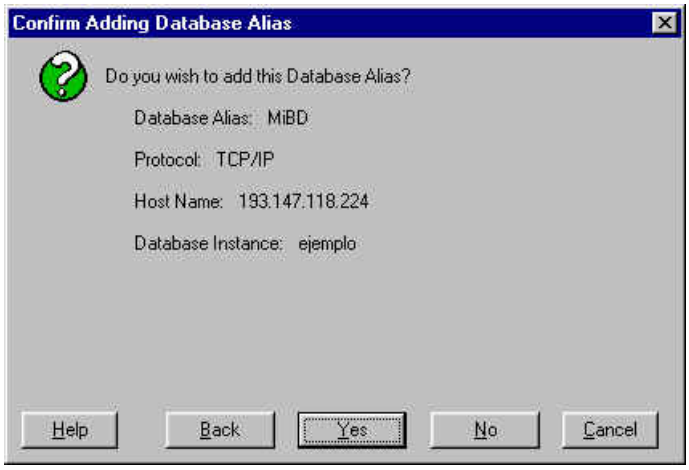

Figura 3.5. Creación de un alias.

Con SQL Net Easy Configuration también puede consultar, modificar y eliminar alias creados anteriormente desde el cuadro de diálogo inicial.

Por tanto, la definición de las conexiones a las distintas bases de datos a las que queramos acceder desde nuestra máquina se realiza desde SQL Net Easy Configuration, y podemos definir tantos alias como queramos para una misma base de datos, y tanto alias para distintas bases de datos como necesitemos. Así pues, es la herramienta que permite incluir en un solo nombre el servidor donde se encuentra Oracle, el nombre de la instancia que queremos utilizar y el protocolo de comunicación que hay que utilizar.

3.3 SQL*Plus

Una vez realizada la introducción a Oracle, pasemos a estudiar SQL*Plus. Aunque en la introducción ya se ha citado a SQL como lenguaje de interacción con Oracle, así como algunos conceptos sobre lenguajes de definición y de manipulación de datos, es en esta sección donde vamos a estudiar con cierta profundidad los aspectos básicos de SQL.

Para ello, utilizaremos SQL*Plus, una herramienta de Oracle que actúa como intérprete de SQL. Así pues, será en SQL*Plus donde introduciremos nuestras sentencias SQL y donde veremos los resultados de nuestras consultas. Concretamente en esta parte de SQL*Plus, describiremos la utilización de la herramienta y estudiaremos con detalle la sintaxis básica de SQL. Todo ello, ilustrado con bastantes ejemplos, de forma que sea sencillo la asimilación de este lenguaje.

3.3.1. Introducción

SQL*Plus, el intérprete de SQL de Oracle, permite la manipulación de órdenes SQL y de bloques PL/SQL para realizar cualquiera de estas tareas:

- introducir, editar, guardar, recuperar y ejecutar órdenes SQL y bloques PL/SQL
- formatear, realizar cálculos, guardar e imprimir informes

SQL*Plus proporciona una interfaz de línea de órdenes en un entorno gráfico (es una aplicación Windows), permitiendo

- abrir y guardar scripts SQL
- realizar algunas operaciones de edición como copiar y pegar texto, o buscar.
- personalizar el entorno

3.3.1.1. Conceptos básicos

A continuación se comentan los términos que se utilizan al hablar de SQL, PL/SQL y SQL*Plus:

- Orden o comando. Instrucción que se da a SQL*Plus o a la base de datos Oracle
- Bloque. Grupo de órdenes SQL y PL/SQL
- Tabla. Unidad básica de almacenamiento de Oracle
- Consulta. Orden SQL de recuperación de información de una o más tablas (SELECT)
- Resultado de una consulta. Datos recuperados por una consulta
- Informe. Resultados de una consulta formateados con órdenes de SQL*Plus

3.3.1.2. Utilización de SQL*Plus

En primer lugar, si nuestra base de datos no es local y está en un servidor remoto, tendremos que cargar los controladores apropiados para poder acceder de forma remota. A continuación bastará con ejecutar SQL*Plus e identificarse como usuario.

Para comenzar a utilizar SQL*Plus, seleccionaremos Inicio|Programas|Oracle for Windows 95|SQL Plus 8.0.

Una vez realizado esto, aparecerá un cuadro de diálogo como el de la figura siguiente para conectarnos a la base de datos. En él introduciremos nuestro nombre de usuario en el cuadro **User name** (alumno*XX*), nuestra contraseña en el cuadro **Password** (alumno*XX*), y el nombre de la base de datos definido en SQL Net Easy Configuration en el cuadro **Host String**. A continuación pulsaremos **OK**.

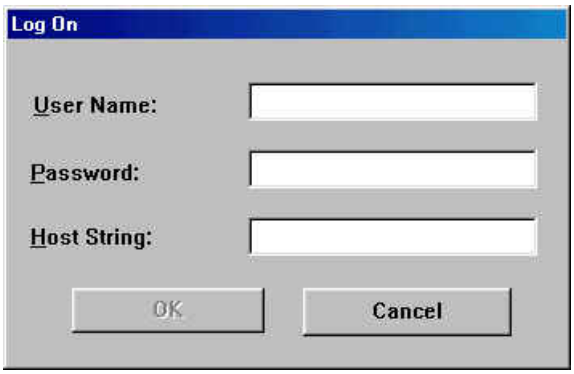

Figura 3.6. Cuadro de diálogo para la conexión a la base de datos

Transcurridos unos instantes (iniciar la instancia, montar la base de datos y abrir la base de datos) aparecerá la ventana el prompt del intérprete de órdenes de SQL*Plus.

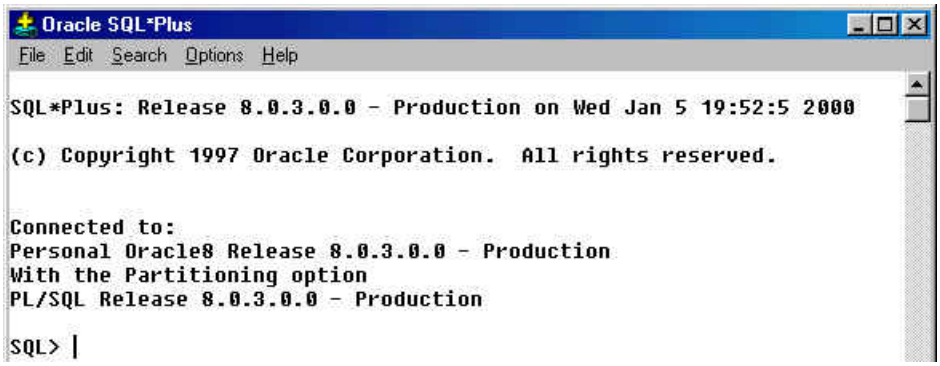

Figura 3.7. El intérprete de órdenes de SQL*Plus.

En esta ventana introduciremos nuestras sentencias SQL, que se irán guardando en un buffer, que posteriormente podremos guardar para luego poder abrirlo. A continuación, describimos las órdenes disponibles en los menús del intérprete SQL*Plus.

Menú File

Open	Abre un archivo de órdenes guardado. Por omisión, tendrán extensión SQL
Save	**Save Create** guarda el contenido del buffer en un archivo de órdenes **Save Replace** sustituye el contenido de un archivo existente con el contenido actual del buffer **Save Append** añade el contenido del buffer al final del archivo que se especifique
Save As	Guarda el contenido del buffer en un archivo de órdenes
Spool	**Spool File** guarda los resultados de las consultas en un archivo. Por omisión, estos archivos tienen extensión LST. **Spool Off** desactiva el envío de resultados a un archivo.
Run	Lista y ejecuta la sentencia SQL o el bloque PL/SQL guardado actualmente en el buffer
Cancel	Cancela el progreso de una operación
Exit	Acepta los cambios pendientes y cierra la aplicación SQL*Plus

Menú Edit

Copy	Copia el texto seleccionado al portapapeles. El método abreviado es CONTROL+C
Paste	Pega el contenido del portapapeles. El método abreviado es CONTROL+V
Clear	Borra el buffer y la pantalla de SQL*Plus. La orden SQL equivalente es CLEAR SCREEN, y el método abreviado es MAYUS+DEL

Editor	**Invoke Editor** carga el contenido del buffer en una ventana del bloc de notas. Por omisión, la orden se guarda en un archivo AFIEDT.BUF que por supuesto se puede cambiar. Si desea que el editor por omisión sea otro distinto tendrá que utilizar la orden *Define Editor*. **Define Editor** define el editor que se utilizará por omisión para ver el contenido del buffer

Menú Search

Find	Busca un carácter, palabra o grupo de palabras en la ventana de SQL*Plus. La búsqueda se realiza desde el principio de la pantalla mostrada, y cuando llega al final, no se continúa buscando desde el principio. El método abreviado es MAYUS+F3
Find Next	Busca la ocurrencia siguiente del elemento buscado

Menú Options

Este menú sólo tiene la orden **Environment**, que sirve para configurar una serie de propiedades de SQL*Plus.

El cuadro de diálogo contiene dos áreas: Set Options y Screen Buffer.

- El área Set Options proporciona una lista de parámetros que puede configurar para establecer ciertos aspectos, como el ancho predefinido para los números, la activación o desactivación de los encabezados, la definición del número de líneas por página, ...
- El área Screen Buffer permite configurar el número de caracteres que se pueden mostrar en una línea (*Buffer Width*) y el número de líneas por pantalla (*Buffer Length*).

3.3.1.3. Carga de los datos de ejemplo

En esta parte veremos cómo cargar una serie de datos de ejemplo que incorpora Oracle. Para ello seguiremos utilizando la cuenta de alumno*XX* con la contraseña alumno*XX*.

Lo único que hay que hacer es cargar la demostración, que en realidad es un script SQL que elimina las tablas en el caso de que existan, crea las tablas y las llena con algunos datos de ejemplo.

Para cargar la demo hay que escribir lo siguiente en el prompt de SQL:

```
@CreaEjer.sql
```

Una vez que introducimos esta orden se carga y se ejecuta (@) el script CreaEjer.sql. Una vez que se haya ejecutado, veremos que se cierra nuestra ventana de SQL*Plus. Esto se debe a que la última orden de este script es una orden EXIT, con lo que se aceptan todos los cambios y se cierra SQL*Plus. Pero, no se preocupe, vuelva a abrir una sesión con SQL*Plus como alumno*XX*/alumno*XX* y allí estarán sus tablas.

Para eliminarlas ejecutaremos el script BorraEje.sql de esta forma:

```
@BorraEjer.sql
```

A continuación se muestran cuales son las tablas con las que vamos a trabajar, que sólo son dos, pese a que en el ejemplo se hayan definido algunas más. Se trata de una tabla de empleados y de una tabla de departamentos, tal y como se muestra en las figuras siguientes.

NUMEMP	NOMBREEMP	EMPLEO	JEFE	FECHAENTR	SUELDO	COMPLEMENTO	NUMDEP
7369	SMITH	ORDENANZA	7902	17-DEC-80	800		20
7499	ALLEN	VENDEDOR	7698	20-FEB-81	1600	300	30
7521	WARD	VENDEDOR	7698	22-FEB-81	1250	500	30
7566	JONES	DIRECTIVO	7839	02-APR-81	2975		20
7654	MARTIN	VENDEDOR	7698	28-SEP-81	1250	1400	30
7698	BLAKE	DIRECTIVO	7839	01-MAY-81	2850		30
7782	CLARK	DIRECTIVO	7839	09-JUN-81	2450		10
7788	SCOTT	ANALISTA	7566	09-DEC-82	3000		20
7839	KING	DIRECTOR		17-NOV-81	5000		10
7844	TURNER	VENDEDOR	7698	08-SEP-81	1500	0	30
7876	ADAMS	ORDENANZA	7788	12-JAN-83	1100		20
7900	JAMES	ORDENANZA	7698	03-DEC-81	950		30
7902	FORD	ANALISTA	7566	03-DEC-81	3000		20
7934	MILLER	ORDENANZA	7782	23-JAN-82	1300		10

Figura 3.8. La tabla de empleados (EMP).

NUMDEP	NOMBREDEP	CIUDAD
10	CONTABILIDAD	NEW YORK
20	INVESTIGACION	DALLAS
30	VENTAS	CHICAGO
40	OPERACIONES	BOSTON

Figura 3.9. La tabla de departamentos (DEPT).

Debido a que SQL es el lenguaje que utilizaremos para interactuar con el SGBD, es importante unificar los conocimientos de SQL. Para ello daremos un breve repaso de la sintaxis SQL tanto como DDL como DML. Decir, que de antemano trabajaremos con definiciones de tablas para DDL y con consultas y actualizaciones. Concretamente en consultas veremos la estructura básica de una sentencia SQL, y estudiaremos las operaciones sobre conjuntos, la operación de producto natural y la agrupación.

3.3.2. Definición de datos en SQL

Para crear una tabla utilizaremos la orden CREATE TABLE, y para modificar la definición de una tabla utilizaremos ALTER TABLE ADD para añadir una columna y ALTER TABLE MODIFY para modificar la descripción de una columna.

Ampliemos nuestra base de datos para que guarde información sobre los proyectos en que participan los empleados, de forma que un empleado sólo trabaja en un proyecto y que los proyectos vienen identificados por un número de proyecto, un nombre de proyecto y un presupuesto.

```
CREATE TABLE Proyecto (NumProy NUMBER(3) NOT NULL,
                       NomProy CHAR(5),
                       Presupuesto NUMBER(7,2));
```

A continuación utilizaremos la orden INSERT para insertar estos tres proyectos:

- Proyecto 1, Alfa, 96000

- Proyecto 2, Beta, 82000
- Proyecto 3, Gamma, 15000

```
INSERT INTO Proyecto VALUES (1, 'ALFA', 96000);
INSERT INTO Proyecto VALUES (2, 'BETA', 82000);
INSERT INTO Proyecto VALUES (3, 'GAMMA', 15000);
```

Ahora vamos a añadir la columna del número del proyecto a la tabla de empleados:

```
ALTER TABLE Emp ADD (NumProy NUMBER(3));
```

A continuación incluimos en el proyecto ALFA a los empleados del departamento número 30 y a los que trabajan de vendedores:

```
UPDATE Emp
SET NumProy = 1
WHERE NumDep = 30 OR
Empleo = 'VENDEDOR';
```

Al resto de los empleados los incluimos en el proyecto BETA:

```
UPDATE Emp
SET NumProy = 2
WHERE NumProy IS NULL;
```

Para obtener una lista alfabética de los empleados junto con el nombre del proyecto en el que trabajan escribiríamos:

```
SELECT NombreEmp, NomProy
FROM  Emp, Proyecto
WHERE Emp.NumProy = Proyecto.NumProy
ORDER BY NombreEmp;
```

Si intenta modificar el presupuesto del proyecto Gamma a 105000 con la orden siguiente obtendrá un mensaje de error

```
UPDATE Proy
SET Presupuesto = 105000
WHERE NomProy = 'Gamma';
```

Para evitar este error tiene que modificar el ancho de la columna, y esto se hace de esta forma:

```
ALTER TABLE Proyecto MODIFY Presupuesto NUMBER(8,2);
```

y verá como al volver a ejecutar la consulta de actualización anterior si tiene éxito.

3.3.3. Consulta de datos en SQL

La recuperación de información a partir de los datos de la base de datos es la operación SQL más común. La recuperación de información de una base de datos se realiza mediante una **consulta**, utilizando la orden SELECT.

La orden SELECT básica tiene dos partes, denominadas **cláusulas**:

SELECT *nombre(s) de columna(s)*

FROM *nombre(s) de tabla(s)*

La cláusula SELECT siempre es la primera y va seguida inmediatamente de la cláusula FROM.

Veamos ahora algunos ejemplos sobre los datos de las tablas DEPT y EMP realizando algunas consultas sencillas.

En primer lugar veamos todas las columnas y todas las filas de la tabla DEPT. La orden SQL que realiza esto es:

```
SELECT NumDep, NombreDep, Ciudad
FROM Dept;
```

En este ejemplo de consulta hemos tenido que indicar los nombres de todas las columnas en la cláusula SELECT. Normalmente, cuando se quieren seleccionar **todas** las columnas de una tabla se utiliza SELECT * como forma abreviada de *proyección*[1] de todas las columnas.

```
SELECT *
FROM Dept;
```

Si en el resultado de una consulta sólo se quieren mostrar (proyectar)[1] algunas columnas en particular, en la cláusula SELECT sólo se incluyen los nombres de las columnas que se desean mostrar.

```
SELECT NombreDep, NumDep
FROM Dept;
```

El orden en que se introduzcan los atributos en la cláusula SELECT determinará el orden en la presentación de columnas en el resultado de la consulta. Cuando se utiliza SELECT *, la secuencia de presentación de las columnas viene determinado por el orden en que se definieron las columnas con la definición de la tabla (*Ya veremos que la definición de una tabla se realiza con la orden CREATE TABLE.*)

Para seleccionar ciertas filas de una tabla hay que añadir una cláusula WHERE después de la orden SELECT, es decir justo después de la cláusula FROM. A la condición de búsqueda se le denomina **predicado de selección** de las filas. Esta consulta selecciona los nombres de los empleados que trabajan en el departamento número 30:

```
SELECT NombreEmp
FROM Emp
WHERE NumDep = 30;
```

A veces es necesario especificar más de un predicado de selección en una misma cláusula WHERE. Para ello se pueden dar los siguientes casos, que a su vez se pueden combinar entre sí.

3.3.3.2. Condiciones compuestas

En el caso de que deseen realizar consultas con varios predicados con significado lógico Y se utiliza el conector lógico AND.

[1] Operación del álgebra relacional que obtiene sólo las columnas que se especifiquen

Seleccionar los directores de la empresa que ganan más de 2800, indicando también su sueldo.

```
SELECT NombreEmp, Sueldo
FROM Emp
WHERE Empleo = 'DIRECTOR' AND Sueldo > 2800;
```

En el caso de que se deseen realizar consultas con varios predicados con significado lógico O se utiliza el conector lógico OR.

Seleccionar nombre y sueldo de los empleados que son directores o que ganan más de 2800

```
SELECT NombreEmp, Sueldo
FROM Emp
WHERE Empleo = 'DIRECTOR' OR Sueldo > 2800;
```

También puede seleccionar los registros que no cumplen un predicado de selección. Para ello se utiliza el conector lógico NOT o bien la combinación !=, (que no es igual).

Seleccionar los nombres de los directores que no trabajen en el departamento número 30, junto con el número de departamento en el que trabajan

i)
```
SELECT NombreEmp, NumDep
FROM Emp
WHERE Empleo = 'DIRECTOR' AND NumDep != 30;
```

ii)
```
SELECT NombreEmp, NumDep
FROM Emp
WHERE Empleo = 'DIRECTOR' AND NOT NumDep = 30;
```

Para seleccionar un rango numérico puede utilizar una combinación de los operadores de comparación <= ó >= o bien el operador BETWEEN, que selecciona las filas que contienen los valores que están incluidos en el rango especificado.

Seleccionar los nombres de los empleados que tienen un sueldo comprendido entre 1200 y 1400. Mostrar también su sueldo

i)
```
SELECT NombreEmp, Sueldo
FROM Emp
WHERE Sueldo >= 1200 AND Sueldo <= 1400;
```

ii)
```
SELECT NombreEmp, Sueldo
FROM Emp
WHERE Sueldo BETWEEN 1200 AND 1400;
```

3.3.3.3. Búsqueda de valores en una lista y comprobación parcial de valores

El operador IN le permite seleccionar filas que contienen un valor que coincide con uno de los valores incluidos en una lista de valores.

Seleccionar una lista de los departamentos cuyo número sea 10 ó 30:

i)
```
SELECT NombreDep
```

```
      FROM Dept
      WHERE NumDep = 10 OR NumDep = 30;
ii)   SELECT NombreDep
      FROM Dept
      WHERE NumDep IN (10, 30);
```

Por último, puede seleccionar filas que coincidan con un patrón de caracteres, y esto se hace con el operador LIKE, en donde el carácter de subrayado (_) sustituye a un carácter cualquiera, y el carácter de tanto por ciento (%) sustituye a un grupo de caracteres.

Seleccionar los nombres de los empleados que su nombre comienza por A

```
SELECT NombreEmp
FROM Emp
WHERE NombreEmp LIKE 'A%';
```

3.3.3.4. Ordenación de resultados

En los ejemplos anteriores, las filas resultantes de una consulta se presentaban en un orden determinado por ORACLE. Puede controlar el orden en que desea que aparezcan las filas de la consulta añadiendo la cláusula ORDER BY al final de orden SELECT.

Obtener una lista ordenada alfabéticamente de los empleados que trabajan en el departamento número 30:

```
SELECT NombreEmp
FROM Emp
WHERE NumDep = 30
ORDER BY NombreEmp;
```

La cláusula ORDER BY efectúa una ordenación ascendente, pero si lo que queremos es realizar una ordenación descendente utilizaremos ORDER BY *atributo* DESC, y además se pueden tener varios criterios de ordenación, que se realizarán de izquierda a derecha según se hayan escrito.

Obtener una lista de los empleados ordenados por trabajo en orden alfabético y por orden decreciente de sueldo:

```
SELECT Empleo, Sueldo, NombreEmp
FROM Emp
ORDER BY Empleo, Sueldo DESC;
```

En casi todas las ocasiones es necesaria la presentación de los resultados de forma única, es decir, eliminando las filas duplicadas. Para ello, escribiremos SELECT DISTINCT en lugar de SELECT, y esto hará que se eliminen las filas resultantes duplicadas.

¿Cuáles son los tipos de trabajo que se desempeñan en su empresa?

```
SELECT DISTINCT Empleo
FROM Emp;
```

3.3.3.5. Consultas sobre varias tablas

Hasta ahora sólo hemos realizado consultas sobre una sola tabla, pero como en las bases de datos relacionales los datos se encuentran guardados en varias tablas, necesitaremos una forma de poder combinar las tablas y poder acceder a los datos que necesitamos.

Para ello se impone que las columnas de las tablas que se refieran a los mismos atributos tomen el mismo valor, tal y como se muestra en el ejemplo siguiente.

¿En qué estado trabaja Allen?

Como en la tabla en la que está guardada la información propia del empleado Allen no se encuentra el estado en el que trabaja, sino el número de departamento en el que trabaja, necesitamos conocer en qué estado se encuentra el departamento con dicho número.

```
SELECT Ciudad
FROM Emp, Dept
WHERE Emp.NumDep = Dept.NumDep AND NombreEmp = 'ALLEN';
```

Siempre que tenga que realizar consultas que utilicen más de una tabla, tenga cuidado al hacer referencia a los atributos que puedan pertenecer a más de una tabla, incluyendo en estos casos antes del nombre del atributo, el nombre de la tabla al que se refiere, y a continuación un punto. De esta forma se eliminan las posibles ambigüedades.

3.3.3.6. Funciones de agregación

Las funciones de agregación son otro ejemplo de la potencia de SQL. Las funciones de agregación le permiten obtener información resumida a partir de grupos de filas. Para ello, se utiliza el operador de agrupación GROUP BY después de la cláusula WHERE (en el caso de que esté) y las funciones de agregación COUNT, MAX, MIN, AVG y SUM.

Obtener el sueldo máximo de cada departamento:

```
SELECT NumDep, MAX(Sueldo)
FROM Dept
GROUP BY NumDep;
```

Ha de tener en cuenta que cuando utilice grupos, sólo puede proyectar las columnas sobre las que agrupe o columnas a las que aplique una función de agregación.

Cuando utilice la cláusula WHERE antes de GROUP BY estará haciendo una selección de filas antes de realizar la agrupación, es decir, sólo se agruparán las filas que cumplan el predicado de la cláusula WHERE.

A la hora de utilizar grupos, puede que esté interesado en que una vez que haya hecho la agrupación y haya obtenido el resultado, sólo aparezcan las filas que cumplan una condición. Esta selección a posteriori (por el contrario de la selección a priori de la cláusula WHERE) se realiza con la cláusula HAVING.

¿Cuáles son los departamentos que tienen un salario máximo superior a 4000?:

```
SELECT NumDep, MAX(Sueldo)
FROM Dept
GROUP BY NumDep
HAVING MAX(Sueldo) > 4000;
```

3.3.3.7. Consultas anidadas

Una de las razones de la potencia de SQL es que se pueden crear consultas complejas a partir de consultas sencillas, sin más que incluir en el predicado de la cláusula WHERE otra consulta (que se denomina subconsulta).

¿Cuáles son los empleados que tienen el mismo trabajo que Jones?

```
SELECT NombreEmp
FROM Emp
WHERE Empleo =  (SELECT Empleo
                 FROM Emp
                 Where NombreEmp = 'JONES');
```

¿Cuáles son los empleados que tienen un sueldo superior al sueldo medio de los empleados?:

```
SELECT NombreEmp, Sueldo
FROM Emp
WHERE Sueldo >  (SELECT AVG(Sueldo)
                 FROM Emp);
```

3.3.4. Creación y eliminación de vistas

Oracle le permite diseñar y guardar **vistas** (o niveles de visión) de los mismos datos guardados en su base de datos. Las vistas son tablas virtuales a través de las que puede ver los datos guardados realmente en tablas reales.

Las vistas se crean fundamentalmente para:

- Simplificar el acceso a los datos
- Proporcionar niveles de acceso o privacidad de los datos

En primer lugar, se crearán las vistas, y como el resultado de una consulta es una tabla, las vistas se pueden usar para definir nuevas consultas, ya que son tablas virtuales

Crear una vista con los datos de empleados del departamento número 10:

```
CREATE VIEW EMP10 AS  SELECT *
                      FROM Emp
                      WHERE NumDep = 10;
```

¿Cuáles son los nombres de los empleados del departamento número 10?

```
SELECT NombreEmp
FROM Emp10;
```

Para eliminar una vista se utiliza la orden DROP VIEW *nombre de la vista*

Eliminar la vista Emp10

```
DROP VIEW Emp10;
```

NOTA: Para modificar una vista hay que eliminar la vista anterior y crear una nueva vista con los cambios necesarios.

3.3.5. Modificación de datos en SQL

Como ya hemos visto, una sentencia SQL le permite recuperar filas de una o varias tablas. SQL también le permite añadir, modificar o eliminar filas mediante estas tres órdenes:

- Orden UPDATE. Cambia los valores guardados en los campos
- Orden INSERT . Añade filas a una tabla
- Orden DELETE. Elimina filas de una tabla

3.3.5.1. Orden UPDATE

La orden UPDATE permite actualizar los valores de un campo (sólo un campo en una orden UPDATE) de algunas filas de una tabla cuando se cumpla un predicado de selección (el predicado que indica las filas que se tienen que modificar)

Supongamos que en esta base de datos de ejemplo tenemos que subir el sueldo en 100 € a todos los ordenanzas, es decir, necesitamos modificar el valor del campo Sueldo de aquellas filas de la tabla Emp que cumplan el predicado correspondiente:

```
UPDATE Emp
SET Sueldo = Sueldo + 100
WHERE Empleo = 'ORDENANZA';
```

- En la cláusula UPDATE se indica el nombre de la tabla que se desea modificar (UPDATE EMP).
- En la cláusula SET se establece el campo que se desea modificar a un valor o expresión (SET Sueldo = Sueldo +100).
- En la cláusula WHERE se especifica el predicado que selecciona las filas que se tienen que modificar (WHERE Empleo = 'ORDENANZA').

3.3.5.2. Orden INSERT

La orden INSERT se utiliza para insertar una o varias filas en una tabla. La operación de inserción se puede hacer especificando explícitamente los valores que se desean insertar (una fila cada vez) Para insertar una fila hay que especificar cada uno de los valores de todos los campos de la tabla, mientras que para insertar una serie de filas puede utilizar una subconsulta que devuelva cada uno de los valores que desea introducir en la tabla.

Insertar el departamento número 30 que está en Chicago y se dedica a las ventas:

```
INSERT INTO DEPT
```

```
VALUES (30, 'VENTAS', 'CHICAGO');
```

Como tenemos una tabla de primas denominada *Primas* que contiene las columnas NombreEmp, Empleo, Sueldo y Complemento, podemos insertar en la tabla *Primas* los datos de los empleados que tienen algún puesto relacionado con la dirección de la empresa

```
INSERT INTO Primas (NombreEmp, Empleo, Sueldo, Comm)
      SELECT NombreEmp, Empleo, Sueldo
      FROM  Emp
      WHERE Empleo LIKE 'DIREC%';
```

3.3.5.3. Orden DELETE

La orden DELETE se utiliza para eliminar las filas de una tabla que cumplan una condición.

Como el departamento número 30 creado anteriormente no existe, vamos a eliminarlo:

```
DELETE
FROM  Dept
WHERE NumDep = 30;
```

3.3.3. El búfer de SQL*Plus

Presentado SQL*Plus y repasado SQL, terminemos esta primera parte dedicada a la introducción de SQL*Plus comentando cómo podemos utilizar el búfer de SQL*Plus para editar órdenes SQL. Con esto conseguiremos realizar modificaciones y evitaremos tener que volver a introducir la orden SQL.

En primer lugar, debemos saber que SQL*Plus cuenta con un búfer en el que se guarda la última sentencia SQL escrita, y siempre se ejecuta la orden del búfer. Por tanto, podemos modificar el contenido de este búfer y ejecutar su contenido.

Para listar el contenido del búfer escribiremos *LIST*, y aparecerá la última sentencia que hayamos introducido con las líneas numeradas. Esta numeración la podemos emplear para listar ciertas líneas (en el caso de sentencias muy largas) escribiendo *LIST m n*, siendo *m* y *n* los dos números que marcan el rango que queremos listar. También podemos listar una línea cualquiera con *LIST n*, y listar la última *LIST LAST*. En la figura siguiente se ilustra la utilización de LIST.

```
SQL> select *
  2  from emp
  3  where empleo = 'ANALISTA';

    NUMEMP NOMBREEMP  EMPLEO          JEFE FECHAENTR     SUELDO COMPLEMENTO     NUMDEP
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7788 SCOTT      ANALISTA        7566 09-DEC-82       3000                    20
      7902 FORD       ANALISTA        7566 03-DEC-81       3000                    20
SQL> LIST
  1  select *
  2  from emp
  3* where empleo = 'ANALISTA'
SQL> LIST 1 2
  1  select *
  2* from emp
SQL> LIST 2
  2* from emp
SQL> LIST LAST
  3* where empleo = 'ANALISTA'
```

Figura 3.10. Utilización de LIST.

Si lo que queremos es modificar una línea lo que haremos es escribir el número de la línea que queremos modificar y a continuación escribiremos la línea completa que sustituye a la línea no deseada.

Por ejemplo, la figura 3.11 muestra cómo modificar una sentencia que obtiene los registros de empleados que trabajan en el departamento número 20 por los que trabajan en el número 30. Para ejecutar el contenido del búfer escribiremos *RUN*.

Para hacer pequeñas sustituciones podemos utilizar la orden *CHANGE /antiguo/nuevo*, que sustituye la cadena que sigue a la primera barra por la cadena que sigue a la segunda barra. La figura 3.12 muestra esto con un ejemplo.

```
SQL> select *
  2  from emp
  3  where numdep = 20;

    NUMEMP NOMBREEMP  EMPLEO          JEFE FECHAENTR     SUELDO COMPLEMENTO     NUMDEP
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      ORDENANZA       7902 17-DEC-80        800                    20
      7566 JONES      DIRECTIVO       7839 02-APR-81       2975                    20
      7788 SCOTT      ANALISTA        7566 09-DEC-82       3000                    20
      7876 ADAMS      ORDENANZA       7788 12-JAN-83       1100                    20
      7902 FORD       ANALISTA        7566 03-DEC-81       3000                    20
SQL> 3 where numdep = 30;
SQL> list
  1  select *
  2  from emp
  3* where numdep = 30
SQL> run
  1  select *
  2  from emp
  3* where numdep = 30

    NUMEMP NOMBREEMP  EMPLEO          JEFE FECHAENTR     SUELDO COMPLEMENTO     NUMDEP
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      VENDEDOR        7698 20-FEB-81       1600        300         30
      7521 WARD       VENDEDOR        7698 22-FEB-81       1250        500         30
      7654 MARTIN     VENDEDOR        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      DIRECTIVO       7839 01-MAY-81       2850                    30
      7844 TURNER     VENDEDOR        7698 08-SEP-81       1500          0         30
      7900 JAMES      ORDENANZA       7698 03-DEC-81        950                    30
```

Figura 3.11. Modificación de una sentencia SQL.

```
SQL> SELECT NOMBREEMP
  2  FROM EMP
  3  WHERE EMPLEO = 'ORRDENANZA';
SQL> CHANGE /ORR/OR
  3* WHERE EMPLEO = 'ORDENANZA'
SQL> CHANGE /EMPLEO = 'ORDENANZA'/NUMDEP = 30
  3* WHERE NUMDEP = 30
SQL> LIST
  1  SELECT NOMBREEMP
  2  FROM EMP
  3* WHERE NUMDEP = 30
```

Figura 3.12. Sustitución de cadenas en una sentencia SQL.

Al utilizar la sustitución hay que tener cuidado con los espacios, ya que se trata de una búsqueda por igualdad total, luego entre el final de la primera cadena y la segunda barra no se escribe ningún espacio. El separador es la propia barra. De lo contrario, también se busca el espacio en blanco en la sentencia SQL.

Por último, podemos eliminar líneas de una sentencia SQL con *DEL*. *DEL* elimina la línea actual, *DEL n* elimina la línea *n*, y *DEL m n* elimina las líneas comprendidas entre *m* y *n*. La figura 3.13 muestra el uso de *DEL*.

```
SQL> select *
  2  from dept;

    NUMDEP NOMBREDEP      CIUDAD
---------- -------------- -------------
        10 CONTABILIDAD   NEW YORK
        20 INVESTIGACION  DALLAS
        30 VENTAS         CHICAGO
        40 OPERACIONES    BOSTON
SQL> del
SQL> list
  1* select *
```

Figura 3.13. Eliminación de líneas de una sentencia SQL.

3.4. Utilización eficiente de SQL*Plus

Para terminar con esta parte dedicada a SQL*Plus veamos cómo podemos utilizar algunas órdenes para mejorar la presentación de resultados y para utilizar SQL*Plus de una forma más eficiente.

Antes de comenzar, decir que todas las operaciones que vamos a ver, las haremos desde la línea de órdenes de SQL*Plus. Muchas de las posibilidades que vamos a presentar en esta última sección (las relacionadas con la configuración del entorno) también son configurables desde la opción *Environment* de SQL*Plus, aunque suele ser más rápido utilizarlas desde el prompt.

Como muchas de las características de SQL*Plus vienen determinadas por los valores de sus variables de entorno, podremos actuar sobre ellas para modificar sus valores (p.e. modificar el número de caracteres por línea). Por tanto, a veces será necesario conocer cuál es el valor de alguna de estas variables. Para ello utilizaremos

SHOW *variable*

Por ejemplo, *SHOW linesize* muestra el contenido de la variable *linesize* que marca el número de caracteres por línea.

Para establecer el número de caracteres por línea utilizaremos

SET LINESIZE *valor*

Veamos un ejemplo en la figura siguiente.

```
SQL> set linesize 100
SQL> select * from emp;

    NUMEMP NOMBREEMP  EMPLEO          JEFE FECHAENTR     SUELDO COMPLEMENTO      NUMDEP
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      ORDENANZA        7902 17-DEC-80        800                    20
      7499 ALLEN      VENDEDOR         7698 20-FEB-81       1600        300         30
      7521 WARD       VENDEDOR         7698 22-FEB-81       1250        500         30
      7566 JONES      DIRECTIVO        7839 02-APR-81       2975                    20
      7654 MARTIN     VENDEDOR         7698 28-SEP-81       1250       1400         30
      7698 BLAKE      DIRECTIVO        7839 01-MAY-81       2850                    30
      7782 CLARK      DIRECTIVO        7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALISTA         7566 09-DEC-82       3000                    20
      7839 KING       DIRECTOR              17-NOV-81       5000                    10
      7844 TURNER     VENDEDOR         7698 08-SEP-81       1500          0         30
      7876 ADAMS      ORDENANZA        7788 12-JAN-83       1100                    20

    NUMEMP NOMBREEMP  EMPLEO          JEFE FECHAENTR     SUELDO COMPLEMENTO      NUMDEP
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7900 JAMES      ORDENANZA        7698 03-DEC-81        950                    30
      7902 FORD       ANALISTA         7566 03-DEC-81       3000                    20
      7934 MILLER     ORDENANZA        7782 23-JAN-82       1300                    10

14 rows selected.
```

Figura 3.14. Configuración del número de caracteres por línea.

SQL*Plus presenta los resultados de las consultas por páginas, y cada una de ellas tiene como encabezado el nombre del campo en la definición de la tabla a la que pertenece. Si lo que queremos es aumentar el número de líneas por página utilizaremos la orden

SET PAGESIZE *valor*

Veamos un ejemplo en la figura siguiente.

```
SQL> set pagesize 40
SQL> select * from emp;

    NUMEMP NOMBREEMP  EMPLEO          JEFE FECHAENTR     SUELDO COMPLEMENTO      NUMDEP
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      ORDENANZA        7902 17-DEC-80        800                    20
      7499 ALLEN      VENDEDOR         7698 20-FEB-81       1600        300         30
      7521 WARD       VENDEDOR         7698 22-FEB-81       1250        500         30
      7566 JONES      DIRECTIVO        7839 02-APR-81       2975                    20
      7654 MARTIN     VENDEDOR         7698 28-SEP-81       1250       1400         30
      7698 BLAKE      DIRECTIVO        7839 01-MAY-81       2850                    30
      7782 CLARK      DIRECTIVO        7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALISTA         7566 09-DEC-82       3000                    20
      7839 KING       DIRECTOR              17-NOV-81       5000                    10
      7844 TURNER     VENDEDOR         7698 08-SEP-81       1500          0         30
      7876 ADAMS      ORDENANZA        7788 12-JAN-83       1100                    20
      7900 JAMES      ORDENANZA        7698 03-DEC-81        950                    30
      7902 FORD       ANALISTA         7566 03-DEC-81       3000                    20
      7934 MILLER     ORDENANZA        7782 23-JAN-82       1300                    10

14 rows selected.
```

Figura 3.15. Configuración del número de líneas por página.

SQL*Plus devuelve los resultados de las consultas con los encabezados de columna correspondientes a los nombres de las columnas de las tablas. Para modificar los encabezados de las columnas escribiremos

COLUMN *columna [tabla]* **HEADING** *nuevaCabecera*

El argumento *tabla* es opcional. Si no lo incluimos, se cambiarán las cabeceras de todos los campos de todas las tablas que coincidan con el nombre especificado en *columna*.

Si queremos desactivar los formatos aplicados a las columnas escribiremos

CLEAR COLUMNS

Para crear encabezados y pies de página utilizaremos

TTITLE *posición cadena*

BTITLE *posición cadena*

para el título y pie de cada página, respectivamente.

El parámetro *posición* puede tomar uno de estos tres valores: LEFT, CENTER o RIGHT.

Para crear encabezados y pies de informe utilizaremos

REPHEADER *posición cadena*

REPFOOTER *posición cadena*

para el título y pie del informe, respectivamente.

Si la definición de nuestro encabezado o pie ocupa más de una línea, podemos utilizar un guión para indicar que la definición continúa en la línea siguiente. (Esto es especialmente útil al definir la parte izquierda, central y derecha de un encabezado o pie.)

Veamos todo esto ilustrado en una figura.

```
SQL> COLUMN NUMEMP HEADING NUMERO
SQL> REPHEADER CENTER 'LISTADO ALFABETICO DE EMPLEADOS'
SQL> TTITLE LEFT 'LISTA'
SQL> SELECT * FROM EMP;

LISTA
                                LISTADO ALFABETICO DE EMPLEADOS
    NUMERO NOMBREEMP  EMPLEO          JEFE FECHAENTR     SUELDO COMPLEMENTO     NUMDEP
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      ORDENANZA       7902 17-DEC-80        800                    20
      7499 ALLEN      VENDEDOR        7698 20-FEB-81       1600        300         30
      7521 WARD       VENDEDOR        7698 22-FEB-81       1250        500         30
      7566 JONES      DIRECTIVO       7839 02-APR-81       2975                    20
      7654 MARTIN     VENDEDOR        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      DIRECTIVO       7839 01-MAY-81       2850                    30
      7782 CLARK      DIRECTIVO       7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALISTA        7566 09-DEC-82       3000                    20
      7839 KING       DIRECTOR             17-NOV-81       5000                    10
PRUEBA

LISTA
    NUMERO NOMBREEMP  EMPLEO          JEFE FECHAENTR     SUELDO COMPLEMENTO     NUMDEP
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7844 TURNER     VENDEDOR        7698 08-SEP-81       1500          0         30
      7876 ADAMS      ORDENANZA       7788 12-JAN-83       1100                    20
      7900 JAMES      ORDENANZA       7698 03-DEC-81        950                    30
      7902 FORD       ANALISTA        7566 03-DEC-81       3000                    20
      7934 MILLER     ORDENANZA       7782 23-JAN-82       1300                    10



PRUEBA
```

Figura 3.16. Utilización de cabeceras y pies.

La numeración de páginas se consigue utilizando la variable de sistema **SQL.PNO**, tal y como se muestra en este ejemplo.

```
TTITLE LEFT 'PAGINA Nº: ' SQL.PNO
```

A menudo es necesario crear scripts SQL con sentencias que realicen ciertas operaciones. Para ello, necesitaremos una forma para poder guardar y recuperar estos archivos.

Podemos crear o modificar nuestro archivo de sentencias SQL desde el editor que tengamos configurado por omisión sin más que escribir

EDIT *archivo*

donde *archivo* hará referencia al nombre completo del archivo, incluyendo la ruta de acceso.

Para crear comentarios en un archivo de sentencias SQL, podemos utilizar **REMARK** al principio de una línea o **- -** (dos guiones seguidos) precediendo al comentario.

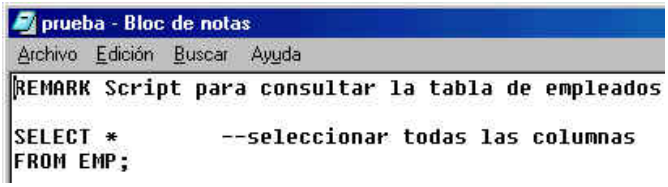

Figura 3.17. Comentarios en SQL*Plus.

Si lo que queremos es crear nuestro archivo de órdenes SQL desde el mismo prompt de SQL*Plus, podemos utilizar

SAVE *archivo*

La orden *SAVE* guarda el contenido del buffer en un archivo, pero como el buffer solo guarda una sentencia SQL o un bloque PL/SQL, sólo utilizaremos *SAVE* cuando sólo queramos guardar una instrucción.

Pero, si lo que queremos hacer es crear un archivo con varias especificaciones de configuración para una consulta, si podremos incluir estas líneas en un archivo. Esto se consigue con la orden

INPUT

Como en el búfer sólo cabe una instrucción SQL, y sólo hablamos de una instrucción SQL (aunque con varias órdenes de configuración), esta opción nos vale perfectamente para lo que deseamos. Con esta orden todo lo que escribamos pasará al búfer, y una vez que hayamos finalizado podremos guardar nuestras órdenes.

Para abrir un archivo escribiremos

GET *archivo*

Lo único que hace esta orden es cargar en el búfer el archivo que hayamos especificado. Si lo que queremos es ejecutarlo escribiremos **RUN**, o bien, **START**.

Por último, cuando deseemos enviar los resultados de nuestras consultas a un archivo escribiremos

SPOOL *archivo*

Para desactivar el envío de resultados al archivo especificado, hay que escribir

SPOOL OFF

Para evitar volver a introducir órdenes de configuración del entorno (p.e. líneas por página), podemos guardar nuestras preferencias en un archivo y recuperarlas cada vez que utilicemos SQL*Plus.

Guardaremos nuestras preferencias (el contenido de todas las variables de entorno) con

STORE SET *archivo*

Para activar nuestras preferencias sólo hay que cargarlas (@).

SQL*Plus permite evitar la presentación de valores repetidos de forma que los resultados sean más legibles. Esto se realiza con

BREAK ON *columna* [**SKIP** *n*|**SKIP PAGE**]

Veamos que quiere decir esto con la ayuda de un ejemplo.

```
SQL> BREAK ON EMPLEO
SQL> SELECT *
  2  FROM EMP
  3  ORDER BY EMPLEO, NOMBREEMP;

    NUMEMP NOMBREEMP  EMPLEO          JEFE FECHAENTR     SUELDO COMPLEMENTO     NUMDEP
---------- ---------- --------- ---------- --------- ---------- ----------- ----------
      7902 FORD       ANALISTA        7566 03-DEC-81       3000                      20
      7788 SCOTT                      7566 09-DEC-82       3000                      20
      7698 BLAKE      DIRECTIVO       7839 01-MAY-81       2850                      30
      7782 CLARK                      7839 09-JUN-81       2450                      10
      7566 JONES                      7839 02-APR-81       2975                      20
      7839 KING       DIRECTOR             17-NOV-81       5000                      10
      7876 ADAMS      ORDENANZA       7788 12-JAN-83       1100                      20
      7900 JAMES                      7698 03-DEC-81        950                      30
      7934 MILLER                     7782 23-JAN-82       1300                      10
      7369 SMITH                      7902 17-DEC-80        800                      20
      7499 ALLEN      VENDEDOR        7698 20-FEB-81       1600        300           30
      7654 MARTIN                     7698 28-SEP-81       1250       1400           30
      7844 TURNER                     7698 08-SEP-81       1500          0           30
      7521 WARD                       7698 22-FEB-81       1250        500           30

14 rows selected.
```

Figura 3.18. Utilización de BREAK para facilitar la comprensión de los resultados.

Los parámetros opcionales *SKIP n* y *SKIP PAGE* se utilizan, respectivamente, para dejar *n* líneas en blanco entre cada cambio, o para insertar un salto de página entre cada cambio.

Tenga en cuenta que BREAK actúa sobre cada cambio de la columna sobre la que se hace el BREAK, por lo que es vital ordenar por dicha columna.

Para eliminar esta presentación organizada basta con escribir

CLEAR BREAKS

Si queremos obtener subtotales o resultados parciales sobre grupos podemos realizarlo con la ayuda de *COMPUTE*, combinándola con *BREAK*. A diferencia de las operaciones sobre grupos del *GROUP BY*, *COMPUTE* sólo actúa a la hora de presentar

los resultados, pero no son los auténticos resultados de la consulta, sino que toman estos resultados como fuente de datos para procesarlos mediante una operación matemática.

Su sintaxis es

COMPUTE *función* [**LABEL** *etiqueta*] **OF** *columna* **ON** *columnaBreak*

donde *función* puede ser SUM, MINIMUM, MAXIMUM, AVG, STD, VARIANCE, COUNT o NUMBER. El parámetro *columna* hace referencia a la columna sobre la que se realiza la operación *función*, mientras que *columnaBreak* es la que indica cuando se realiza la operación (a cada cambio de valor de esa columna).

Veámoslo con un ejemplo.

```
SQL> break on numdep skip 1
SQL> compute sum label SUMA of sueldo on numdep
SQL> select numdep, numemp, nombreemp, sueldo
  2  from emp
  3  order by numdep;

    NUMDEP     NUMEMP NOMBREEMP      SUELDO
---------- ---------- ---------- ----------
        10       7782 CLARK            2450
                 7839 KING             5000
                 7934 MILLER           1300
********** ----------
SUMA                                   8750

        20       7369 SMITH             800
                 7876 ADAMS            1100
                 7902 FORD             3000
                 7788 SCOTT            3000
                 7566 JONES            2975
********** ----------
SUMA                                  10875

        30       7499 ALLEN            1600
                 7698 BLAKE            2850
                 7654 MARTIN           1250
                 7900 JAMES             950
                 7844 TURNER           1500
                 7521 WARD             1250
********** ----------
SUMA                                   9400


14 rows selected.
```

Figura 3.19. Utilización de subtotales.

Para realizar una de estas operaciones sobre todo el informe haremos lo siguiente

BREAK ON REPORT

COMPUTE *función* **[LABEL** *etiqueta***] OF** *columna* **ON REPORT**

También podemos realizar varias operaciones sobre una misma columna de esta forma

COMPUTE *funciones* **OF** *columna* **ON** *columnaBreak*

```
COMPUTE SUM AVG OF SUELDO ON NUMDEP
```

Aunque en esta sentencia también se puede utilizar una etiqueta, no tendría mucho sentido, ya que no sabríamos a qué operación hace referencia.

Por último, también podemos realizar subtotalizar varias columnas. En este caso, utilizaríamos algo así

COMPUTE *función* **OF** *columna1 columna 2 ...* **ON** *columnaBreak*

```
COMPUTE SUM OF SUELDO COMPLEMENTO ON NUMDEP
```

Para ver los subtotales definidos escriba

COMPUTE

Y para eliminar los subtotales definidos escriba

CLEAR COMPUTES

3.4. Developer 2000

Developer 2000 (D2K) es un conjunto de herramientas de desarrollo cliente/servidor de Oracle que permite la creación escalable de aplicaciones que se pueden ejecutar en la mayoría de las plataformas. Concretamente, incorpora un conjunto de herramientas para la creación de formularios, informes, gráficos, consultas, esquemas y procedimientos, y lo mejor de todo, es que la programación se realiza a través de una interfaz gráfica declarativa, por lo que se minimiza la escritura de código. Comencemos comentando brevemente las herramientas que integran Developer 2000.

3.4.1. Introducción a las herramientas de Developer 2000

Antes de comenzar a estudiar con detalle las herramientas de Developer 2000 que vamos a ver, comentemos brevemente las principales herramientas de D2K. Esto nos permitirá obtener una panorámica de D2K y nos ayudará a conocer las herramientas que lo integran y el propósito de cada una de ellas.

Project Builder le ayuda a organizar y mantener los distintos archivos de los que consta una aplicación. Además, puede utilizar las herramientas que desee a través esta herramienta, lo que facilita el proceso de desarrollo y documentación de su aplicación.

3.4.1.1. Conceptos básicos

Un proyecto es un conjunto de enlaces a los archivos que forman su aplicación. En un proyecto podrá definir editores, compiladores y otras herramientas que necesite en su proyecto.

El Asistente para proyectos (*Project Wizard*) proporciona una forma sencilla y rápida de crear un proyecto. Este asistente se utiliza para crear un archivo de registro del proyecto, en el que se guardan los archivos asociados a la aplicación. El asistente le guía a través de una serie de pasos, en los que indicará si se trata de un proyecto nuevo o se trata de un subproyecto, denominará a su proyecto, le asignará un directorio y especificará una conexión por omisión. Una vez completados estos pasos, ya podrá añadir los archivos que desee.

Form Builder es una herramienta de desarrollo de aplicaciones que permite a los usuarios acceder a los datos guardados en la base de datos.

En Form Builder se puede trabajar con tres tipos de módulos: formularios, menús y librerías. Un formulario es un conjunto de objetos y datos con los que interactúan los usuarios para manipular la base de datos. Un módulo de menú está formados por menús y código para estos menús, y los usuarios seleccionarán elementos del menú para realizar ciertas operaciones sobre la aplicación. Un módulo de librería es un código del cliente que puede ser compartido por varios módulos y aplicaciones.

El Explorador de objetos (*Object Navigator*) proporciona una vista jerárquica de los objetos de la aplicación.

El Asistente para bloques de datos (*Data Block Wizard*) le permite crear o modificar fácilmente los bloques de su aplicación.

Las Librerías de objetos (*Object Libraries*) proporcionan un método sencillo para la reutilización de objetos y ayudan al seguimiento de estándares dentro del equipo de desarrollo.

El Asistente de presentación (*Layout Wizard*) le permite organizar de forma sencilla los elementos de un bloque de datos. El asistente muestra los elementos en un marco (*frame*) o en un lienzo (*canvas*) y permite visualizarlos de acuerdo con varios estilos que posteriormente podrá personalizar.

El Cuadro de Propiedades (*Property Palette*) le permite configurar propiedades de los objetos de un módulo de un formulario o un menú. El Cuadro de Propiedades se actualiza automáticamente cada vez que se selecciona un objeto en el Explorador de objetos.

El Editor integrado de PL/SQL le permite escribir código PL/SQL desde Form Builder. Ofrece una interfaz gráfica para la edición y depuración de funciones y procedimientos cliente/servidor.

Graphics Builder le permite generar gráficos dinámicos que representen gráficamente los datos y ayuden a una mejor comprensión visual.

Una aplicación creada con Graphics Builder se denomina *display*. Un *display* contiene todos los componentes que se utilizan en la aplicación, incluyendo las definiciones de los orígenes de datos, los elementos visuales y su comportamiento. Cada *display* tiene un esquema, sobre el que se crean los objetos gráficos. El esquema consiste en una o varias capas que contienen los distintos elementos del *display*. Una vez que esté ejecutando el *display* podrá ocultar, mostrar y organizar estas capas de forma que el usuario pueda obtener distintas vistas. También son parte de los *displays* las construcciones PL/SQL y las consultas en las que están basados los *displays*.

Hay unos cincuenta tipos de gráficos definidos, y además puede utilizar un conjunto de herramientas de dibujo para la creación de *displays* personalizados, como puede ser un mapa interactivo. Los gráficos se pueden modificar con el Editor de presentación (*Layout Editor*).

El Explorador de objetos (*Object Navigator*) proporciona una representación jerárquica de los objetos del *display*.

El Asistente para gráficos (*Chart Wizard*) le permite crear fácilmente un gráfico, seleccionar una fuente de datos, y crear una consulta para especificar los datos que hay que representar en el gráfico.

El Cuadro de Propiedades (*Property Palette*) le permite configurar las propiedades de los objetos de los *displays*, y se actualiza automáticamente cada vez que se selecciona un objeto en el Explorador de objetos.

Report Builder le permite crear informes de calidad en un entorno cliente/servidor o en un entorno Web. Los informes pueden ser independientes o estar incluidos en formularios o *displays* del Graphic Builder.

Un informe es una colección de objetos que definen sus datos, aspecto e interfaz. Para crear un informe se utiliza el Asistente de informes (*Report Wizard*). Este asistente le guía en el proceso de selección de un tipo de informe, definición del origen de los datos y configuración de la presentación.

Una vez que termine con el Asistente de informes, el informe aparecerá en el Visualizador (*Live Previewer*). El Visualizador es una herramienta de edición WYSIWYG para facilitar las tareas de modificación del aspecto del informe. Para generar una salida HTML o PDF de su informe utilice el Asistente para Web (*Web Wizard*).

Para ejecutar informes dinámicamente en un cliente Web, se utiliza el Servidor de informes (*Reports Server*) junto con el *Reports Web Cartridge* o el *Web CGI*. Estos últimos le permiten enviar a su servidor Web enviar sus peticiones para ejecutarlas y recibir los resultados en el cliente.

Procedure Builder es un entorno integrado de desarrollo y mantenimiento de código para aplicaciones cliente/servidor.

Procedure Builder proporciona una interfaz gráfica para la creación, edición y compilación de código PL/SQL en el cliente y en el servidor.

El Explorador de objetos proporciona una representación jerárquica de los objetos que forman el procedimiento (unidades de programa, librerías y disparadores).

El Editor de unidades de programa (*Program Unit Editor*) proporciona un entorno de edición para el código de su procedimiento, donde podrá compilar y depurar código PL/SQL.

El Editor de disparadores (*Database Trigger Editor*) de la base de datos proporciona una interfaz gráfica para la creación de disparadores de la base de datos.

El Intérprete (*Interpreter*) es el lugar donde podrá depurar y probar unidades de programa simulando su comportamiento. En el Intérprete podrá utilizar puntos de ruptura para detener la ejecución de una unidad de programa, de forma que pueda depurar su código.

Las Librerías son conjuntos de unidades de programa que puede guardar en el cliente o en el servidor para su posterior reutilización por otras aplicaciones. Las librerías aumentan el rendimiento ya que no se cargan en memoria hasta que no se las llame (a una de sus unidades de programa).

Query Builder proporciona una forma sencilla de acceder a la información de las bases de datos de su organización de forma que pueda analizar su estructura. Query Builder presenta las tablas gráficamente, lo que proporciona una idea intuitiva para la creación gráfica de una consulta.

En Query Builder cada consulta aparece como un rectángulo en la ventana Consulta (*Query*). Las columnas de la tabla aparecen en la ventana Consulta debajo del nombre de la tabla.

La ventana Consulta está dividida en dos secciones, la de Condiciones (*Conditions*) y la de Origen de datos (*Datasource*). En la sección de Origen de datos se especifican las tablas y las columnas que participan en la consulta. Query Builder entiende que dos tablas están relacionadas si hay una línea que las une.

Con Query Builder podrá establecer condiciones múltiples, ordenar sus resultados y generar subtotales. Además podrá realizar cálculos sobre los datos para crear tablas de hipótesis y estimación (*what-if scenarios* y *forecasts*), cambiar tipos de fuente y formatos, imprimir y guardar los datos recuperados, así como exportarlos a otras aplicaciones (p.e Excel).

Los resultados de la consulta se presentan en la ventana de resultados (*Results*).

El Editor de datos (*Data Editor*) le permite realizar operaciones de actualización de la base de datos (inserción, actualización y eliminación). Sin embargo, la opción del Editor de datos suele estar desactivada, y no podrá acceder a ella si no tiene los privilegios suficientes.

Schema Builder es una herramienta gráfica que le permite crear, copiar, modificar y eliminar objetos y relaciones de una base de datos.

Un esquema es un conjunto de objetos de una base de datos a nivel lógico, como son tablas, relaciones, vistas e índices.

La interfaz de Schema Builder tiene un aspecto similar al de la ventana de Consulta en Query Builder, pero en lugar de permitirle realizar consultas sobres las tablas, Schema Builder le permite realizar operaciones de DDL. Sin embargo, como Schema Builder le permite modificar la estructura de la base de datos, no podrá acceder a él si no tiene los privilegios suficientes.

3.4.2. El proceso de desarrollo de una aplicación

Antes de pasar a estudiar las herramientas en las que nos vamos a centrar, es importante conocer cuál es el proceso que hay que seguir para desarrollar una aplicación. Este proceso está basado en técnicas de la Ingeniería del Software y cubre aspectos sobre el análisis, la planificación, el diseño, el desarrollo y la prueba del software. Por tanto, el seguimiento de este proceso permite generar un software de calidad, fácil de mantener y minimiza las posibilidades de rechazo de la aplicación.

La figura siguiente muestra las etapas de este proceso, y que serán comentadas a continuación

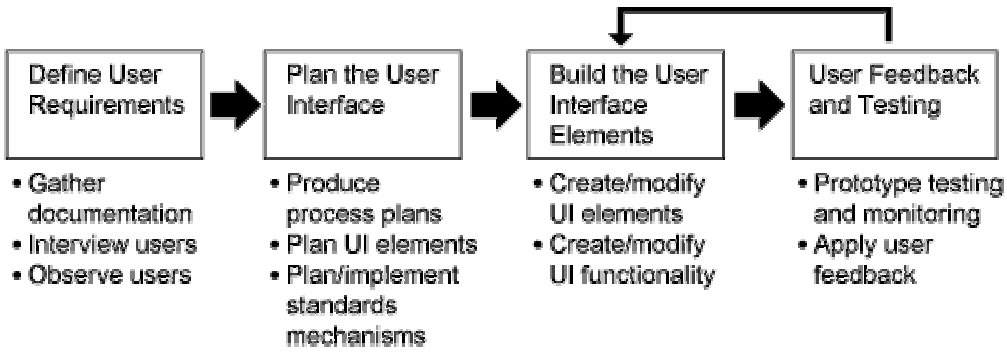

Figura 3.20. Proceso de desarrollo de una aplicación

Antes de comenzar con el desarrollo de una aplicación, hay que determinar lo que el usuario espera de la aplicación. Este análisis permitirá desarrollar una aplicación basada en las necesidades del usuario, lo que supone realizar un esfuerzo extra por comprender a los propios usuarios. Esto supone realizar una análisis del usuario y de las tareas que realiza el usuario. Para ello, podemos crear un modelo basado en las tareas que realiza cada usuario, indicando la secuencia de operaciones de cada tarea junto con los objetos que se utilizan para el desarrollo de cada una de ellas. Si no comprendemos realmente a los usuarios, nunca podremos desarrollar la aplicación que ellos necesitan.

Para definir los requerimientos del usuario:

- Recopilaremos información. Técnicas, procedimientos y manuales existentes del sistema (si ya estaba informatizado) que ayuden a comprender el sistema, de forma que se puedan realizar las entrevistas con una comprensión previa del sistema actual.
- Observaremos a los usuarios en su trabajo. Hay que confeccionar una lista de las tareas que realizan los usuarios y determinar el orden en que las realizan
- Entrevista a los usuarios. A la hora de realizar las entrevistas:
    - No le pregunte sólo lo que hacen, sino cómo lo hacen (p.e. sabemos que trabajan con informes, pero, ¿necesitan trabajar con más de un informe a la vez?)
    - Encontrar a qué usuarios no les gusta el sistema actual
    - Preguntar a los usuarios cómo esperan que sea el modo de operación de la aplicación
    - Determinar si los usuarios tienen algún tipo de discapacidad o hay alguna circunstancia especial que hay que considerar (p.e. idioma, red)
- Entrevista a varios tipos de usuarios y de varios tipos o perfiles

En la siguiente etapa, debemos planificar el desarrollo de la interfaz de forma que se ajuste a las necesidades del usuario. Esto implica:

- Desarrollar estándares a seguir en el equipo de desarrollo

- Considerar requerimientos y restricciones derivados de la plataforma
- Hacer un diseño previo de la interfaz (prototipo) indicando los tipos de elementos que se van a utilizar y comprobar que se ajusta a las necesidades del usuario

Una vez recopilada la información suficiente sobre los usuarios y las tareas que realizan y una vez que haya definido los estándares y los haya utilizado para desarrollar el modelo de su aplicación, podrá comenzar a desarrollar un prototipo de la interfaz de su aplicación.

Una vez que haya creado un prototipo, en papel o con D2K, muestre a los usuarios su trabajo y déjeles trabajar con él. Para ello puede:

- Crear algún cuestionario que permita la evaluación
- Elija un conjunto representativo de usuarios
- Capte las impresiones del usuario
- Utilice varias personas para interpretar los resultados del usuario

Tras captar las impresiones del usuario, si es necesario, vuelva a revisar la interfaz de usuario para que se ajuste a las críticas del usuario.

3.4.3. Diseño de un formulario con Form Builder

En esta sección veremos cómo crear un formulario sencillo y cómo crear un formulario maestro-detalle con Form Builder. Pero antes de eso, comentemos algunos conceptos básicos.

Los formularios creados con Form Builder están compuestos de lo que se denominan módulos. Concretamente hay cuatro tipos de módulos:

- Módulo Formulario (*Form*). Colección de objetos como ventanas, cuadros de texto, casillas de verificación, botones, cuadros de lista y bloques de código PL/SQL denominados disparadores.
- Módulo Menú. Colección de menús y elementos de menús (órdenes).
- Módulo Librería PL/SQL (*PL/SQL Library*). Colección de procedimientos, funciones y paquetes que pueden ser llamados por otros módulos de la aplicación.
- Módulo Librería de objetos (*Object Library*). Colección de objetos que se pueden utilizar para el desarrollo de una aplicación.

3.4.3.1. Formularios, bloques, elementos, regiones y marcos

Un **formulario** (o módulo formulario) es una aplicación que proporciona acceso a una fuente de datos. Al ver un formulario, se observan elementos de la interfaz como cuadros de texto y casillas de verificación, lo que permite interactuar con la fuente de datos. Estos elementos de la interfaz pertenecen a lo que se denomina un **bloque**.

En la figura siguiente, los campos Número, Ciudad y Nombre pertenecen al bloque Departamento, mientras que los campos Número, Nombre, Empleo, Jefe, Fecha entrada, Sueldo y Complemento pertenecen al bloque Empleados.

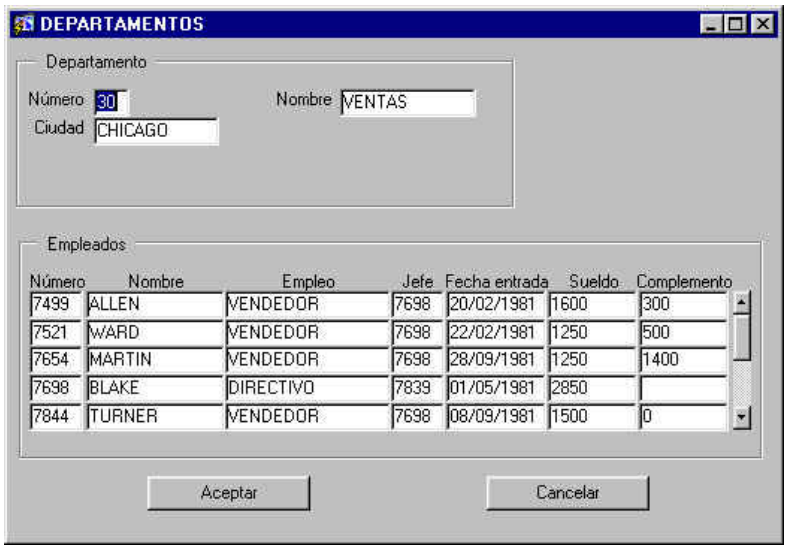

Figura 3.21. Un formulario

Hay dos tipos de bloques: *bloques de datos*, que se utilizan como puente entre el usuario y la fuente de datos, y *bloques de control*, que no están asociados a ninguna fuente de datos. Cada bloque de datos permite a un usuario acceder a los datos de una tabla, consulta o vista de la fuente de datos. Además, los bloques pueden tener un solo registro (como el bloque Departamento de la figura anterior), lo que significa que sólo se puede acceder a una fila en un instante, o pueden tener varios registros (como el bloque Empleados de la figura anterior), lo que significa que se pueden ver varios registros a la vez.

Una **región** es un rectángulo que agrupa de forma lógica ciertos campos dentro de un bloque, mientras que un **marco** es una forma de organización predefinida de elementos en un bloque. Concretamente el marco configura aspectos como los márgenes y distancias entre elementos.

3.4.3.2. Ventanas y *canvas*

Una **ventana** es el contenedor en el que se muestran todos los objetos visuales de un formulario. Un formulario puede constar de varias ventanas, pero lo más común es tener un formulario para cada ventana.

En Form Builder tenemos los siguientes tipos de ventanas:

- Contenedor (MDI). Contiene al resto de las ventanas. Suele contener a la barra de herramientas y al menú principal.
- No modales. Permiten al usuario interactuar con cualquier otra ventana, así como con la barra de herramientas y el menú.
- Modales. Obligan al usuario a trabajar con una única ventana, de forma que sólo pueda aceptar o cancelar los cambios que realice. La barra de herramientas y el

menú no son accesibles. Se suelen utilizar en los cuadros de diálogo de configuración de propiedades y en los asistentes.

Un **canvas** (*lienzo* o *superficie*) es el objeto implícito sobre el que aparecen los elementos de la interfaz. Un formulario puede tener varios *canvas*, como si se tratasen de las páginas de un formulario. Un *canvas* puede mostrar elementos de uno o más bloques. Para poder ver los elementos de un *canvas*, tiene que colocar el *canvas* en una ventana.

Existen cuatro tipos de lienzos:

- *Canvas* de contenido. Ocupan toda la ventana. Cada ventana tiene al menos un lienzo de contenido.
- *Canvas* apilados. Aparecen apilados sobre el lienzo de contenido. Se utilizan para ocultar zonas de un lienzo de contenido.
- *Canvas* con fichas. Un conjunto de fichas que le permiten agrupar y presentar una gran cantidad de información relacionada en el espacio de un solo lienzo.
- *Canvas* de barras de herramientas. Se utilizan para crear barras de herramientas.

Cada ventana puede presentar un *canvas* o más, y además se pueden presentar de forma condicional, dependiendo de si se cumplen ciertas condiciones.

3.4.3.3. Creación de un formulario

Puede crear un formulario de varias formas.

- Ejecutando Form Builder. Esto le lleva al cuadro de diálogo de bienvenida de Form Builder que se muestra en la figura siguiente. En este cuadro de diálogo podrá:
    - Utilizar el Asistente para bloques de datos, que le asiste en el proceso de creación de un bloque de datos.
    - Crear un formulario de forma manual, que muestra directamente el Explorador de objetos para cree un formulario a partir de uno vacío.
    - Crear un formulario a partir de una plantilla.

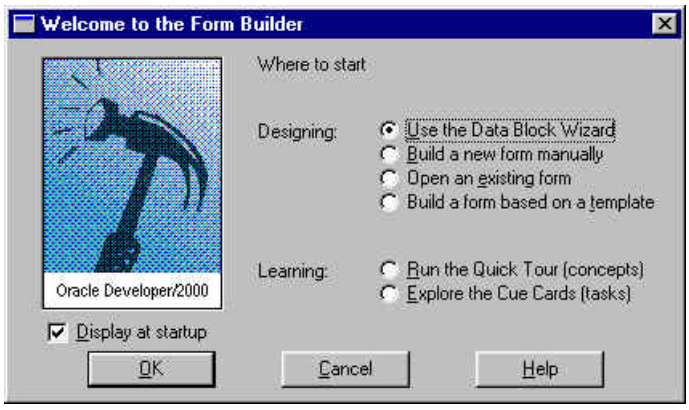

Figura 3.22. Cuadro de diálogo de bienvenida de Form Builder.

- Si ya está ejecutando Form Builder, puede crear un nuevo formulario de alguna de estas formas:
  - Seleccionar **File|New|Form** del menú
  - Situar el ratón sobre **Forms** en el Explorador de objetos, y pulsar sobre el botón **Create**. 

Bueno, comencemos a crear nuestro formulario, y comencemos aprendiendo a cambiar su nombre.

Para cambiar el nombre de un formulario, basta con pulsar una vez sobre el nombre del formulario en el Explorador de objetos, una vez que esté seleccionado. El método es el mismo que el que se sigue para cambiar el nombre de un archivo en Windows 9x.

Una vez que hemos aprendido a cambiar el nombre de un formulario, veamos cómo podríamos acceder al Cuadro de Propiedades.

Para cada objeto que veamos en Form Builder, y en general en el resto de las herramientas de desarrollo de D2K, podemos manipular sus propiedades a través del Cuadro de Propiedades, que aparece en la ilustración siguiente.

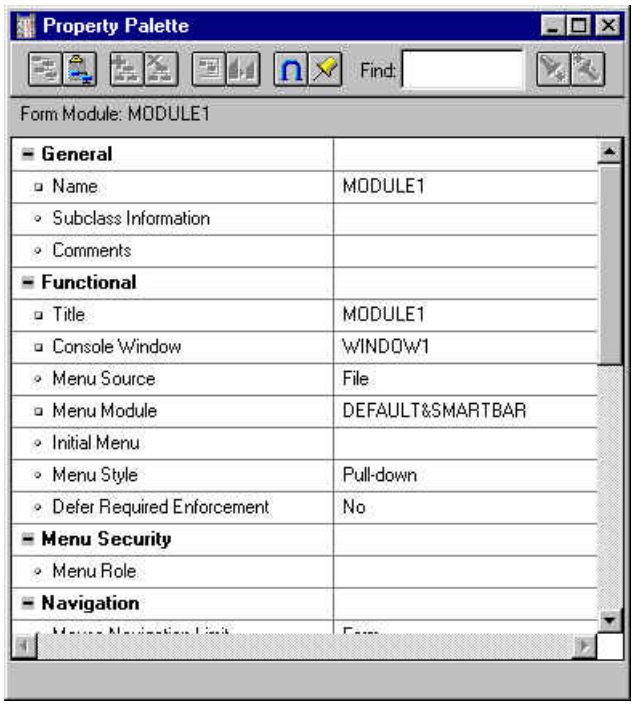

Figura 3.23. Cuadro de Propiedades.

Para activar el Cuadro de Propiedades, basta con pulsar dos veces en el Explorador de objetos sobre el objeto cuyas propiedades queremos configurar o consultar. Esta acción abrirá el Cuadro de Propiedades y mostrará las propiedades del objeto activo junto con sus valores, de forma que si en el Explorador de objetos seleccionamos otro objeto, el Cuadro de Propiedades mostrará las propiedades del nuevo objeto seleccionado.

Veamos ahora cómo crear un bloque de datos nuevo.

Un formulario consta de uno o más bloques de datos y bloques de control. Ahora que ya sabemos crear módulos de formulario, vamos a ver cómo podemos crear los bloques de datos del formulario.

NOTA: La creación de un bloque de datos supone la creación del propio bloque de datos y la creación de su diseño para su presentación en el formulario. Puede crear los bloques de datos de forma manual o utilizando los asistentes de Form Builder.

Veamos cómo crear un bloque de datos y su presentación para la tabla de departamentos utilizando el Asistente para bloques de datos (*Data Block Wizard*) y el Asistente de presentación (*Layout Wizard*).

Lo primero que tenemos que hacer, a no ser que ya lo hayamos hecho, es conectarnos a la base de datos. Para ello, introduciremos nuestro nombre de usuario, contraseña y nombre de la base de datos.

A continuación iniciaremos el Asistente para bloques de datos seleccionado **Tools|Data Block Wizard**. Una vez pasada la pantalla de bienvenida, hay que indicar al asistente si nuestro bloque de datos está basado en una tabla o vista, o bien está basado en un procedimiento, tal y como muestra la figura siguiente.

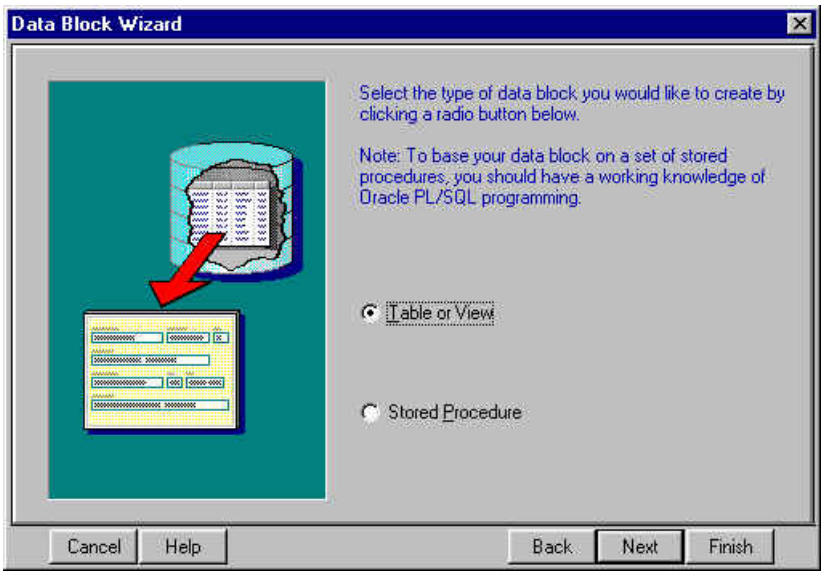

Figura 3.24. Selección del origen de datos del bloque de datos.

Para nuestro ejemplo, seleccionaremos **Table or View**, ya que vamos a crear un bloque de datos para la tabla de departamentos.

A continuación, aparecerá un nuevo paso del asistente en el que indicaremos cuál es la tabla o vista sobre la que vamos a construir el bloque de datos, y cuáles son las columnas que vamos a incluir en nuestro bloque de datos.

Para ello, pulsaremos el botón **Browse** para examinar las tablas de nuestra base de datos y elegir la tabla de departamentos. A continuación aparecerán en el cuadro de **Available Columns** las columnas de nuestra tabla, y podremos seleccionar cuáles son las columnas que queremos que formen nuestro bloque de datos. En este ejemplo pasaremos todas al cuadro de **Database Items**, tal y como muestra la figura siguiente.

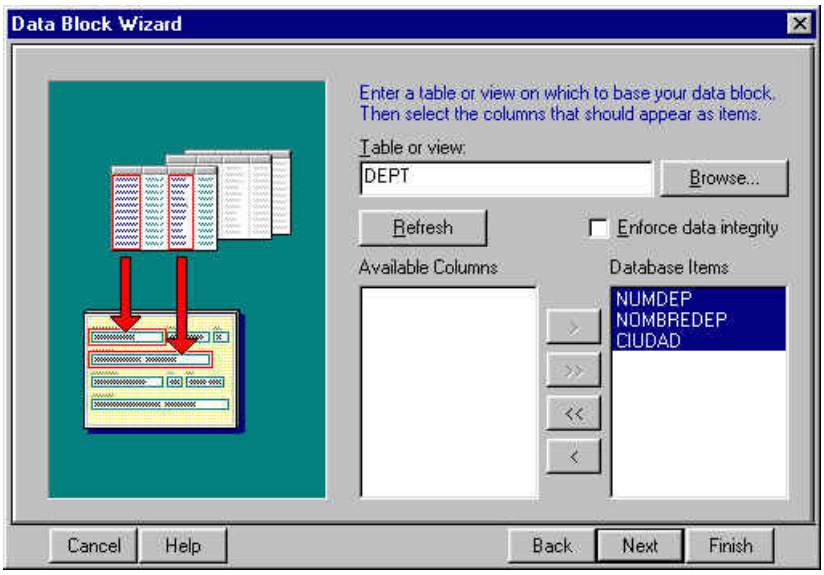

Figura 3.25. Selección de los elementos del bloque de datos.

Una vez seleccionados los campos del bloque de datos, pasamos al paso siguiente en el que podemos elegir entre lanzar el Asistente de presentación de nuestro bloque de datos, o detener el proceso. En nuestro caso elegiremos seguir con el Asistente de presentación, tal y como muestra la figura siguiente.

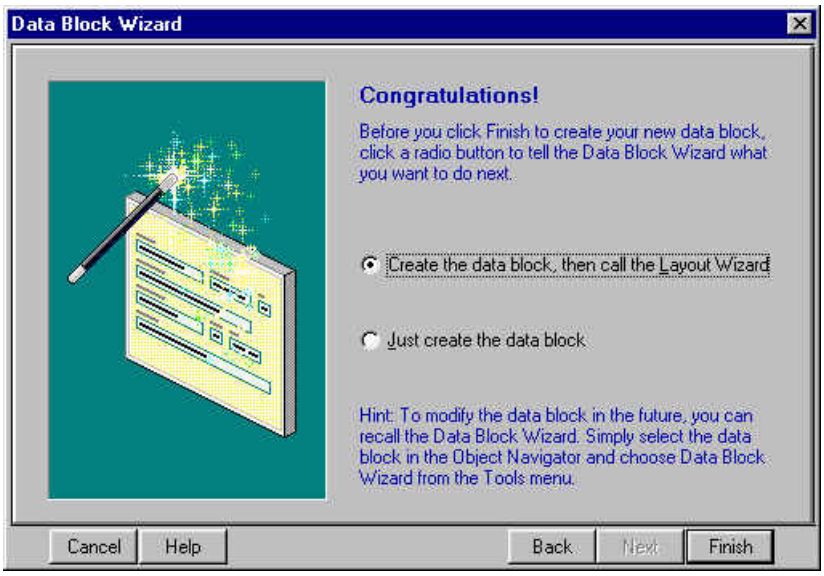

Figura 3.26. Finalización de la creación del bloque de datos y llamada al Asistente de presentación.

A continuación vamos a crear una presentación para nuestro bloque de datos, y comenzaremos seleccionando el tipo de *canvas* que vamos a crear. El tipo puede ser *Content*, para ser uno principal o continente, *Stacked, Vertical Toolbar* u *Horizontal Toolbar* para la creación de barras de herramientas, o bien del tipo *Tab* si lo que queremos es crear un *canvas* con fichas. En nuestro caso elegiremos el tipo *Content*, tal y como muestra la figura y a continuación pasaremos al paso siguiente.

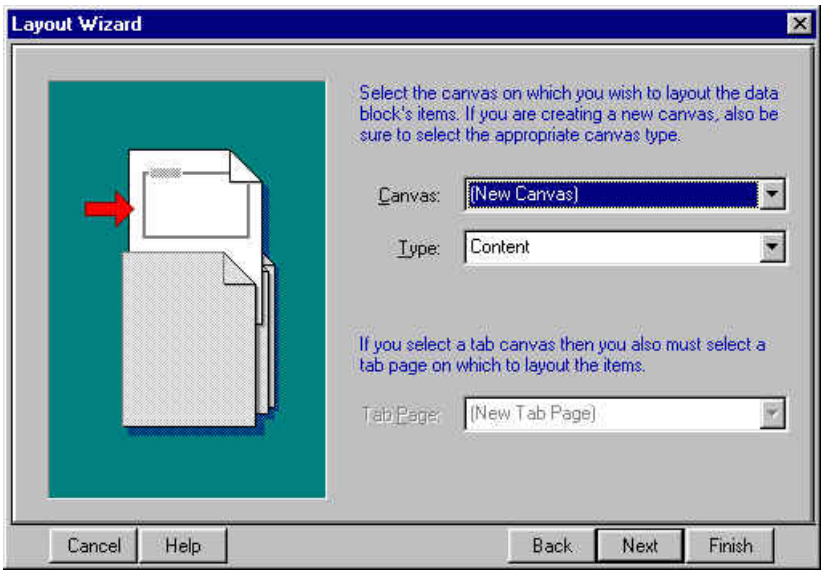

Figura 3.27. Elección del tipo de *canvas*.

En este paso tendremos que seleccionar los elementos del bloque de datos que queremos incluir en el *canvas*, o lo que es lo mismo, los campos que queremos mostrar. Para ello, basta con seleccionar los elementos del cuadro **Available Items** y pasarlos al cuadro **Displayed Items**, tal y como muestra la figura siguiente.

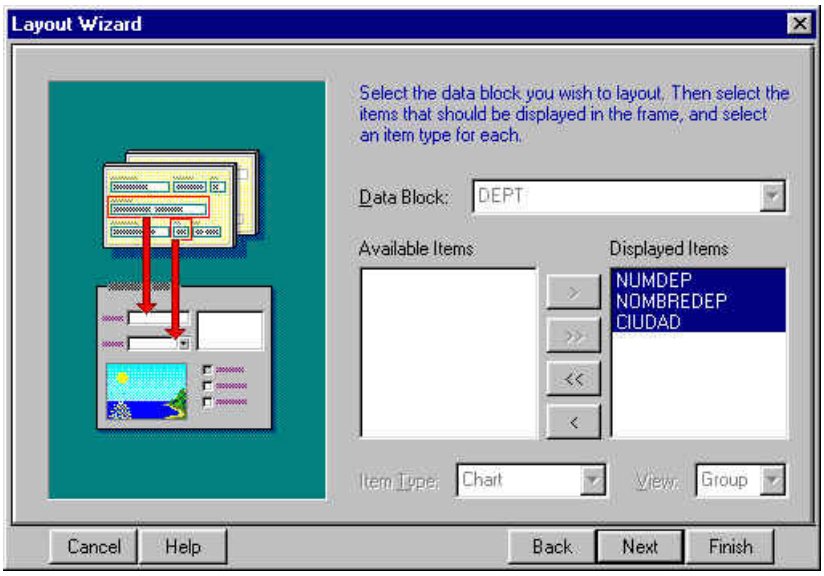

Figura 3.28. Selección de los elementos mostrados en el *canvas*.

A continuación, pulse el botón **Finish**, y obtendrá un resultado como el siguiente.

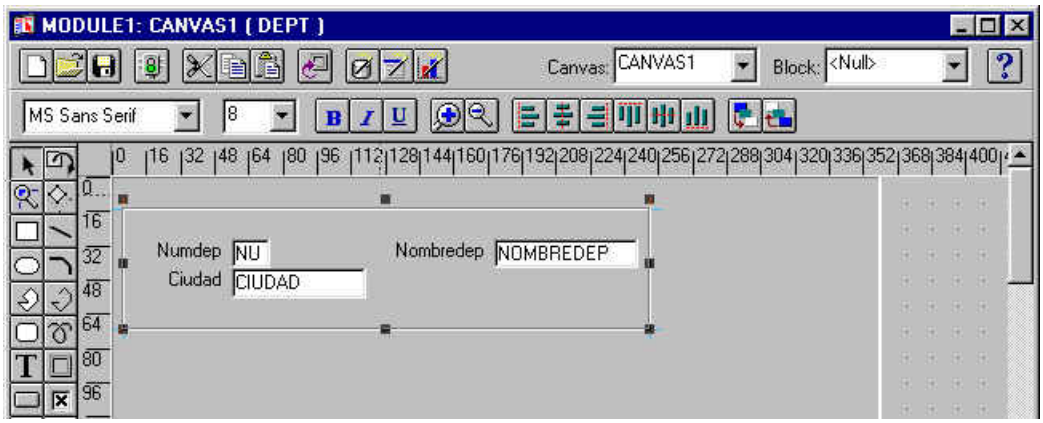

Figura 3.29. Resultado de crear un bloque de datos y definirle una presentación.

En esta ventana podemos modificar el diseño de nuestro formulario, y desde el menú podemos guardarlo con **File|Save** y ejecutarlo con **Program|Run Form**. Antes de ejecutarlo cambiemos las etiquetas de los campos de forma que sean más descriptivas. Esto lo podemos hacer modificándolas directamente o bien a través de la propiedad **Prompt** del Cuadro de Propiedades. Cambie las etiquetas a *Número* y *Departamento*.

Si ahora ejecuta el formulario verá que el aspecto se ajusta a lo que aparecía en la ventana de diseño, pero hay algunas cosas que habría que modificar, tal y como el nombre de la ventana, y cómo hacer que aparezcan los registros.

Para modificar el nombre de la ventana, tiene que cambiar la propiedad **Name** del objeto Ventana en el que esté definido el *canvas* (p.e. cambiar **WINDOW1** por **DEPARTAMENTOS**). En cuanto a la razón por la que no aparecen los registros, lo vamos a ver a continuación.

Cuando esté ejecutando un formulario con el runtime, verá que sus registros no aparecen automáticamente al presentarse el formulario. Esto se debe a que el runtime de Form Builder está esperando a que introduzca un nuevo registro o a que introduzca una consulta para mostrar la información solicitada.

Lo que hay que hacer para ver todos los registros de un formulario es introducir una consulta y ejecutarla. Para ejecutar una consulta sencilla bastará con introducir los parámetros en los cuadro de edición correspondientes y lanzar la consulta pulsando el botón **Execute Query** de la barra de herramientas. En esta barra de herramientas también nos encontramos los botones de **Enter Query** para especificar una nueva consulta, y el botón **Cancel Query** que cancela la introducción de una consulta. La figura siguiente muestra estos tres botones.

Execute Query **Enter Query** **Cancel Query**

Figura 3.30. Botones de consultas.

Para mostrar en el formulario todos los registros, pulsaremos el botón de ejecutar consulta, y se ejecutará sin ninguna condición, por lo que se mostrarán todos los registros, tal y como se muestra en la figura siguiente.

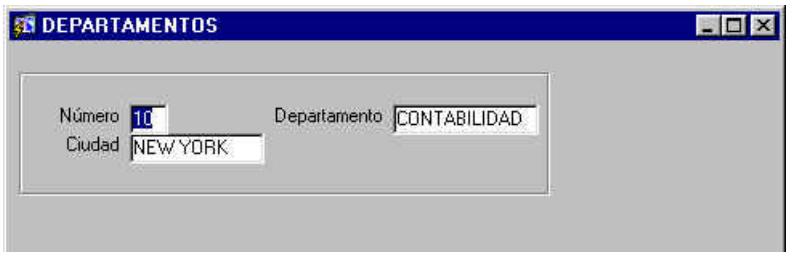

Figura 3.31. Presentación de todos los registros.

Ahora puede utilizar los botones de navegación por los registros para desplazarse por los resultados.

Para introducir consultas que muestren información concreta, habilitaremos el modo de introducción de consultas del runtime pulsando el botón de introducir consulta, y escribiremos en el formulario los valores que queremos que cumplan los registros resultantes, teniendo en cuenta que, en principio, la consulta se hace con comparación exacta de cadenas (sensible a las mayúsculas) y equivale a un AND lógico. Por tanto, para conocer cuáles son los departamentos de Dallas, habilitaremos el modo de introducción de consultas en el formulario, escribiremos DALLAS en el campo Ciudad, y pulsaremos el botón **Execute Query**. Este proceso se muestra en la figura siguiente.

Activación del modo de introducción de consultas

Introducción de parámetros

Presentación de resultados

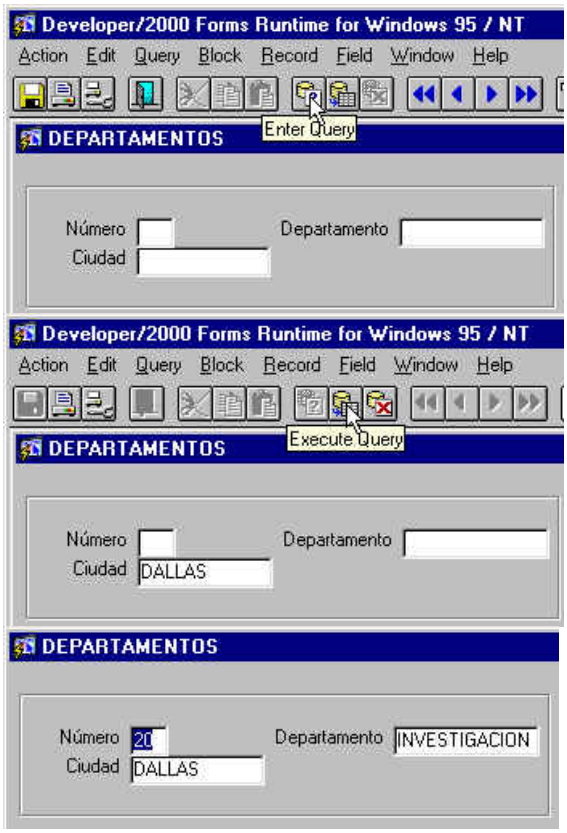

Figura 3.32. Información de departamentos de Dallas.

Si desea especificar varias condiciones para que se cumplan simultáneamente en varios campos, tendrá que introducir los valores deseados en cada uno de los campos correspondientes. Este tipo de consulta que hemos visto, sólo permite especificar condiciones AND y con un solo valor de comparación por campo.

A continuación vamos a ver cómo podemos especificar condiciones OR y varias condiciones por campo.

El proceso consta de dos partes:

- Definición de 'variables' en los campos sobre los que queremos definir las condiciones escribiendo en cada campo (*:nombreVariable*)
- Especificación de los predicados del WHERE en el Editor de predicados WHERE (*Query/Where*), utilizando los nombres de variable anteriores.

Por ejemplo, para obtener la información sobre los departamentos de los estados de Dallas y Nueva York, introduciremos en el campo **Ciudad** :C. Esto define una variable (para eso se incluyen los dos puntos) denominada *C*, que es la que vamos a utilizar en el Editor de predicados WHERE.

Por tanto, el resultado de nuestro primer paso sería algo así.

Figura 3.33. Definición de una variable para especificar una condición múltiple sobre Ciudad.

Si ahora pulsa el botón **Execute Query** aparecerá el cuadro de diálogo Query/Where, en el que debe especificar su predicado WHERE en SQL, utilizando la variable junto los dos puntos que ha definido en el formulario, tal y como se muestra en la figura siguiente. Además, en este editor podrá hacer búsquedas y sustituciones de cadenas.

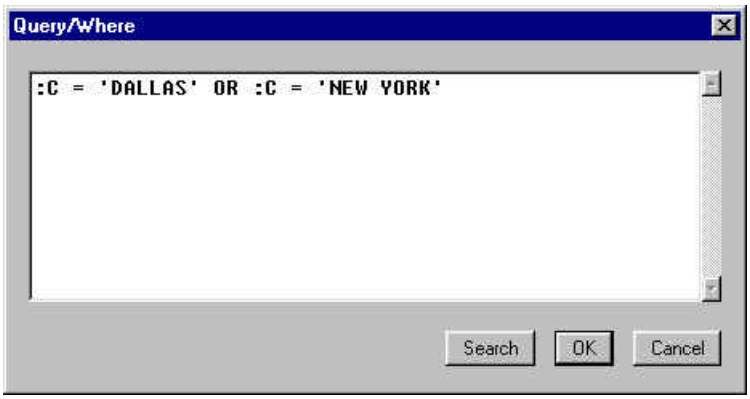

Figura 3.34. Definición de predicados múltiples.

Una vez que finalice la introducción de la consulta, pulse **OK** y aparecerán en su formulario los registros que cumplen la condición que ha especificado.

Para editar la consulta basta con volver a introducir una consulta (puede ser la misma) que utilice variables de este tipo, y el cuadro de diálogo que aparece es el mismo. La figura siguiente muestra una modificación realizada a la consulta para que presente los resultados ordenados por Estado. En cuando a las ordenaciones, las puede realizar directamente sobre el nombre de la columna, sin necesidad de utilizar un nombre de variable, tal y como se muestra en la figura siguiente.

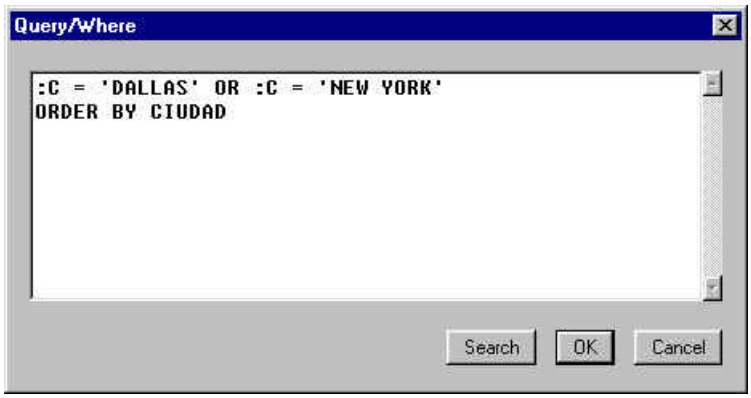

Figura 3.35. Especificación de un predicado WHERE junto con una cláusula de ordenación.

Un formulario no vale exclusivamente para la visualización de datos y ejecución de consultas, sino que también se utilizan para la modificación de la base de datos. Por tanto, las operaciones de inserción, modificación y eliminación de SQL se pueden hacer de una forma más intuitiva a través de la interfaz de un formulario.

Antes de pasar a ver cómo se realizan estas tres operaciones, decir que los cambios que se realicen tienen carácter temporal, y que sólo se hacen persistentes (se guardan realmente en la base de datos) cuando se pulse el botón **Save** de la barra de herramientas, o bien seleccione **Action|Save**. Si lo que desea es cancelar los cambios, seleccione **Action|Clear All**.

Para insertar un registro, basta con situarse sobre un registro en blanco (el siguiente al último), o bien pulsar el botón **Insert Record** de la barra de herramientas, o bien seleccionar **Record|Insert**. A continuación, inserte los valores que desee en los campos correspondientes.

Para eliminar un registro, primero deberá situarse sobre el registro que desea eliminar (p.e. realizando una consulta) y luego pulse el botón **Remove Record**, o bien seleccione **Record|Remove**.

Para actualizar un registro, sólo tendrá que situarse sobre el registro que desee modificar (p.e. haciendo una consulta) y realizar las modificaciones necesarias.

3.4.3.4. Creación de formularios Maestro-Detalle

A menudo es necesario la presentación o la introducción de detalles para un grupo de registros iterando sobre un mismo registro, como puede ser el caso de presentar en un formulario todos los departamentos, y en el que para cada departamento podamos iterar sobre cada uno de sus empleados.

Esta idea facilita la interacción con la base de datos y a estos formularios se les denomina Maestro-Detalle, ya que el formulario está basado en un formulario Maestro, en el que podemos ver cada uno de sus registros de detalle. Para la interconexión de la parte maestro y de la parte detalle se utilizan las claves externas, es decir se presentan todos los registros de la parte detalle en los que los valores de la clave externa en la tabla que actúa como detalle coincida con los valores de la clave en la tabla que actúa como maestro. Esta idea de formularios no tiene una configuración única, ya que el formulario detalle, puede ser a su vez maestro de otro formulario, o un formulario

maestro puede ser maestro de formularios detalle, tal y como se ilustra en la figura siguiente.

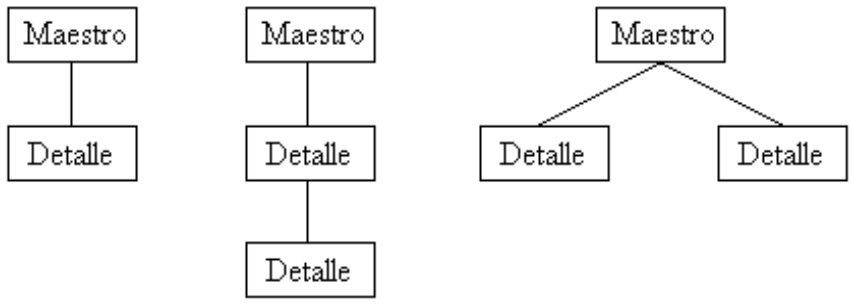

Figura 3.36. Relaciones Maestro-Detalle.

Una vez introducida la idea de formularios Maestro-Detalle, veamos cómo podemos realizar esto con Form Builder.

Para ello necesitaremos haber definido el bloque maestro, que en nuestro caso nos basta con la definición del bloque de datos DEPT que definimos anteriormente. A continuación crearemos un nuevo bloque de datos para la parte detalle utilizando el Asistente para Bloques de datos.

En el cuadro de diálogo en el que hay que seleccionar el origen de los datos, seleccionaremos la tabla EMP, y en el cuadro **Database Items** incluiremos todos los campos de la tabla. A continuación, en el cuadro de diálogo siguiente marcaremos la casilla de verificación **Auto-join data blocks**, para que una vez seleccionado el bloque maestro, realice la conexión de la parte detalle con la parte maestro a través de la clave externa. Ahora pulsaremos el botón **Create Relationship** y en el cuadro de diálogo que aparece, seleccionaremos la tabla DEPT (a la que hace referencia la clave externa de EMP), y veremos que el cuadro de **Join condition** se rellena automáticamente con la condición de producto natural, quedando como la figura siguiente.

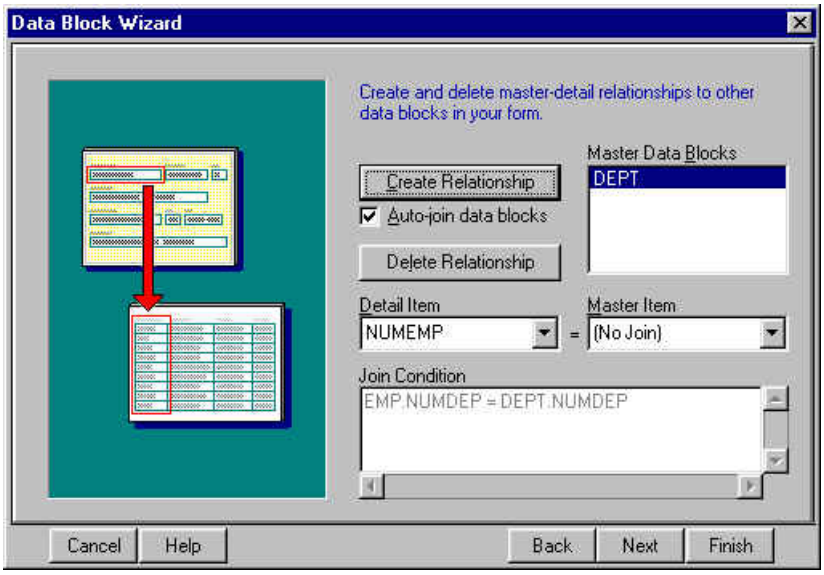

Figura 3.37. Creación de una relación Maestro-Detalle.

Una vez finalizada la creación del bloque de datos, falta crear el diseño de su presentación, y a continuación detallamos los pasos que vamos a seguir.

El primer paso importante es elegir si queremos presentar nuestro detalle junto a la parte maestro (en el mismo *canvas*), o lo queremos presentar en un cuadro aparte (en otro *canvas* nuevo). Nosotros elegiremos el mismo *canvas* que es la forma más común.

En el paso siguiente, nos aseguraremos que el bloque de datos seleccionado es el de EMP, y a continuación pasaremos al cuadro de **Displayed Items** todos los campos excepto NUMDEP, puesto que ya aparece en la parte maestro, de forma que nuestro cuadro de diálogo quedaría así.

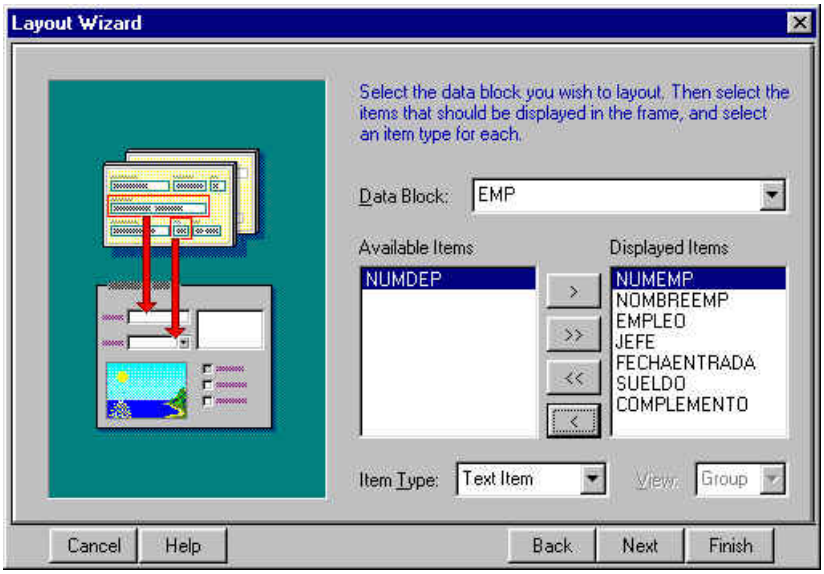

Figura 3.38. Definición de los campos a presentar en la parte detalle.

El paso siguiente se utiliza para modificar el tamaño de los cuadros de texto, que dejaremos con los valores por omisión, y en el cuadro de diálogo siguiente podemos elegir si la parte detalle se presenta en formato de formulario (ficha por cada detalle), o bien en formato tabular. Nosotros elegiremos el formato tabular que es el más común para formularios Maestro-Detalle, ya que permite ver varios registros de detalle por cada registro del maestro.

A continuación, el asistente muestra un cuadro de diálogo para que introduzcamos el nombre que deseamos que aparezca en la parte detalle, el número de registros que queremos ver simultáneamente en la parte detalle, y si deseamos presentar una barra de desplazamiento para poder ver registros que no quepan en el área definida. En nuestro caso, elegiremos como título **Empleados**, como número de registros presentados **5**, y además presentaremos la barra de desplazamiento.

Una vez que haya finalizado el diseño del formulario Maestro-Detalle, podrá modificar su diseño, y el resultado final deberá ser similar al que se muestra en la figura siguiente.

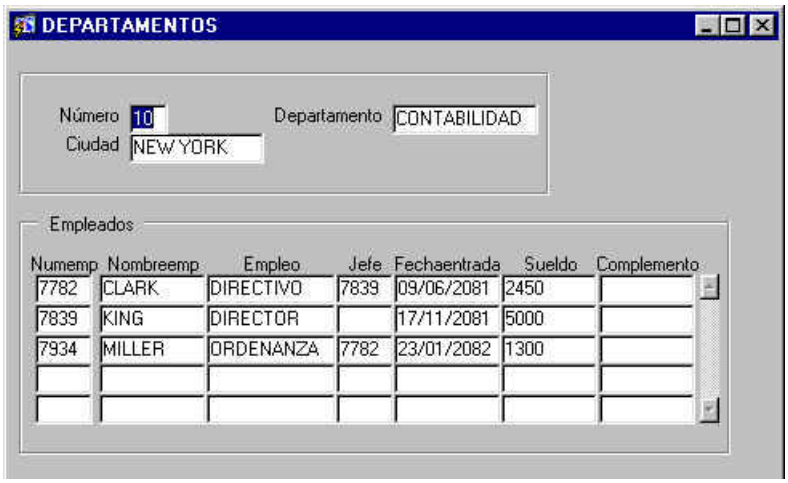

Figura 3.39. Formulario Maestro-Detalle.

Al crear una relación Maestro-Detalle entre dos bloques de datos, se crea un objeto Relation en el bloque de datos que actúa como maestro. Las propiedades más importantes de este objeto son:

- *Detail Data Block*: Indica cuál es el bloque de datos que actuará como detalle.
- *Join Condition*: Indica la condición de producto natural entre los bloques de datos.
- *Delete Record Behavior*: Especifica cómo afecta al detalle la eliminación de un registro del bloque de datos maestro.
    - *Non-Isolated*: Previene la eliminación de un registro maestro si hay registros detalle.
    - *Isolated*: Permite la eliminación de registros del bloque maestro sin afectar a los registro del detalle.
    - *Cascading*: Realiza una eliminación en cascada de los registros del bloque de datos del detalle al eliminar un registro del bloque de datos maestro. En una relación maestro-detalle de varios niveles, las eliminaciones no se propagan de forma automática.
- *Prevent Masterless Operations*: Previene la inserción de registros en el detalle si no hay registros relacionados en el maestro. Además, impide que se realicen consultas sobre el detalle sin especificar condiciones en el maestro, debido a las relaciones M:N.

Para crear nuevas relaciones maestro-detalle, basta con repetir la creación del bloque detalle desde el principio, teniendo en cuenta que el bloque maestro siempre tendrá que estar creado previamente.

3.4.3.5. Control de las propiedades de los bloques de datos

Hasta ahora, nos hemos limitado a crear bloques de datos indicando cuál era el origen del bloque de datos, y las columnas que queríamos incluir, pero no hemos hablado nada del resto de las propiedades.

Pese a que los bloques de datos tienen muchísimas propiedades, no es nuestro objetivo conocer cada una de ellas, pero sí conocer el uso de las más frecuentes e importantes. Es por ello que vamos a dedicarle a continuación un pequeño espacio a dichas propiedades.

- Propiedades sobre el estilo de navegación
    - *Navigation Style*: Indica a dónde se dirige el enfoque cuando se abandona el primer o el último campo de un registro. Puede seguir en el mismo registro (*Same Record*), cambiar de registro (*Change Record*), o cambiar de bloque de datos (*Change Data Block*).
    - *Previous/Next Navigation Data Block*: En el caso de que en la propiedad *Navigation Style* elija *Change Data Block*, con estas propiedades puede indicar cuáles serán los bloques que recibirán el enfoque al cambiar.
- Propiedades sobre la base de datos
    - *Database Data Block*: Indica si el bloque es un bloque de datos o es un bloque de control.
    - *Enforce Primary Key*: Indica a Form Builder que se encargue en la medida de lo posible de comprobar que los registros que se están introduciendo no tengan el mismo valor en su clave primaria. Esto evita enviar a la base de datos registros erróneos, lo que reduce el tráfico de la red.
    - *Query/Insert/Update/Delete Allowed*: Permite o impide que se realicen consultas, inserciones, actualizaciones o eliminaciones en el bloque de datos.
    - *WHERE Clause*: Especifica una cláusula WHERE por omisión. Cualquier condición que imponga el usuario se añadirá con un AND a esta cláusula WHERE por omisión.
    - *ORDER BY*: Especifica una clave de ordenación por omisión.
    - *Update Changed Columns Only*: Modifica solamente las comunas que se han modificado, lo que reduce el tráfico de la red.

- *Maximun Query Time*: Limita el tiempo máximo que se le puede conceder a la ejecución de una consulta. Si se alcanza este límite se aborta su ejecución.
- *Maximun Records Fetched*: Limita el número máximo de registros que puede devolver una consulta.

- Propiedades sobre los registros
  - *Query Array Size*: Especifica el número máximo de registros que entrega la base de datos a Form Builder cada vez. Un valor pequeño implica un menor tiempo de comunicación, mientras que un número grande reduce el número de operaciones de comunicación, por lo habrá una solución de compromiso.
  - *Query All Records*: Indica si la base de datos debe enviar a Form Builder todos los registros cuando se ejecute una consulta.

3.4.3.6. Creación de elementos de texto

Aunque en principio todos los campos que aparecen en los formularios son elementos de texto, nosotros vamos a utilizar los elementos de texto para mostrar campos calculados.

Para ello tendremos que crear un elemento de texto en la representación de un bloque de datos (ya que el cálculo hará referencia a una combinación de sus campos). A continuación, en la propiedad **Calculation Mode**, elegiremos **Formula**, y en la propiedad **Formula**, introduciremos la fórmula.

Las fórmulas se introducen utilizando como operandos con esta sintaxis:

:nombreTabla.nombreColumna

Por tanto, si queremos calcular el sueldo total de los empleados de cada departamento, escribiremos en el campo **Formula** del elemento de texto :emp.sueldo + :emp.complemento. Si además añadimos alguna característica de formato, podemos obtener un resultado como el de la figura siguiente.

Figura 3.40. Utilización de elementos de texto para la realización de cálculos.

3.4.3.7. Presentación de indicaciones y uso de casillas de verificación

En las interfaces que necesitan la introducción de algún tipo de datos por parte del usuario, es necesario informar al usuario del tipo de acción que tiene realizar, ya que no basta con las etiquetas que aparecen junto a los cuadros de introducción de datos.

Para ello se utilizan mensajes explicativos en la barra de estado para que aparezcan de forma automática cuando el elemento reciba el enfoque. Esto se consigue con las propiedades **Hint** y **Display Hint Automatically**. La propiedad **Hint** se configura con el mensaje que queramos mostrar en la barra de estado.

Por ejemplo, para indicar al usuario que el número de empleado ha de ser un número de cuatro dígitos, crearemos esta indicación "Introduzca un número de cuatro dígitos", de forma que cuando el número de empleado reciba el enfoque, aparezca dicho mensaje en la barra de estado, tal y como muestra esta figura.

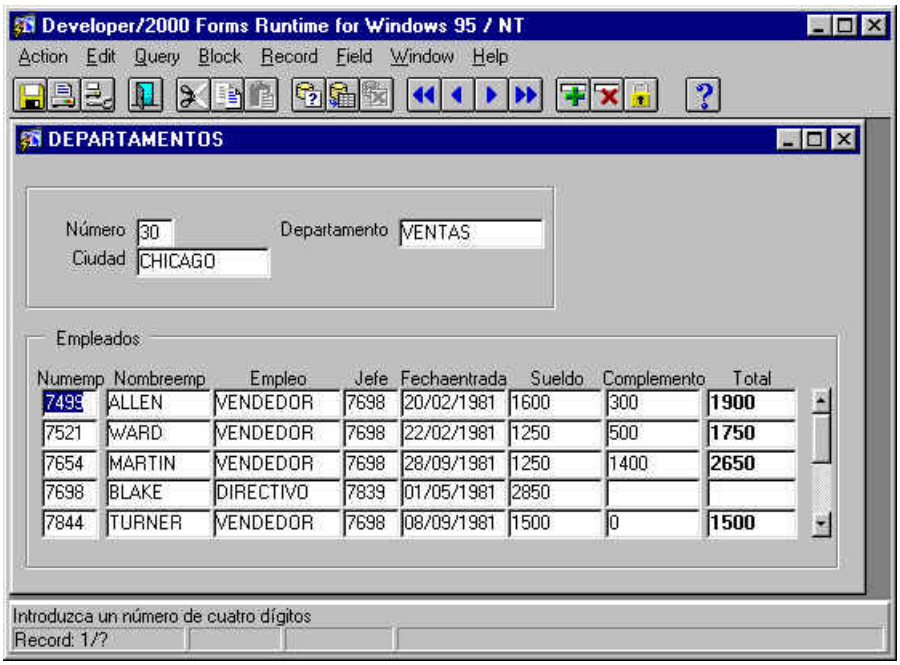

Figura 3.41. Formulario con indicaciones para el número de empleado.

Si lo que desea es que aparezca una sugerencia cuando el puntero del ratón se sitúe sobre un elemento de la interfaz, debe configurar la propiedad **Tooltip** con el texto que desea que aparezca (p.e. **Un número de cuatro dígitos que identifica al empleado en la empresa**).

Figura 3.42. Ejemplo de utilización de las sugerencias para el número de empleado.

Ahora vamos a ver cómo crear casillas de verificación. Para ello, necesitamos que hagan referencia a una columna de una tabla, por lo que tendremos que añadir alguna columna a alguna de las tablas con las que estamos trabajando.

Concretamente vamos a añadir dos columnas nuevas a la tabla de departamentos. Se trata de dos columnas que indican si el departamento tiene cobertura nacional y cobertura internacional. Por tanto, desde SQL*Plus modificaremos la estructura de la tabla para añadir estas dos columnas, y lo haremos con la instrucción ALTER TABLE de la forma siguiente.

```
ALTER TABLE EMP
      (ADD NACIONAL CHAR,
      ADD INTERNACIONAL CHAR);
```

Una vez que hemos creado estas dos nuevas columnas, necesitamos incorporarlas al bloque de datos de departamentos, y a continuación añadir dos nuevos campos a la presentación del bloque de datos. A continuación se muestra como podemos realizar estos dos pasos.

Para añadir un nuevo elemento al bloque de datos de departamentos, seleccionaremos el bloque de datos de departamentos en el Explorador de objetos, y a continuación seleccionaremos **Tools|Data Block Wizard**, para activar el asistente de bloques de datos y poder modificar el bloque de datos de departamentos.

En el cuadro de diálogo que aparece, seleccionaremos la ficha **Table**, y pulsaremos el botón **Refresh** para que muestre las columnas que acabamos de añadir a la tabla. Una vez que aparezcan estas columnas, las llevaremos al cuadro **Database Items**, tal y como muestra la figura siguiente. Una vez hecho esto pulse el botón **Finish.**

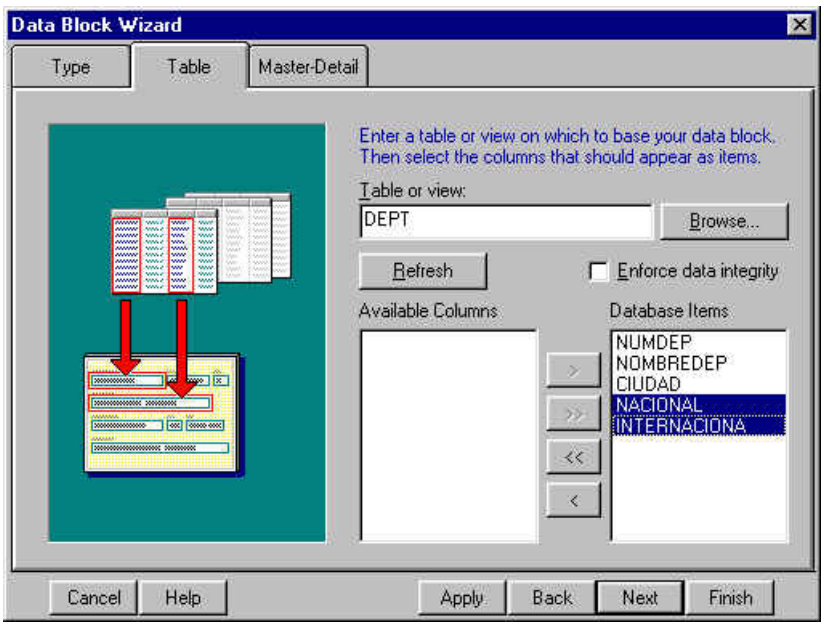

Figura 3.43. Modificación de un bloque de datos.

Ahora, una vez que se ha modificado el bloque de datos, tenemos que modificar el diseño de la presentación, de forma que podamos acceder a los nuevos elementos del bloque de datos. Para ello, seleccione el marco de departamentos en la vista de diseño del formulario y seleccione **Tools|Layout Wizard**.

Una vez que aparezca el asistente, tiene que incluir en el cuadro **Displayed Items** de la ficha **Data Block**, los dos elementos nuevos del bloque de datos. Una vez que haya añadido los dos elementos, deberá especificar para cada uno que se trata de una casilla de verificación, y no de un cuadro de edición de texto (valor por omisión). Para ello, seleccione **Check box** en la lista desplegable **Item Type**, de forma que su cuadro de diálogo sea similar al de la figura siguiente.

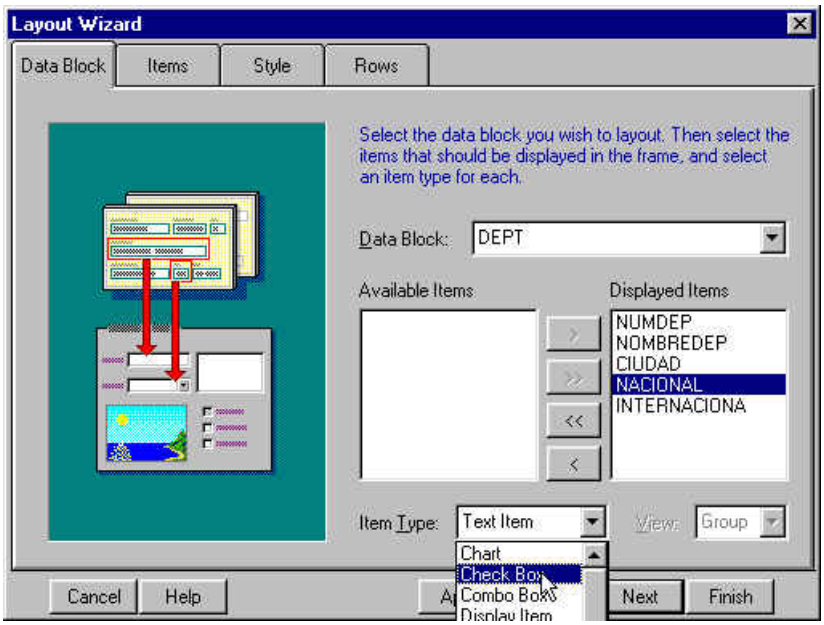

Figura 3.44. Modificación del diseño de un bloque de datos.

Una vez seleccionados los dos nuevos elementos de bloque de datos, y una vez que se han configurado como casillas de verificación, acepte los cambios.

Por último, lo único que falta por hacer es asignarle a cada uno de ellos un valor para cuando la casilla esté marcada y otro valor para cuando la casilla no esté marcada. Esto se realiza escribiendo un valor en las propiedades **Value When Checked** y **Value When Unchecked**, que si bien pueden tomar cualquier valor permitido por el tipo de datos del elemento, se suelen utilizar los valores V y F, o T y F, ya que se trata de variables lógicas. Observe cómo se ha realizado esto para la columna *internacional* en la figura siguiente.

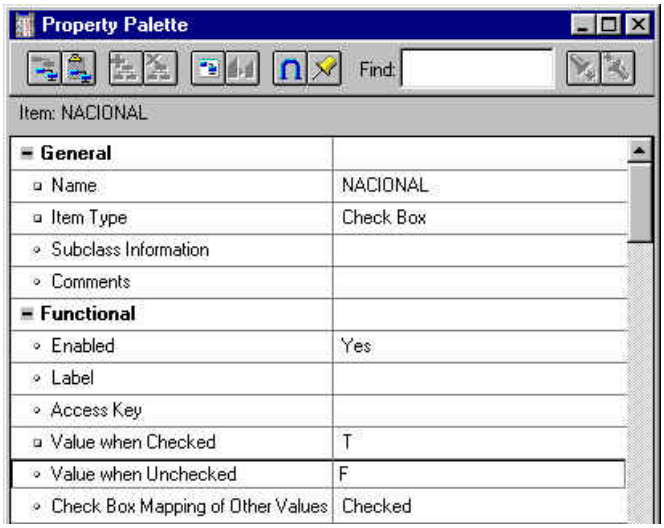

Figura 3.45. Asignación de valores a los dos estados de una casilla de verificación.

3.4.3.8. Utilización de listas de valores y cuadros de lista

A menudo es necesario utilizar campos en los formularios en los que la entrada de datos esté limitada a una serie de valores. Con esto se consigue que el usuario no tenga que recordar exactamente los valores permitidos, sino que le basta con seleccionar uno entre una lista. Además evita que se cometan errores de escritura, ya que el valor es seleccionado de una lista.

Form Builder ofrece dos mecanismos para la creación de listas de valores. El primero que veremos son las LOVs (Listas de valores, *List Of Values*), y luego veremos los conocidos cuadros de lista o cuadros combinados.

Las LOVs son listas que aparecen en una ventana aparte, donde el usuario selecciona un valor, y este valor seleccionado es el que se pasa al campo al que está ligado la LOV. Las LOVs están basadas en grupos de registros (*Record groups*) obtenidos a partir de consultas SQL.

Para crear una LOV, hay que añadir un nuevo objeto en la categoría LOVs del Explorador de objetos. A continuación, en la ventana que aparece indicaremos el origen de los valores que mostrará la LOV. Esto se realiza especificando un grupo de registros definido previamente o bien creando uno nuevo escribiendo una consulta en el cuadro **Query Text**.

Nosotros vamos a crear una LOV para mostrar los nombres de los departamentos, y así no tener que conocer cuáles son los nombres disponibles, y además evitar errores. Por tanto, tenemos que seleccionar los nombres de los departamentos de la tabla de departamentos. A continuación se muestra cómo quedaría la definición de la LOV.

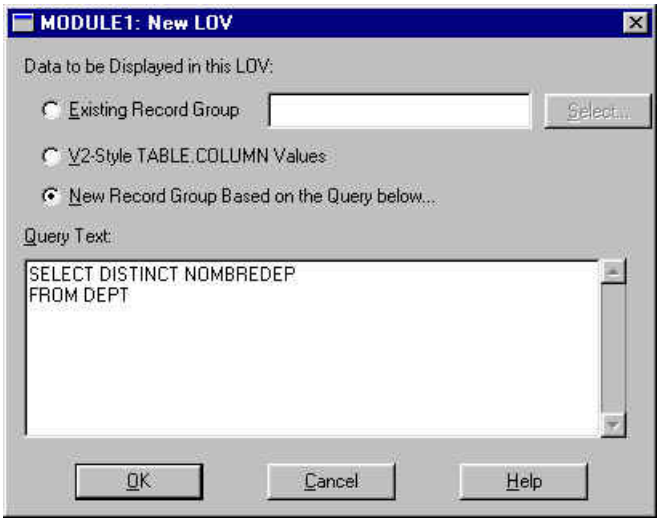

Figura 3.46. Definición de una LOV.

Esta operación ha creado una lista de valores y un grupo de registros. Observe que los nombres que ha generado Form Builder han sido LOV*xx*. Vamos a cambiar estos nombres a algo más significativo, como es DEPARTAMENTOS.

Para cambiar el nombre del grupo de registros puede hacerlo pulsando dos veces sobre el nombre del grupo de registros, o bien, mostrando sus propiedades y modificando la propiedad **Name**. Si lo hace de esta forma, verá que en **Record Group**

Query se encuentra definida la consulta que seleccionar los valores de la lista. Podrá observar también que el número de valores está limitado por la propiedad **Record Group Fetch Size**.

A continuación, cambiaremos el nombre de la LOV, y también lo podemos hacer de las dos formas, aunque lo haremos desde el Cuadro de Propiedades, ya que tenemos que modificar más propiedades aparte del nombre.

Concretamente, cambiaremos el nombre por DEPARTAMENTOS, y observaremos que ha cambiado el nombre del grupo de registros al que estaba ligado por el de DEPARTAMENTOS. A continuación, veremos algunas de las propiedades más importantes de la categoría **Functional**.

- **Column Mapping Properties**: Permite definir un campo del formulario para que tome el valor seleccionado en la LOV. Concretamente tiene que escribir el nombre del campo en **Return Item**. Para nuestro ejemplo escribiremos NOMBREDEP.
- **Automatic Display**: Indica si se activa automáticamente la LOV al seleccionar el campo al que está ligada.
- **Automatic Refresh**: Indica si se realiza una actualización automática de la lista al insertar nuevos valores.
- **Automatic Position**: Indica si la lista aparece cerca del campo.

Además, la propiedad **Title** define el título de la ventana de la LOV, de forma que quede descrita por un texto sugerente, como puede ser **Departamentos disponibles**.

Una vez configuradas las propiedades de la lista, sólo falta configurar el campo del formulario para que esté ligado a la lista. Esto se hace configurando la propiedad **List of Values** del campo correspondiente, en este caso NOMBREDEP, tal y como muestra la figura siguiente.

Figura 3.47. Configuración de una LOV para que devuelva el valor seleccionado

Así pues, ahora cada vez que seleccione el campo del nombre del departamento se abrirá la LOV para que seleccione un valor. Al pulsar **OK** el valor seleccionado pasará al campo del nombre del departamento. Si quiere añadir un valor nuevo, cancele la selección de valores de la lista, y edite directamente el valor del campo. Una vez que guarde el registro modificado este valor formará parte de la lista, gracias a la propiedad **Automatic Refresh**.

Figura 3.48. Utilización de una LOV.

Una vez vistas las LOVS (listas de valores basadas en consultas SQL), pasemos a ver los cuadros de lista. A diferencia de las LOVs, los cuadros de lista contienen una lista de valores definidos previamente, pero no están basadas en consultas SQL. Por tanto, no son tan flexibles, pero son útiles cuando se conocen previamente los valores del dominio del campo sobre el que se van a definir.

Form Builder permite la utilización de tres tipos de cuadros de lista:

- **Listas emergentes** (*Pop-Up Lists*): Se trata de un campo con una flecha a la derecha para mostrar la lista de valores. Este tipo de listas no permite la introducción de valores desde teclado.
- **Cuadros combinados** (*Combo Boxes*): Son listas emergentes que permiten la introducción de datos por parte del usuario.
- **Listas rectangulares** (*TLists*): Se trata de cuadros que tienen una lista de valores que pueden ser recorridos mediante una barra de desplazamiento.

La figura siguiente muestra distintas presentaciones del nombre del departamento para que observe la diferencia entre las listas emergentes y las listas rectangulares.

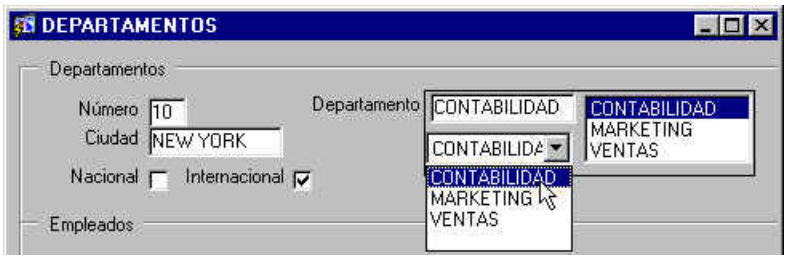

Figura 3.49. Cuadros de lista.

Para definir un cuadro de lista, sólo hay que especificar en la propiedad **Item Type** del elemento correspondiente del bloque de datos que se trata de un cuadro de lista (**List Item**). A continuación, en la propiedad **List Style** se debe especificar el tipo de cuadro de lista.

Una vez especificado el tipo de cuadro de lista, hay que definir los valores de la misma, y esto se hace configurando la propiedad **Elements in List**. Al seleccionar esta propiedad aparece un cuadro de diálogo en el que introduciremos en el cuadro **List Elements** los valores que queremos que aparezcan en el cuadro de lista, mientras que en el cuadro **List Item Value** introduciremos el valor real con el que se corresponde dicho elemento del cuadro de lista, que no tiene por que ser el mismo.

3.4.3.9. Utilización de botones de opción y botones de orden

A veces se suelen utilizar botones de opción para limitar la selección de valores. Los botones de opción permiten la selección de un único valor entre una serie de valores, por lo que son similares a los cuadros de lista. La diferencia está en que los botones de opción se suelen utilizar cuando la lista de valores es reducida.

Para ello, vamos a añadir una nueva columna a la tabla de departamentos para indicar si un departamento está inscrito o no en cierto registro escribiríamos desde SQL*Plus lo siguiente:

```
ALTER TABLE DEPT ADD (INSCRITO CHAR)
```

que luego podríamos eliminar con

```
ALTER TABLE DEPT DROP (INSCRITO)
```

A continuación utilizaríamos el Asistente para bloques de datos para añadir este nuevo campo al bloque de datos, pero indicaremos que su tipo es **Radio Group**.

Una vez creado el grupo de botones hay que crear cada uno de los botones, que en este caso serán dos, uno para inscrito y otro para no inscrito.

Una vez creados los dos botones de opción, hay que configurar algunas propiedades. Concretamente configuraremos:

- **Name:** INSCRITO
- **Label:** Sí/No
- **Radio Button Value:** T/F

Observe la diferencia entre la propiedad de la etiqueta, que indica lo que se muestra en la interfaz, y la propiedad del valor, que contiene lo que realmente se guarda en la base de datos.

A continuación puede crear un rectángulo para crear un grupo lógico con los dos botones de opción, y cambiar las propiedades del fondo (**none**) y las del título (**label**), de forma que su formulario sea similar al de la figura siguiente.

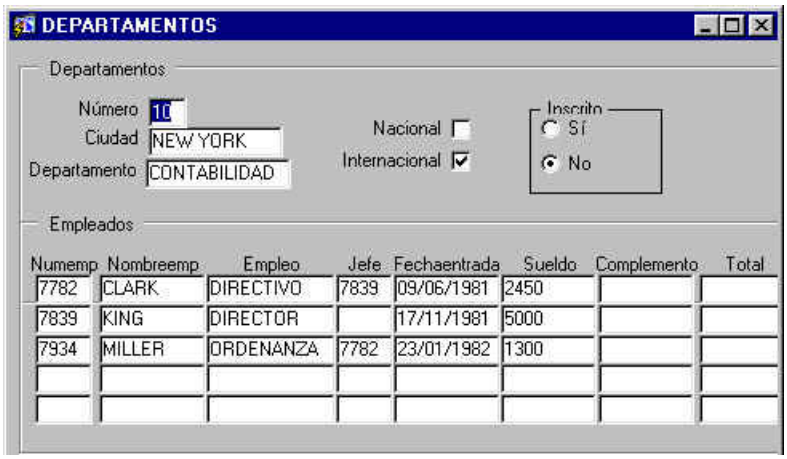

Figura 3.50. Formulario con botones de opción.

La creación de botones de orden es algo muy sencillo, pero el problema es que la creación de un botón de orden supone la asignación de acciones al evento que se produce cuando se pulsa el botón, y esto se hace mediante instrucciones PL/SQL que no hemos estudiado. No obstante veremos algo muy sencillo para ilustrar cómo crear dos botones de orden, uno para aceptar los cambios y otro para salir del formulario.

En primer lugar crearemos los botones seleccionando la herramienta **Button** de la barra de herramientas del diseño del formulario y crearemos dos botones. A continuación modificaremos su nombre y la etiqueta que muestran a través de las propiedades **Name** y **Label**. Concretamente le asignaremos los valores ACEPTAR y SALIR.

Ahora, sobre cada botón pulsaremos con el botón derecho, y en el menú contextual seleccionaremos la opción **Smart Triggers**, que lo que hace es mostrar los procedimientos de evento asociados al objeto seleccionado. Nosotros seleccionaremos **WHEN-BUTTON-PRESSED**, que definirá un procedimiento de evento asociado a la pulsación del botón.

Una vez que seleccionemos esta opción, aparecerá un editor de PL/SQL en el que introduciremos las órdenes PL/SQL que queremos que se ejecuten al pulsar los botones. Concretamente escribiremos **COMMIT;** para el procedimiento del botón ACEPTAR y **EXIT_FORM** para el botón SALIR.

De esta forma habrá creado dos botones de opción elementales en una interfaz de los que podrá probar su funcionamiento.

3.4.3.10. Creación de una barra de herramientas

Una barra de herramientas no es más que un tipo de *canvas* especial, por lo que su creación se limita a crear un *canvas*, a definir los botones que la forman y a asignar acciones a los botones (mediante bloques PL/SQL). En esta sección crearemos a modo de ejemplo una barra de herramientas con dos botones a los que faltaría asociarles el código PL/SQL apropiado.

Para crear una barra de herramientas basta con añadir un nuevo *canvas*. Para ello, seleccione la categoría **Canvases** en el Explorador de objetos y pulse el botón de Crear objeto. A continuación, active el Cuadro de Propiedades del nuevo *canvas*,

modifique su nombre y en la propiedad **Canvas Type** seleccione **Horizontal Toolbar** para indicar que se trata de una barra de herramientas horizontal. Por último, asegúrese de que la propiedad **Window** contiene la ventana en la que desea que aparezca la barra de herramientas. Si lo desea puede cambiar la presentación de la barra con la propiedad **Bevel**.

Una vez definido el *canvas* de la barra de herramientas, hay que añadirle los botones tal y como se hizo para añadir los botones de ACEPTAR y SALIR al *canvas* anterior (de hecho la barra es un *canvas* a todos los efectos). Añada dos botones asignándoles un nombre y una etiqueta apropiada y reduzca el tamaño del *canvas* para que se ajuste a los botones que haya añadido.

NOTA: Para añadir los botones, tenga seleccionado alguna categoría interior al *canvas* de la barra de herramientas en el Explorador de objetos.

Además, sería conveniente que asignara una sugerencia (propiedad *Tooltip*) a cada botón de modo que apareciese al situar el ratón sobre el ratón indicando de forma escueta la acción asociada a dicho botón. Si ahora ejecuta su formulario, el resultado debe ser similar al de la figura siguiente.

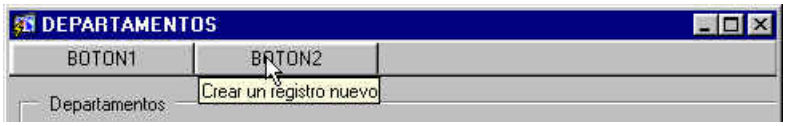

Figura 3.51. Creación de una barra de herramientas.

3.4.3.11. Formularios basados en fichas

Los formularios basados en fichas son una solución ideal para crear formularios con gran cantidad de datos en un espacio mínimo. Para ello se crean un conjunto de fichas superpuestas que pueden ser seleccionadas pulsando una pestaña. En este tipo de formularios, cada ficha contiene información relacionada, por lo que en el formulario la información queda dividida en grupos lógicos correspondientes a cada una de las fichas.

Antes de continuar es conveniente que cree otro módulo nuevo para el formulario basado en fichas que vamos a crear en esta sección, de forma que pueda comparar la funcionalidad de uno respecto al otro. Siguiendo con nuestro ejemplo vamos a crear un formulario basado en fichas para departamentos y empleados, de forma que tengamos una ficha para cada uno de ellos.

Crear un formulario basado en fichas es una tarea sencilla, y lo único que hay que hacer es indicar en la propiedad **Canvas Type** que el tipo de *canvas* es del tipo **Tab**. Esto creará un formulario con dos fichas o páginas por omisión. Si desea añadir más páginas, basta con pulsar el icono Crear objeto del Explorador de objetos.

Una vez creado el *canvas* cambiaremos su nombre a FICHA y modificaremos algunas de las propiedades de la fichas, como son el nombre y la etiqueta (p.e. Departamentos y Empleados).

A continuación crearemos los bloques de datos para cada una de las fichas, y este proceso es similar al que se siguió en su momento para crear el formulario Maestro-Detalle. Lo único que hay que tener en cuenta ahora en el Asistente para diseño (*Layout*

Wizard) es situar el bloque de datos en la ficha correspondiente, tal y como se muestra en la figura siguiente.

NOTA: Si tiene problemas al ejecutar la consulta y sólo puede consultar por la parte detalle, cambie el orden de los bloques de datos, ya que siempre se pregunta por el que está en primer lugar.

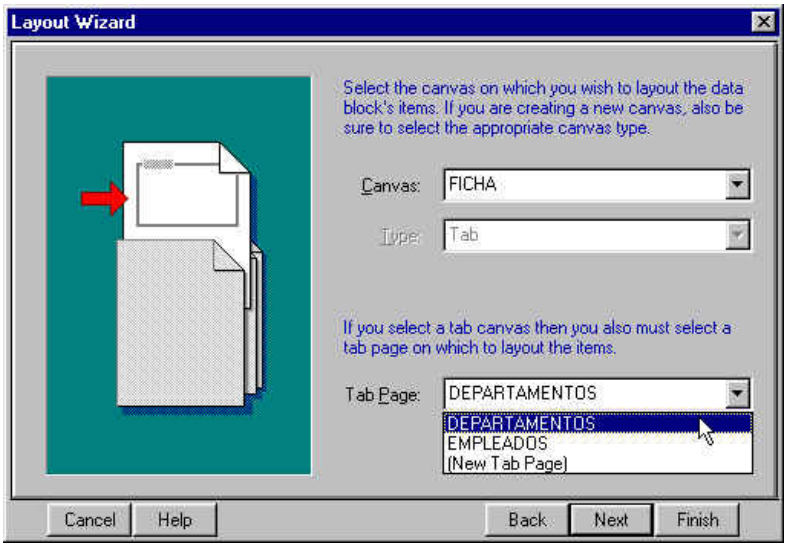

Figura 3.52. Asignación de un bloque de datos a una ficha de un *canvas*.

Al final, su formulario deber ser similar al de la figura siguiente.

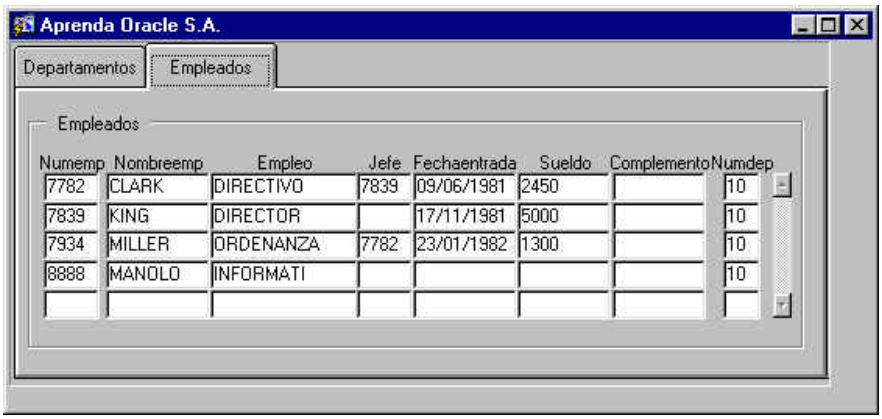

Figura 3.53. Formulario basado en fichas.

Pese a que este ejemplo es un poco forzado, piense en las posibilidades que ofrecen este tipo de formularios, como puede ser el de presentar en una ficha información resumida y en otra ficha información más detallada.

Además, recuerde que multiplica el espacio útil del formulario por el número de fichas que tenga su formulario.