UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INFORMÁTICA

PREPARANDO LA INSTALACION DE UNA BASE DE DATOS ORACLE 9i

TESINA

QUE PARA OBTENER EL TÍTULO DE
LICENCIADO EN INFORMÁTICA

PRESENTA
ANA LILIA MARTÍNEZ DOMÍNGUEZ

DIRIGIDA POR
I.S.C. JABEL RESÉNDIZ GONZÁLEZ

SANTIAGO DE QUERÉTARO, QRO. A 3 DE DICIEMBRE DEL 2004.

CARTA DE ACEPTACIÓN

Por este medio, se otorga constancia de aceptación de tesina para obtener el título de Licenciado en Informática, que presenta el pasante *ANA LILIA MARTÍNEZ DOMÍNGUEZ*, con el tema denominado *"OPTIMIZACIÓN DE LA BASE DE DATOS ORACLE 8i"*.

Este trabajo fue desarrollado como una investigación derivada del curso de titulación *"Administración de Base de Datos Oracle 8i"*, dando cumplimiento a uno de los requisitos contemplados en el artículo 34 del reglamento de titulación vigente, en lo referente a la opción de titulación por realización y aprobación de cursos de actualización.

Se extiende la presente para los fines legales a que haya lugar y para su inclusión en todos los ejemplares impresos de la tesina, a los seis días del mes de diciembre de dos mil cuatro.

ATENTAMENTE

I.S.C. JABEL RESÉNDIZ GONZÁLEZ
PROF. CURSO DE TITULACIÓN

ÍNDICE

AGRADECIMIENTOS

A mi Madre...
Porque ha forjado un futuro digno para mí con amor y esfuerzo.

A mi Familia...
Gracias por su apoyo incondicional, por creer en mí y darme ese entusiasmo para lograr cada objetivo, los quiero muchísimo.

A mis Amigos...
Aquellas personas que me han brindado momentos agradables, gracias por compartir alegrías, emociones, secretos, problemas, esfuerzos y amistad dentro y fuera de esta facultad.

A mis Profesores...
Con profundo respeto y admiración, porque nos han impulsado a que siempre seamos mejores seres humanos y profesionistas, por compartir su tiempo y espacio.
Ante todo gracias por su apoyo, estimulo y amistad.

GRACIAS

INTRODUCCIÓN

En la versión Oracle 9i, se han añadido muchas características nuevas y se han cambiado muchas funciones. Al mismo tiempo, Oracle ha añadido nuevas herramientas para simplificar las tareas de administración de la base de datos. En el capítulo 1 verá ejemplos de los componentes de una base de datos Oracle y los conceptos de implementación básicos que determinan su uso. Para administrar una base de datos Oracle, es necesario conocer como interactúan sus distintos componentes, dónde encajan dentro del conjunto y cómo personalizar el sistema en función de las necesidades propias.

Aunque todas las bases de datos Oracle están construidas a partir de los mismos componentes básicos, las opciones de las que se disponga en cada caso dependerán de la plataforma de hardware y del sistema operativo que se utilice. El capítulo 2, describe los tipos de arquitectura estándar disponibles, es decir, las formas en las que se suelen ensamblar esos componentes en los entornos más comunes.

Independientemente de la configuración que utilice, tiene que ser posible garantizar la disponibilidad, recuperabilidad y seguridad de su base de datos. Al añadir nuevos componentes a su entorno, añade posibles puntos de fallo. Sus estrategias de copias de seguridad y de recuperación, seguridad, de supervisión y de optimización deben tener en cuenta todos los componentes de su configuración hardware.

La configuración lógica de una base de datos influye enormemente en su rendimiento y puede también facilitar grandemente las tareas de administración. En el capítulo 3, se ofrecen algunas directrices para elegir el diseño adecuado de los espacios de tablas de cualquier base de datos Oracle.

La distribución efectiva de los objetos lógicos de las bases de datos forma parte de la *arquitectura flexible óptima* (OFA, Optimal Flexible Architecture), una pauta de configuración diseñada para simplificar la administración de la base de datos y optimizar la flexibilidad. Dependiendo de la opción de instalación que se elija, una configuración de tipo OFA estándar de espacios de tablas se crea automáticamente al utilizar el software de instalación de Oracle. Encontrará explicaciones sobre esa configuración, así como sobre los diferentes tipos y estados de los espacios de tablas. Incluso en el caso de que no pueda garantizar la ubicación física de los archivos en los sistemas RAID grandes, la creación de los espacios de tablas que cumplan los requisitos de la arquitectura OFA simplificará la administración.

En el capítulo 4 veremos la forma en que Oracle gestiona el almacenamiento físico de los datos, además del diseño físico de la base de datos recomendado para diferentes tipos de aplicaciones y entornos. Con demasiada frecuencia, no se planifica el diseño físico de la base de datos y sólo se tiene en cuenta cuando se comienzan a experimentar problemas de rendimiento. Al igual que es necesario planificar el diseño lógico, también hay que diseñar e implementar la disposición física de los archivos de la base de datos para conseguir nuestros objetivos. Si no se planifica dicha disposición antes de crear la base de datos, se producirá un ciclo recurrente de problemas relacionados con el diseño y los esfuerzos de ajuste del rendimiento. Aquí veremos dónde hay que colocar los archivos relacionados con la base de datos unos respecto de otros, para garantizar una capacidad de recuperación y un rendimiento óptimos. También veremos la estructura de directorios recomendada para los discos del sistema y una panorámica de la utilización del espacio de la base de datos.

CAPÍTULO 1
INTRODUCCIÓN A LA ARQUITECTURA DE ORACLE

1.1 Introducción a las bases de datos e instancias

Para comprender la arquitectura de Oracle, es fundamental entender dos conceptos básicos: bases de datos e instancias.

1.1.1 Bases de datos

Una *base de datos* es un conjunto de datos. Oracle proporciona la capacidad de almacenarlos y acceder a ellos de una forma coherente con un modelo definido y conocido como el modelo relacional. Debido a esto, Oracle constituye lo que se conoce como un sistema de gestión de bases de datos relacionales (RDBMS o Relational Database Management System) Cuando hablamos de una <<base de datos>> no sólo nos estamos refiriendo a los datos físicos, sino también a la combinación de objetos físicos, de memoria y de proceso.

Los datos de una base de datos se almacenan en tablas. Las tablas relacionales están de finidas por sus *columnas* y se les asigna un nombre. Los datos se almacenan como *filas* de la tabla. Las tablas pueden estar relacionadas entre sí y la base de datos puede utilizarse para hacer que se apliquen esas relaciones. En la figura 1.1 se muestra un ejemplo de la estructura de una tabla.

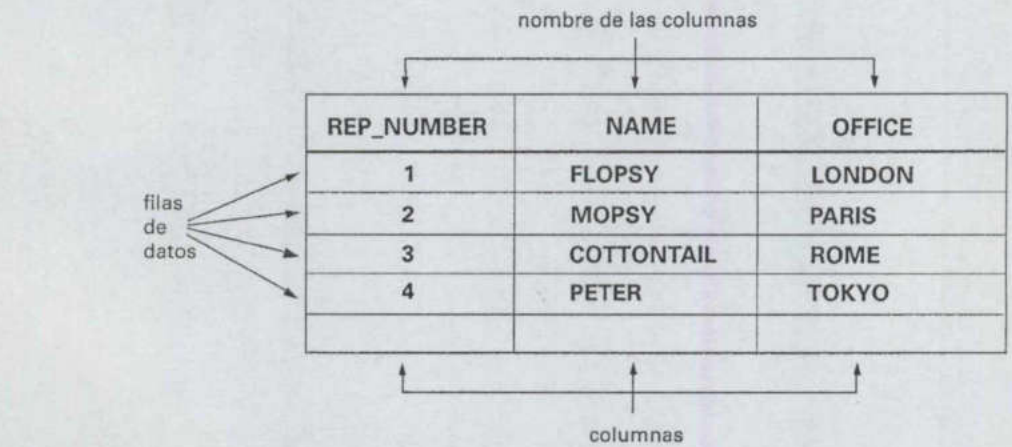


Fig. 1.1 Ejemplo de la estructura de una tabla.

1.1.1.1 Espacios de tablas

Un espacio de tablas es una división lógica de una base de datos. Toda base de datos consta, al menos, de un espacio de tablas (llamado espacio de tablas SYSTEM). Se pueden utilizar otros espacios de tablas para agrupar a los usuarios o aplicaciones, con el fin de facilitar el mantenimiento y mejorar el rendimiento. Ejemplos de estos tipos de espacios de tablas serían USERS, para uso general, y UNDO para segmentos para deshacer cambios. Un espacio de tablas sólo puede pertenecer a una base de datos.

1.1.1.2 Archivos de datos

Cada espacio de tablas consta de uno o más archivos, llamados archivos de datos, que se almacenan en disco. Un archivo de datos sólo puede pertenecer a un único espacio de tablas. El tamaño de los archivos de datos se puede modificar después de su creación. Para crear nuevos espacios de tablas hay que crear nuevos archivos de datos. Una vez que un archivo de datos se ha añadido a un espacio de tablas, ese archivo de datos no puede ser eliminado del espacio de tablas, ni puede ser asociado con otro espacio de tablas.

Si almacena objetos de base de datos en espacios de tabla diferentes, puede separarlos físicamente colocando los archivos de datos correspondientes en discos distintos. Esta separación de datos es una herramienta importante para la planificación y optimización de la forma en que la base de datos maneja las solicitudes de E/S que recibe. La relación entre bases de datos, espacios de tablas y archivos de datos se ilustran en la figura 1.2.

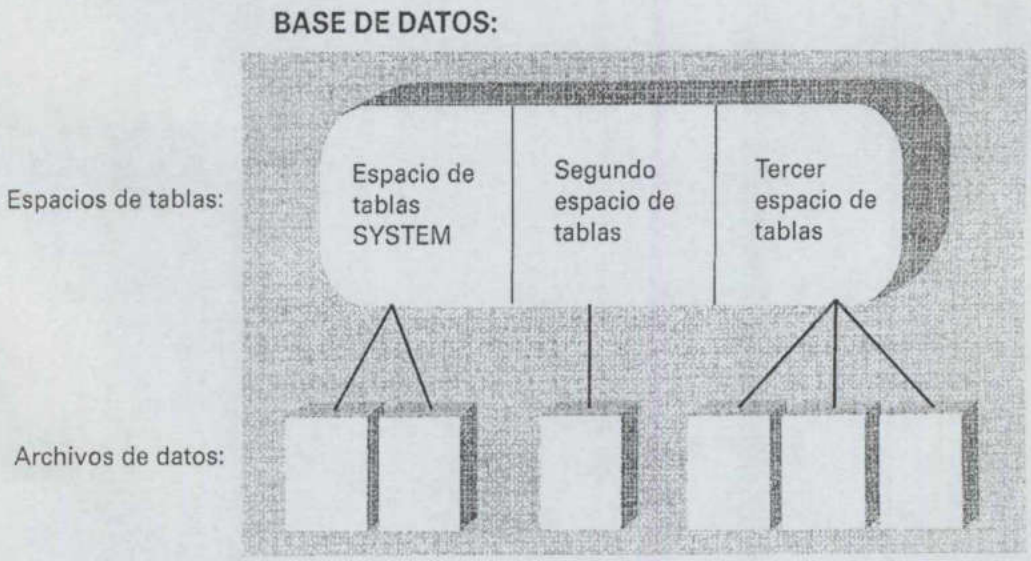


Fig. 1.2 Relación entre las bases de datos, los espacios de tablas y los archivos de datos.

1.1.2 Otros archivos

Los archivos de datos de la base de datos proporcionan el almacenamiento físico de los datos de la base de datos. Por tanto, son estructuras tanto <<internas<<, por estar ligadas directamente a los espacios de tablas, como <<externas<<, por tratarse de archivos físicos.

Los siguientes tipos de archivos, aunque están relacionados con la base de datos, están separados de los archivos de datos. Entre estos archivos se incluyen los siguientes:

- Los registros de reconstrucción.
- Los archivos de control.
- Los archivos de traza y el registro de alertas.

1.1.2.1 Registros de reconstrucción

Oracle conserva registros de todas las transacciones realizadas en la base de datos. Estas transacciones se registran en archivos denominados archivos de registro de reconstrucción en línea. Estos archivos de reconstrucción se utilizan para recuperar las transacciones de la base de datos en el orden adecuado, en caso de que se produzca un fallo en la base de datos.

Los archivos de registro de reconstrucción también permiten que Oracle coordine la forma en que se escriben los datos en el disco. Cuando se produce una transacción en la base de datos, ésta se introduce en los buffers del registro de reconstrucción, mientras que los bloques de datos afectados por la transacción no se escriben de manera inmediata en el disco.

Todas las bases de datos Oracle tienen tres o más archivos de registro de reconstrucción en línea. Oracle escribe en ellos de manera cíclica: después de llenar el primer archivo de registro, escribe en el segundo hasta llenarlo. Cuando se han llenado todos los archivos de registro de reconstrucción en línea, vuelve al primero y empieza a sobrescribir su contenido con nuevos datos de transacciones. Si la base de datos se esta ejecutando en modo ARCHIVELOG, hará una copia de los archivos de registro de reconstrucción en línea antes de sobrescribirlos. Estos archivos de registro de reconstrucción salvaguardados pueden utilizarse para devolver a cualquier parte de la base de datos al estado en que se encontrara en un determinado momento.

La base de datos puede duplicar en espejo los archivos de registro de reconstrucción. Esta operación permite al DBA duplicar los archivos de registro de reconstrucción sin depender del sistema operativo o de la funcionalidad que ofrezca el hardware del entorno de funcionamiento.

1.1.2.2 Archivos de control

La arquitectura física global de una base de datos se mantiene por medio de sus archivos de control, en los que registra la información de control referente a todos los archivos de la base de datos. Se utilizan para conservar la coherencia interna y guiar las operaciones de recuperación.

Como los archivos de control son fundamentales para la base de datos, se almacenan varias copias en línea. Estos archivos suelen almacenarse en discos separados, con el fin de minimizar el daño potencial derivado de posibles fallos de los discos. La base de datos creará y mantendrá los archivos de control que se hubieran especificado en el momento de la creación de la base de datos.

Los nombres de los archivos de control de la base de datos se especifican a través del parámetro de inicialización CONTROL_FILES. Si tiene que añadir un nuevo archivo de control a la base de datos, puede cerrar la instancia, copiar uno de los archivos de control existentes en una nueva ubicación, añadir la nueva ubicación a la configuración del parámetro CONTROL_FILES y reiniciar la instancia.

1.1.2.3 Archivos de traza y registros de alerta

Todos los procesos asociados a una instancia que se ejecutan en segundo plano tienen a su vez un archivo de traza asociado. el archivo de traza contiene información relativa a los sucesos significativos con los que se encuentran dichos procesos. Además de estos archivos, Oracle conserva un archivo llamado registro de alertas. Este registro almacena los comandos y los resultados de los comandos correspondientes a los sucesos más importantes que ocurren en la vida de la base de datos. Por ejemplo, la creación de espacios de tablas, el paso de un registro de reconstrucción a otro, las operaciones de recuperación y los inicios de la base de datos se graban en el registro de alertas. El registro de alertas es una fuente de información fundamental para la gestión cotidiana de una base de datos. Los archivos de traza son muy importantes para tratar de descubrir la causa de un fallo importante.

1.1.3 Archivos administrados por oracle

En las versiones de Oracle anteriores a Oracle 9i, cuando se quitaba un espacio de tablas de la base de datos, los archivos de datos subyacentes utilizados para dar soporte al espacio de tablas no se eliminaban automáticamente. Del mismo modo, cuando se eliminaba un archivo de registro de reconstrucción en línea o un archivo de control, los archivos físicos asociados no se eliminaban. En Oracle9i, se puede habilitar la herramienta de archivos administrados por Oracle

(Oracle Managed Files, OMF) PARA CONTRIBUIR A ALIVIAR LA CARGA DE LA ADMINISTRACIÓN DE ARCHIVOS. OMF INCLUYE TAMBIÉN UNA CARACTERÍSTICA que permite a Oracle crear y borrar automáticamente archivos del sistema operativo, según sea necesario, para permitir la realización de las actividades de la base de datos.

Cuando se activa OMF, Oracle utiliza directorios del sistema de archivos en lugar de nombres de archivo para crear archivos secundarios como, por ejemplo, los archivos de control, los archivos de registro de reconstrucción y los archivos de datos. Además, crea los nombres apropiados para los archivos automáticamente. Cuando se eliminan los archivos, los archivos subyacentes del sistema operativo se borran automáticamente. Por tanto, con OMF, no se dejan olvidados archivos obsoletos que puedan causar confusión o desperdiciar espacio.

1.1.4 Instancias

Para acceder a los datos de la base de datos, Oracle utiliza un conjunto de procesos en segundo plano que comparten todos los usuarios. Además, existen estructuras de memoria (conocidas por todos como SGA) que sirven para almacenar los datos de la base de datos que se han consultado más recientemente. Las secciones más grandes de la SGA (System Global Area o Área Global del Sistema), la caché de buffers de bloques de datos, el área compartida SQL, el área de bloques de gran tamaño y el área Java, constituyen generalmente más del 95 por 100 de la memoria asignada a la SGA. Estas áreas de memoria ayudan a mejorar el rendimiento de la base de datos, ya que reducen la cantidad de operaciones de E/S realizadas sobre los archivos de datos.

Una instancia de la base de datos (conocida, también, como servidor) es un conjunto de estructuras de memoria y procesos en segundo plano que acceden a un conjunto de archivos de bases de datos. Es posible que múltiples instancias accedan a una única base de datos (ésta es la opción conocida como Real Application Cluster) La figura 1.3 ilustra la relación entre instancias y bases de datos.

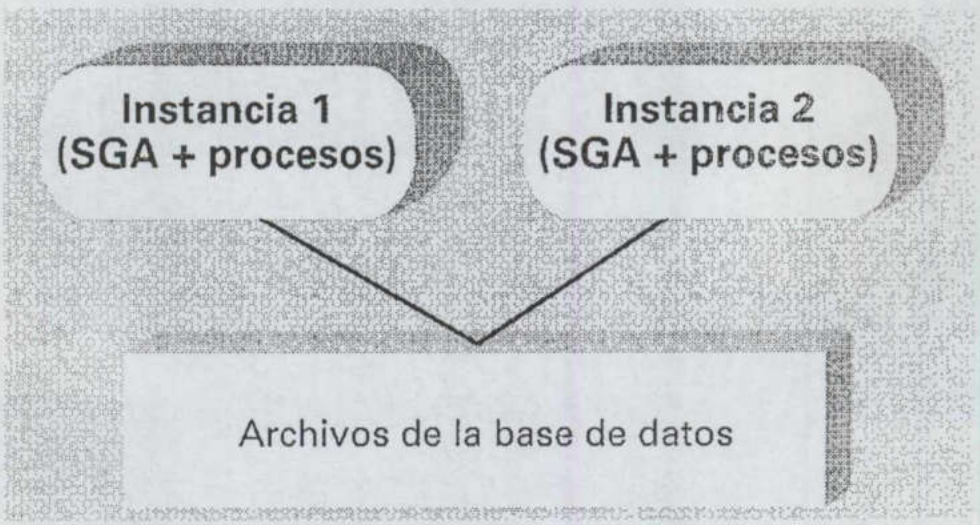


Fig. 1.3 Instancias y archivos de datos de Oracle.

Los parámetros que determinan el tamaño y la composición de una instancia se almacenan, bien, en un archivo de inicialización denominado init.ora o, bien, residen dentro de la base de datos, en un archivo de parámetros de servidor, conocido como SPFILE, el cual está almacenado en spfile.ora.

El archivo de inicialización se lee durante el inicio de la instancia y el administrador de la base de datos puede modificarlo. Las modificaciones que se realizan en este archivo no surten efecto hasta la siguiente operación de inicio de la instancia. El nombre del archivo de inicialización de una instancia suele incluir el nombre de la instancia. Si la instancia se llama ORCL, el archivo init.ora se llamará initorcl.ora. Oracle crea cada vez más y más

parámetros dentro de la base de datos que se pueden ajustar dinámicamente. Los cambios de los parámetros pueden continuar en efecto después de las operaciones sucesivas de apagado e inicio, independientemente del contenido del archivo de parámetros de inicialización.

1.2 Instalación del software

El software de Oracle actual puede resultar en ocasiones todo un desafío en lo referente a instalarlo correctamente la primera vez que se intenta. Por ejemplo el producto Oracle9iAS Portal necesita que se lleven a cabo un buen número de pasos de preinstalación incluso antes de ejecutar los discos de instalación. Para instalar con éxito el paquete, o paquetes, de productos Oracle que se hayan adquirido, se debe comenzar por examinar las instrucciones de instalación incluidas en el CD-ROM del software. Los requisitos y las instrucciones de instalación pueden ser diferentes para cada plataforma, especialmente en el área de los parches y del sistema operativo. Por ejemplo, Oracle9i en una plataforma Windows NT necesita que el sistema operativo sea Windows NT versión 4.0, con el Service-Pack 6.0a instalado, mientras que Windows 2000 requiere la versión 2. Por tanto, hay que asegurarse de comprobar qué nivel de sistema operativo y qué parches se necesitan antes de comenzar con la instalación. Algunas versiones de Oracle no se generarán y ejecutarán correctamente si las bibliotecas del sistema operativo adecuadas no se encuentran disponibles.

Puesto que la instalación del software de Oracle es diferente para cada plataforma, queda fuera del ámbito de esta tesina la descripción detallada del proceso de instalación para cada caso.

Debería encontrar ayuda para el proceso de instalación en la documentación de instalación que proporciona Oracle y en los sitios web de MetaLink(http: \\metalink.oeacle.com) y Technology Network(http: \\technet.oracle.com.

1.2.1 Opciones y componentes de la instalación de oracle

Independientemente de cuál sea la versión del sistema operativo, existen disponibles tres niveles diferentes del software de Oracle dependiendo del producto que haya comprado su compañía. Los tipos disponibles son:

- Enterprise Edition. La edición para empresas incluye la instalación de: una base de datos inicial preconfigurada; servicios de red; Oracle options con licencia; herramientas del entorno de base de datos; el entorno de trabajo Oracle Enterprise Manager de herramientas de administración, entre las que se incluyen: Console, Management Server e Intelligent Agent; utilidades de Oracle y la documentación en línea. También se incluyen los productos que se utilizan más comúnmente para el diseño de almacenes de datos y el procesamiento de transacciones.

- Standard Edition. Esta edición, la estándar, incluye la instalación de los siguientes componentes: una base de datos inicial preconfigurada, servicios de red, Oracle Enterprise Manager Console y utilidades de Oracle.

- Personal Edition. La edición personal está compuesta por: una base de datos inicial preconfigurada que da soporte a un entrono de implementación y desarrollo monousuario que requiere compatibilidad absoluta con Oracle 9i Enterprise Edition y Oracle9i Standar Edition.

Puede utilizar la opción personalizada (Custom) para instalar o reinstalar los productos individuales.

También hay disponible una aplicación de interfaz de base de datos que da soporte a las características de red y de interacción entre el cliente de Oracle y una base de datos ubicada en un servidor diferente. Los tipos de instalación cliente disponibles son:

- Administrador. La opción administrador incluye: Oracle Enterprise Manager Console, con las herramientas de administración empresarial, los servicios de red, utilidades, software básico de cliente y la documentación en línea.

- Runtime. La opción de tiempo de ejecución consta de los servicios de red y archivos auxiliares.

Puede utilizar la opción Custom para instalar o reinstalar componentes individuales disponibles en las opciones de Administrador y Runtime.

Encontrará disponibles tres tipos de instalación de administración y son los siguientes:

- Oracle Management Server. Consta de los siguientes elementos: Oracle Management Server, el cual procesa todas las tareas de administración del sistema y gestiona la distribución de estas tareas a Intelligent Agents (agentes inteligentes), repartidos por nodos administrados en toda la empresa, además del software cliente básico y la documentación en línea.

- Oracle Internet Directory. Está compuesto de: un servidor de directorio de Internet de Oracle compatible con LDPA (Lightweight Directory Access Protocol, Protocolo Ligero de Acceso a Directorios), herramientas de cliente compatibles con LDAP y el esquema de base de datos del directorio de Internet de Oracle.
- Oracle Integration Server. Consta de: componentes compatibles con XML, incluyendo una JVM (Máquina Virtual Java) de Oracle9i, un motor de flujos de trabajo (workflow) y servicios avanzados de gestión de colas.

Una vez más, puede utilizar la opción personalizada para instalar o reinstalar productos individuales.

Aunque se puede instalar el software de administración durante la instalación normal de Enterprise Edition, debería instalar el conjunto del producto de la base de datos teniendo primero preconfigurada la base de datos y, a continuación, volver a ejecutar los discos de instalación para instalar los productos de administración. Durante la instalación de Enterprise Edition, se instala el entorno de trabajo del software del servidor de

administración. El software de administración requiere una base de datos para su soporte y se puede utilizar la base de datos preconfigurada creada durante la instalación inicial de Enterprise Edition para dicho propósito, en lugar de dejar que el programa de instalación del software de administración cree una base de datos adicional para cada una de las opciones de administración.

1.2.1.1 Otras opciones con licencia que se distribuyen separadamente

Además de los productos incluidos en los discos de distribución de Oracle9i, hay disponibles otras opciones de software con un coste adicional. La lista actual de opciones de coste añadido incluye las siguientes:

- Oracle Advanced Security.
- Oracle Change management Pack.
- Oracle Data Mining.
- Oracle Diagnostics Pack.
- Oracle Label Security.
- Oracle Management Pack para SAP.
- Oracle Management Pack para Oracle Applications.
- Oracle OLAP.
- Oracle Partitioning.
- Oracle Real Application Cluster (anteriormente, Oracle Parallel Server).
- Oracle Spatial.
- Oracle Tuning Pack.

1.3 Creación de una base de datos

En función del tipo de instalación de software de Oracle que haya seleccionado, se le ofrecerá o no la oportunidad de crear una base de datos preconfigurada. Le recomendamos encarecidamente que permita a Oracle crear una base de datos preconfigurada por varias razones. Con una base de datos preconfigurada, podrá:

- Cerciorarse de que el software esta lo suficientemente bien instalado para poder crear una base de datos.
- Examinar las nuevas características de Oracle en un entorno limpio.

- Determinar qué cosas son diferentes en la base de datos de la versión nueva, en el caso de que tenga bases de datos de Oracle de versiones anteriores.

Una vez instalado el software de Oracle , el asistente para la configuración de la base de datos Oracle se ejecuta para recopilar la información necesaria para crear una base de datos. Puesto que puede ejecutar el asistente para la configuración después de haber instalado el software, los pasos para la creación de una base de datos que se incluyen aquí reflejan los pasos y las descripciones del proceso de creación posterior a la instalación.

1.3.1 Uso del asistente para la configuración de bases de datos oracle

Cuando se inicia el asistente para la configuración de bases de datos Oracle (Oracle Database Configuration Assistant), aparece una pantalla de bienvenida, seguida de una pantalla en la que se pide que se especifique la operación que se desea realizar. Las opciones son las siguientes:

- Create a Database: crea una base de datos nueva.

- Configure Database Options in a Database: cambia la configuración para que pase de ser un servidor dedicado a ser un servidor compartido o añadir opciones que no se incluían anteriormente en la creación de la base de datos, como, por ejemplo, incluir las opciones de coste añadido citadas anteriormente.

- Delete a Database: elimina la base de datos y todos los archivos asociados del sistema operativo.

- Manage Templates: modifica las plantillas existentes de base de datos o crea otras nuevas. La ventaja de contar con plantillas de creación de bases de datos es que se pueden crear bases de datos duplicadas rápida y fácilmente, sin necesidad de especificar los parámetros necesarios más de una vez. Dentro de la opción de administración de administración de plantillas, se puede crear una plantilla a partir de una existente, crear una plantilla basada en una estructura de base de datos local o remota existente, o crear una plantilla a partir de una base de datos existente incluyendo los datos de la base de datos. La última opción es excelente para crear una base de datos nueva que duplique en espejo una base de datos de producción para utilizarla en un entorno de desarrollo de aplicaciones o como punto de partida para disponer de una base de datos de reserva.

1.3.1.1 Selección de una plantilla de base de datos

Cuando se opta por crear una base de datos nueva, la pantalla siguiente muestra las plantillas de bases de datos que proporciona Oracle según un modelo proyectado de interacción del usuario dentro de la nueva base de datos, incluyendo:

- General Purpose (propósito general): se realiza una gran variedad de tareas de base de datos, incluyendo transacciones simples, ya sea leer, escribir o eliminar datos de la base de datos, es una operación relativamente sencilla que procesa una pequeña cantidad de información, pero hay numerosos usuarios que realizan una gran cantidad de transacciones simultáneamente.

- Data Werehousing o Decisión support system (Sistema de almacén de datos o Sistema de ayuda a la toma de decisiones): se llevan a cabo numerosas consultas complejas, para las que se procesan grandes cantidades de datos. Para ello, se necesita una alta disponibilidad, un tiempo de respuesta excelente y precisión. En un

entorno DSS (Decisión Support system o sistema de ayuda a la toma de decisiones), las consultas pueden ser simples, devuelven un número pequeño de registros, o complejas, ordenan varios miles de registros repartidos entre varias tablas.

- Customized (personalizada): esta opción permite crear una base de datos personalizada para el entorno específico de procesamiento de los usuarios.

- Software Only (sólo software): esta opción solicita al usuario que proporcione información exhaustiva sobre la creación de la base de datos.

En la figura 1.4 se muestra una imagen de la pantalla Database Templates (plantillas de base de datos) del asistente para configuración de bases de datos con la opción General Purpose (propósito general) seleccionada. En la parte inferior de la pantalla, hay un botón que sirve para visualizar los detalles de la configuración de la base de datos. Puede examinar las opciones propuestas, pero no puede, en este momento del proceso, modificar ninguno de los valores. Las opciones mostradas son opciones comunes.

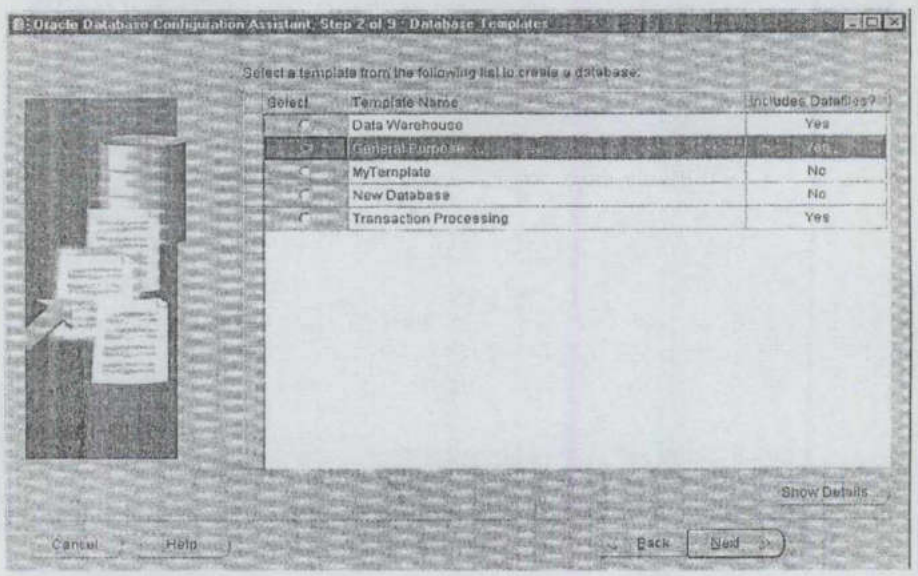


Fig. 1.4 Pantalla Database Templates (plantillas de bases de datos)

1.3.1.2 Declaración del nombre de la base de datos y registro de la base de datos

Se pide que se especifique un nombre de base de datos global y un nombre de instancia (SID) para la nueva base de datos. La extensión estándar de base de datos global es .world y puede proporcionar este valor a menos que posea un nombre de dominio establecido para usarlo en este caso. Cuando escriba el nombre de la base de datos global, aparecerá automáticamente en el campo SID ese nombre hasta el punto (.). Por ejemplo, si introduce el nombre MYDB.world en el campo Global Database Name (nombre de la base de datos global), el valor MYDB aparecerá escrito automáticamente en el campo SID. En la

figura 1.5 se muestra la pantalla Database identification (identificación de la base de datos) con el valor MYDB.world.

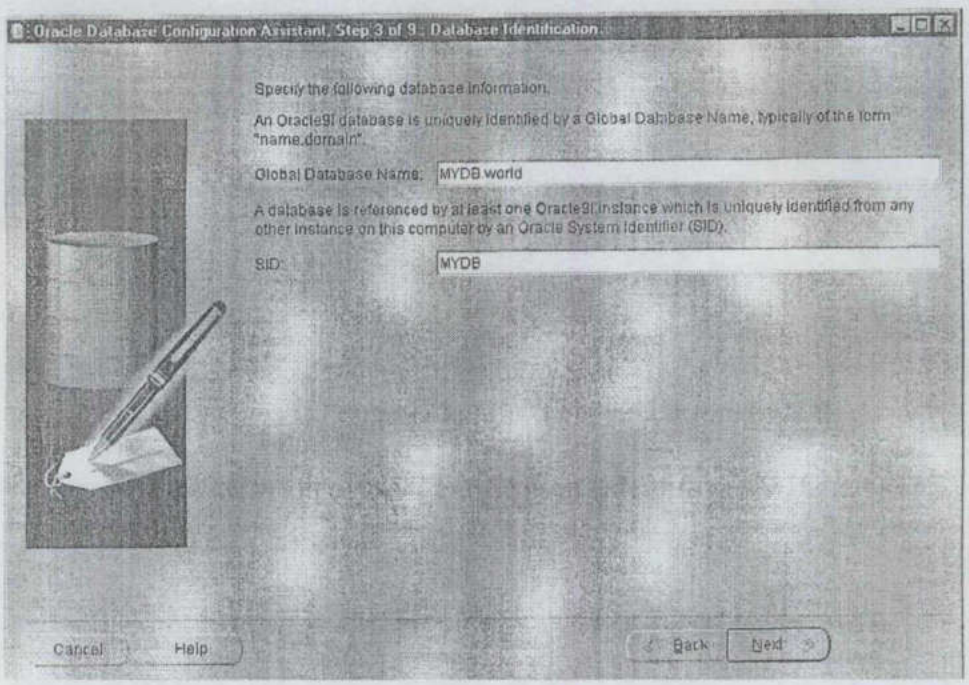


Fig. 1.5 Pantalla Database Identification.

La opción siguiente esta relacionada con la configuración de red de la nueva base de datos y permite registrar la base de datos en un servicio de directorio. Si ya ha configurado un servidor de directorio como Oracle Internet Directory o Microsof Access, puede optar por registrar la base de datos en ese servidor. El valor predeterminado es no registrar la base de datos.

1.3.1.3 Opción de servidor dedicado o compartido

La pantalla Database Connection Options (opciones de conexión a la base de datos), que se muestra en la figura 1.6, permite determinar la forma en que cada cliente se conectará a la base de datos. Si la nueva base de datos va a dar soporte a un número reducido de clientes o los clientes van a permanecer conectados a la base de datos durante largos períodos de tiempo, seleccione la opción Dedicated Server Mode (modo de servidor dedicado). En la figura 1.6 se muestra esta opción seleccionada.

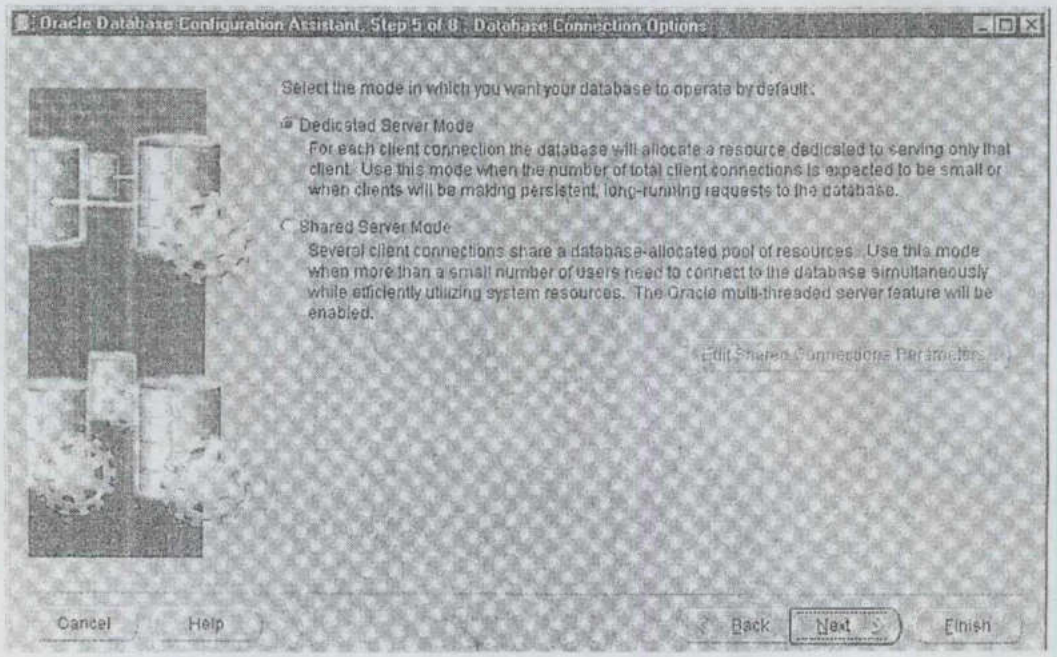


Fig. 1.6 Pantalla Database connection Options.

Sin embargo, si su base de datos va a dar soporte a un gran volumen de clientes, seleccionando la opción Shared Server Mode (modo de servidor compartido), puede permitir a estos clientes que compartan cierto número de recursos. Cuando se selecciona esta opción, se ofrece la oportunidad de modificar la configuración predeterminada de shared Server (servidor compartido) haciendo clic en el botón Edit Shared Connection Parameters (editar parámetros de conexiones compartidas). Entonces se muestran las fichas de configuración Basic (opciones básicas) y Advanced (opciones avanzadas) del modo de servidor compartido, como puede ver en la figura 1.7.

Se puede cambiar el protocolo de conexión TCP, opción predeterminada, por SPX, NMP, TCPS (la opción TCP segura) o IPC, además de modificar los aspectos siguientes:

- Number of Dispatchers: especifica el número de expedidores. Los expedidores se utilizan como mensajeros para pasar las consultas a la base de datos y devolver las respuestas a los clientes. La opción predeterminada es 1 expedidor.

- Maximum Number of Connections per Dispatcher: define el número máximo de conexiones de red que se permiten por cada expedidor.

- Maximum Number of Dispatchers: determina el número máximo de expedidores. Se trata de un valor estático, es decir, una vez configurado, hay que reiniciar la base de datos para modificar el valor. El valor predeterminado será el que resulte mayor de entre 5 y el valor que el usuario establezca.

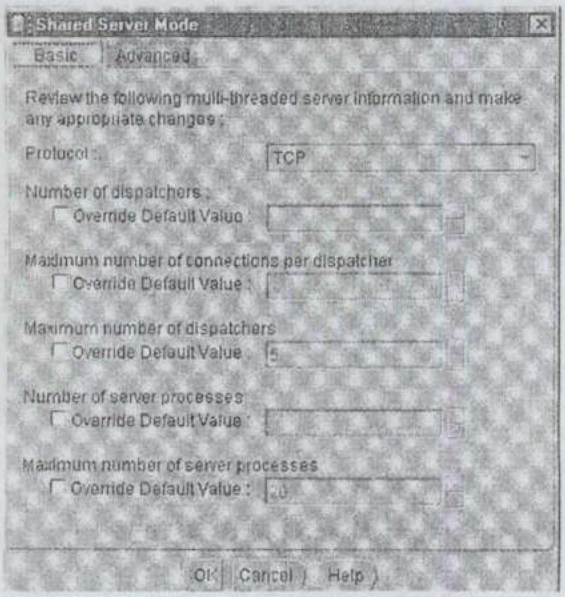


Fig. 1.7 Ficha de las opciones básicas de Shared Server Mode (modo de servidor compartido).

- Number of Server Processes: indica el número de procesos de servidor. Se trata de un parámetro dinámico que define cuántos procesos de servidor se crearán al iniciar una instancia. El valor varía entre el número designado y el número máximo de procesos de servidor declarado. El valor nunca será inferior al número de procesos de servidor definido por esta variable.

- Maximum Number of Server Proceses: define el número máximo de procesos de servidor. Es un parámetro estático definido de forma predeterminada en el número que sea más alto de entre 20 o el doble del valor de MAX_SERVERS.

La ficha Advanced (opciones avanzadas) de Shared Server Mode (modo de servidor compartido) incluye las opciones mostradas en la figura 1.8. entre las opciones avanzadas se encuentra la posibilidad de activar la multiplexación de conexiones de servidor compartido. La multiplexación permite la concentración de conexiones del administrador de conexiones de Oracle (Oracle Connection Manager Concentration), característica por la que se canalizan múltiples sesiones de red de cliente a través de una única conexión del protocolo de transporte puede definir el modo en que se desea activar la concentración de conexiones. Las opciones son:

- Off: desactiva la característica tanto para las conexiones entrantes como para las salientes.

- On: activa la característica para todas las conexiones.

- Incoming Connections: activa la característica sólo para las conexiones entrantes.
- Outgoing Connections: activa la característica sólo para las conexiones salientes.

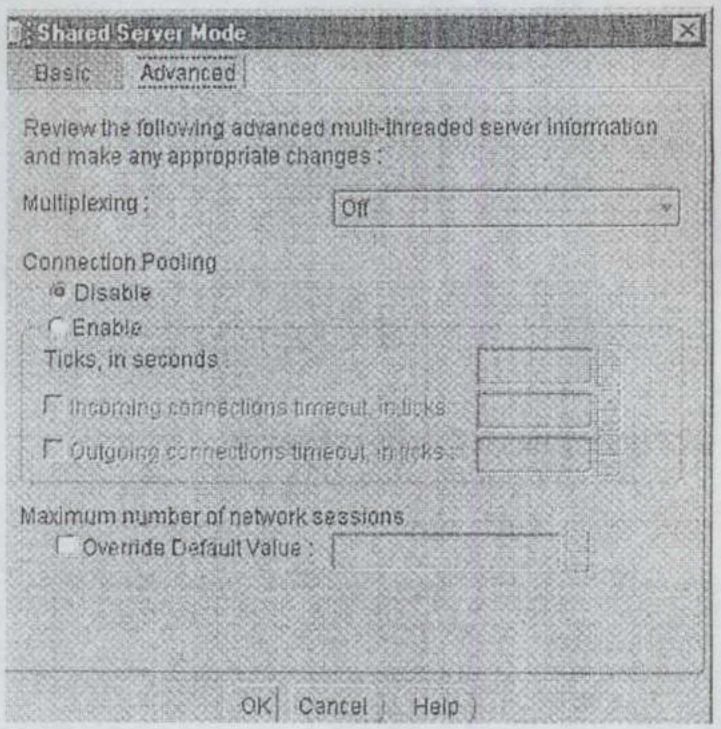


Fig. 1.8 Ficha de las opciones avanzadas de Shared Server Mode.

Utilizando Connection Pooling (batería de conexiones), se puede maximizar el número de conexiones de red físicas para un servidor compartido. Se pueden especificar las opciones siguientes:

- Disable: desactiva la opción Connection Pooling tanto para las conexiones de red entrantes como para las salientes.
- Enable: activa la opción Connection Pooling para las conexiones de red entrantes y para las salientes.

Si activa la opción Connection Pooling, debe especificar el tamaño de la medida de red conocida como *tic*. El tamaño de un tic se especifica en segundos. Después de establecer el tamaño de un tic, se especifica la cantidad de tiempo, expresada en tics, que una conexión entrante deberá esperar antes de que finalice el tiempo de espera límite. También se puede especificar el tiempo de espera límite de una conexión saliente.

El último valor de la ficha de opciones de configuración avanzadas de Shared Server Mode es el número máximo de sesiones de red que se permiten por cada expedidor.

1.3.2 Configuración de los parámetros de inicialización: memoria

Oracle utiliza áreas de memoria compartida para gestionar sus estructuras de archivos y de memoria. Antes de ver los parámetros de inicialización que se pueden modificar en la pantalla Initializacion Parameters (parámetros de inicialización), se presenta aquí información detallada sobre las estructuras básicas de memoria que Oracle utiliza y cómo las utiliza. Las estructuras descritas son las siguientes:

- Área Global del Sistema (SGA, System Global Area): el tamaño de la SGA depende del tamaño de otros valores de parámetros de estructuras de memoria.
- Caché de buffers de bloques de datos.
- Caché de diccionario.
- Buffer de registros de reconstrucción.
- Área SQL compartida.
- Área Java.
- Múltiples conjuntos de buffers.
- Áreas de contexto.
- Área Global del Programa (PGA).

1.3.2.1 Área global del sistema (SGA)

El Área Global del Sistema (SGA) de una base de datos Oracle facilita la transferencia de información entre usuarios. Además, almacena la mayor parte de la información sobre la propia estructura de la base de datos que es consultada con más frecuencia.

1.3.2.2 Cachés de buffers

Tradicionalmente, la SGA ha sido una zona estática de memoria previamente reservada y compartida por todos los procesos de Oracle. El tamaño de memoria se calcula en función de los valores de los parámetros del archivo INIT.ora y, una vez reservada, el área de memoria no se puede ampliar ni reducir. Para cambiar el tamaño de los valores en los que se basa la SGA, es necesario cerrar primero la base de datos, cambiar el valor del parámetro del archivo INIT.ora y, a continuación, reiniciar la base de datos. Cerrar una base

de datos diseñada para tener una muy alta (casi del 100 por 100) disponibilidad con el fin de cambiar un parámetro, puede causar un impacto negativo en los sistemas de producción.

Para solucionar este problema, Oracle ha introducido un área de SGA dinámica en Oracle9i. Ahora se puede cambiar la configuración de la SGA mientras la base de datos se encuentra en funcionamiento, de modo que se puede cambiar la caché de buffer y el área compartida sobre la marcha sin necesidad de cerrar la base de datos. También se pueden imponer límites durante el inicio de la base de datos sobre cuánta memoria física se va a utilizar para la SGA. De esta forma, puede asignar menos memoria para la caché de buffers, el área compartida y el área de bloques de gran tamaño al arrancar y, a continuación, podrá aumentar o reducir el tamaño según convenga en función de la carga de trabajo que soporten. Basta con establecer un tamaño global máximo para la SGA y Oracle asigna áreas de la memoria a los diferentes componentes.

Para llevar a cabo la gestión de l asignación de memoria, Oracle proporciona una nueva unidad de asignación de la SGA denominada *gránulo* que representa una cantidad de memoria virtual contigua y se basa en el valor del parámetro SGA_MAX_SIZE. Para determinar el tamaño de un gránulo, puede fijarse en el tamaño total de la SGA. Si el tamaño total de la SGA es inferior a 128 MB, cada gránulo tendrá un tamaño de 4MB. Por el contrario, si la SGA supera los 128 MB, el tamaño predeterminado del gránulo será de 16 MB.

Puede determinar los gránulos asignados para los componentes de la caché de buffers que posee gránulos. La caché de buffers puede aumentar o disminuir su tamaño basándose en incrementos de gránulos. Inicialmente, la cantidad mínima de gránulos que se asigna es tres: uno para la SGA fija (incluyendo los buffers de reconstrucción), uno para la caché de buffers y otro para el área compartida. El área compartida debe tener un tamaño de al menos 8 MB de modo que la cantidad mínima es cuatro gránulos cuando cada gránulo sea de 4 MB.

Oracle 9i admite diferentes tamaños de bloques de base de datos dentro de una base de datos y cada tamaño de bloque debe contar con una caché de memoria asociada.

1.3.2.3 Caché de diccionario

La información sobre los objetos de base de datos se almacena en las tablas del diccionario de datos. Por ejemplo, la información del diccionario de datos incluye los datos de las cuentas de usuarios, los nombres de los archivos de datos, los nombres de los segmentos, la ubicación de las extensiones, las descripciones de las tablas y los privilegios. Cuando la base de datos necesita esta información, se leen las tablas del diccionario de datos y los datos devueltos se almacenan en *la caché de diccionario* de la SGA.

Esta memoria caché también se gestiona mediante un algoritmo LRA (Least Recently Used o menos recientemente usado). La base de datos gestiona internamente el tamaño de la caché de diccionario. Esta memoria caché forma parte del área SQL compartida, cuyo tamaño se define mediante el parámetro SHARED_POOL_SIZE del archivo de parámetros de inicialización de la base de datos.

Si la caché de diccionario es muy pequeña, la base de datos ha de consultar repetidamente las tablas del diccionario de datos para obtener la información que necesita.

Estas consultas se denominan llamadas recursivas y se resuelven de forma más lenta que las consultas que puede manejar por sí sola la caché de diccionario.

1.3.2.4 Buffer de registros de reconstrucción

Los elementos de los archivos de registro de reconstrucción describen los cambios realizados en la base de datos. Estos elementos se escriben en los archivos de registro de reconstrucción en línea, de forma que se pueden utilizar en las operaciones de re-ejecución de las transacciones durante las repercusiones de la base de datos. No obstante, antes de escribirse en los archivos de registro de reconstrucción en línea, las transacciones se registran en un área de la SGA llamada *buffer de registros de reconstrucción.* Después, periódicamente, la base de datos escribe de forma agrupada varias de estas transacciones de reconstrucción en los archivos de registro de reconstrucción en línea, optimizando así esta operación.

El tamaño (en bytes) de los buffers de registro de reconstrucción se especifica mediante el parámetro LOG_BUFFER del archivo de parámetros de inicialización.

1.3.2.5 Área compartida

El área compartida almacena la caché de diccionario de datos y la caché de biblioteca (información sobre las instrucciones ejecutadas contra la base de datos). Mientras la caché de buffers de bloques de datos y la caché de diccionario permiten que los usuarios de la base de datos compartan la información sobre los datos y su estructura, la caché de biblioteca permite que se compartan las instrucciones SQL que se emplean con más frecuencia.

El área SQL compartida contiene el plan de ejecución y el árbol de análisis de las instrucciones SQL que se ejecutan en la base de datos. La segunda vez que cualquier usuario ejecuta una misma instrucción, puede aprovecharse de la información de análisis disponible en el área SQL compartida, para acelerar su ejecución.

El área compartida se gestiona mediante el algoritmo LRU, Cuando un área SQL compartida se llena, los planes de ejecución utilizados menos frecuentemente y los árboles de análisis asociados se suprimen de la caché de biblioteca con el fin de hacer sitio para las nuevas entradas. Si un área SQL compartida es demasiado pequeña, las instrucciones se han de cargar continuamente en la caché de biblioteca y el rendimiento se resiente. Puede modificar dinámicamente el tamaño del área compartida revisando el parámetro SHARED_POOL_SIZE (en bytes). El parámetro SHARED_POOL_SIZE inicial se establece en el archivo de parámetros de inicialización. El valor debe ser un número entero múltiplo del tamaño del gránulo.

El área compartida incluye la caché de biblioteca, la caché de diccionario de datos y el área SQL compartida, como puede comprobar observando la vista V$SQLSTAT. A menos que haya configurado para el parámetro de inicialización SGA_MAX_SIZE un tamaño mayor que el total de la memoria que requieren los parámetros de inicialización, no podrá cambiar el tamaño de ningún componente. Alternativamente, podrá reducir el tamaño del área compartida; de esta forma, puede dar memoria a la caché de buffers o reducir el tamaño de la caché de buffers y dar memoria al área compartida. Si define un valor bastante

grande para SGA_MAX_SIZE, tendrá la flexibilidad necesaria para cambiar uno, cualquiera de estos componentes o ambos.

1.3.2.6 El área de bloques de gran tamaño

El área de bloques de gran tamaño es un área de memoria opcional. Si utiliza la opción servidor compartido o si lleva a cabo operaciones de copia de seguridad o restauración con frecuencia, estas operaciones podrían resultar más eficientes si creara un área de bloques de gran tamaño. Otra aplicación del área de bloques de gran tamaño es la de albergar los buffers de mensajes que se utilizan en las consultas paralelas. El tamaño del área de bloques de gran tamaño, en bytes, se establece inicialmente mediante el parámetro de inicialización LARGE_POOL_SIZE.

1.3.2.7 El área java

El área Java, como su nombre indica, se ocupa de satisfacer los requisitos de análisis de los comandos Java. El tamaño de área Java se define, en bytes, mediante el parámetro de inicialización JAVA_POOL_SIZE, introducido por primera vez en Oracle8i. Para el parámetro de inicialización JAVA_POOL_SIZE se debe definir un tamaño mínimo basado en el tamaño del gránulo de la base de datos. Por ejemplo, el tamaño predeterminado debería ser 24 MB en UNÍS, si el tamaño de gránulo es 4 MB y 32 MB, si el tamaño de gránulo es 16 MB.

1.3.2.8 Cachés de buffers múltiples

Puede crear cachés de buffers múltiples dentro de la SGA con el fin de dar soporte a múltiples tamaños de bloques de base de datos. Puede utilizar cachés de buffers múltiples para separar grandes conjuntos de datos del resto de la aplicación, reduciendo así la posibilidad de que compitan por los mismos recursos dentro de la caché de buffers. Es necesario especificar un tamaño para cada caché de buffers que se cree.

Cuando se crean cachés de buffers, se establecen los tamaños en el archivo de parámetros de inicialización. El ejemplo siguiente muestra entradas de una caché de tamaños de bloque de 4 KB y 16 KB. En este ejemplo, el tamaño predeterminado de bloque de base de datos es de 8KB.

```
DB_CACHE_SIZE = 128M
DB_4K_CACHE_SIZE = 64M
DB_16K_CACHE_SIZE = 128M
DB_BLOCK_SIZE = 8K
```

En este ejemplo, la memoria asignada para admitir bloques de 4 KB es de 64 MB, mientras que la asignación para soportar bloques de 16 KB es de 128 MB. Los valores establecidos no deben exceder el límite MAX_SGA_SIZE definido para todos los componentes. Si el conjunto de valores excede el límite de la SGA, la base de datos no se

abrirá. Se pueden tener hasta un máximo de cinco tamaños diferentes de bloques dentro de la base de datos Oracle9i.

1.3.2.9 Pantallas de parámetros de inicialización en el asistente de configuración de la base de datos

En la figura 1.9 se muestra la ficha Memory (memoria) de la pantalla Initialization Parameteers (parámetros de inicialización) de Database Configuration Assistant (asistente para la configuración de la base de datos) con los valores predeterminados. Se pueden visualizar todos los parámetros de inicialización haciendo clic en el botón All Initializacion Parameters de la parte inferior de la pantalla. Si ha limitado los recursos de memoria, puede volver a ampliar los valores del área compartida y del área Java, pero no pude asignar al área Java un valor inferior a un factor del tamaño del gránulo y debe asegurarse de que es un valor lo suficientemente alto para que el área compartida satisfaga los requisitos de la estructura de la memoria que haya definido. Si no tiene demasiado código Java, 20 MB deberían ser suficientes.

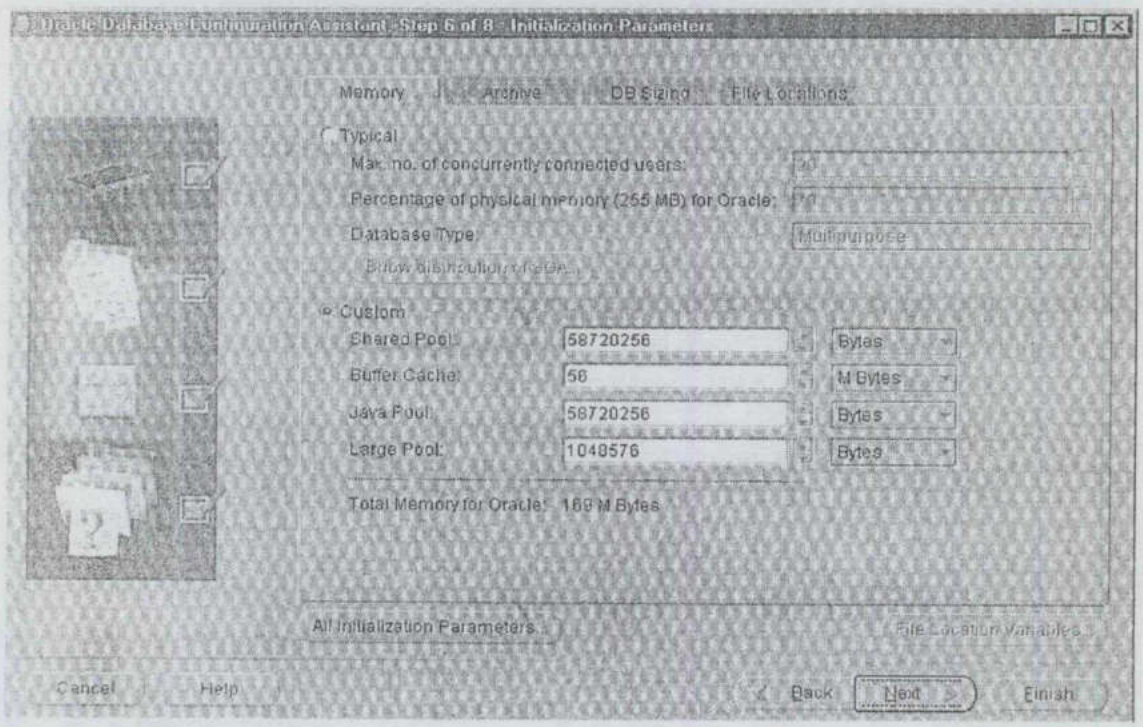


Fig. 1.9 Ficha Memory de la pantalla Initializacion Parameters.

La segunda ficha de la pantalla Initialization Parameters es Archive (archivado definitivo), mostrada en la figura 1.10. Puesto que la creación de una base de datos implica ejecutar un gran número de scripts que carguen grandes cantidades de datos en las estructuras de la base de datos, se recomienda no activar el registro de archivado mientras

Oracle crea la nueva base de datos. Se puede activar el registro de archivado cuando ya se haya creado la base de datos y se hayan rellenado las estructuras de base de datos.

En la tercera ficha de la pantalla Initialization Parameters se puede especificar el tamaño del área de ordenación y el juego de caracteres que se vaya a utilizar en la nueva base de datos. En la figura 1.11 se muestra la ficha DB Sizing (dimensionamiento de la base de datos). El valor de Sort Area Size (tamaño del área de ordenación) se puede cambiar dinámicamente o en el archivo de parámetros de inicialización cuando se haya creado la base de datos.

Una vez declarado el valor del juego de caracteres (character set) de la base de datos, sólo se puede modificar el valor si se ha elegido un juego de caracteres que incluya a otros juegos de caracteres de entre la lista de juegos de caracteres disponible. Se puede configurar un juego de caracteres de base de datos y además un juego de caracteres internacional para la nueva base de datos. El juego de caracteres define como traducirá y presentará Oracle los valores de caracteres cuando se devuelvan los datos de una consulta. Cuando se insertan datos en la base de datos, Oracle utiliza un método transparente y no evalúa los caracteres de entrada para asegurarse de que podrán ser reconocidos posteriormente cuando sean devueltos.

En la última ficha de Initialization Parameters, mostrada en la figura 1.13, se puede modificar la ubicación del archivo de parámetros init.ora, activar el uso de un archivo SPFILE (Persistent Stored Parameter File, archivo de parámetros almacenados persistentes) y modificar la ubicación de los archivos de traza.

Observe el uso de los valores de parámetros globales en los diferentes nombres de archivo. Haciendo clic en el botón File Location Variables (variables de ubicación de archivo), situado en la parte inferior de la pantalla, se puede ver la traducción de los valores.

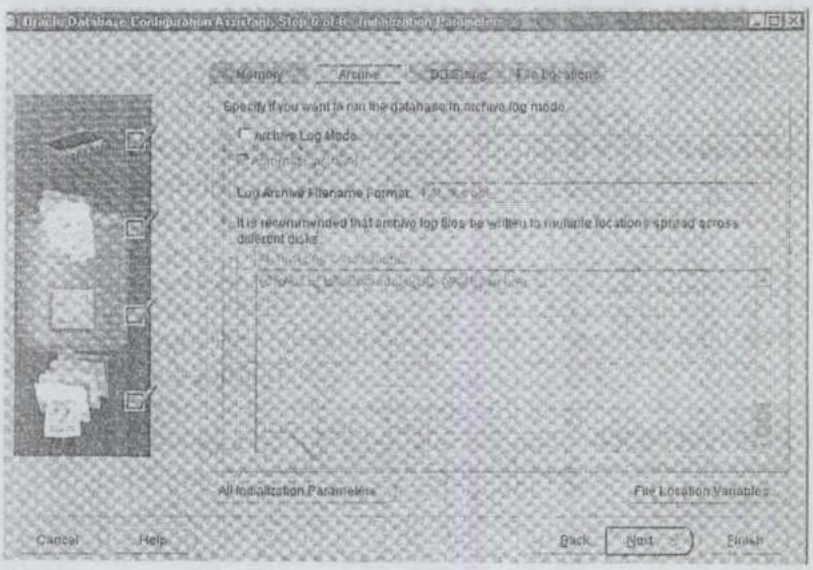


Fig. 1.10 Ficha Archive de la pantalla Initializacion parameters.

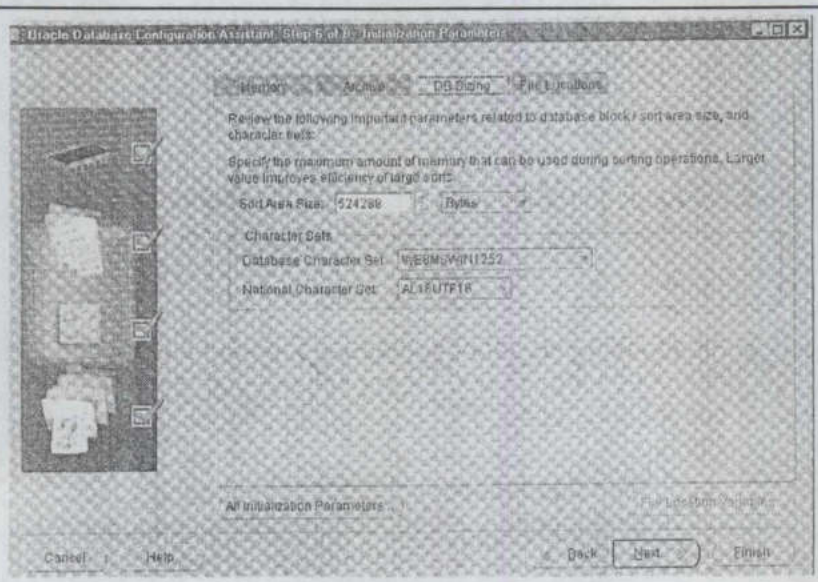


Fig. 1.11 Ficha DB Sizing de la pantalla Initializacion Parameters.

En la figura 1.13 se muestran los valores de este ejemplo. No puede cambiar los valores que se muestran en esta pantalla a menos que vaya a crear una base de datos utilizando la opción New Database Template (nueva platilla de base de datos). Con opción de plantilla, se ofrece la oportunidad de cambiar los valores en la siguiente pantalla del asistente.

Para modificar las ubicaciones de los grupos de registros de reconstrucción, de los archivos de control y de los archivos de datos se puede utilizar la opción Database Storage (almacenamiento de base de datos), mostrada en la Figura 1.14.

En la parte izquierda de la pantalla aparece un árbol de carpetas y, en la derecha, una explicación. Todavía no se pueden cambiar las variables de archivo predeterminadas, como puede ver en la figura 1.13, pero se pueden modificar interactivamente las opciones de ubicación de los archivos y especificar el tamaño de cada uno de los archivos.

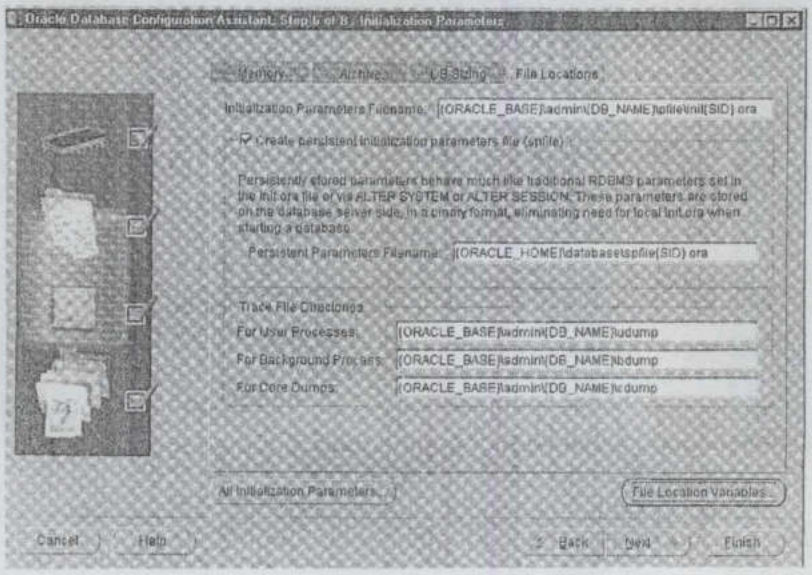


Fig. 1.12 Ficha File Locations de la pantalla Initializacion Parameters.

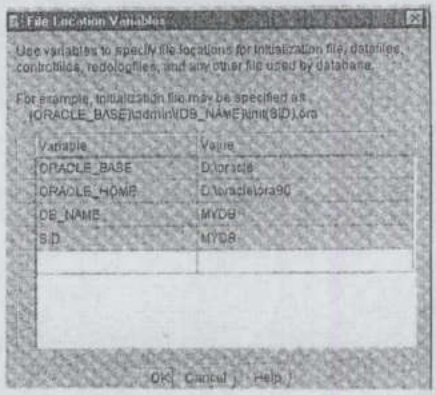


Fig. 1.13 Pantalla File Location Variables.

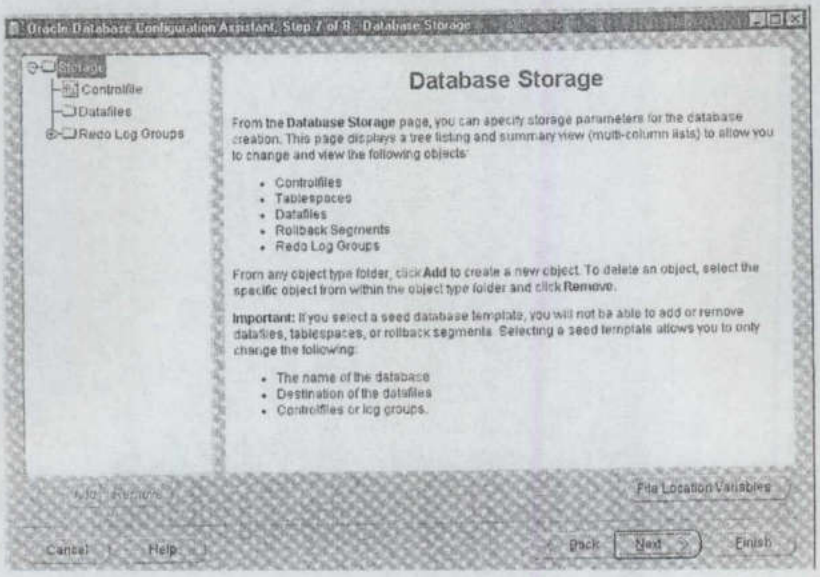


Fig. 1.14 Pantalla Database Storage.

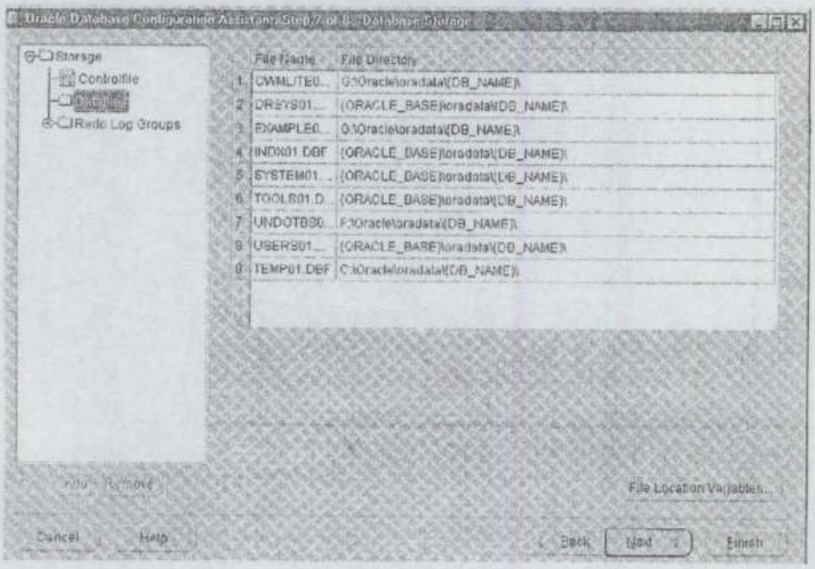
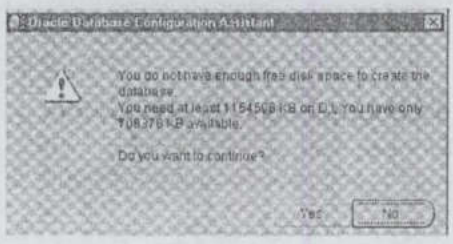


Fig. 1.15 Pantalla Data Files de Database Storage.

El siguiente paso del proceso de configuración de la base de datos consiste en indicar a Oracle que cree la base de datos, que almacene la plantilla de base de datos o ambas cosas. En la figura 1.15 se muestra la pantalla Creation Options (opciones de creación) con ambas opciones seleccionadas, además de un nombre y un comentario para la nueva plantilla de creación de base de datos, donde se explica que esta plantilla se utilizará para la creación de bases de datos en desarrollo.

Cuando decida guardar la plantilla, se le presentará un resumen de la misma para que lo revise. Entonces podrá continuar con la operación o cancelarla.

Si decide crear la base de datos y no dispone de espacio en disco suficiente para ello, recibirá un mensaje de advertencia, donde se le preguntará si desea continuar con la operación o cancelarla, como puede ver en la ilustración siguiente:



1.3.2.10 Análisis de la nueva base de datos

Una de las cosas que observará cuando Oracle cree una base de datos es que los usuarios predefinidos de la base de datos tienen las cuentas bloqueadas y las contraseñas caducas de antemano. Oracle está haciendo un esfuerzo notable para garantizar una mayor seguridad para la base de datos desde el mismo momento de la creación.

En una versión futura de Oracle9i, todas las cuentas que se creen en la base de datos estarán bloqueadas de forma predeterminada y las contraseñas caducadas, con la salvedad de las cuentas SYS y SYSTEM. Se solicitará al usuario que proporcione contraseñas para estas cuentas durante la creación de la base de datos. En la versión actual, cuando se ordena a Oracle que se cree una base de datos, inmediatamente después de la creación de la base de datos, se ofrece la oportunidad de modificar las contraseñas para las cuentas predeterminadas que se crean desbloqueadas con sus contraseñas estándar. Había cinco cuentas iniciales desbloqueadas con sus conocidas contraseñas en Windows NT Oracle9i, versión 9.0.1.0.0. Las cuentas que se encontraban inmediatamente disponibles en una base de datos Oracle9i eran:

- SYS
- SYSTEM
- DBSNMP
- AURORAJISUTILITY$

- AURORAORBUNAUTHENTICATED
- OUTLN
- SCOTT
- OSE$HTTP$ADMIN

Cuando se creaba una base de datos con el proceso habitual utilizando un script **create database,** los usuarios predeterminados, todos ellos con ACCOUNT_STATUS=OPEN, eran:

- SYS
- SYSTEM
- DBSNMP
- OUTLN

1.3.3 Cómo crear una base de datos manualmente

El proceso para la creación manual de una base de datos consta de varios pasos. Algunos de los pasos dependerán del sistema operativo. Por ejemplo, en un entorno Windows, es necesario ejecutar primero oradim, un programa ejecutable utilizado para crear un servicio de base de datos, antes de poder crear la base de datos.

Para crear manualmente una base de datos, debe seguir estos pasos:

1. Escriba un script de creación de base de datos. En el paso 6 se muestra un ejemplo de este tipo de scripts.

2. Cree la estructura de directorios que albergará la nueva base de datos. Siga las directrices de la arquitectura flexible óptima descritas en el capítulo 4.

3. Modifique un archivo init.ora existente o un ejemplo de los que proporciona Oracle para que refleje los parámetros de la nueva base de datos.

4. Declare el nombre SID de Oracle. En una plataforma Windows, se escribe el siguiente comando en el símbolo del sistema operativo:

```
set ORACLE_SID = mydb
```

En UNIX:

```
export ORACLE_SID=mydb
```

5. Conéctese a la base de datos mediante SQL*Plus, bien como **SYSTEM/MANAGER** as **sysdba** o como **/ as sysdba** y ejecute el comando para iniciar la base de datos en modo no montada, como puede ver a continuación:

```
startup nomount
pfile=<<D:\oracle\admin\mydb\scripts\initMYDB.ora<<;
```

Sustituya el valor del parámetro **pfile** por el de su archivo de parámetros.

6. Una vez iniciada la base de datos, utilice el script de creación de base de datos que haya escrito. A continuación se presenta un ejemplo de un script de creación de base de datos:

```
create database MYNEW
maxinstances 1
maxloghistory 1
maxlogfiles 5
maxlogmembers 5
maxdatafiles 100
datafile 'D:\oracle\oradata\mydb\system01.dbf'
     size 325M reuse autoextend on next 10240K maxsize
unlimited
character set WE8MSWIN1252
national character set AL16UTF16
logfile group 1 (d:\oracle\oradata\mydb\redo01.log') size
100M
     group 2 (d:\oracle\oradata\mydb\redo02.log') size
     100M
        group 3 (d:\oracle\oradata\mydb\redo03.log') size
     100M
default temporary tablespace TEMP
tempfile 'd:\oralce\oradata\mydb\temp01.dbf'
     extent management local uniform size 1M
undo tablespace UNDO_TS datafile
'd:\oracle\oradata\mydb\temp01.dbf'
     size 150M reuse autoextend on next 10240K maxsize
unlimited
```

7. Una vez creada la base de datos, ejecute los siguientes scripts: catalog.sql, catproc.sql, catexp.sql y cualquier otro script necesario para los productos que haya instalado. Los scripts residen en el directorio $ORACLE_HOME/rdbms/admin., en un sistema UNÍS, y en ORACLE_HOME\rdbms\admin. En un entorno Windows. Revise los scripts antes de ejecutarlos, porque muchos scripts de catálogo invocan a su vez a otros scripts de catálogo.

8. Para garantizar una seguridad más estricta, debe, como mínimo, sustituir las constraseñas de SYSTEM y SYS por cualquier otra que no sean sus valores predeterminados, MANAGER y CHANGE_ON_INSTALL, respectivamente.

En el script de ejemplo mostrado en el paso 6, se crea el espacio de tablas UNDO. Los parámetros de inicialización necesarios para dar soporte a la funcionalidad de deshacer cambios son:

```
undo_management=AUTO
undo_tablespace=UNDOTBS
```

El único parámetro que no se puede cambiar después de la creación de la base de datos es el tamaño de bloque de la base de datos que se ha declarado en el archivo init.ora antes de la creación de la base de datos. Para configurar este valor, se utiliza el parámetro DB_BLOCK_SIZE. Por ejemplo, la línea siguiente establece un tamaño predeterminado de bloque de base de datos de 8 KB.

```
DB_BLOCK_SIZE=8k
```

Para ver los parámetros que se han definido para la base de datos, consulte la vista dinámica V$PARAMETER:

```
select Name, Value, IsDefault
from V$PARAMETER;
```

1.4 Procesos en segundo plano

Las relaciones entre las estructuras físicas y de memoria de la base de datos se mantienen y aplican mediante *procesos en segundo plano.* Estos procesos son propios de la base de datos y su número varía en función de la configuración de la base de datos. La base de datos gestiona estos procesos y necesitan muy poco trabajo administrativo.

Los archivos de traza se crean únicamente cuando surge un problema. El convenio utilizado para asignar nombres y la ubicación de los archivos de traza de los procesos en segundo plano varían según las versiones de la base de datos y del sistema operativo. Por regla general, los nombres de archivo de traza contienen el nombre del proceso en segundo plano o, bien, el identificador del proceso del sistema operativo correspondiente al proceso en segundo plano. Se puede definir el parámetro de inicialización BACKGROUND_DUMP_DEST para especificar una ubucación para los archivos de traza de los procesos de segundo plano. Los archivos de traza juegan su papel más importante cuando surgen problemas de depuración en la base de datos. Los problemas serios que afectan a los procesos en segundo plano suelen reflejarse en el registro de alertas de la base de datos.

El registro de alertas suele residir en el directorio BACKGROUND_DUMP_DEST, que por lo general, es el directorio /admin./NOMBRE_INSTANCIA/bdump, situado debajo del directorio ORACLE_BASE.

Puede utilizar V$BGPROCESS para obtener una lista completa de los procesos en segundo plano disponibles en su base de datos.

1.4.1 SMON

Cuando se inicia la base de datos, el proceso SMON (System Monitor, monitor del sistema) realiza la recuperación de la instancia, cuando sea preciso. Además, limpia la base de datos, eliminando objetos transaccionales que el sistema ya no necesita.

SMON cumple una segunda función: agrupa extensiones libres contiguas en extensiones de mayor tamaño. El proceso de fragmentación del espacio libre se describe en el capítulo 4. en algunos espacios de tablas, los administradores de bases de datos deben realizar la agrupación del espacio libre manualmente. SMON sólo agrupa el espacio libre en espacios de tablas cuyo valor de almacenamiento predeterminado **pctincrease** sea diferente de cero.

1.4.2 PMON

El proceso en segundo plano PMON (Process Monitor, monitor de procesos) realiza una limpieza de los procesos de usuarios que han fallado. PMON libera los recursos que estuviera utilizando el usuario. Sus efectos se manifiestan cuando se elimina un proceso que mantiene un bloqueo: PMON libera el bloqueo y lo pone a disposición de otros usuarios. Al igual que SMON, PMON se activa de forma periódica para comprobar si es necesaria su intervención.

1.4.3 DBWR

El proceso en segundo plano DBWR (Database Writer, escritor de la base de datos) se encarga de gestionar los contenidos de la caché de buffers de los bloques de datos y de la caché de diccionario. El proceso DBWR realiza escrituras por lotes de los bloques modificados en los archivos de datos.

Aunque sólo hay un proceso SMON y otro PMON en ejecución por cada instancia de base de datos, es posible tener múltiples procesos DBWR ejecutándose a la vez, dependiendo de la plataforma y del sistema operativo. La utilización de varios procesos DBWR ayuda a minimizar la contienda dentro del propio DBWR durante operaciones de gran tamaño que afecten a varios archivos de datos.

El número de procesos DBWR en ejecución se establece mediante el parámetro DB_WRITER_PROCESSES del archivo de parámetros de inicialización de la base de datos. Si su sistema no soporta operaciones de E/S asíncronas, entonces puede crear un único proceso DBWR con múltiples procesos esclavos DBWR de E/S. El número de procesos esclavos DBWR de E/S se define mediante el parámetro de inicialización BDWR_IO_SLAVES. Con E/S asíncrona, puede tener múltiples procesos DBWn y sin E/S asíncrona puede utilizar procesos esclavos DBWR de E/S.

Si crea múltiples procesos DBWR, los procesos no se denominarán DBWR, sino que incluirán un componente numérico. Si crea, por ejemplo, cinco procesos DBWR, el sistema operativo les asignará los nombres DBW0, DBW1, DBW2, DBW3 y DBW4 respectivamente.

1.4.4 LGWR

El proceso en segundo plano LGWR (Log Writer, escritor del registro) gestiona la escritura del contenido del buffer del registro de reconstrucción en los archivos de registro de reconstrucción en línea. LGWR realiza dicha escritura por lotes. Los elementos del buffer del registro de reconstrucción siempre contienen el estado más actualizado de la base de datos, ya que es posible que el proceso DBWR espere antes de escribir los bloques modificados desde los buffers de los bloques de datos a los archivos de datos.

LGRW es el único proceso que escribe en los archivos de registro de reconstrucción en línea y el único que lee directamente los buffers del registro de reconstrucción durante el funcionamiento normal de la base de datos. Los archivos de registro de reconstrucción en línea se escriben de modo secuencial, al contrario que los accesos, bastante aleatorios, que realiza el proceso DBWR en los archivos de datos. Si loas archivos de registro de reconstrucción están duplicados en espejo, LGRW escribe simultáneamente en los conjuntos de registros duplicados. En el capítulo 2 se detalla esta funcionalidad de duplicación de espejo.

1.4.5 CKPT

Los puntos de comprobación ayudan a reducir la cantidad de tiempo necesaria para realizar la recuperación de las instancias. Los puntos de comprobación hacen que el proceso DBWR escriba en los archivos de datos todos los bloques que se hayan modificado desde el último punto de comprobación. El proceso CPT actualiza los archivos de control y las cabeceras de los archivos de datos para salvaguardar el punto de comprobación. Los puntos de comprobación se producen de forma automática cuando se llena un archivo de registro de reconstrucción en línea. Para configurar un punto de comprobación más frecuente, pueden utilizarse los parámetros LOG_CHECKPOINT_INTERVAL y LOG_CHECKPOINT_TIMEOUT especificados en el archivo de inicialización de la instancia de la base de datos.

El proceso en segundo plano CPT separa las dos funciones del proceso LGWR de versiones anteriores (señalar los puntos de comprobación y copiar los elementos de reconstrucción) en dos procesos en segundo plano.

1.4.6 ARCH

El proceso en segundo plano LGWR escribe en los archivos de registro de reconstrucción en línea de forma cíclica. Después de llenar el primer archivo, empieza a escribir en el segundo, hasta que lo llena, y empieza entonces a escribir en el tercero. Una vez que el último archivo de registro de reconstrucción en línea esta lleno, LGWR empieza a sobrescribir los contenidos del primero.

Cuando Oracle se ejecuta en modo ARCHIVELOG, la base de datos realiza una copia de cada uno de los archivos de registro de reconstrucción cuando se llenan. La copia de dichos archivos suele escribirse en un dispositivo de disco. La función de archivado la realiza el proceso en segundo plano ARCH. Las bases de datos que utilizan esta opción suelen experimentar problemas de contienda en su disco del registro de reconstrucción

durante los períodos con altas tasas de transacciones de datos, ya que el proceso LGWR intenta escribir en un archivo de registro de reconstrucción mientras ARCH intenta leer de otro. Si se llena el disco de destino del archivo, también pueden producirse bloqueos en la base de datos. En este caso, ARCH se detiene, lo que impide que LGWR siga escribiendo, lo que, a su vez, impide que se realicen más transacciones en la base de datos hasta que se libere espacio para los archivos de registro de reconstrucción que deben pasar a archivado definitivo.

Se pueden establecer múltiples áreas de destino para los registros de reconstrucción archivados y se puede especificar el número de áreas apropiado para que la instancia continúe.

1.4.7 RECO

El proceso en segundo plano RECO corrige los fallos de las bases de datos distribuidas dudosas e intenta resolver esas transacciones. Este proceso sólo se crea si la plataforma de trabajo admite la opción distribuida (Distributed Option) y se asigna un valor mayor que cero al parámetro DISTRIBUTED_TRANSACTIONS del archivo init.ora.

1.4.8 CJQn

La gestión de la cola de trabajos de Oracle depende de si se apoya en los procesos en segundo plano para la ejecución de actualizaciones de datos y otros trabajos programados. El proceso coordinador, denominado CJQ0, selecciona los trabajos que han de ejecutarse y lanza la ejecución de los procesos de cola de trabajo (de J000 a J999) para ejecutarlos. El número máximo de procesos de trabajos creados para una instancia se define mediante el parámetro JOB_QUEUE_PROCESSES del archivo de inicializa-ción de la base de datos.

1.4.9 LMSn

Se utilizan varios procesos *LMS,* con nombres entre LCK0 y LCK9, para que se encarguen de los bloqueos entre instancias cuando se utiliza la opción Oracle Real Application Clusters.

1.4.10 Dnnn

Los procesos *expedidores* forman parte de la arquitectura del servidor compartido. Al manejar múltiples conexiones, ayudan a minimizar las necesidades de recursos. Debe crearse al menos un proceso expedidor para cada protocolo que admita el servidor de base de datos. Los procesos expedidores se crean durante el inicio de la base de datos, basándose en el parámetro de inicialización DISPATCHERS y pueden crearse o eliminarse mientras la base de datos permanezca abierta.

1.4.11 Server: *Snnn*

Los procesos de *servidor* se crean para gestionar las conexiones con la base de datos que necesiten un servidor dedicado. Estos procesos de servidor pueden realizar operaciones de E/S en los archivos de datos. El número máximo de procesos de servidor se especifica en el parámetro de inicialización SHARED_SERVERS.

1.4.12 Procesos de servidor de consultas en paralelo: *Pnnn*

Si se activa la opción Parallel Query (consulta en paralelo) en la base de datos, los requisitos de recursos de una consulta tendrán que distribuirse entre múltiples procesadores.

El número de procesos de servidor de consultas en paralelo que se activan al iniciarse la instancia se determinan mediante el parámetro PARALLEL_MIN_SERVERS. Cada uno de estos procesos aparecerá directamente en el nivel del sistema operativo; cuantos más procesos se necesiten para operaciones en paralelo, más procesos de servidor de consultas en paralelo se activarán. Cada uno de los procesos de servidor de consultas en paralelo posee un nombre, como puede ser P000, P001 y P002, en el nivel del sistema operativo. El número máximo de procesos de servidor de consultas en paralelo se define con el parámetro de inicialización PARALLEL_MAX_SERVERS.

Los procesos de servidor de consultas en paralelo no generan archivos de traza a menos que se produzca algún error.

1.5 Estructuras internas de la base de datos

Una vez creada la base de datos, podrá crear estructuras internas que den soporte a las aplicaciones. Entre los elementos internos de la base de datos se incluyen los siguientes: Tablas, columnas, restricciones y tipos de datos (incluyendo los tipos abstractos de datos).

- Particiones y subparticiones.
- Usuarios y esquemas.
- Índices, clusters y clusters hash.
- Vistas.
- Secuencias.
- Procedimientos, funciones, paquetes y disparadores.
- Sinónimos.
- Privilegios y roles.
- Enlaces de base de datos.

- Segmentos, extensiones y bloques.
- Segmentos de anulación.
- Instantáneas y vistas materializadas.

1.5.1 Tablas, columnas y tipos de datos

Las tablas son el mecanismo de almacenamiento de los datos en una base de datos Oracle. Como se mostró anteriormente en la Figura 1.1, contienen un conjunto fijo de columnas. Las columnas de una tabla describen los atributos de la entidad que se representa con la tabla. Cada columna tiene un nombre y unas características específicas.

Una columna tiene un tipo de datos y una longitud. En las columnas que utilizan el tipo de datos NUMBER, pueden especificarse las características de precisión y escala. La *precisión* determina el número de cifras significativas de un valor numérico. La *escala* determina la posición de la coma decimal. La especificación NUMBER (9,2) en una columna, indica que tiene un total de nueve cifras, de las que dos son decimales. La precisión predeterminada es de 38 cifras, que además coincide con la precisión máxima. En la Tabla 1.1 se enumeran los tipos de datos disponibles.

Además de los tipos de datos que se enumeran en la Tabla 1.1, Oracle proporciona alternativas para tipos de datos ANSI estándar. Para los tipos de datos ANSI CHARACTER y CHAR, se utiliza el tipo de datos CHAR de Oracle. Para los tipos de datos ANSI CHARACTER VARYING y CHAR VARYING, se utiliza el tipo de datos VARCHAR2 de Oracle. Para los tipos de datos ANSI NUMERIC, DECIMAL, DEC, INTEGER, INT y SMALLINT, se utiliza el tipo de datos NUMBER de Oracle. Para los tipos de datos ANSI estándar FLOAT, REAL y DOUBLE PRECISIÓN, Oracle admite un tipo de datos FLOAT dentro de PL/SQL.

Tabla 1.1 Tipos de datos de Oracle

Tipo de datos	Descripción
CHAR	Campo de caracteres de longitud fija, con un máximo de 2.000 caracteres.
NCHAR	Campo de longitud fija para juegos de caracteres multibyte. El tamaño máximo es de 2.000 caracteres o de 2.000 bytes por fila, dependiendo del juego de caracteres, siendo la configuración predeterminada de 1 byte.
VARCHAR2	Campo de caracteres de longitud variable, con una longitud máxima de 4.000 caracteres.

Tipo de datos	Descripción
NVARCHAR2	Campo de longitud variable para juegos de caracteres multibyte. El tamaño máximo es de 4.000 caracteres o 4.000 bytes por fila, dependiendo del juego de caracteres, siendo la configuración predeterminada de 1 byte.
DATE	Campo de longitud fija, de 7 bytes. que se utiliza para almacenar todas las fechas. La hora se almacena como parte de la fecha. En las consultas, la fecha aparece en el formato DD-MON-YY como, por ejemplo, 22-APR-01 para el 22 de abril de 2001, a menos que modifique el formato de fecha configurando el parámetro de inicialización NLS_DATE_FORMAT.
INTERVAL DAY TO SECOND	Período de tiempo, almacenado en 11 bytes, representado como días, horas, minutos y segundos. Los valores de precisión especifican el número de dígitos de los campos DAY (día) y la parte decim al del campo SECOND (segundos) de la fecha. La configuración predeterminada de la precisión es 2 para los días y 6 para los segundos.
INTERVAL YEAR TO MONTH	Período de tiempo, almacenado en 5 bytes, representado como años y meses. El valor de precisión especifica el número de dígitos del campo YEAR (años) de la fecha. La configuración predeterminada de la precisión es 2 para años, pero puede ser cualquier valor entre 0 y 9.
TIMESTAMP	Valor que puede ocupar entre 7 y 11 bytes y que representa la fecha y hora, incluyendo fracciones de segundos. Se basa en el valor del reloj del sistema operativo. El valor de precisión especifica el número de dígitos de la parte de fracciones de segundo del campo de fecha SECOND (segundos). La configuración predeterminada de la precisión es 6, pero podría ser cualquier valor entre 0 y 9.
TIMESTAMP WITH TIME ZONE	Un valor, almacenado en 13 bytes, que representa una fecha y hora, además de una configuración de zona horaria. La zona horaria puede expresarse en relación al estándar UTC, como, por ejemplo, '-5:0', o utilizando un nombre de región, como 'US/Pacific'.
TIMESTAMP WITH LOCAL TIME	Este tipo de datos, que puede ocupar entre 7 y 11 bytes, es similar a TIMESTAMP WITH TIME ZONE con la salvedad de que los datos se normalizan con la zona horaria de la base de datos cuando se almacenan y se ajustan para que coincidan con la zona horaria del cliente cuando se consulten.

Tipo de datos	Descripción
NUMBER	Columna numérica de longitud variable. Los valores permitidos son cero y números positivos y negativos. El espacio de almacenamiento interno necesario para el valor NUMBER es aproximadamente la mitad del número de los dígitos significativos del valor. Para calcularlo, se toma la longitud total, por ejemplo, 9, se divide por 2, se redondea el resultado a un número entero y se suma 1 para un número positivo.
LONG	Campo de longitud variable, con una longitud máxima de 2 GB.
RAW	Campo de longitud variable para datos binarios, de una longitud máxima de 2.000 bytes.
LONG RAW	Campo de longitud variable para datos binarios, de una longitud máxima de 2 GB.
BLOB	Objeto binario de gran tamaño, de hasta 4 GB de longitud.
CLOB	Objeto de caracteres de gran tamaño, de hasta 4 GB de longitud.
NCLOB	Tipo de datos CLOB para juegos de caracteres multibyte, de hasta 4 GB de longitud.
BFILE	Archivo binario externo. El sistema operativo dicta el tamaño máximo.
ROWID	Datos binarios que representan un identificador de fila (RowID). Todos los RowID ocupan 6 bytes para índices normales de tablas no particionadas, índices locales en tablas particionadas y punteros de fila utilizados para filas encadenadas o migradas. El RowID ocupa 10 bytes sólo en los índices globales de tablas particionadas.
UNROWID	Datos binarios utilizados para el díreccionamiento de datos, de una longitud máxima de 4.000 bytes que admiten valores de RowID tanto lógicos como físicos, además de tablas externas a las que se acceda a través de una pasarela.

A partir de la versión Oracle8 puede crear sus propios tipos abstractos de datos al instalar la opción Object (objeto). También puede utilizar tipos de datos especiales REF que hagan referencia a objetos de fila situados en cualquier lugar de la base de datos al crear tablas de objetos.

Las tablas que posee el usuario SYS se denominan *tablas del diccionario de datos* y proporcionan un catálogo del sistema que la base de datos utiliza para gestionarse a sí misma. El diccionario de datos lo crea un conjunto de scripts de catálogo incluido en Oracle. Cada vez que se instala o actualiza una base de datos, es necesario ejecutar scripts que creen o que modifiquen las tablas del diccionario de datos. Cuando se instala una nueva

opción en la base de datos, es probable que haya que ejecutar scripts de catálogo adicionales. El usuario SYSTEM posee vistas en las tablas del diccionario de datos.

Las tablas se relacionan entre sí mediante las columnas que tienen en común. La base de datos puede utilizarse para asegurar que estas relaciones estén correctamente establecidas por medio de la *integridad referencia!.* Si utiliza las funciones orientadas a objetos de Oracle, las filas podrían estar relacionadas entre sí por medio de referencias internas llamadas identificadores de objeto. La integridad referencial se impone en el nivel de base de datos por medio de restricciones.

1.5.1.1 Tablas temporales

Al igual que una tabla normal, una tabla temporal es un mecanismo de almacenamiento de datos que reside en una base de datos Oracle. Una tabla temporal está compuesta de columnas para las que se han definido tipos de datos y longitudes. A diferencia de una tabla normal, la configuración de una tabla temporal se conserva, mientras que los datos insertados en la tabla permanecen en ella o bien durante una sesión, o bien, durante una transacción. Al crear la tabla temporal como una tabla temporal *global,* nos aseguramos de que todas las sesiones que se conecten a la base de datos podrán ver y utilizar la tabla. Varias sesiones pueden insertar filas de datos en la tabla temporal. Sin embargo, las filas de datos de la tabla sólo se pueden visualizar en la sesión en la que se ha insertado esa fila en la tabla.

Los datos contenidos en una tabla temporal son específicos de una sesión o específicos de una transacción, dependiendo de las palabras claves que se utilicen en la cláusula **on commit** asociada a la tabla temporal. Se puede especificar **on commit delete rows** (al confirmar eliminar filas) o bien **on commit preserve rows** (al confirmar conservar filas).

Las tablas temporales se introdujeron con Oracle8i y proporcionan un espacio para almacenar temporalmente unos resultados o un conjunto de resultados cuando las aplicaciones tienen que ejecutar varias instrucciones DML durante una transacción o una sesión.

1.5.2 Restricciones

Se pueden crear restricciones en las columnas de una tabla; cuando esto ocurre, todas las filas de la tabla deben cumplir las condiciones que se especifiquen en la definición de la restricción. Con el siguiente comando **create table,** se crea una tabla llamada EMPLOYEE con varias restricciones:

```
create table EMPLOYEE
(EmpNo          NUMBER(10)      PRIMARY KEY,
 Name           VARCHAR2(40)    NOT NULL,
 DeptNo         NUMBER(2)       DEFAULT 10,
 Salary         NUMBER(7,2)     CHECK (salary < 1000000),
 Birth_Date     DATE,
 Soc_Sec_Num    CHAR(9)         UNIQUE,
 foreign key    (DeptNo) references DEPT(DeptNo))
```

```
tablespace USERS;
```

Observe, en primer lugar, que la tabla recibe un nombre (EMPLOYEE), al igual que cada una de sus columnas (EmpNo, Name, etc.) Se especifica, para cada columna, un tipo de datos y una longitud. A la columna EmpNo se le ha asignado el tipo de datos NUMBER, sin valor de escala (lo que equivale a un entero). Para la columna Name se ha establecido el tipo VARCHAR2(40). Es decir, será una columna de longitud variable de hasta 40 caracteres.

La *clave primaria* de la tabla es la columna o conjunto de columnas que hacen que cada fila de la tabla sea única. Una columna de clave primaria se definirá dentro de la base de datos como *NOT NULL,* lo que significa que toda fila de dicha tabla deberá tener un valor para esa columna, no pudiendo dejarse en blanco (NULL). La restricción NOT NULL puede aplicarse a cualquier columna de la tabla, tal y como se hace con la columna Name en el ejemplo anterior.

Una columna puede tener una restricción *DEFAULT.* Este tipo de restricción se utiliza para generar un valor en una columna cuando se inserta **(insert)** una fila en una tabla sin especificarse un valor para dicha columna.

La restricción *CHECK* se utiliza para asegurarse de que los valores de una determinada columna cumplan un cierto criterio (en este caso, que el valor de la columna Salary sea menor que 1.000.000). Una restricción CHECK no puede hacer referencia, a una tabla diferente. La base de datos trata una restricción NOT NULL como si se tratara de una restricción CHECK.

Otra restricción, *UNIQUE,* garantiza la unicidad de columnas que deben ser unívocas, pero que no forman parte de la clave primaria. En el ejemplo anterior, la columna Soc_Sec_Num tiene una restricción UNIQUE, por lo que todos los registros de esta tabla han de contener un valor unívoco para esta columna.

Para especificar la naturaleza de la relación entre tablas, se utiliza una restricción de *clave externa.* Una clave externa de una tabla hace referencia a una clave primaria que se haya definido previamente en cualquier otro lugar de la base de datos. Por ejemplo, si otra tabla, llamada DEPT, tuviera una clave primaria denominada DeptNo, los registros de esa tabla contendrían todos los valores válidos de DeptNo. La columna DeptNo de la tabla EMPLOYEE hace referencia a esa columna DEPT.DeptNo. Al especificar EMPLOYEE. DeptNo como clave externa relacionada con DEPT.DeptNo, se garantiza que no se introduzca ningún valor DeptNo en la tabla EMPLOYEE sin que previamente exista en la tabla DEPT.

Las restricciones de la base de datos ayudan a garantizar la integridad referencial de los datos. De esta forma, uno puede estar seguro de que todas las referencias de la base de datos son válidas y de que se cumplen todas las restricciones.

1.5.3 Tipos abstractos de datos

En Oracle8 uno puede definir sus propios tipos de datos al instalar la opción Object (objeto) en la base de datos. Por ejemplo, se puede crear un tipo de datos que contenga todas las partes del nombre de una persona (nombre, segundo nombre, apellido y segundo

apellido) como un único tipo de datos. En el siguiente listado, se crea el tipo de datos NAME_TY:

```
create type NAME_TY as object
(First_Name          VARCHAR2(25),
Middle_Initial       CHAR(l) ,
Last_Name            VARCHAR2(30),
Suffix               VARCHAR2(5)) ;
```

Pueden utilizarse tipos de datos definidos por el usuario para estandarizar la utilización de datos en las aplicaciones. Por ejemplo, puede usarse el tipo de datos NAME__TY en cualquier sitio en el que se usaría otro tipo de datos. En el siguiente listado, se vuelve a crear la tabla EMPLOYEE, pero, esta vez, se emplea el tipo de datos NAME_TY para la columna EMPLOYEE.Name:

```
crate table EMPLOYEE
(EmpNo          NUMBER(10)          PRIMARY KEY,
 Name           NAME_TY,
 DeptNo         NUMBER(2)           DEFAULT 10,
 Salary         NUMBER(7,2)         CHECK (salary<1000000) ,
 Birth_Date     DATE,
 Soc_Sec_Num    CHAR(9)             UNIQUE,
 foreign key (DeptNo) references DEPT(DeptNo)
tablespace USERS;
```

La columna Name de la tabla EMPLOYEE contiene cuatro atributos, tal y como se indica en la instrucción de creación de NAME_TY. Si define *métodos* (programas que modifican los atributos de los tipos de datos) en el tipo de datos NAME_TY puede aplicar esos métodos a los valores de la columna Name en la tabla EMPLOYEE.

1.5.3.1 Métodos constructores

Cuando se crea un tipo abstracto de datos, Oracle crea, automáticamente, un método constructor para dar soporte a la manipulación de datos (DML) en la columna que utiliza dicho tipo de datos. Para NAME_TY, el método constructor se llama NAME_TY, y los parámetros de este método son los atributos del tipo de datos

1.5.3.2 Tablas de objetos

Una tabla de objetos es una tabla cuyas filas son todas objetos. Es decir, todas contienen valores del tipo identificador de objeto (OID). Para crear una tabla de objetos, puede utilizar el comando **create table.** Por ejemplo, puede utilizar el comando **create table** del siguiente listado para crear una tabla NAME basada en el tipo de datos NAME_TY:

```
create  table NAME of NAME_TY;
```

Así podrá crear referencias desde otras tablas a los objetos de fila de la tabla de objetos NAME. Si crea referencias a los objetos de fila de NAME, podrá seleccionar filas de NAME por medio de referencias, sin consultar directamente la tabla NAME.

1.5.3.3 Tablas anidadas y matrices variables

Una tabla anidada es una columna (o columnas) de una tabla que contiene múltiples valores para una misma fila. Por ejemplo, si tiene varias direcciones de una persona, puede crear una fila dentro de una tabla que contenga varios valores para la columna Address y un único valor para el resto de las columnas. Las tablas anidadas pueden contener varias columnas y un número ilimitado de filas. Existe un segundo tipo de colección de datos, llamado matriz variable, que puede contener un número de filas limitado. Las tablas anidadas ofrecen más flexibilidad en la administración de datos que las matrices variables. Tanto si utiliza tablas anidadas como si utiliza matrices variables, tiene que modificar la sintaxis SQL utilizada para acceder a los datos. Normalmente, se pueden simular las relaciones de las tablas anidadas por medio de tablas relacionales relacionadas entre sí.

1.5.4 Particiones y subparticiones

A medida que aumenta el tamaño de las tablas, el mantenimiento se hace cada vez más difícil. En las bases de datos de muy gran tamaño, podría facilitar enormemente las actividades de administración de la base de datos dividiendo los datos de una tabla grande en varias tablas más pequeñas. Por ejemplo, podría dividir una tabla en otras más pequeñas utilizando como criterio los valores de hora, departamento o producto contenidos en la tabla.

Se puede especificar que la base de datos utilice determinados rangos a la hora de dividir una tabla de muy gran tamaño en otras más pequeñas. Estas tablas de menor tamaño se llaman *particiones y* suelen ser más fáciles de gestionar que las tablas más grandes. Por ejemplo, se pueden truncar **(truncate)** los datos de una partición sin truncar los datos de las restantes. Oracle trata una tabla particionada como si fuera una sola tabla de mayor tamaño, pero las particiones se pueden gestionar como objetos independientes.

Las particiones también pueden mejorar el rendimiento de una aplicación. Dado que el optimizador conoce los valores de rango utilizados como base de las particiones, podrá hacer que las consultas sólo utilicen determinadas particiones durante los accesos a las tablas. Como durante el procesamiento de las consultas se leen menos datos, el rendimiento de las mismas debería ser mejor.

Además de las tablas, también se pueden particionar los índices. Los rangos de valores de las particiones de un índice particionado pueden coincidir con los rangos utilizados para la tabla indexada, en cuyo caso, el índice se llama *índice local.* Si las particiones del índice no coinciden con los rangos de valores utilizados para las particiones de la tabla, el índice se llama *índice global.*

A partir de Oracle8i las particiones se pueden dividir en otras particiones, creando de este modo, *subparticiones.* Por ejemplo, se puede dividir una tabla en particiones utilizando como criterio una serie de valores y, a continuación, volver a particionar las particiones utilizando un segundo método de partición. A partir de Oracle9i, se pueden *crear particiones por listas,* además de las particiones basadas en rangos y hash.

1.5.5 Usuarios

Una cuenta de *usuario* no es una estructura física de la base de datos, pero sí que tiene importantes relaciones con los objetos de la base de datos: los usuarios son propietarios de los objetos de la base de datos. El usuario SYS es propietario de las tablas del diccionario de datos, en las que se almacena información sobre el resto de las estructuras de la base de datos. El usuario SYSTEM posee las vistas que permiten acceder a estas tablas del diccionario de datos, para que las utilicen los restantes usuarios de la base de datos.

Cuando se crean objetos en la base de datos, se crean bajo cuentas de usuario. Cada una de estas cuentas se puede personalizar para que utilice un espacio de tablas específico de manera predeterminada.

Las cuentas de la base de datos se pueden asociar a cuentas del sistema operativo, permitiendo así a los usuarios el acceso a la base de datos desde el sistema operativo, sin tener que introducir una contraseña para el sistema operativo y luego otra para la base de datos. Los usuarios pueden acceder a los objetos que poseen o a aquellos a los que se les ha concedido acceso.

1.5.6 Esquemas

El conjunto de objetos que posee una cuenta de usuario se denomina *esquema* del usuario. Se pueden crear usuarios que no tengan la capacidad de iniciar una sesión en la base de datos. Tales cuentas de usuario proporcionan un esquema que puede utilizarse para guardar un conjunto de objetos de base de datos separado de otros esquemas de usuario.

1.5.7 Índices

Para que sea posible encontrar los datos, todas las filas de todas las tablas se etiquetan con un identificador *RowID (identificador de fila).* Este identificador de fila indica a la base de datos la ubicación exacta de la fila (archivo, bloque del archivo y fila del bloque).

Una tabla organizada con un índice no posee los tradicionales RowID de Oracle, sino que la clave primaria actúa como un RowID lógico.

Un índice es una estructura de base de datos que utiliza el servidor para localizar rápidamente una fila de una tabla. Existen tres tipos de índices: índices de cluster, índices de tabla e índices de mapa de bits. Los índices de cluster almacenan los valores de clave de cluster en clusters. Un índice de tabla almacena los valores de las filas de una tabla junto con la ubicación física de la fila, es decir, su RowID. Un índice de mapa de bits es un tipo especial de índice de tabla diseñado para dar soporte a consultas de tablas de gran tamaño con columnas que contengan pocos valores distintos.

Los índices constan de un valor clave y de un identificador de fila (RowID). Se puede indexar una sola columna o un conjunto de columnas. Oracle almacena los elementos de los índices utilizando un mecanismo de árbol binario que garantiza una ruta de acceso corta hasta el valor clave. Cuando una consulta accede al índice, busca las entradas del índice que coincidan con los criterios de consulta. El valor RowID que coincida con la consulta proporciona a Oracle la ubicación física de la fila asociada, reduciendo así la carga de E/S necesaria para localizar los datos.

Los índices se utilizan tanto para mejorar el rendimiento como para garantizar (opcionalmente) la unicidad de una columna. Oracle crea un índice, de forma automática, siempre que se especifique una cláusula de restricción UNIQUE o PRIMARY KEY en un comando **create table** (crear tabla). El comando **create index** (crear índice) sirve para crear índices de forma manual.

Los índices se pueden crear a partir de una o de varias columnas de una tabla. En el ejemplo anterior de la tabla EMPLOYEE, Oracle creará de forma automática índices unívocos para las columnas EmpNo y Soc_Sec_Num, ya que se han especificado como PRIMARY KEY y como UNIQUE, respectivamente. La eliminación de un índice no afecta a los datos contenidos en la tabla previamente indexada.

A partir de Oracle7.3 pueden crearse *índices de mapas de bits.* Este tipo resulta muy útil cuando los datos no son muy selectivos (cuando hay muy pocos valores distintos en una columna). Los índices de mapas de bits aceleran las búsquedas en las que se utilizan esas columnas poco selectivas como limitaciones para las consultas. Los índices de mapas de bits son muy efectivos con los datos muy estáticos.

En Oracle8 pueden crearse índices que *invierten* el orden de los datos antes de almacenarlos. Es decir, un elemento cuyo valor de datos sea de '1002' se indexará como '2001'. La inversión del orden de los datos antes de la indexación ayuda a mantener los datos mejor distribuidos dentro del índice. Como estos índices invierten los valores de los datos, sólo se usan para realizar operaciones de equivalencia en las consultas, tales como:

```
where key_col_value  =  1002
```

Para realizar búsquedas de rango, como:

```
where  key_col_value  >  1000
```

los índices de orden inverso no satisfarán de forma eficaz sus necesidades porque los valores consecutivos no se almacenarán uno al lado del otro. Puesto que las filas consecutivas se almacenarán en ubicaciones separadas, las consultas basadas en rangos no pueden utilizar los índices de clave inversa.

En Oracle8 puede crear una tabla organizada mediante índice. En este tipo de tabla, que se especifica mediante la cláusula **organization index** del comando **create table,** toda la tabla se almacena dentro de una estructura de índice, con los datos ordenados por la clave primaria de la tabla. Para crear una tabla organizada mediante un índice, hay que especificar una restricción de clave primaria para la tabla. La tabla de sólo índice no tendrá identificadores de fila (RowID) para las filas, Oracle utiliza el valor de clave primaria como

un valor de identificador de fila lógico. A partir de Oracle8i se pueden crear índices secundarios en tablas organizadas con índices.

Se pueden crear índices basados en funciones utilizando funciones de Oracle o funciones definidas por los usuarios. Por ejemplo, podemos crear un índice basado en UPPER(Name), en lugar de utilizar simplemente la columna Name si sabemos que las cláusulas **where** de las consultas siempre utilizan la función **UPPER.**

1.5.8 Clusters

Las tablas a las que se suele acceder conjuntamente pueden almacenarse físicamente juntas. Para almacenarlas juntas, se crea un *cluster* que contenga las tablas. Los datos de las tablas se almacenan entonces juntos con el fin de minimizar el número de operaciones de E/S que deben realizarse y mejorar así el rendimiento.

Las columnas relacionadas de las tablas se denominan *clave de cluster.* Esta clave de cluster se indexa utilizando un *índice de cluster,* y su valor se almacena una única vez para las diversas tablas del cluster. Debe crear el índice de cluster antes de insertar **(insert)** cualquier fila en las tablas del cluster.

Los clusters pueden resultar ventajosos en las tablas que se consultan con frecuencia. Dentro del cluster, filas de tablas diferentes se almacenan en los mismos bloques, de forma que las consultas que combinen estas tablas no necesitan llevar a cabo tantas operaciones de E/S como si las tablas se almacenaran en ubicaciones diferentes. Sin embargo, el rendimiento de las operaciones de inserción **(insert),** actualización **(update)** y eliminación **(delete)** de tablas agrupadas puede ser notablemente inferior que en las mismas operaciones realizadas en tablas no agrupadas. Antes de agrupar tablas, calcule la frecuencia con la que se consultan estas tablas conjuntamente. Si siempre se consultan juntas, debería considerar la posibilidad de combinar las dos tablas en una sola en lugar de agruparlas.

1.5.9 Clusters hash

Existe un segundo tipo de cluster: los *clusters hash.* Estos clusters utilizan *funciones de tipo hash* sobre la clave de cluster de la fila para determinar la ubicación física en la que debe almacenarse una fila. Así se obtiene el mayor rendimiento en las consultas de equivalencia, como la que se muestra a continuación:

```
select Name
from EMPLOYEE
where EmpNo = 123;
```

En este ejemplo, se busca en la tabla EMPLOYEE una coincidencia exacta de la columna EmpNo. Si la tabla EMPLOYEE forma parte de un cluster hash y EmpNo forma parte de la clave de cluster, la base de datos puede utilizar la función de hash para determinar rápidamente la ubicación física de los datos. La mejora de rendimiento no suele ser la misma si, en la cláusula **where,** se especifica un rango de valores, como en el listado siguiente:

```
select Name
from EMPLOYEE
where EmpNo > 123;
```

Buscar una fila en una tabla indexada normal puede necesitar múltiples operaciones de E/S, una o más para encontrar el valor clave del índice y otra para leer la fila de la tabla. El uso de un algoritmo hash reduce el número de operaciones E/S necesario para devolver la fila en consultas de equivalencia.

1.5.10 Vistas

Una *vista* se asemeja a una tabla con columnas y se consulta de la misma forma que una tabla. Conceptualmente, podemos imaginarnos una vista como una máscara que recubre a una o más tablas, de forma que las columnas de la vista se encuentran realmente en una o varias tablas subyacentes. Así, las vistas no almacenan físicamente los datos. La definición de una vista (en la que se incluye la consulta en la que se basa, la disposición de sus columnas y los privilegios concedidos) se almacena en el diccionario de datos.

Cuando se consulta una vista, ésta consulta las tablas en que se basa y devuelve los valores en el formato y orden especificados en la definición de la vista. Las vistas no pueden indexarse, ya que no hay datos físicos directamente asociados a ellas.

A menudo, las vistas se utilizan para aplicar mecanismos de seguridad en el nivel de fila de datos. Por ejemplo, se podría conceder a un usuario acceso a una vista que sólo mostrara las filas de ese usuario contenidas en una tabla, y no concederle acceso al resto de las filas. De forma similar, se puede limitar el número de columnas que puede ver el usuario utilizando una vista.

En Oracle8 se pueden definir *vistas objeto* para crear un nivel orientado a objetos por encima de las tablas. Las vistas objeto también sirven para simular tipos abstractos de datos, identificadores de objeto (object identifier, OID) y referencias.

1.5.10.1 Vistas objeto

Cuando se utilizan tipos abstractos de datos, pueden surgir problemas de coherencia durante la implementación. El acceso a los atributos de tipos abstractos de datos requiere el uso de sintaxis que no se utiliza para acceder a columnas normales. Como resultado, es probable que haya que cambiar los estándares de codificación SQL de la empresa para que se puedan utilizar los tipos abstractos de datos. También es preciso recordar qué tablas utilizan tipos abstractos de datos cuando se llevan a cabo transacciones y consultas sobre las tablas.

Las vistas objeto constituyen un importante puente hacia los tipos abstractos de datos. Se pueden utilizar vistas objeto para proporcionar una presentación de tipo objeto-relacional a los datos relacionales. Las tablas subyacentes no se modifican, pero las vistas admiten la configuración de tipos abstractos de datos. Desde la perspectiva de un DBA, hay muy pocos cambios: las tablas se gestionan como se gestionaría cualquier otra tabla de la base de datos. Desde el punto de vista de un desarrollador, las vistas de objeto proporcionan acceso de tipo objeto-relacional a los datos de las tablas.

1.5.11 Secuencias

Las definiciones de *secuencias* también se almacenan en el diccionario de datos. Las secuencias proporcionan una lista consecutiva de números unívocos que sirve para simplificar las tareas de programación.

La primera vez que una consulta llama a una secuencia, se devuelve un valor predeterminado. En cada consulta subsiguiente, se obtendrá un valor incrementado según el incremento especificado. Las secuencias pueden ser cíclicas o seguir creciendo hasta alcanzar un valor máximo especificado.

Cuando se utiliza una secuencia, no se ofrecen garantías de que se pueda generar una cadena de valores que no esté rota. Por ejemplo, si busca el valor siguiente de una secuencia para utilizarlo en un comando **insert,** la suya es la única sesión que puede utilizar ese valor de secuencia. Si no confirma su transacción, el valor de secuencia no se insertará en la tabla y las inserciones posteriores utilizarán los valores subsiguientes de la secuencia.

1.5.12 Procedimientos

Un *procedimiento* es un bloque de instrucciones PL/SQL que se almacena en el diccionario de datos y al que pueden llamar las aplicaciones. Los procedimientos permiten almacenar dentro de la base de datos la lógica de las aplicaciones que se emplea con más frecuencia. Cuando se ejecuta el procedimiento, sus instrucciones se ejecutan como una unidad. Los procedimientos no devuelven ningún valor al programa que los llama.

Se pueden utilizar procedimientos que ayuden a forzar la seguridad de los datos. En lugar de conceder a los usuarios acceso directamente a las tablas de una aplicación, se les puede conceder la capacidad de ejecutar un procedimiento que acceda a las tablas. Cuando se ejecuta el procedimiento, lo hará con los privilegios de su propietario. Los usuarios no podrán acceder a las tablas, si no es por medio del procedimiento.

1.5.13 Funciones

Las funciones, al igual que los procedimientos, son bloques de código que se almacenan en la base de datos. A diferencia de éstos, las funciones pueden devolver valores al programa que las llama. Puede crear sus propias funciones e invocarlas desde las instrucciones SQL, de igual forma que se ejecutan las funciones que proporciona Oracle.

Por ejemplo, Oracle proporciona una función llamada **SUBSTR** que realiza funciones de «tratamiento de cadenas>> en cadenas de caracteres. Si crea una función denominada **MY_SUBSTR** que realice operaciones de «tratamiento de cadenas>> personalizadas, puede invocarla desde un comando SQL.

```
select MY_SUBSTR('texto') from DUAL;
```

En caso de no ser el propietario de la función **MY_SUBSTR,** tendrán que haberle concedido el permiso EXECUTE sobre la función. Sólo se puede utilizar una función

definida por el usuario en una instrucción SQL si la función no modifica ninguna fila de la base de datos.

1.5.14 Paquetes

Los *paquetes* se pueden utilizar para organizar los procedimientos y las funciones en agrupaciones lógicas. Las especificaciones y el contenido de los paquetes se almacenan en el diccionario de datos. Los paquetes son muy útiles en las labores administrativas necesarias para gestionar los procedimientos y las funciones.

Dentro de un paquete hay elementos que pueden definirse como «públicos>> y otros, como «privados>>. Los elementos públicos son accesibles para el usuario del paquete, mientras que los elementos privados no están a su disposición, sino que están ocultos. Entre los elementos privados se incluyen procedimientos que son invocados por otros procedimientos contenidos en el paquete.

El código fuente de las funciones, los paquetes y los procedimientos se almacena en las tablas del diccionario de datos. Si sus aplicaciones utilizan paquetes con mucha frecuencia, tal vez tenga que aumentar considerablemente el tamaño del espacio de tablas SYSTEM, para adaptarlo al aumento de tamaño del diccionario de datos.

El número y la complejidad de los paquetes que se estén utilizando influyen directamente en el tamaño del área SQL compartida de la SGA.

1.5.15 Disparadores

Los *disparadores* son procedimientos que se ejecutan cuando se produce un suceso de base de datos determinado en una tabla específica. Pueden utilizarse para aumentar la integridad referencial, imponer requisitos de seguridad adicionales o mejorar las opciones de auditoría disponibles. Existen dos tipos de disparadores:

Disparadores de instrucción	Se activan una vez por cada instrucción de disparo.
Disparadores de fila	Se activan una vez por cada fila de una tabla afectada por las instrucciones.

Por ejemplo, un disparador de instrucción se activa una sola vez para un comando **delete** que elimina 10.000 filas. Un disparador de fila se activaría 10.000 veces para esa misma transacción.

Para cada tipo de disparador, puede crearse un disparador BEFORE (antes) y otro AFTER (después) para cada tipo de suceso de disparo. Entre los sucesos de disparo se encuentran las operaciones **insert** (insertar), **update** (actualizar) y **delete** (eliminar).

Los disparadores de instrucción son útiles si el código del disparador no depende de los datos afectados. Por ejemplo, se puede crear un disparador de instrucción BEFORE INSERT en una tabla, para impedir que se efectúen operaciones de inserción **(insert)** en dicha tabla excepto durante determinados períodos de tiempo.

Los disparadores de fila son útiles si la acción del disparador depende de los datos afectados por la transacción. Por ejemplo, puede crearse un disparador de fila AFTER INSERT que introduzca filas nuevas en una tabla de auditoría, así como en la tabla base del disparador.

A partir de Oracle8 se pueden crear disparadores INSTEAD OF. Un disparador INSTEAD OF se ejecuta en lugar de la acción que hizo que se iniciase. Es decir, si se creara un disparador INSTEAD OF INSERT en una vista objeto, el código del disparador se ejecutaría y la operación de inserción **(insert)** que hizo que se ejecutara el disparador no se produciría nunca. Si una vista combina varias tablas en una consulta, un disparador INSTEAD OF puede redirigir las acciones de Oracle en caso de que un usuario trate de actualizar **(update)** las filas utilizando directamente la vista.

A partir de Oracle8i se pueden crear disparadores que actúen asociados a sucesos que tengan lugar a nivel del sistema. Puede configurar un disparador para que ejecute un código cuando se empleen los comandos **create, alter o delete.** También se pueden utilizar sucesos del sistema del tipo conexión y desconexión de la base de datos e inicio y apagado de la propia base de datos como sucesos de disparo.

Un nuevo disparador denominado **on logon** (al conectar) permite la configuración de valores de entorno y la asociación de estos valores a un usuario desde el momento en que ese usuario se conecta a la base de datos.

1.5.16 Sinónimos

Para realizar una identificación completa de un objeto de base de datos (como una tabla o una vista) en una base de datos distribuida, es necesario especificar el nombre de la máquina host, el nombre de la instancia, el propietario del objeto y el nombre del objeto. En función de la ubicación del objeto, serán necesarios entre uno y cuatro de estos parámetros. Para ocultar este proceso al usuario, los desarrolladores pueden crear sinónimos que apunten al objeto adecuado; de este modo, los usuarios sólo necesitan saber el nombre del sinónimo. Los sinónimos públicos son los que comparten todos los usuarios de una base de datos determinada. Los sinónimos privados pertenecen a los propietarios de cuentas de base de datos individuales.

Por ejemplo, la tabla EMPLOYEE descrita anteriormente debe ser propiedad de una cuenta (digamos que el propietario es HR). Se podría hacer referencia a dicha tabla como HR.EMPLOYEE desde otra cuenta de usuario de la misma base de datos. Sin embargo, para ello es necesario que la segunda cuenta sepa que la cuenta HR es la propietaria de la tabla EMPLOYEE. Para evitar esto, se puede crear un sinónimo público llamado EMPLOYEE que apunte a HR.EMPLOYEE. Siempre que se haga referencia a este sinónimo, apuntará a la tabla adecuada. La siguiente instrucción SQL permite crear dicho sinónimo:

```
createpublic synonym EMPLOYEE for HR.EMPLOYEE;
```

Los sinónimos pueden emplearse para proporcionar punteros a las tablas, vistas, procedimientos, funciones, paquetes y secuencias. Pueden apuntar a objetos de una base de

datos local o de bases de datos remotas. Para esto último es necesario utilizar enlaces de base de datos, como se describe en breve.

No se pueden crear sinónimos para tipos abstractos de datos. Además, Oracle no comprueba la validez de un sinónimo cuando lo crea. Debería probar los sinónimos antes de crearlos con el fin de asegurarse de que son válidos.

1.5.17 Privilegios y roles

Para que una cuenta pueda acceder a un objeto propiedad de otra cuenta, tiene que habérsele concedido previamente el *privilegio* de acceso a dicho objeto. Lo habitual es que a las cuentas que no son propietarias se les concedan los privilegios de insertar **(insert),** seleccionar **(select)**, actualizar **(update)** o borrar **(delete)** filas de una tabla o de una vista. También puede concedérseles el privilegio de seleccionar **(select)** valores de secuencias y de ejecutar **(execute)** procedimientos, funciones, paquetes y tipos abstractos de datos. No se conceden privilegios sobre los índices ni los disparadores, ya que la base de datos los utiliza durante la actividad de la tabla. Se puede conceder el privilegio de leer **(read)** en los directorios (para tipos de datos BFILE y las tablas externas) y de ejecutar **(execute)** en las bibliotecas (para programas externos llamados por el código de su aplicación). Los privilegios pueden concederse a usuarios individuales o a PUBLIC, lo que concede el privilegio a todos los usuarios de la base de datos.

Se pueden crear *roles,* es decir, grupos de privilegios, para simplificar el proceso de gestión de los privilegios. Pueden concederse privilegios a un rol, y éste, a su vez, concederse a varios usuarios. Así, el proceso de añadir nuevos usuarios a las aplicaciones resulta mucho más fácil de gestionar, ya que se reduce simplemente a conceder roles al usuario o revocárselos.

En la Figura 1.17 se muestra la relación entre privilegios y roles. En la Figura 1.17a, se representan mediante líneas los privilegios que se requieren para conceder acceso **select** sobre dos tablas a cuatro usuarios. En la Figura 1.17b, la capacidad de los roles se emplea para simplificar la administración de privilegios. Los privilegios se conceden a un único rol, y ese rol se concede a los cuatro usuarios. Los roles se pueden activar y desactivar dinámicamente dentro de una aplicación.

Los roles también pueden emplearse para conceder privilegios de nivel de sistema, como **createtable.**

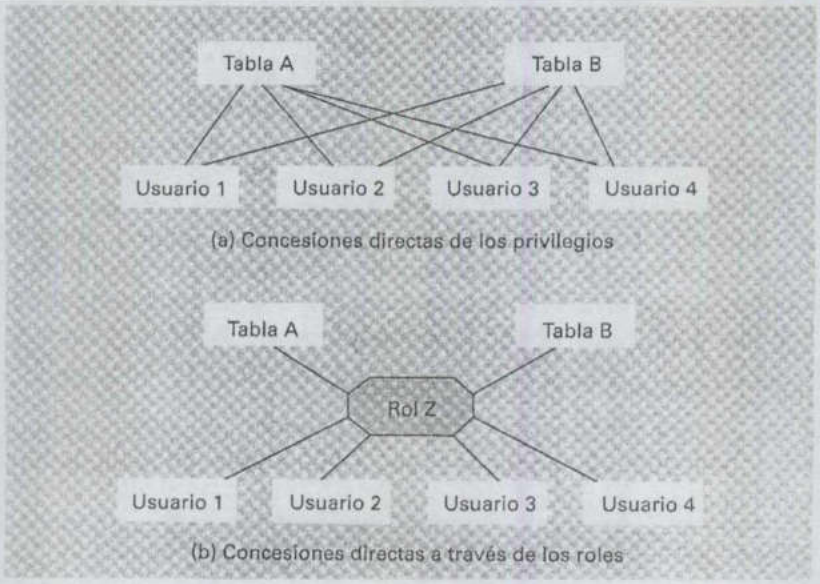


Fig. 1.17 Relaciòn entre los privilegios y los roles.

1.5.18 Enlaces de base de datos

Las bases de datos Oracle tienen la capacidad de hacer referencia a datos almacenados fuera de la base de datos local. Al hacer referencia a este tipo de datos, hay que especificar el nombre completo del objeto remoto. En el ejemplo de la sección dedicada a los sinónimos, sólo se han especificado dos partes del nombre completo: el propietario y el nombre de la tabla. ¿Qué ocurre si la tabla se encuentra en una base de datos remota? Para especificar una ruta de acceso a un objeto almacenado en una base de datos remota, hay que crear un *enlace de base de datos.* Estos enlaces pueden ser públicos (disponibles para todas las cuentas de la base de datos) o privados (creados por un usuario para el uso exclusivo de esa cuenta). Cuando se crea un enlace de base de datos, hay que especificar el nombre de la cuenta a la que hay que conectarse, la contraseña de la cuenta y el nombre de servicio asociado a la base de datos remota. En caso de no especificar el nombre de la cuenta a la que se va a conectar, Oracle intentará utilizar el nombre y la contraseña de la cuenta local para realizar la conexión con la base de datos remota. En el listado siguiente, se crea un enlace llamado MY_LINK:

```
create public database link MY_LINK
connect to HR identified by PUFFINSTUFF
using 'DB1';
```

En este ejemplo, el enlace especifica que, cuando se utilice, se iniciará una sesión en la base de datos identificada por el nombre de servicio DB1. Cuando se abra la sesión en la instancia DB1, se iniciará la sesión con la cuenta de usuario HR y la contraseña PUFFINSTUFF. Los nombres de servicio de las instancias se almacenan en archivos de configuración que utiliza Oracle Net. El archivo de configuración para los nombres de

servicio se denomina tnsnames.ora y, en este archivo, se especifica la máquina host, el puerto y la instancia asociada con cada nombre de servicio.

Para utilizar este enlace para acceder a una tabla, se tiene que especificar en la cláusula **from,** como en el ejemplo siguiente:

```
Select * from  EMPLOYEE@MY_LINK;
```

La consulta anterior accederá a la tabla EMPLOYEE mediante el enlace de base de datos MY_LINK. Además, puede crear un sinónimo para esta tabla, tal y como se indica en el siguiente comando SQL:

```
create synonym EMPLOYEE for EMPLOYEE@MY_LINK;
```

Observe que se ha definido el nombre completo del objeto de base de datos: su host y su instancia (mediante su nombre de servicio), su propietario (HR) y su nombre (EMPLOYEE).

La ubicación de la tabla EMPLOYEE es completamente transparente para el usuario final. Se puede mover la tabla EMPLOYEE a un esquema diferente o a otra base de datos; cuando se cambia la definición del enlace de base de datos, se redirige *el* sinónimo a la nueva ubicación.

Si un procedimiento almacenado, un paquete o un disparador contiene referencias a un enlace de base de datos, ese enlace debe existir para poder compilar correctamente el código PL/SQL.

1.5.19 Segmentos, extensiones y bloques

Los *segmentos* son la contrapartida física de los objetos de base de datos lógicos. Los segmentos almacenan datos. Los segmentos de índice, por ejemplo, almacenan los datos asociados con los índices. Para gestionar eficazmente los segmentos, es necesario que el DBA (administrador de bases de datos) sepa qué objetos utiliza una aplicación, cómo se introducen los datos en dichos objetos y de qué formas se recuperarán.

Dado que un segmento es una entidad física, debe estar asignado a un espacio de tablas de la base de datos (y, por tanto, debe encontrarse en uno de los archivos de datos de dicho espacio de tablas). Un segmento se compone de varias secciones llamadas *extensiones* (conjuntos contiguos de bloques Oracle). Cuando las extensiones de un segmento ya no pueden contener nuevos datos, el segmento tendrá que obtener otra extensión. El proceso de ampliación continuará mientras sea necesario, hasta agotar el espacio libre disponible en los archivos de datos del espacio de tablas o hasta alcanzar el número máximo de extensiones por segmento.

Cada segmento puede tener un límite que controle el número máximo de extensiones que puede adquirir. Aunque el límite teórico de extensiones en un segmento es miles de millones, la mayor parte de las operaciones de mantenimiento funcionan mejor si se limitan las extensiones por tabla a un número inferior a 4.000. Puede especificar el número máximo de extensiones por segmento para cada segmento de forma individualizada o puede utilizar la configuración predeterminada del espacio de tablas del segmento.

Cuando se elimina un segmento, se liberan las extensiones que éste utilizaba. Oracle puede reutilizar las extensiones liberadas para nuevos segmentos o para extensiones de segmentos existentes.

1.5.20 Segmentos para deshacer cambios y segmentos de anulación

Para mantener la coherencia de lectura entre varios usuarios de una base de datos y poder anular transacciones, Oracle ha de disponer de un mecanismo para reconstruir cualquier <<imagen anterior>> de los datos para las transacciones no confirmadas. Oracle utiliza *segmentos de anulación* contenidos en la base de datos para proporcionar una imagen anterior de los datos. En Oracle9i, se ofrece una opción adicional para almacenar una imagen anterior de los datos en un espacio de tablas para deshacer cambios gestionado por Oracle; se trata de los *segmentos para deshacer cambios.*

Las transacciones utilizan los segmentos de anulación o los segmentos para deshacer cambios a fin de registrar una imagen de los datos anterior a la introducción de los cambios. Por ejemplo, una operación **delete** importante en cuanto a tamaño requiere un segmento de anulación o de reconstrucción grande para guardar los registros que se van a eliminar. Si la transacción **delete** se anula, Oracle utilizará el segmento de anulación o el segmento para deshacer cambios con la finalidad de reconstruir los datos.

Las actualizaciones sólo guardan la imagen anterior de las columnas que se van a actualizar, no la fila completa.

Las consultas también utilizan los segmentos para deshacer cambios y los segmentos de anulación. Oracle realiza consultas de coherencia de lectura, de modo que la base de datos debe reconstruir los datos devolviéndolos al estado en que se encontraban cuando se inició una consulta. Si una transacción concluye después de que se inicie una consulta, Oracle continuará utilizando los elementos del segmento para deshacer cambios o del segmento de anulación de esa transacción para reconstruir las filas modificadas. Por regla general, debería evitar programar la ejecución simultánea de consultas de larga duración y de transacciones.

Los segmentos para deshacer cambios y de anulación aumentarán su tamaño hasta alcanzar el tamaño de las transacciones a las que dan soporte.

1.5.21 Vistas materializadas

Las *vistas materializadas* se pueden utilizar para proporcionar copias locales de datos remotos a los usuarios o para almacenar datos duplicados en la misma base de datos. Una vista materializada se basa en una consulta que puede utilizar un enlace de base de datos para seleccionar datos desde una base de datos remota. Los usuarios pueden hacer consultas entonces en la vista materializada o el optimizador podría redirigir dinámicamente las consultas para utilizar la vista materializada en lugar de la tabla de origen. Esta característica se denomina *reescritura de consultas* y se puede activar mediante un parámetro de inicialización. Las vistas materializadas se pueden implementar para que sean de sólo lectura o actualizables. Para mejorar el rendimiento, se puede indexar la tabla subyacente que utiliza la vista materializada.

Dependiendo de la complejidad de la consulta base de la vista materializada, podría utilizar un *registro de vistas materializadas* con el fin de mejorar el rendimiento de las operaciones de duplicación. Este tipo de operaciones se pueden llevar a cabo automáticamente, según el programa que se especifique para cada vista materializada.

1.5.22 Áreas de contexto

Dentro del área SQL compartida, existen tanto áreas públicas como privadas. Todas las instrucciones SQL que ejecuta un usuario necesitan un área SQL privada, que sigue existiendo hasta que se cierra el cursor correspondiente a la instrucción. En Oracle8, también se utiliza una memoria caché de objetos privada cuando se emplean características objeto-relacionales.

1.5.23 Área global de programa (PGA)

El *área global de programa (PGA)* es un área de memoria utilizada por un único proceso de usuario de Oracle. La memoria de la PGA no se comparte.

Con Oracle9i, Oracle ofrece un nuevo enfoque de la gestión de la memoria SQL: la característica de gestión de la memoria de ejecución SQL automática (Automated SQL Execution Memory Management). Con esta característica, el área de memoria SQL se puede optimizar sin necesidad de desconectarse de la base de datos y los ajustes se llevan a cabo automáticamente.

Para conseguir la optimización automática de la PGA, la memoria se divide en dos tipos: la memoria que se puede optimizar y la que no. La memoria que se puede optimizar es la que consume el área de trabajo SQL y la que no se puede optimizar es el resto. El administrador de la base de datos (DBA) debe activar esta característica de optimización automática, además de determinar y asignar la cantidad de memoria de la PGA de destino. La ecuación que utiliza Oracle para determinar cuánta memoria tiene disponible para la optimización es:

TAMAÑO_MEMORIA_NO_AJUSTABLE + TAMAÑO_MEMORIA_AJUSTABLE <=
TAMAÑO_TOTAL_DESEADO_DE LA_PGA

Cuando la característica de ajuste automático de la PGA está activada, el sistema sólo puede controlar la parte de la memoria que se puede ajustar; de modo que si esta parte es demasiado pequeña, Oracle nunca podrá aplicar la ecuación. En realidad, el área que se puede ajustar suele ser muy pequeña (<1 por 100 del PGA) para sistemas OLTP y la optimización SQL automática se presta mucho mejor a las cargas de trabajo de ayuda a la toma de decisiones (almacenes de datos) (>90 por 100 del PGA), por lo que los entornos de almacenes de datos son más apropiados para el uso de esta característica.

Para sacar provecho de esta opción, se deben desactivar dos parámetros: PGA_AGGREGATE_TARGET y WORKAREA_SIZE_POLICY. El parámetro PGA_AGGREGATE_TARGET se configura en el archivo de parámetros de inicialización, mientras que WORKAREA_ SIZE_POLICY es un parámetro dinámico de nivel del sistema y de sesión que se puede definir utilizando los comandos **alter system o alter**

session. El parámetro WORKAREA_SIZE_POLICY tiene uno de estos dos valores: MANUAL, donde el ajuste se lleva a cabo utilizando el parámetro *_AREA_SIZE (por ejemplo, SORT_AREA_SIZE) y AUTO, donde el ajuste se lleva a cabo dinámica o automáticamente.

El parámetro PGA_AGGREGATE_TARGET es un valor entre 10 MB y 400 GB que se puede definir en el archivo de parámetros de inicialización y modificar dinámicamente para alcanzar un rendimiento óptimo si resulta ser demasiado grande o demasiado pequeño. Se pueden utilizar bytes, kilobytes (K), megabytes (M) o gigabytes (G) para definir este valor. Si PGA_AGGREGATE_TARGET no está definido, el valor predeterminado de WORKAREA_SIZE_POLICY es MANUAL. En este caso, se utilizarían los parámetros *_AREA_SIZE. Cuando el ajuste automático está activado, los objetivos son asegurar que la memoria PGA global no exceda nunca el tamaño de PGA_AGGREGATE_TARGET. El área de memoria que se puede ajustar se regula de tal modo que nunca se agote la memoria para un proceso; regulando las áreas de trabajo, la tasa de transferencia y el tiempo de respuesta deberían quedar optimizados. Si utiliza el servidor compartido, una parte de la UGA (área global de usuario) debe almacenarse en la SGA. La arquitectura del servidor compartido permite que múltiples procesos de usuario utilicen el mismo proceso de servidor, reduciendo así los requisitos de memoria de la base de datos. Si se utiliza el servidor compartido, la información de sesión del usuario se almacena en la SGA, en lugar de almacenarse en la PGA. Si se utiliza el servidor compartido, se debe aumentar el tamaño del área SQL compartida en respuesta a los requisitos de memoria compartida. La UGA es parte del área de bloques de gran tamaño si está definida.

1.5.23 Capacidades de copia de seguridad y recuperación

La base de datos Oracle incorpora varias opciones de copia de seguridad y de recuperación. Las opciones disponibles se exponen en los apartados siguientes.

1.5.24.1 Exportar e Importar (Export/Import)

La utilidad *Export* (exportar) extrae datos de la base de datos y almacena el resultado en un archivo binario. Puede especificarse qué partes de la base de datos deben exportarse. Se puede exportar la base de datos entera, el esquema o esquemas de un usuario o de un conjunto de usuarios o un conjunto específico de tablas.

Las *exportaciones completas del sistema* leen las tablas completas del diccionario de datos, además de los datos de la aplicación. Las exportaciones completas del sistema pueden utilizarse para reconstruir completamente una base de datos, puesto que el diccionario de datos registra información sobre los usuarios, los archivos de datos y los objetos de base de datos.

La utilidad Export realiza una lectura lógica de la base de datos. Para leer la información contenida en el archivo de volcado binario creado por la operación de exportación, hay que emplear la utilidad *Import* (importar). Con Import se pueden elegir de manera selectiva los objetos o usuarios del archivo de volcado que se deseen importar. Después, Import tratará de insertar estos datos en la base de datos (en lugar de sobreescribir los registros existentes).

Las utilidades Export e Import forman parte de la mayoría de los planes de copias de seguridad y de recuperación, tanto para bases de datos pequeñas como para bases de datos de desarrollo. Puesto que la utilidad Export lee los datos de un determinado instante, sólo se pueden utilizar los datos exportados para restaurar los datos al estado en que se encontraban en ese instante. Por tanto, las exportaciones resultan de máxima eficacia para las copias de seguridad de datos que no sean demasiado volátiles. En los entornos en los que se llevan a cabo numerosas transacciones, las exportaciones rara vez constituyen el principal método de copias de seguridad; por lo general, se prefieren las copias de seguridad en línea.

1.5.24.2 Copias de seguridad fuera de línea

Además de las copias de seguridad lógicas de la base de datos, también pueden realizarse copias de seguridad físicas de sus archivos. Para ello, se pueden utilizar las *copias de seguridad en línea* y las *copias de seguridad fuera de línea.* Para realizar copias de seguridad fuera de línea hay que cerrar primero la base de datos; entonces puede realizarse una copia de seguridad de los archivos que componen la base de datos en un dispositivo de almacenamiento (copias de disco a disco o escrituras en cinta). Una vez completada la copia de seguridad, se puede volver a abrir la base de datos. Aunque las copias de seguridad fuera de línea no constituyan la opción más utilizada de recuperación y seguridad, conviene realizar una copia de seguridad de este tipo de la base de datos de manera periódica (como, por ejemplo, cuando la máquina host en que se encuentra tiene que someterse a las operaciones de mantenimiento rutinarias).

1.5.24.3 Copias de seguridad en línea

En las bases de datos que se ejecutan en modo ARCHIVELOG pueden realizarse copias de seguridad en línea. Este tipo de copias permiten realizar copias de seguridad físicas mientras la base de datos está abierta. Durante una copia de seguridad en línea, hay que poner los espacios de tablas temporalmente en un estado de copia de seguridad y devolverlos a su estado original una vez que los archivos se han copiado.

1.5.24.4 Recovery Manager (RMAN)

En Oracle9i se puede utilizar el administrador de recuperación, Recovery Manager (RMAN), para llevar a cabo copias físicas de la base de datos. En lugar de realizar una copia de seguridad de un archivo de datos entero, RMAN puede llevar a cabo copias de seguridad físicas increméntales de los archivos de datos. Durante una copia de seguridad completa *(nivel 0)* de un archivo de datos, se hace una copia de seguridad de todos los bloques que se hayan utilizado alguna vez en el archivo de datos. Durante una copia de seguridad acumulativa *(nivel 1)* de un archivo de datos, se copian todos los bloques utilizados desde la última copia de seguridad completa. Una copia de seguridad incremental *(nivel 2)* de un archivo de datos copia únicamente los bloques que se han modificado desde la copia de seguridad completa o acumulativa más reciente. Se pueden definir los niveles utilizados para las copias de seguridad increméntales.

El Recovery Manager realiza el seguimiento de las copias de seguridad bien a través de un catálogo de recuperación o, bien, colocando la información requerida en el archivo de control de la base de datos de la que se va a realizar una copia de seguridad. El número de días que abarquen los registros del RMAN almacenados en un archivo de control se define mediante el parámetro de inicialización CONTROL_FILE_ RECORD_KEEP_TIME.

La capacidad de llevar a cabo copias de seguridad increméntales y acumulativas de los archivos de datos puede mejorar notablemente el rendimiento de las copias de seguridad en el caso de bases de datos de muy gran tamaño con áreas aisladas de actividad de transacción.

1.5.25 Características de seguridad

Entre las carasterísticas relacionadas con la seguridad en Oracle se encuentran:

1.5.25.1 Seguridad de las cuentas

Las cuentas de las bases de datos pueden protegerse mediante una contraseña. También se pueden crear cuentas con capacidad de inicio de sesión automático, lo que permite a los usuarios que han accedido a una cuenta host acceder a la cuenta de la base de datos relacionada sin introducir una contraseña para dicha base de datos. El hecho de disponer de una cuenta o de privilegios en una base de datos no le otorga al usuario una cuenta o privilegios en ninguna otra base de datos.

1.5.25.2 Privilegios de nivel del sistema

Para ampliar el conjunto básico de roles de nivel del sistema, pueden crearse nuevos roles a partir de todo el conjunto de privilegios de nivel del sistema (tales como CREATE TABLE, CREATE INDEX y SELECT ANY TABLE). CONNECT, RESOURCE y DBA son roles estándar para los usuarios de aplicaciones, desarrolladores y administradores de bases de datos de una aplicación, respectivamente.

1.5.25.3 Seguridad de los objetos

Los usuarios que han creado objetos pueden conceder privilegios sobre ellos a otros usuarios por medio del comando **grant.** Asimismo, pueden conceder **(grant)** a un usuario acceso a tablas con la opción de concesión **(with grant option),** en cuyo caso ese usuario (el que posee el permiso) puede conceder acceso a las tablas a otros usuarios.

1.5.25.4 Auditoría

Las actividades de los usuarios que afectan a objetos de la base de datos pueden auditarse por medio del comando **audit.** Entre dichas actividades, se encuentran los accesos a tablas, los intentos de conexión a la base de datos y las actividades efectuadas con privilegio de DBA. Los resultados de estas auditorías se almacenan en una tabla de auditoría

dentro de la base de datos. Además de las capacidades de auditoría, se pueden crear disparadores de base de datos para registrar los cambios en los valores de los datos.

1.5.25.5 Auditoría de granularidad fina

Un defecto de la auditoría de objetos es que, a pesar de que uno puede ver que alguien a accedido al objeto y quién ha sido, no puede ver qué valores se han modificado ni dónde se encontraban los valores antiguos. Oracle9i ofrece un paquete PL/SQL para activar la auditoría de granularidad fina con el fin de contribuir en el seguimiento de la información a la que se ha accedido y de la forma en que se ha modificado dicha información.

1.5.25.6 Base de datos privada virtual

Con Oracle8i, versión 3, se introdujo la base de datos privada virtual (Virtual Private Database o VPD) con el fin de proporcionar control de acceso de granularidad fina junto con un contexto de aplicación seguro. Con VPD se pueden establecer directrices en forma de predicados (cláusulas **where)** que se añaden a cada consulta que los usuarios presentan a la base de datos. Una organización que utilice VPD sólo necesita crear una estructura de seguridad una vez en el servidor de datos. Dado que las directrices de seguridad se añaden a los datos en vez de a la aplicación, las reglas de seguridad se aplican siempre y sea cual sea la forma de acceso. De este modo, los datos presentados desde una consulta son idénticos independientemente del modo de conexión: desde una aplicación, SQL*Plus o un controlador ODBC. VPD se basa en varios mecanismos para asegurar la privacidad de los datos de todos los usuarios. Para lograr esta separación de los datos, asegúrese de que sus tablas están diseñadas para permitir la restricción del acceso a los datos de acuerdo con los valores de una o más columnas.

1.5.26 Oracle Enterprise Manager

A partir de Oracle7.3 se incluye el Oracle Enterprise Manager (OEM), una herramienta de interfaz gráfica de usuario (GUI), para permitir a los DBA administrar sus bases de datos desde una computadora personal. Con la versión Oracle9i, la herramienta OEM proporciona una interfaz mucho más robusta para la administración de bases de datos remotas. Con la versión 2 de OEM, incluida en Oracle8i, todos los DBA pueden utilizar el mismo repositorio central para llevar a cabo su trabajo. Además de estos cambios, OEM versión 2 incluye las características de asignación y planificación de tareas para permitir una cobertura de la base de datos de 24 horas. Con todas las versiones de OEM, hay que tomar varias decisiones clave antes de instalar y configurar la herramienta. Es necesario decidir dónde se ha de crear el repositorio de OEM y cómo y dónde se van a realizar las copias de seguridad para proteger este repositorio. Puesto que se puede utilizar OEM como una interfaz para Oracle Recovery Manager (RMAN), la información de recuperación se puede almacenar en el repositorio de OEM. Aunque nada impide que se cree el repositorio de OEM en una base de datos de producción, en caso de que se perdiera esta base de datos, ¿dónde se conseguiría la información de recuperación? Probablemente quiera crear otra

base de datos pequeña en la que almacenar el repositorio de OEM. Debería asegurarse de que se hace una copia de seguridad de ese repositorio con frecuencia para garantizar la recuperación del repositorio. Además, una vez que declare una base de datos como base de datos de repositorio, ésta dejará de estar disponible desde la herramienta OEM Server Manager, aunque se podrá iniciar y detener si se abre la consola OEM en modo de aplicación independiente.

En el caso de que sólo haya un DBA trabajando con el juego de herramientas OEM, no será necesario pensar quién se encargará de la administración de bases de datos específicas del entorno. Si, por el contrario, existen varios DBA en el sitio, habrá que definir las tareas, las responsabilidades de la base de datos y la planificación. Con OEM, se pueden conceder niveles de acceso y privilegio a cada DBA del grupo según las tareas que desempeñen. Se puede configurar OEM para que permita a un DBA enviar asignaciones de tareas y solicitudes por correo electrónico a otros DBA o tomar el control de un problema para agilizar su resolución.

Si tiene una versión anterior de OEM en el sistema, migre el repositorio a la versión más reciente con el fin de sacar provecho de las nuevas características. Si dispone de más de un repositorio en el sistema, tendrá que tener cuidado para asegurarse de que migra cada versión del repositorio sin dañar la información actualmente almacenada.

Con la versión de OEM incluida en Oracle9i se puede examinar el contenido de los archivos de registro de reconstrucción utilizando la característica de minería de registros.

Oracle utiliza el protocolo SNMP (Simple Network Management Protocol, protocolo simple de gestión de red). Dando soporte a SNMP, los productos Oracle se pueden integrar fácilmente en herramientas de supervisión para sistemas y redes. El hecho de que admita este protocolo permite a Oracle ser supervisado por herramientas existentes en las redes corporativas.

Entre las principales ventajas de SNMP se incluyen las siguientes:

- Integración fácil en las redes corporativas administradas utilizando herramientas SNMP.
- Supervisión central de todos los servicios necesarios para el acceso a la base de datos.
- Soporte para alertas automáticas en situaciones críticas.
- Soporte para reacciones automáticas ante condiciones de alerta.

A pesar de que no es obligatorio utilizar OEM para administrar una base de datos, esta herramienta ofrece una interfaz común para las bases de datos. Cuando una empresa crece (y el número de bases de datos y DBA crece con ella), la coherencia de la interfaz de DBA facilitará la implementación coherente de los procesos de control de producción y control de cambios.

CAPÍTULO 2
CONFIGURACIONES Y CONSIDERACIONES ACERCA DEL HARDWARE

1.1 Introducción a la arquitectura

Una base de datos Oracle consta de archivos físicos, áreas de memoria y procesos. La distribución de estos componentes varía en función de la arquitectura de base de datos que hayamos elegido.

Los datos de la base de datos se almacenan en archivos físicos (llamados *archivos de datos)* en un disco. A medida que se van utilizando, los datos se almacenan en memoria. Oracle utiliza las áreas de memoria para mejorar el rendimiento y gestionar la compartición de datos entre los usuarios. La principal área de memoria de una base de datos se denomina *área global del sistema (SGA).* Para leer o escribir los datos que se mueven entre la SGA y los archivos de datos, Oracle utiliza un conjunto de procesos en segundo plano que comparten todos los usuarios.

Un *servidor* de base de datos (también conocido como *instancia)* es un conjunto de estructuras de memoria y procesos en segundo plano que tienen acceso a un conjunto de archivos de base de datos. En la Figura 2.1 se ilustra la relación entre servidores y bases de datos.

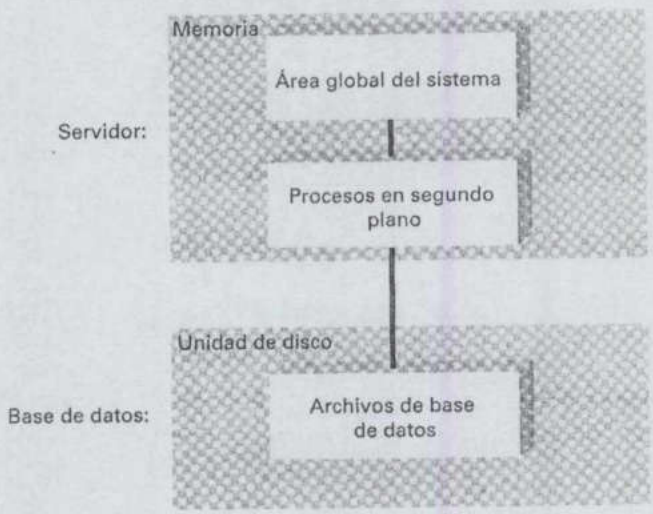


Fig. 2.1 Un servidor ùnico en una màquina host independiente.

Las características del servidor de base de datos, como son el tamaño de la SGA y el número de procesos en segundo plano, se especifican durante el inicio del servidor.

Estos parámetros se almacenan en un archivo denominado init.ora. El archivo init.ora de una base de datos suele incluir el nombre de la instancia como parte del nombre de archivo; una base de datos que se llame ORA1 tendrá, por lo general, un archivo init.ora denominado initoral.ora.

El archivo init.ora puede, a su vez, invocar a un archivo config.ora asociado. Por lo general, un archivo config.ora contiene sólo los valores de parámetros correspondientes a la información que nunca cambia, tales como, por ejemplo, el tamaño de los bloques de la base de datos o el nombre de la base de datos. Los archivos de inicialización sólo se leen durante el inicio; cualquier modificación que se introduzca en estos archivos no se aplicará hasta la siguiente vez que se inicie la base de datos. A partir de Oracle9i, se pueden utilizar los *archivos de parámetros de servidor* en lugar de los archivos de parámetros init.ora tradicionales. Un archivo de parámetros de servidor es un archivo binario que crea Oracle en el nivel del sistema operativo para almacenar las definiciones de los parámetros de la base de datos. El comando **create spfile** crea un archivo de parámetros de servidor basado en las definiciones de los parámetros actuales de la base de datos; el archivo se actualiza automáticamente cuando se utiliza el comando **alter system** para modificar la configuración de los parámetros. Cuando se inicia la base de datos, Oracle comprueba en primer lugar si existe un archivo de parámetros de sistema denominado *spfilenombre_instancia.ora;* a continuación, comprueba si existe un archivo de parámetros de sistema denominado spfile.ora en el directorio $ORACLE_HOME/dbs, en el caso de un sistema UNIX; si no existe ninguno de los dos, busca el archivo init.ora tradicional. Puede utilizar la cláusula **pfile** durante los comandos **startup** para forzar el uso de un archivo init.ora en lugar de un archivo de parámetros de sistema existente. Para crear un archivo init.ora a partir de un archivo de parámetros de sistema existente, se utiliza el comando **create pfile.**

2.2 Máquinas host independientes

La configuración conceptual más sencilla de una base de datos consta de un único servidor, que accede a una única base de datos en una máquina host independiente con un único disco. En esta configuración, mostrada en la Figura 2.1, todos los archivos se almacenan en el único dispositivo del servidor, y sólo hay un área global del sistema (SGA) y un conjunto de procesos en segundo plano de Oracle en el servidor.

La arquitectura que se muestra en la Figura 2.1 representa la configuración mínima. El resto de configuraciones de bases de datos son modificaciones de esta estructura básica.

Entre los archivos que se almacenan en el disco se incluyen los archivos de datos de la base de datos, los archivos de registro en línea, los archivos de control y el archivo de parámetros del host. Como se muestra en la Figura 2.1, existen dos interfaces principales en la base de datos:

- Entre los archivos de la base de datos y los procesos en segundo plano.
- Entre los procesos en segundo plano y el área global del sistema (SGA).

Las tareas de optimización consisten, fundamentalmente, en mejorar el rendimiento de estas interfaces. Si el área de memoria dedicada a la base de datos es lo bastante grande, se realizan menos lecturas repetitivas en los archivos de la base de datos. Dado que todos los archivos se almacenan en el único dispositivo de disco disponible en esta configuración, convendría minimizar el número de accesos a los archivos de datos.

2.2.1 Máquinas host independientes con matrices de discos

Cuando se dispone de varios discos, los archivos de la base de datos pueden separarse físicamente. Al separar los archivos, se mejora el rendimiento de la base de datos, ya que se reduce la contienda entre sus archivos. Durante el funcionamiento de la base de datos, lo habitual es que se necesite información de varios archivos para resolver una transacción o una consulta. Si los archivos no se distribuyen entre múltiples discos, el sistema tendría que leer varios archivos del mismo disco de forma concurrente. En la Figura 2.2 se ilustra la separación de archivos distribuyéndolos entre varios discos. Una base de datos utiliza varios tipos de archivos.

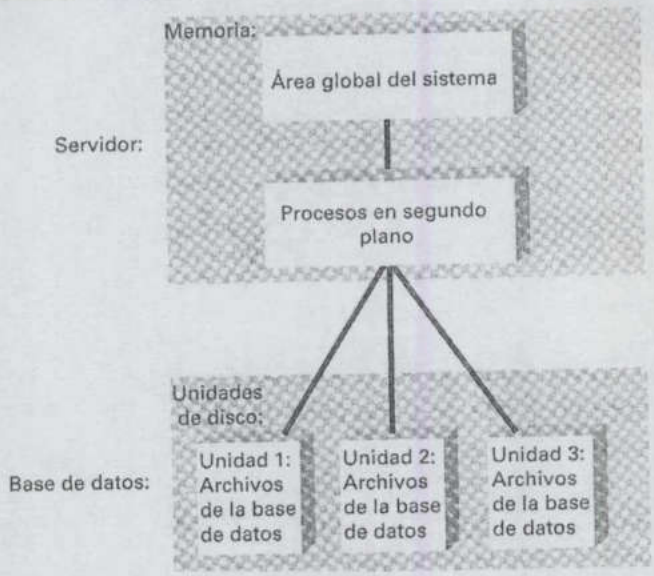


Fig. 2.2 Un servidor ùnico en una màquina host independiente con varios discos.

2.2.1.1 Duplicación en espejo de los archivos de control

El archivo de parámetros correspondiente al servidor que accede a la base de datos se almacena en los directorios del software de Oracle, normalmente en un directorio contenido en el directorio base del software Oracle. En la configuración predeterminada de directorios, el archivo init.ora se almacena en un directorio denominado (para UNIX) /orasw/app/oracle/admin/nombre_instancia/pfile. En el caso de los sistemas NT, el archivo init.ora se ubica en el directorio \admin\nombre_instancia\pfile dentro del directorio raíz del software Oracle. Si se utiliza un archivo init.ora en lugar de un archivo de parámetros de servidor y el nombre de la instancia es ORA1, el nombre del archivo init.ora será initoral.ora y se ubicará en /orasw/app/oracle/admin/ORAl/pfile. El archivo initora no incluye los nombres de los archivos de datos ni de los archivos de registro de reconstrucción en línea de la base de datos; dichos nombres se almacenan en el archivo de control y en el diccionario de datos. Sin embargo, el archivo init.ora contiene los nombres

de los archivos de control de la base de datos. En una máquina host con varios discos, conviene almacenar los archivos de control en discos distintos, que estén conectados a controladores diferentes. La base de datos los mantendrá sincronizados. Al almacenar archivos de control duplicados en espejo en varios discos, se reduce notablemente el riesgo de que se produzcan problemas en la base de datos debidos a fallos de los soportes físicos.

El listado siguiente muestra la entrada correspondiente al parámetro CONTROL_FILES en el archivo de parámetros de inicialización:

```
control_files    =    (/db01/oracle/ORA1/ctrl1ora1.ctl,
                       /db02/oracle/ORA1/ctrl2ora1.ctl,
                       /db03/oracle/ORA1/ctrl3ora1.ctl )
```

El listado anterior asigna nombres a los tres archivos de control. Si la entrada anterior se utiliza durante la creación de la base de datos, la base de datos creará automáticamente los tres archivos de control mostrados aquí. Si desea añadir nuevos archivos de control a una base de datos existente, siga este procedimiento:

1. Cierre la base de datos.

2. Copie uno de los archivos de control actuales en la nueva ubicación.

3. Modifique el archivo init.ora, añadiendo el nombre del nuevo archivo de control al elemento CONTROL_FILES.

4. Reinicie la base de datos.

El nuevo archivo de control se activará en ese momento.

Un único archivo init.ora puede invocar a múltiples archivos de inicialización a través del parámetro IFILE. También se pueden anidar las llamadas a los archivos de inicialización. Por ejemplo, init.ora podría tener una entrada IFILE para incluir un archivo denominado tuning.ora. El archivo tuning.ora, a su vez, podría contener una entrada IFILE para un archivo denominado disktuning.ora que contuviera parámetros de optimización relacionados con las operaciones de E/S del disco. Puede servirse de la característica de soporte de múltiples archivos de inicialización para agrupar parámetros relacionados; no obstante, debe tener cuidado para no sustituir la definición de un parámetro utilizándola en más de uno de los archivos incluidos.

2.2.1.2 Duplicación en espejo de los archivos de registro de reconstrucción

Tal como se ha indicado en la sección anterior, la base de datos sincroniza automáticamente los archivos de control, duplicando la información que contienen. La base de datos también puede duplicar en espejo los archivos de registro de reconstrucción en línea. Para ello, hay que utilizar *grupos de registros de reconstrucción.* Al utilizarlos, la base de datos duplica en espejo los archivos.

Cuando se duplican en espejo los registros de reconstrucción, el proceso en segundo plano LGWR (Log Writer o escritor de registros) escribe simultáneamente en todos los miembros del grupo de registros de reconstrucción en línea que esté activo en ese momento. Así, en lugar de ir utilizando de forma cíclica los archivos de registro de reconstrucción, va utilizando de forma cíclica los *grupos* de registros de reconstrucción. Dado que los miembros de un grupo suelen estar ubicados en unidades de disco diferentes, no existe ninguna contienda por los discos entre los archivos, por lo que el proceso LGWR apenas sufre cambios en su rendimiento.

Los grupos de registros de reconstrucción pueden crearse por medio del comando **create database,** aunque también pueden añadirse a la base de datos después de su creación, mediante el comando **alter database.** En el siguiente listado se muestra un ejemplo de adición de un grupo de registros de reconstrucción a una base de datos existente. El grupo en cuestión se denomina "GROUP 4". La utilización de números en los nombres de grupo facilita su administración, por lo que conviene numerarlos secuencialmente, empezando con el 1.

```
add logfile group  4
('/db01/oracle/CCl/log_lc.dbf'
 '/db02/oracle/CCl/log_2c.dbf')    size 5M;
```

Para añadir un nuevo archivo de registro de reconstrucción a un grupo existente, se utiliza el comando **alter database,** tal y como se muestra en el siguiente listado. El ejemplo siguiente añade un tercer miembro al grupo de registros de reconstrucción GROUP 4.

```
alter database
add logfile member /db03/oracle/CCl/log_3c.dbf
to group 4;
```

Cuando se utiliza la opción **add logfile member** del comando **alter database**, no se especifica ninguna información sobre el tamaño del archivo. Esto se debe a que todos los miembros del grupo han de tener el mismo tamaño. Dado que el grupo ya existe, la base de datos ya sabe qué tamaño ha de tener el nuevo archivo.

2.2.1.3 Duplicación en espejo de archivos de registro de reconstrucción archivados

Se puede indicar a la base de datos que escriba múltiples copias de cada uno de los archivos de registro de reconstrucción archivados (a partir de la versión Oracle 8) en el mismo momento en que se están archivando. En init.ora, el parámetro LOG_ ARCHIVE_DEST define la ubicación principal de almacenamiento para los archivos de registro de reconstrucción archivados. En Oracle8 se puede utilizar el parámetro LOG_ARCHIVE_DUPLEX_DEST para especificar una segunda ubicación para los registros de reconstrucción archivados. Cuando escriba los archivos de registro de reconstrucción archivados, Oracle escribirá en ambas ubicaciones. La escritura en la ubicación principal debe llevarse a cabo siempre; de lo contrario, la base de datos no se encontraría disponible cuando el proceso LGWR intentara escribir en el archivo de registro

de reconstrucción. La actividad de la base de datos continuará cuando esa escritura se lleve a cabo con éxito.

La escritura en la segunda área de destino para los registros archivados, especificada en LOG_ARCHIVE_DUPLEX_DEST, puede ser opcional. Si se define el valor 1 para LOG_ARCHIVE_MIN_SUCCEED_DEST, la escritura en una sola ubicación (la primera, especificada en LOG_ARCHIVE_DEST) se realizará con éxito. Si la escritura en la ubicación secundaria fallara, la disponibilidad de la base de datos no se interrumpiría.

Los parámetros de init.ora correspondientes a las áreas de destino para archivar registros han cambiado a partir de Oracle8i. En Oracle8i y versiones posteriores, el parámetro LOG_ARCHIVE_DEST queda obsoleto y se sustituye por LOG_ARCHIVE_DEST_n. Se pueden especificar hasta cinco áreas de destino para los registros archivados en Oracle8i y hasta diez áreas de destino en Oracle9i, sustituyendo n por el número de destino. Por ejemplo, se pueden especificar dos áreas diferentes de destino de los registros de reconstrucción archivados utilizando los parámetros LOG_ARCHIVE_DEST_1 y LOG_ARCHIVE_DEST_2, como se puede ver en el listado siguiente:

```
log_archive_dest_1  = '/db00/arch'
log_archive_dest_2  = '/db01/arch'
```

El parámetro LOG_ARCHIVE_MIN_SUCCEED_DEST de Oracle8 ha quedado obsoleto en Oracle8i y versiones posteriores. En lugar de ese parámetro, puede utilizar el parámetro LOG_ARCHIVE_DEST_STATE_n para activar o desactivar las áreas de destino de registros archivados. Por ejemplo, para desactivar la segunda área de destino de registros archivados, asigne a LOG_ARCHIVE_DEST_STATE_2 el valor DEFER (diferido). De manera predeterminada, los valores de estado de las áreas de destino están configurados en ENABLE.

Si no utiliza la funcionalidad interna de Oracle8 para duplicar en espejo los archivos de registro de reconstrucción archivados, debería duplicarlos en espejo utilizando los recursos del propio sistema operativo. Puede utilizar técnicas RAID como parte del método de duplicación en espejo por hardware. En general, debería inclinarse por la solución de duplicación en espejo por hardware del sistema operativo frente a la que ofrece Oracle, siempre y cuando el hardware de su sistema permita la duplicación en espejo sin sufrir un deterioro importante de su rendimiento. Las soluciones de duplicación en espejo que utilicen los métodos de Oracle o los métodos del sistema operativo sin apoyo hardware resultan más portables en las plataformas pero pueden hacer que el uso de la CPU se resienta.

Las soluciones de duplicación en espejo proporcionadas por el sistema operativo no le protegen frente a posibles problemas. Si se llevara a cabo una escritura equivocada en el archivo de registros, la duplicación en espejo proporcionada por el sistema operativo copiaría el error. Con los métodos de duplicación que ofrece Oracle, los archivos se escriben por separado, de modo que si fallara la escritura en un archivo de registro de reconstrucción, la base de datos continuaría operativa. Además, el proceso de archivado se lleva a cabo más rápidamente si se lee de un grupo de varios archivos de registro de reconstrucción mientras se están archivando.

2.2.2 Máquinas host independientes con duplicación de disco

Muchos sistemas operativos ofrecen la posibilidad de mantener duplicados sincronizados de los archivos mediante un proceso conocido como *duplicación de disco* o *duplicación de volumen.* Esta práctica también se denomina *duplicación en espejo.* El hecho de duplicar los discos presenta dos ventajas. En primer lugar, el conjunto de discos duplicados sirve como copia de seguridad en caso de que se produzca un fallo de disco. En la mayoría de los sistemas operativos, un fallo de disco hace que el disco correspondiente del juego de duplicados ocupe, automáticamente, el lugar del disco que ha fallado. La segunda ventaja consiste en un mejor rendimiento. La mayoría de los sistemas operativos que admiten la duplicación de volumen pueden redirigir las solicitudes de E/S de modo que se utilice el conjunto de archivos duplicados en lugar del principal. Mediante esta redirección, se reduce la carga de E/S en el conjunto de archivos principal y se mejora el rendimiento de las operaciones de E/S en los archivos. En la Figura 2.3 se ilustra el uso de la duplicación de discos.

La duplicación de disco a nivel del sistema operativo puede hacer que el rendimiento de la escritura se resienta si el sistema operativo no admite escrituras asincronas.

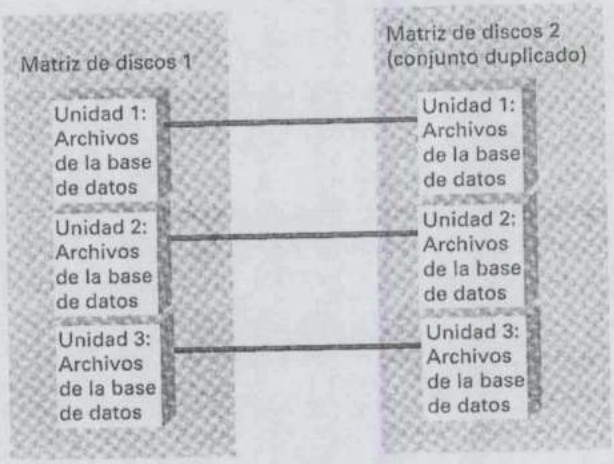


Fig. 2.3 Duplicaciòn de discos.

El tipo de duplicación que se muestra en la Figura 2.3 se denomina RAID-1 (Redundant Array of Independent Disks, matriz redundante de discos independientes). En este tipo de duplicación, cada uno de los discos del conjunto principal tiene su correspondiente pareja en el conjunto de duplicados. Dependiendo del sistema operativo, puede haber disponibles otras opciones de duplicación. En la duplicación RAID-3 y RAID-5, por ejemplo, cada conjunto de discos se trata como una sola unidad lógica y cada archivo se distribuye de forma transparente en pequeños fragmentos entre los distintos discos. En RAID-3 y RAID-5, un sistema de comprobación de paridad permite recuperar un miembro dañado o que haya fallado del conjunto de discos.

El método de duplicación que se utilice repercutirá en la distribución de los archivos entre los dispositivos. Por ejemplo, los archivos de datos que almacenan tablas y los

archivos de datos que almacenan los índices de dichas tablas suelen guardarse en discos distintos. Sin embargo, si se utiliza RAID-3 o RAID-5, no existe tal distinción de discos. Acceder a un archivo de datos cuando se usa una de estas opciones, casi siempre va a requerir acceder a todos los discos del conjunto. La magnitud de la posible contienda no debería ser grave. En RAID-5, por ejemplo, el primer bloque de datos se almacena en el primer disco de un conjunto, y el segundo bloque, en el siguiente disco. En esta configuración, la base de datos sólo tiene que leer un único bloque de un disco antes de pasar al siguiente disco. Cualquier contienda provocada por un gran número de accesos al mismo disco sólo debería durar el tiempo que se tarde en leer un único bloque.

El aspecto de optimización más importante relacionado con los entornos RAID es el rendimiento de la escritura. Durante una operación de escritura, un sistema RAID-5 debe leer el bloque original, leer el bloque de paridad, calcular la nueva paridad de datos y escribir el bloque modificado y el bloque de paridad en el disco. El gran volumen de actividad de disco asociada a las escrituras de un sistema RAID-5 hace que RAID-5 sea mucho más efectivo para las aplicaciones del tipo <<escribir una vez, leer muchas>> (por ejemplo, los almacenes de datos), que para las aplicaciones OLTP.

Las bases de datos muy grandes tienden a utilizar formas más avanzadas de RAID, como, por ejemplo, el sistema RAID-S utilizado para gestionar los dispositivos EMC. En estos entornos, los datos se distribuyen entre un gran número de dispositivos de disco, reduciendo así la posibilidad de contienda. La capacidad y la tecnología de distribución por fragmentos resuelven con frecuencia la mayor parte de los problemas de E/S que surgen. Incluso en entornos autogestionados, se recomienda el uso de un enfoque OFA (Optimal Flexible Architecture o arquitectura flexible óptima) para separar los objetos por entre los espacios de tablas con la finalidad de simplificar el mantenimiento.

2.2.3 Máquinas host independientes con múltiples bases de datos

En una misma máquina host se pueden crear varias bases de datos. Cada una de ellas dispondrá de un conjunto de archivos diferente y habrá que acceder a ellas desde un servidor distinto.

En la Figura 2.4 se muestra un único host que alberga dos bases de datos. Como cada servidor requiere un área global del sistema (SGA) y unos procesos en segundo plano, el host ha de ser capaz de cumplir los requisitos de memoria y de procesos que le impone esta configuración.

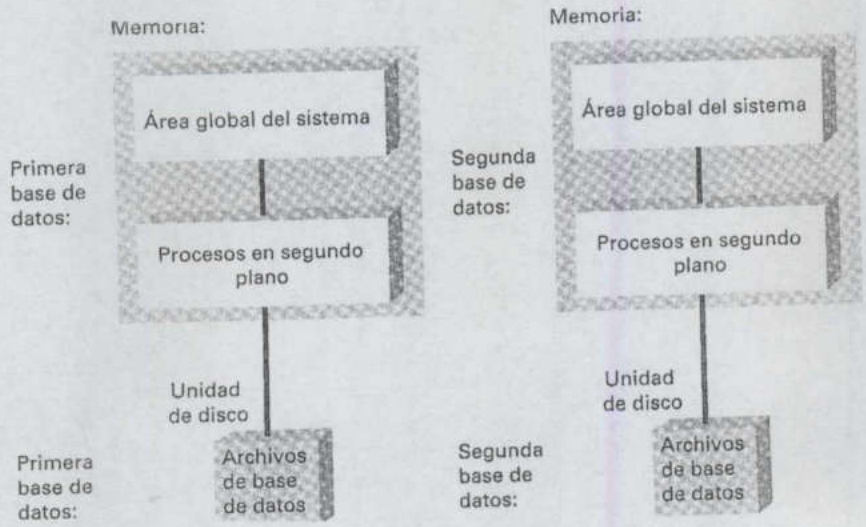


Fig. 2.4 Màquina host independiente con mùltiples bases de datos.

Como ya se ha indicado anteriormente, estas configuraciones son modificaciones de la arquitectura básica de una base de datos. En este caso, basta con crear una segunda base de datos que reproduzca la estructura de la primera. Observe que, aunque las dos bases de datos se encuentran en la misma máquina host, no se comunican entre sí (en este caso). Los procesos en segundo plano y los procesos de usuario de la primera base de datos no pueden acceder a los archivos de la segunda base de datos.

Normalmente, las diversas bases de datos que haya en un mismo servidor comparten los mismos directorios de código fuente de Oracle. Los archivos de parámetros de las dos bases de datos que se muestran en la Figura 2.4 se almacenan en directorios distintos, ya que el nombre de la instancia forma parte de la estructura del directorio. Por ejemplo, si los nombres de los dos servidores fueran ORA1 y ORA2, los archivos init.ora asociados se llamarían initoral.ora e initora2.ora, respectivamente. El primero se almacenaría en /orasw/app/oracle/admin/ORAl/pfile, y el segundo, en orasw/app/oracle/admin/ORA2/pfile. Los parámetros de sus servidores, al igual que sus archivos de datos, son totalmente independientes entre sí.

Aunque las bases de datos compartan los mismos directorios de código fuente, sus archivos de datos deberían guardarse en directorios distintos, y, a ser posible, en discos separados. Si múltiples bases de datos tienen archivos de datos almacenados en el mismo dispositivo, ninguna estadística de E/S de la base de datos reflejará exactamente la carga de E/S en dicho dispositivo. En su lugar, habrá que sumar las operaciones de E/S que atribuya cada base de datos a cada disco.

2.2.3.1 Simplificación del proceso de actualización

Si su sistema operativo puede dar soporte a varias versiones del software de Oracle, el proceso para la actualización de la versión de base de datos Oracle se simplificará en gran medida. Podrá actualizar el software de una versión incremental a la siguiente con un esfuerzo de administración mínimo. Oracle proporciona scripts para actualizar el diccionario de datos de una versión a otra o para volver al diccionario de datos de una versión anterior. Consulte el archivo README.doc (ubicado en el subdirectorio /rdbms/doc, dentro del directorio principal del software de Oracle) para obtener los nombres de los scripts de actualización del catálogo que debe utilizar.

2.3 Máquinas host en red

Cuando las máquinas host que albergan bases de datos Oracle se conectan a través de una red, dichas bases de datos pueden comunicarse por medio de Oracle Net (que antes se denominaba SQL*Net y Net8). Como se muestra en la Figura 2.5, los controladores de Oracle Net se apoyan en el protocolo de red local para lograr la conectividad entre dos servidores. A partir de ahí, Oracle Net permite la comunicación entre los dos servidores en el nivel de aplicación.

Las opciones de configuración de base de datos disponibles en un entorno de red dependen de la configuración y de las opciones de que disponga dicha red. En las secciones siguientes se describen los principales tipos de arquitecturas:

- Redes de bases de datos, utilizadas para consultas remotas.
- Bases de datos distribuidas, utilizadas para transacciones remotas.
- Real Application Clusters, en el que múltiples servidores acceden a la misma base de datos.
- Operaciones de consulta en paralelo, en las que varias CPU dan servicio a una única operación.
- Bases de datos cliente/servidor.
- Arquitecturas de tres niveles.
- Bases de datos accesibles desde la Web.
- Acceso a Oracle Enterprise Transparent Gateway.
- Bases de datos de reserva.
- Bases de datos duplicadas.

- Acceso a archivos externos.
- Acceso a tablas externas.

Fig. 2.5 Arquitectura de Oracle Net.

2.3.1 Redes de bases de datos

Oracle Net permite que las bases de datos Oracle se comuniquen con otras bases de datos que sean accesibles por medio de una red. Todos los servidores implicados deben estar ejecutando Oracle Net. En la Figura 2.6 se ilustra esta configuración.

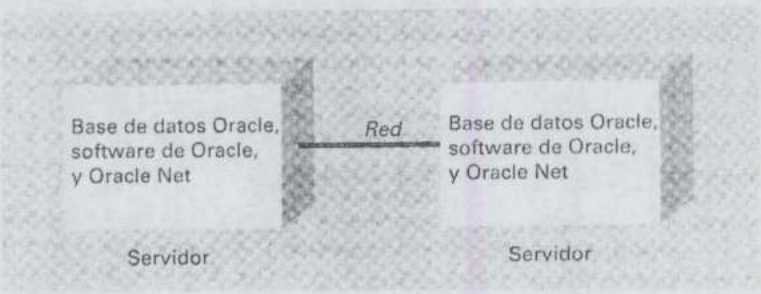


Fig. 2.6 Màquinas de host en red con bases de datos.

En esta figura se muestran dos máquinas host. Cada una de ellas puede dar soporte a una base de datos independiente, como se muestra en las Figuras 2.1 y 2.2. Cada máquina host de este ejemplo mantiene una copia del software de Oracle y una o más bases de datos Oracle.

Para que las bases de datos puedan comunicarse, sus servidores han de poder comunicarse entre sí. Como vemos en la Figura 2.6, las capas de comunicación de la base de datos se apoyan en el software y el hardware de red para establecer el enlace de

comunicaciones entre los servidores. Una vez que se crea dicho enlace, el software de la base de datos puede utilizarlo para transportar paquetes de datos entre bases de datos remotas.

El software de Oracle que se emplea para transferir datos entre bases de datos se denomina Oracle Net. En su configuración más sencilla, consta de un proceso que espera a que se produzcan conexiones a través de una ruta de conexión específica. Cuando detecta dichas conexiones, el proceso sigue las instrucciones que se pasan a través de la conexión y devuelve los datos solicitados.

Para que Oracle Net reciba y procese las comunicaciones, el host ha de ejecutar un proceso llamado *proceso escucha (listener).* Este proceso ha de estar ejecutándose en todos los host que estén implicados en las comunicaciones entre bases de datos. Cada servidor ha de estar configurado para asociar este proceso a un puerto de comunicaciones específico.

En las siguientes secciones se muestran ejemplos de conexiones entre bases de datos. Las aplicaciones de base de datos en red incluyen las consultas y las transacciones con bases de datos remotas.

2.3.1.1 Consultas remotas

Las consultas a bases de datos Oracle remotas utilizan *enlaces de base de datos* para identificar la ruta que debe seguir la consulta para localizar los datos. Un enlace de base de datos especifica, ya sea directa o indirectamente, el host, la base de datos y la cuenta que debe utilizarse para acceder a un objeto específico. El enlace de la base de datos identifica el host y la base de datos a la que se va a acceder haciendo referencia al *nombre de servicio* que se tenga que utilizar. Cuando una instrucción SQL hace referencia a un enlace de base de datos, abre una sesión en la base de datos especificada y ejecuta una instrucción SQL en ella. A continuación, se devuelven los datos y la sesión remota puede permanecer abierta (permanece abierta de manera predeterminada) por si volviera a ser necesaria. Los enlaces de base de datos se pueden crear como enlaces públicos, (si los DBA hacen que el enlace esté disponible para todos los usuarios de la base de datos local), o como enlaces privados, disponibles únicamente para el usuario que haya creado el enlace de base de datos.

En el ejemplo siguiente, se crea un enlace de base de datos público denominado HR_LINK:

```
create public database link HR_LINK
connect to HR identified by PUFFINSTUFF
using 'hg';
```

El comando **create database link,** como se puede ver en este ejemplo, tiene varios parámetros:

- La palabra clave opcional **public,** que permite que los administradores de bases de datos creen enlaces para todos los usuarios de la base de datos.

- El nombre del enlace (en este caso, HR_LINK).

- La cuenta a la que conectarse (si no se especifica ninguna, en la base de datos remota se utilizarán el nombre de usuario y la contraseña locales).

- El nombre de servicio ('hq'), que se define en el archivo tnsnames.ora.

Para utilizar este enlace, basta con añadirlo como un sufijo a los nombres de las tablas en los comandos. En el ejemplo siguiente, se consulta una tabla remota mediante el enlace de base de datos HR_LINK:

```
select * from EMPLOYEE@HR_LINK
where Office='ANNAPOLIS';
```

Los enlaces de base de datos permiten que las consultas accedan a bases de datos remotas y también permiten que la información relativa a la ubicación física de los datos (el host, la base de datos y el esquema) sea transparente para el usuario. Por ejemplo, si un usuario de una base de datos local crea una vista basada en un enlace de base de datos, cualquier acceso a la vista local consultará automáticamente la base de datos remota. El usuario que realice la consulta no tendrá que saber dónde residen los datos.

Esto se ilustra en el siguiente listado. En este ejemplo, se crea una vista utilizando el enlace de base de datos HR_LINK que se ha definido anteriormente. Luego, se pueden conceder privilegios de acceso a esta vista para los usuarios de la base de datos local, tal y como se muestra a continuación:

```
create view LOCAL_EMP
as select * from EMPLOYEE@HR_LINK where
Office='ANNAPOLIS';

grant select on LOCAL_EMP to PUBLIC;
```

Cuando un usuario consulta la vista LOCAL_EMP, Oracle utiliza el enlace HR_LINK para abrir una sesión usando la información de conexión especificada para el enlace de base de datos HR_LINK. La consulta se ejecutará sobre los datos de la base de datos remota y el resultado se devolverá al usuario local. El usuario local de LOCAL_EMP no sabrá que los datos proceden de una base de datos remota.

2.3.2 Actualizaciones remotas: la opción de duplicación avanzada

Además de poder consultar datos almacenados en bases de datos remotas, puede actualizar **(update)** bases de datos ubicadas en máquinas host remotas. Las actualizaciones de estas bases de datos pueden combinarse con actualizaciones de la base de datos local en una misma unidad lógica de trabajo: se confirman **(commit)** todas o se anulan todas.

En la Figura 2.7 se muestra un ejemplo de un conjunto de transacciones. Una de las transacciones se realiza en una base de datos remota y otra transacción en el host local. En este ejemplo, se actualiza **(update)** como parte de la misma transacción una tabla local denominada EMPLOYEE y una tabla remota también llamada EMPLOYEE, de una base

de datos definida por el enlace HR_LINK. Si fallara alguna de las actualizaciones, se anularían ambas transacciones. La coordinación de las transacciones remotas se logra gracias a la implementación de Oracle de la característica de confirmación en dos fases (Two-Phase Commit, 2PC).

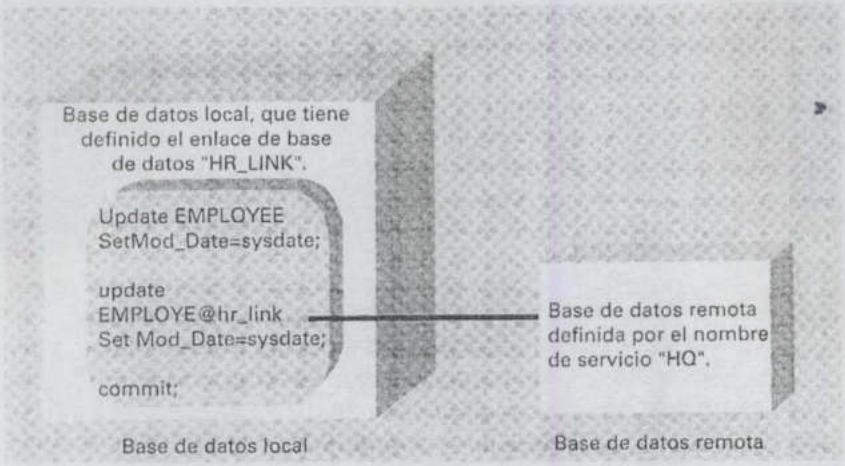


Fig. 2.7 Ejemplo de transacciòn distribuida.

Todas las máquinas host implicadas en la operación de duplicación o en las transacciones distribuidas deben ejecutar Oracle Net, estar configuradas para dar soporte a la duplicación de Oracle y para permitir las comunicaciones de host a host. Una vez que se ha configurado Oracle Net para ambos hosts, hay que configurar correctamente cualquier archivo asociado a él. Estos archivos de configuración permiten a la base de datos que interprete los nombres de servicio mostrados en el comando **create database link** (crear enlace de base de datos) mencionado más arriba.

Cada host que esté ejecutando Oracle Net debe mantener un archivo denominado tnsnames.ora. Este archivo define los descriptores de conexión para los nombres de servicio a los que se puede acceder desde dicho host. Por ejemplo, en el siguiente listado se muestra el fragmento del archivo tnsnames.ora para el nombre de servicio <<HQ>> utilizado en el ejemplo del enlace HR_LINK:

```
HQ  =(DESCRIPTION=
      (ADDRESS=
          (PROTOCOL=TCP)
          (HOST=HQ)
          (PORT=1521))
      (CONNECT_DATA=
          (SID=loc)))
```

Oracle8i y versiones posteriores utilizan SERVICE_NAME en lugar del SID en el archivo tnsnames.ora:

```
S817.us.oracle.com =
 (DESCRIPTION =
      (ADDRESS_LIST =
           (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = sdg-lap)
                (PORT = 1521))
       )
       (CONNECT_DATA =
           (SERVICE_NAME = S817.us.oracle.com)
       )
  (
```

En el listado anterior se muestran varios aspectos del proceso de conexión (los parámetros son específicos de TCP/IP, pero las necesidades de conexión básicas son las mismas para todas las plataformas). En primer lugar, está la información de direccionamiento hardware (el protocolo, el nombre del host y el puerto de comunicaciones que se van a utilizar). La segunda sección define el nombre de la instancia, en este caso,<<loc>>. Dado que el archivo tnsnames.ora le dice a la base de datos todo lo que necesita saber para conectarse a bases de datos remotas, es importante que el contenido de este archivo sea coherente entre las distintas máquinas host.

La unidad de trabajo lógica de las transacciones distribuidas se procesa mediante la implementación de Oracle de la confirmación en dos fases (2PC). Si se produce un fallo de red o de servidor que impida que se complete satisfactoriamente la unidad de trabajo, es posible que los datos de las bases de datos afectadas por las transacciones queden desincronizados. Existe un proceso en segundo plano, RECO, que busca automáticamente las transacciones incompletas y las resuelve en cuanto dispone de todos los recursos necesarios para ello.

El número máximo de transacciones distribuidas concurrentes para una base de datos se define mediante el parámetro DISTRIBUTED_TRANSACTIONS del archivo init.ora. Si se asigna a este parámetro el valor 0, no se podrán realizar transacciones distribuidas y el proceso en segundo plano encargado de la recuperación (RECO) no será activado cuando se inicie la instancia.

Existen dos vistas del diccionario de datos que resultan prácticas a la hora de diagnosticar las transacciones distribuidas incompletas. La vista DBA_2PC_NEIGHBORS contiene información sobre las conexiones entrantes y salientes para transacciones pendientes. DBA_2PC_PENDING contiene información sobre transacciones distribuidas que aguardan una operación de recuperación. Si la transacción distribuida se encuentra con un error, busque más información en DBA_2PC_NEIGHBORS y DBA _2PC _PENDING.

2.3.3 Real Application Clusters

Las configuraciones que se han analizado hasta este punto del capítulo utilizaban bases de datos a las que accedía un único servidor. Sin embargo, dependiendo de la configuración del hardware, es posible utilizar varios servidores para acceder a una misma

base de datos. Esta configuración, llamada anteriormente *servidor paralelo de Oracle (OPS, Oracle Parallel Server) y* conocida ahora como *Real Application Clusters* (RAC), se ilustra en la Figura 2.8.

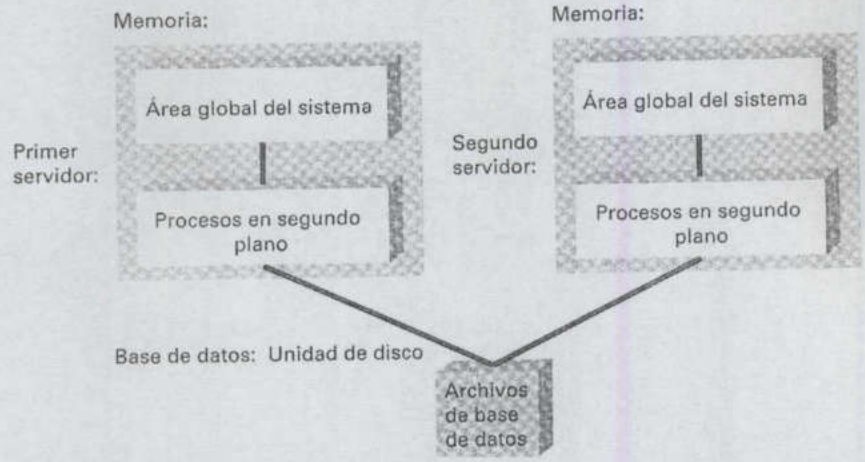


Fig. 2.8 Real Application Clusters (RAC).

Tal y como muestra la Figura 2.8 dos servidores distintos comparten el mismo conjunto de archivos de datos. Normalmente, estos servidores se encuentran en distintas máquinas host de un cluster hardware. Un cluster es normalmente un conjunto de máquinas host individuales que se han conectado con un cable de fibra óptica de baja latencia y gran ancho de banda, a través del cual las máquinas host se pasan mensajes entre sí, trabajando como una sola entidad. Esta configuración ofrece las siguientes ventajas:

- Se dispone de más recursos de memoria, puesto que se utilizan dos máquinas.
- Si una de las máquinas host dejase de funcionar, la otra podría acceder a los archivos de datos, lo que permitiría la recuperación de los datos frente a fallos catastróficos.
- Los usuarios pueden clasificarse según el tipo de procesamiento que realicen, y mantenerse a aquellos usuarios que hagan un uso intensivo del procesador en un host independiente del que lleva a cabo las transacciones de procesamiento en línea normales.

Al configurar un conjunto de servidores para que utilicen la tecnología RAC, hay que especificar una serie de estructuras y parámetros específicos de RAC. En primer lugar, para activar la tecnología RAC, se debe asignar al parámetro CLUSTER_DATABASE (anteriormente, PARALLEL_SERVER) el valor TRUE (verdadero). También hay que asignar un valor a CLUSTER_DATABASE_INSTANCES (parámetro que se denominaba anteriormente PARALLEL_SERVER_INSTANCES) en el archivo de parámetros init.ora.

Asigne a CLUSTER_DATABASE_INSTANCES un valor que se corresponda con el número más alto de bases de datos que desee iniciar al mismo tiempo para una base de datos Real Application Cluster; siempre puede asignar el valor 10 y, luego, iniciar sólo cinco bases de datos.

La base de datos central debe configurarse para gestionar distintos servidores, con un conjunto de segmentos de anulación que cada instancia de base de datos puede utilizar. Para gestionar estos segmentos de anulación de la mejor manera posible, cree un espacio de tablas de segmentos de anulación distinto para cada servidor, utilizando el nombre de la instancia como parte del nombre del espacio de tablas. Por ejemplo, si los nombres de los servidores son ORA1 y ORA2, los espacios de tablas de segmentos de anulación deberían ser RBS_ORA1 y RBS_ORA2. Sólo puede haber activo un espacio de tablas para deshacer cambios.

2.3.4 Procesadores múltiples: las opciones de consulta en paralelo y de carga en paralelo

Para realizar transacciones y consultas uno puede servirse de varios procesadores. El trabajo necesario para resolver una sola solicitud de una base de datos puede ser realizado por varios procesadores coordinados. La distribución de la carga de trabajo entre varios procesadores contribuye a mejorar el rendimiento de las transacciones y de las consultas. La arquitectura de la opción de consulta en paralelo (Parallel Query Option, PQO) de Oracle permite que, prácticamente la totalidad de las operaciones, se procesen en paralelo. Entre las operaciones que pueden realizarse con esta opción se incluyen **create table as select, create index,** exploraciones completas de tabla, exploraciones de índices, operaciones de ordenación, de inserción **(insert),** de actualización **(update),** de borrado **(delete)** y la mayoría de las consultas.

Las opciones de actualización en paralelo y de borrado en paralelo sólo se encuentran disponibles cuando la opción de partición está instalada.

La medida en la que la base de datos utiliza el paralelismo depende de los parámetros **degree e instances** de la cláusula **parallel** que se utilice con estos. El parámetro **degree** especifica el grado de paralelismo (el número de servidores de consulta utilizados) para una determinada operación. El parámetro **instances** especifica cómo debe dividirse la tabla entre las instancias de una instalación RAC. Podemos especificar las reglas de paralelismo de una instancia (como, por ejemplo, el número mínimo de servidores de consulta) en el archivo init.ora de la instancia. El número máximo de procesos concurrentes de servidor para consultas en paralelo se fija mediante el parámetro PARALLEL_MAX_SERVERS en el archivo init.ora, mientras que el número mínimo se define mediante el parámetro PARALLEL_MIN_SERVERS. El número de discos en los que se almacenan los datos de una tabla y el número de procesadores disponibles en el servidor se utilizan para generar los parámetros predeterminados de paralelización de una consulta.

Puede utilizar el parámetro PARALLEL_AUTOMATIC_TUNING de init.ora para configurar automáticamente muchos de los parámetros que pueden definirse en init.ora relacionados con las consultas en paralelo. Uno de los parámetros que se configuran automáticamente, PARALLEL_ADAPTIVE_MULTI_USER, reduce el paralelismo de las operaciones si hay varios usuarios activos procesando operaciones en paralelo en la base de

datos. Dado que hay un límite para el número de procesos disponibles de servidor de consultas en paralelo, reducir el paralelismo para una operación evita que utilice todos los recursos disponibles. También puede utilizar el administrador de recursos de la base de datos (Database Resource Manager) para limitar el paralelismo. Además de procesar las consultas en paralelo, utilizando el cargador SQL*Loader Direct Path, se pueden procesar en paralelo las cargas de datos

2.3.5 Aplicaciones de base de datos cliente-servidor

En una configuración de tipo host a host, como ya hemos apuntado en este capítulo, existe una base de datos Oracle en cada host, y las bases de datos se comunican por medio de Oracle Net. No obstante, un host que no albergue una base de datos puede acceder a una base de datos remota; esto se logra haciendo que los programas de la aplicación de un host accedan a una base de datos ubicada en otro host. En esta configuración, el host que ejecuta la aplicación se denomina *cliente* y el otro host, *servidor*. En la Figura 2.9 se ilustra esta configuración.

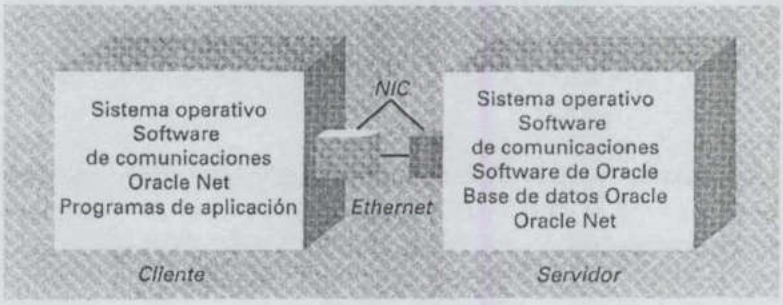


Fig. 2.9 Configuración cliente-servidor.

Tal y como vemos en la Figura 2.9 el cliente ha de tener la capacidad de comunicarse con el servidor a través de la red. Como los programas de la aplicación se ejecutan en el lado del cliente, la base de datos se utiliza, principalmente, para operaciones de E/S. Los costes de procesamiento derivados de la ejecución de los programas de aplicación pasan entonces al PC cliente, en lugar de quedar en el servidor. Para que esta configuración funcione, el cliente ha de estar ejecutando Oracle Net. Cuando el programa de aplicación del cliente le solicita al usuario información sobre la conexión de base de datos, éste debe especificar el nombre de servicio. Entonces, la aplicación abre una sesión en la base de datos remota.

La utilización de una configuración cliente-servidor ayuda a reducir la cantidad de trabajo que realiza el servidor. No obstante, el hecho de convertir la configuración de una aplicación en una configuración cliente-servidor no significa una mejora automática en el rendimiento del sistema, por dos razones:

- Tal vez el consumo de recursos de CPU nunca haya sido el problema. Normalmente, estos recursos se utilizan con frecuencia durante el día y con poca frecuencia después de la jornada de trabajo. Puede ser conveniente modificar la planificación del uso del procesador, ejecutando procesos por lotes o programas de gran tamaño después de la jornada laboral en lugar de introducir una configuración cliente-servidor como solución para un sistema con problemas de consumo de recursos de CPU.

- La aplicación puede no haber sido rediseñada. Para diseñar en un entorno cliente-servidor hay que tener en cuenta el volumen de datos que se envía a través de la red en cada acceso a una base de datos. En las aplicaciones basadas en servidor, esto no supone ningún problema. En las aplicaciones cliente-servidor, hay que tener en cuenta el tráfico en la red durante la planificación y la optimización.

Existen muchas formas distintas de implementar una configuración cliente-servidor, en función del hardware que se utilice. La implementación que se muestra en la Figura 2.9 es bastante habitual; se trata de la implementación que utilizaría una herramienta de consultas *ad hoc* que se ejecutase en un PC para acceder a una base de datos Oracle que se ejecute en un servidor.

2.3.6 Arquitectura de tres niveles

Una arquitectura de tres niveles es una extensión del modelo cliente-servidor. La función que desempeña cada nivel depende de la implementación, pero los niveles son normalmente los siguientes:

- Un cliente, utilizado para la presentación de la aplicación.

- Un servidor de aplicaciones, utilizado para el procesamiento de la lógica de negocio de la aplicación.

- El servidor de base de datos, utilizado para almacenar y recuperar los datos.

En una arquitectura de tres niveles los requisitos de procesamiento de la aplicación se trasladan del nivel del cliente al nivel del servidor de aplicaciones. Normalmente, el servidor de aplicaciones es más potente que el cliente, de modo que el rendimiento de la aplicación se puede beneficiar. El rendimiento también se puede mejorar reduciendo la cantidad de tráfico entre el servidor de base de datos y el cliente. Se han de enviar menos datos al cliente, puesto que una mayor parte de la interacción se realiza entre el servidor de aplicaciones y el servidor de base de datos.

Las ventajas a largo plazo de una configuración de tres niveles se deben a la simplificación del mantenimiento del cliente. Puesto que la aplicación reside en el servidor de aplicaciones, la mayor parte de las actualizaciones se centrarán en la actualización del servidor de aplicaciones. Si no posee un amplio control de la configuración en las máquinas

cliente, una arquitectura de tres niveles puede reducir los costes de mantenimiento de las aplicaciones.

No obstante, migrar una arquitectura cliente-servidor a una arquitectura de tres niveles no es siempre un proceso sencillo, fácil o ventajoso. Si tiene una red bien configurada, máquinas cliente controladas y una aplicación bien optimizada, cambiar a una arquitectura de tres niveles podría no traer ninguna ventaja y, en cambio, acarrear posibles gastos. Por ejemplo, tendrá que rediseñar su aplicación para reducir el tráfico de red entre los clientes y el servidor de aplicaciones; de lo contrario, el rendimiento de la aplicación se vería afectado. Además, el proceso de actualización de la aplicación cambia; en lugar de actualizar los clientes, tendrá que actualizar un servidor de aplicaciones utilizado por muchos clientes. Dado que el uso del servidor de aplicaciones está centralizado, encontrará más dificultades a la hora de probar los cambios realizados en las aplicaciones sin que ello afecte a todos los usuarios de las aplicaciones. Sin embargo, para introducir modificaciones en una aplicación, sólo necesitará modificar la aplicación en un servidor.

En la Figura 2.10 se muestra una configuración de tres niveles genérica para una aplicación de Oracle. El servidor de aplicaciones y el servidor de base de datos se comunican a través de Oracle Net. El protocolo utilizado para la comunicación entre los clientes y el servidor de aplicaciones depende del entorno en que tengan lugar; se puede utilizar Oracle Net o un protocolo de Internet como HTTP.

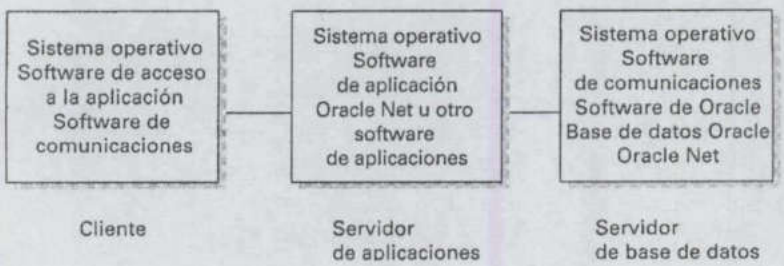


Fig 2.10

En cualquier configuración la aplicación debería evitar el tráfico innecesario entre el cliente y la base de datos. En una aplicación de tres niveles, debería preocuparse especialmente por el tráfico registrado entre la base de datos y el cliente, ya que hay más componentes entre ellos. Los componentes adicionales entre el cliente y la base de datos podrían afectar negativamente al rendimiento de cada consulta de base de datos que se realice. En muchas aplicaciones de tres niveles, la validación de datos en el lado del cliente es escasa, con lo que se reduce el número de búsquedas en la base de datos necesarias durante el procesamiento de la transacción. La falta de consultas de validación de datos puede compensarse dentro de la aplicación con el uso de cuadros de lista desplegable de valores o casillas de verificación de selección múltiples.

2.3.6 Bases de datos accesibles desde la Web

Se puede configurar una base de datos para que se pueda acceder a ella desde la World Wide Web. Los clientes externos se comunican con el software de aplicación (nivel intermedio) y luego, a través del encaminador cortafuegos, que a su vez pasa solicitudes de datos al servidor de base de datos. Las bases de datos accesibles desde la Web, en lo que a la arquitectura se refiere, se basan en un modelo de tres niveles mostrado en la Figura 2.10. Por ejemplo, el nivel correspondiente al cliente debe ser un PC, donde se ejecuta un explorador web. El explorador se comunica con el software de aplicación del nivel intermedio mediante el protocolo HTTP. El nivel intermedio se comunica con el encaminador cortafuegos, el cual pasa las solicitudes al servidor de base de datos mediante Oracle Net o mediante el protocolo IIOP (Internet Inter-ORB Protocol) si se utiliza Java en la base de datos. La arquitectura ideal para acceder desde la Web con seguridad consiste en ubicar el software de la aplicación web (nivel intermedio) detrás de un cortafuegos y el servidor de base de datos detrás de un segundo cortafuegos. Las configuraciones web utilizan generalmente Oracle9iAS, el servidor de aplicaciones de Oracle de nivel intermedio.

2.3.7 Acceso a Oracle Enterprise Gateway

Se puede crear un enlace de base de datos a un servicio que no sea una base de datos Oracle. De este modo, se podrán consultar los datos a través del enlace de base de datos y los datos se consultarán como si los datos de origen fueran una base de datos Oracle.

Para acceder a unos datos que no sean Oracle, es necesario utilizar el producto Oracle Enterprise Gateway (conocido anteriormente como Oracle Transparent Gateway). Se necesita una pasarela distinta para cada tipo de motor de base de datos al que se vaya a acceder. La pasarela se ejecuta en el host de origen de los datos a los que se vaya a acceder. Por ejemplo, si los datos de origen se almacenan en una base de datos AS/400, el software Oracle Enterprise Gateway para AS/400 se instalará en el servidor AS/400. Cuando se ejecute, el software de pasarela creará un proceso escucha en el servidor de origen que actuará como un proceso escucha de Oracle Net. A continuación, se podrá acceder a determinados objetos de datos de la base de datos AS/400, siempre y cuando se disponga del nombre de usuario y de la contraseña necesarios para esa base de datos.

Oracle Enterprise Gateway es una extensión de la configuración servidor-servidor mostrada en la Figura 2.6; la diferencia es que la base de datos y el software de la base de datos de uno de los host no son Oracle.

2.3.8 Bases de datos de reserva

Puede configurar una segunda base de datos como una copia de reserva de una base de datos primaria. Una base de datos de reserva es un caso especial de una configuración de servidor-servidor. Cada servidor posee una copia completa del software Oracle y las estructuras de archivos de la base de datos suelen ser idénticas (de lo contrario, tendrá que crear un archivo de control independiente para la base de datos de reserva). Tendrá que

crear siempre un archivo de control independiente para la base de datos de reserva utilizando el comando **alter database create standby controlfile as *'nombrearchivo'*** y, a continuación, copiarlo en la base de datos de reserva. Los dos host deberían utilizar la misma versión de sistema operativo y de software de base de datos.

En el caso de que tenga lugar un desastre en la base de datos de producción, puede abrir la base de datos de reserva con una pérdida de datos mínima, (generalmente se pierde el contenido de los archivos de registro de reconstrucción en línea). Ya que la instancia de producción genera también archivos de registro de reconstrucción archivados, se deberán copiar en el sistema de reserva bien manualmente o bien utilizando el proceso ARCH. La base de datos de reserva permanece en modo de recuperación hasta que sea necesaria, aunque, a partir de Oracle8i, se puede abrir en modo de sólo lectura y, a continuación, volver a ponerla en modo de recuperación. Cuando la base de datos de reserva se abre en modo de escritura y lectura, se convierte en la base de datos principal y ya no se puede reconfigurar fácilmente como base de datos de reserva. En Oracle9i, un DBA puede utilizar la metodología de recuperación de errores para pasar de la base de datos primaria a la de reserva y, después, cuando sea necesario, volver fácilmente a la primaria sin necesidad de volver a crear ninguna de las dos bases de datos.

Para automatizar la transferencia de archivos de registro de reconstrucción archivados a la base de datos de reserva, se utilizan los parámetros LOG_ARCHIVE_DEST_n. En lugar de un nombre de directorio, especifique un nombre de servicio de los que están definidos en el archivo tnsnames.ora local como el destino que se debe utilizar para la copia correspondiente a la base de datos de reserva de los archivos de registro de reconstrucción archivados.

Por ejemplo, el archivo de parámetros de la base de datos de producción puede contener las siguientes entradas:

```
log_archive_dest_1  =   '/db00/arch'
log_archive_dest_state_1  =  enable
```

La primera área de destino de los registros archivados es el destino principal para los nuevos archivos de registro de reconstrucción archivados. El segundo conjunto de parámetros indica a la base de datos que escriba copias de los archivos de registro de reconstrucción archivados en el área de destino de la base de datos de reserva. Para este ejemplo, el nombre de servicio de la base de datos de reserva es STBY:

```
log_archive_dest_2="service=stby.world mandatory reopen=60"
log_archive_dest_state_2 = enable
```

La definición LOG_ARCHIVE_DEST_2 especifica el nombre de servicio de la base de datos de reserva. La instancia STBY debe ser accesible a través de Oracle Net. Si en un momento dado hubiera algún problema de conectividad, el proceso ARCH intentaría volver a abrir la conexión 60 segundos más tarde.

En Oracle8i, versión 3, se introdujo una implementación basada en scripts denominada *Oracle8i Data Guard* para facilitar la administración de la base de datos de reserva. Tareas como la administración de la base de datos de producción, la base de datos de

reserva física, las aplicaciones asociadas y los servicios de transporte de registros se llevan a cabo a través de una interfaz de línea de comandos. Oracle9i Data Guard se utiliza para automatizar muchas de las tareas gestionadas por la versión Oracle8i utilizando tanto una interfaz GUI como la interfaz de línea de comandos DGMGRL.

Cada herramienta se utiliza para tareas diferentes. Se utiliza la herramienta GUI, el Data Guard Manager, para las tareas de configuración, de instalación y otras tareas relacionadas con el funcionamiento, mientras que la interfaz de la línea de comandos, DGMGRL, se utiliza para supervisión básica mediante algunas vistas V$; para introducir cambios de roles al migrar de una máquina a otra tras un error, además de para configurar el entorno de Data Guard.

2.3.9 Bases de datos duplicadas

Puede utilizar las características de duplicación que ofrece Oracle para copiar y propagar los cambios introducidos en los datos de una base de datos a otra. Puede seleccionar las tablas, columnas y filas que desee que sean duplicadas desde una base de datos a su correspondiente réplica. En cuanto a la arquitectura, un entorno de duplicación presenta una configuración servidor/servidor, como puede ver en la Figura 2.6. Las bases de datos duplicadas se utilizan, generalmente, por las siguientes razones:

- Para permitir el acceso OLTP para múltiples instalaciones. Los requisitos de introducción de datos de gran volumen en ubicaciones remotas se pueden beneficiar de la duplicación puesto que varias instalaciones repartidas por todo el mundo pueden llevar a cabo la introducción de datos simultáneamente. Las transacciones se pueden enviar a las distintas bases de datos para que el contenido de todas las bases de datos relacionadas esté relativamente actualizado.

- Para crear bases de datos de sólo lectura o de informes y almacenes de datos. Puede utilizar las técnicas de duplicación para separar la base de datos OLTP de la base de datos utilizada para dar soporte a los requisitos de generación de informes.

La primera opción de la lista anterior se conoce como *duplicación o replicación multimaestro:* múltiples bases de datos pueden introducir cambios en los datos y esos datos deben difundirse por las demás bases de datos de la red. Los sistemas de bases de datos multimaestro deben tener en cuenta la resolución de los conflictos que puedan surgir durante el procesamiento de las transacciones.

La segunda opción de la lista anterior es la *duplicación o replicación de sólo lectura.* En ella, los datos sólo se duplican en una dirección: de una base de datos de origen a una o más bases de datos de destino. En general, la replicación de sólo lectura resulta mucho más fácil de administrar y configurar que la replicación multimaestro. Puede utilizar la copia de sólo lectura de la base de datos para generar informes, que sirvan de base para un almacén de datos o como base de datos de prueba. Puesto que la réplica es una base de datos distinta, puede crear esquemas de indexación, tablas y procesos diferentes en la base de datos duplicada.

2.3.10 Acceso a archivos externos

Oracle proporciona numerosos métodos para interactuar con archivos externos. Puede utilizar archivos externos como orígenes de datos, como código fuente de scripts o como archivos de salida. Los usos de archivos externos más comunes son los siguientes:

- Un código fuente para scripts, escrito en SQL*Plus, SQL o PL/SQL.
- La información de salida de un script SQL*Plus, generada mediante el comando **spool.**
- Como información de entrada o salida de un programa PL/SQL, a la que se accede mediante el paquete UTL_FILE.
- Como información de salida de un script PL/SQL, generada utilizando el paquete DBMS_OUTPUT.
- Como datos externos a los que se hace referencia desde la base de datos mediante el tipo de datos BFILE. El tipo de datos BFILE almacena un puntero a un archivo binario externo. En primer lugar, hay que utilizar el comando **create directory** para crear un puntero de directorio dentro de Oracle al directorio en el que se almacena el archivo.
- Como un programa externo al que se accede mediante DBMS_PIPE. El programa debe estar escrito en un lenguaje de tercera generación, 3GL, para el que Oracle ofrezca soporte, como, por ejemplo, C, Ada o COBOL.

Siempre que la aplicación utilice archivos externos, la seguridad es algo que hay que tener en cuenta. Las consideraciones sobre seguridad incluyen las siguientes:

- ¿Aparecen las contraseñas como parte del código de alguno de los archivos? Los scripts que se ejecutan repetidamente pueden tener alguna forma de conectarse a la base de datos.
- ¿Son los archivos seguros? ¿Puede otro usuario leer los archivos?
- Si utiliza UTL_FILE, ¿es seguro el directorio que utiliza, identificado mediante el parámetro UTL_FILE_DIR? Los usuarios podrán ver todos los archivos del directorio.
- Si utiliza los tipos de datos BFILE, ¿es seguro el directorio asociado? ¿Están los archivos protegidos contra posibles modificaciones?

Si la aplicación utiliza archivos externos, debe asegurarse de que se hace una copia de seguridad de estos archivos utilizando el mismo esquema de planificación que utiliza para

las copias de seguridad de la base de datos. En caso de que la base de datos se perdiera a causa de un desastre, las copias de seguridad de la base de datos se utilizarían durante la recuperación, pero si no se ha hecho una copia de seguridad de los archivos externos asociados, es probable que la aplicación deje de ser utilizable.

2.3.11 Acceso a tablas externas

A partir de Oracle9i se puede establecer una determinada equivalencia entre estructuras externas de archivos que permitan manejarlos como si se trataran de tablas de sólo lectura. Los comandos DDL para los datos externos especifican las definiciones de columnas de los datos e indican a Oracle cómo establecer esas equivalencias entre estas columnas y los datos del archivo. No puede llevar a cabo operaciones DML en .as tablas externas y no se pueden crear índices sobre ellas.

CAPÍTULO 3
PLANIFICACIÒN Y ADMINISTRACIÒN DE LOS ESPACIOS DE TABLAS

3.1 El producto final

El objetivo del diseño de la base de datos que se describe en este capítulo es configurar la base de datos de forma que los objetos estén separados en función del tipo de objeto y el tipo de actividad. Esta configuración reduce enormemente la cantidad de trabajo de administración que debe realizarse en la base de datos, a la vez que disminuye las necesidades de supervisión. De este modo, los problemas que puedan surgir en una determinada área no afectarán al resto de la base de datos.

Los objetos lógicos de la base de datos deben clasificarse en función de la forma en la que vayan a usarse y de la influencia que tenga su estructura física en la base de datos. El proceso de clasificación consiste en separar las tablas de sus índices y las tablas con mucha actividad de aquellas con poca actividad. Aunque el volumen de actividad de los objetos sólo puede determinarse durante el uso en el entorno de producción, normalmente suele ser posible aislar un conjunto básico de tablas muy utilizadas.

Otros puntos que hay que tener en cuenta son los datos de sólo lectura, los datos transportables y el tipo de administración que se desee aplicar.

3.2 Arquitectura flexible óptima (OFA)

En las siguientes secciones encontrará las categorías de objetos que se definen en OFA. A continuación de estas descripciones, verá cómo se utilizan algunas opciones avanzadas como, por ejemplo, los espacios de tablas de sólo lectura.

3.2.1 El punto de partida: el espacio de tablas SYSTEM

Aunque no es recomendable, es posible almacenar todos los objetos de la base de datos en un único espacio de tablas, lo que equivaldría a almacenar todos los archivos en el directorio raíz. El espacio de tablas SYSTEM, que es el equivalente en Oracle a un +directorio raíz, es el lugar donde se guardan las tablas del diccionario de datos (propiedad de SYS). Aquí es donde también se encuentra el segmento de anulación SYSTEM, y, durante la creación de la base de datos, el espacio de tablas SYSTEM se utiliza, temporalmente, para almacenar un segundo segmento de anulación (que después se desactiva o se elimina).

No hay ninguna razón para almacenar en el espacio de tablas SYSTEM cualquier otra cosa que no sean las tablas del diccionario de datos y el segmento de anulación SYSTEM. El hecho de almacenar otros tipos de segmentos en SYSTEM aumenta la probabilidad de que se produzcan problemas de gestión del espacio, que obligarían a reconstruir el espacio de tablas. Dado que la única forma de reconstruir el espacio de tablas SYSTEM consiste en volver a crear la base de datos, se recomienda sacar de SYSTEM todos los elementos que se puedan sacar.

Las *tablas del diccionario de datos* almacenan toda la información relativa a todos los objetos de la base de datos. Los *segmentos del diccionario de datos,* el almacenamiento

físico de las tablas del diccionario de datos, se guardan en el espacio de tablas SYSTEM y son bastante estáticos, a menos que se realicen grandes cambios estructurales en las aplicaciones de la base de datos. Los segmentos del diccionario de datos se crean durante el proceso de creación de la base de datos y, generalmente, son bastante pequeños.

El espacio de tablas SYSTEM necesita más de 300 MB para una base de datos Oracle9i.

Cuantos más objetos procedimentales (tales como disparadores y procedimientos) se creen y más tipos abstractos de datos y características orientadas a objetos se utilicen, más grandes serán los segmentos del diccionario de datos. Los objetos procedimentales almacenan código PL/SQL en la base de datos, y sus definiciones respectivas se almacenan en las tablas del diccionario de datos.

De manera predeterminada, cualquier usuario nuevo que se cree en la base de datos dispondrá de SYSTEM como espacio de tablas predeterminado. Para impedir que los usuarios creen objetos en el espacio de tablas SYSTEM, hay que revocar todas las cuotas de usuarios sobre SYSTEM (las cuotas les otorgan la capacidad de crear objetos en dicho espacio de tablas):

```
alter user  USUARIO quota  0  on  SYSTEM;
```

Cuando se crea un usuario (mediante el comando **create user),** se puede especificar un espacio de tablas predeterminado:

```
Create user NOMBRE_USUARIO identified by CONTRASEÑA
default  tablespace NOMBRE_ESPACIOTABLAS;
```

Una vez creado el usuario, se puede utilizar el comando

```
alter user NOMBRE_USUARIO default tablespace
NOMBRE_ESPACIOTABLAS
```

para asignar un nuevo espacio de tablas predeterminado a un usuario. Si se especifica un espacio *de* tablas predeterminado para usuarios y desarrolladores, los objetos creados sin una cláusula **tablespace** se almacenarán fuera del espacio de tablas SYSTEM.

Si se concede a un usuario el privilegio de sistema UNLIMITED TABLESPACE (espacio de tablas ilimitado) o el rol RESOURCE (recurso), esa concesión de privilegio, tendrá prioridad sobre cualquier restricción de cuota que haya sido establecida para el usuario.

En las secciones siguientes se describe cada tipo de objeto de base de datos, su uso y la razón por la que debería almacenarse aparte del resto de la base de datos. Desarrollando una arquitectura lógica coherente, se puede simplificar el desarrollo de una arquitectura física coherente para las bases de datos. Cuanto más coherente sea la arquitectura física de la base de datos, más sencillas resultarán las actividades de administración de la base de datos.

3.2.2 Separación de los segmentos de datos de la aplicación: DATA

Los *segmentos de datos* son las áreas físicas en las que se almacenan los datos asociados con las tablas y clusters. La base de datos suele acceder a estos segmentos con mucha frecuencia, por lo que experimentan un gran número de transacciones de manipulación de datos. El principal objetivo de una base de datos de producción es gestionar las solicitudes de acceso a los segmentos de datos.

Normalmente, un espacio de tablas DATA contiene todas las tablas asociadas a una aplicación. El alto volumen de tráfico de E/S que registran estas tablas las convierte en candidatas ideales para aislarlas en su propio espacio de tablas. Si se aislan las tablas de la aplicación en un espacio de tablas DATA, se pueden separar los archivos de datos de ese espacio de tablas del resto de los archivos de datos de la base de datos. Esta separación de los archivos de datos en distintas unidades de disco puede mejorar el rendimiento (gracias a que la contienda por los recursos de E/S es menor) y simplificar la gestión de archivos.

Es probable que los segmentos de un espacio de tablas DATA estén fragmentados. El hecho de que los segmentos de datos estén fragmentados significa que no se les asignó el tamaño adecuado durante su creación. La gestión de la fragmentación es otro de los motivos que hacen necesario separar el espacio de tablas DATA de otros espacios de tablas, a fin de poder aislar y resolver mejor los problemas de fragmentación.

En el espacio de tablas DATA, se pueden incluir múltiples tipos de datos. Las tablas pequeñas y estáticas tienen diferentes características de almacenamiento y requisitos de administración que las tablas grandes y activas. Consulte la sección dedicada al espacio de tablas DATA_2, descrito más adelante en este capítulo, para conocer los detalles de la administración de los diferentes tipos de tablas.

3.2.3 Espacios de tablas administrados localmente

A partir de la versión Oracle8i los espacios de tablas distintos de SYSTEM se pueden gestionar localmente o utilizando el diccionario. En un espacio de tablas gestionado por un diccionario, Oracle mantiene en el diccionario de datos varias tablas en las que se registra el uso que se hace del espacio dentro de los espacios de tablas. En un espacio de tablas gestionado localmente, la información de uso del espacio se almacena, de forma predeterminada, en un segmento de tipo mapa de bits en los primeros 64 KB del primer archivo de datos del espacio de tablas. Es probable que se necesiten, y que se vayan añadiendo según convenga, segmentos de mapa de bits adicionales.

En versiones anteriores a Oracle9i los espacios de tablas se creaban de manera predeterminada como espacios de tablas gestionados por el diccionario. A partir de Oracle9i, la opción predeterminada indica que se creen los nuevos espacios de tablas como espacios de tablas gestionados localmente. Este tipo de espacios de tablas suele requerir menos reorganización y resulta perfecto para operaciones como cambiar un espacio de tablas de una base de datos a otra.

3.2.4 Separación de los segmentos de datos de poco uso: DATA_2

Al revisar la lista de tablas de datos, es probable que puedan clasificarse fácilmente en dos o más grupos, en función de sus características: algunas contendrán datos muy dinámicos y otras, muy estáticos. Estas últimas pueden contener una lista de países, por ejemplo. Las tablas de datos estáticos suelen experimentar menos operaciones de E/S que las tablas de datos activos. Cuando se consultan, el acceso a una tabla de datos estáticos suele ser concurrente con un acceso a una tabla de datos dinámicos.

Esta E/S concurrente puede distribuirse entre varios archivos (y, por tanto, entre varios discos, para reducir la contienda por los recursos de E/S) colocando todas las tablas de datos estáticos en un espacio de tablas dedicado. Las funciones administrativas que se realizan en el espacio de tablas DATA, como la desfragmentación, sólo ocurrirán en las tablas que tengan mayor probabilidad de requerir intervención administrativa. Mientras tanto, el espacio de tablas de las tablas de datos estáticos, DATA_2, debería permanecer estático y ser fácil de mantener.

Según el tamaño de la base de datos y las características que se utilicen, puede haber varios tipos de espacios de tablas DATA. Además de tener un espacio de tablas DATA_2 para las tablas estáticas, también podría tener varios espacios de tablas de tipo DATA para las categorías que se indican a continuación. Es recomendable tener un espacio de tablas distinto para cada área funcional (contabilidad, ventas, etc.) o para cada aplicación:

- **Agregaciones (vistas materializadas).** Si posee un almacén de datos, lo más probable es que guarde las agregaciones en tablas separadas. Puesto que estas tablas se basan en datos derivados y se suelen eliminar y crear otras nuevas con frecuencia, debería aislarlas de las tablas primarias de transacciones.

- **Datos duplicados (vistas materializadas).** Al igual que las agregaciones, la operación de duplicación utiliza vistas materializadas y otros objetos para distribuir los datos. Las filas duplicadas se basan en datos derivados y es muy probable que se eliminen o creen otras filas nuevas con más frecuencia que en las propias tablas de transacciones de la aplicación.

- **Tablas temporales de trabajo.** Si carga con frecuencia datos de otros sistemas operativos en su base de datos, podría cargar esos datos en tablas temporales antes de pasarlos a las tablas de transacciones. Dichas tablas temporales de trabajo, conocidas también como tablas intermedias, deberían separarse del resto de las tablas de la base de datos.

- **Particiones.** Si utiliza con frecuencia el particionamiento, debería dividir las particiones por espacios de tablas (y, por tanto, por archivos de datos). Dividir las particiones por espacios de tablas permite separar los datos actualmente en uso de la tabla de los datos archivados. Si necesita traspasar una tabla particionada a una base de datos nueva, tendrá que trasladar todos los espacios de tablas de las particiones de índices y de tabla.

¿Cómo puede saber si una tabla es estática? Si no está familiarizado con la aplicación, probablemente tenga que recopilar datos estadísticos (auditar) sobre los accesos a las tablas para poder determinar qué tablas se utilizan más a menudo. Las operaciones de auditoría registran cada acceso a una tabla, además del tipo de acceso que se realiza (insert, select, etc.) Auditar los accesos a varias tablas genera un gran número de registros de auditoría, por lo que debería realizar las auditorias durante una sola semana (o un número de días corto pero suficientemente representativo) y, a continuación, desactivar esta característica. Un solo día en que se utilice la aplicación normalmente debería ser suficiente para determinar el modelo de uso. Al analizar los resultados, debe centrarse en las tablas a las que se accede con mayor frecuencia, así como en los tipos de acceso más comunes que se realicen en cada tabla. Si no se lleva a cabo ninguna operación de inserción (**insert**), actualización (update) o eliminación (delete) en la tabla durante el uso normal, debería considerarse la posibilidad de migrarla desde el espacio de tablas DATA a un espacio de tablas más apropiado.

3.2.5 Separación de los segmentos de índices de la aplicación: INDEXES

Los índices asociados a tablas están sujetos a los mismos problemas de E/S y crecimiento/fragmentación que hacen aconsejable sacar los segmentos de datos del espacio de tablas SYSTEM. No conviene almacenar los segmentos de índices en el mismo espacio de tablas que las tablas de datos a las que estén asociados, ya que registran un gran volumen de operaciones de E/S concurrentes durante la manipulación de datos y las consultas. Si los datos no se insertan en la tabla ordenada por las columnas inde-xadas, el tráfico de E/S de los segmentos de índices podría exceder el tráfico de E/S de la tabla.

Los segmentos de índices también son objeto de fragmentación debida a un dimensionamiento inadecuado o a un crecimiento de la tabla imprevisto. El aislamiento de los índices de la aplicación en un espacio de tablas independiente reduce considerablemente las labores administrativas relacionadas con la desfragmentación de los espacios de tablas DATA o INDEXES.

Para separar los índices existentes de sus tablas, puede utilizarse la opción **rebuild** del comando **alter índex.** Si se ha creado un índice en el mismo espacio de tablas que la tabla a la que indexa, se le puede llevar a otro espacio de tablas con un solo comando. En el ejemplo siguiente, el índice EMPLOYEE$DEPT_NO es trasladado al espacio de tablas INDEXES y se le asignan nuevos valores de almacenamiento con la cláusula **storage:**

```
alter index EMPLOYEE$DEPT_NO rebuild
tablespace INDEXES
storage (initial 2M next 2M pctincrease 0);
```

Si se utilizan particiones, conviene separar los índices de partición de las particiones de tablas. Por ejemplo, se puede crear una tabla con diez particiones. Si se crean índices locales de partición, habrá diez índices locales de partición, es decir, uno para cada una de las particiones de la tabla. Es recomendable almacenar estos índices de partición en una ubicación diferente de las particiones que indexan. Si se crea un índice global en una tabla particionada, las entradas de ese índice podrían hacer referencia a cualquiera de las

particiones de la tabla. Si se utiliza un índice global, se debería separar ese índice de todas las particiones de tabla de la tabla.

Por ejemplo, si la tabla SALES está particionada, se podrán crear índices locales en las particiones de la tabla SALES. Cada una de las particiones del índice local debería almacenarse independientemente de sus respectivas particiones de tabla. Se pueden truncar **(truncate)** particiones individuales y los índices locales correspondientes también serán truncados. Si se crea un índice global que abarque la tabla SALES entera, ese índice debería almacenarse en una ubicación diferente a las del resto de las particiones de la tabla SALES.

Cuando se crea una tabla, la base de datos crea dinámicamente un índice para cualquier restricción de clave primaria (PRIMARY KEY) o unívoca (UNIQUE) que se especifique (a menos que se cree la tabla sin restricciones, después se creen los índices y, por último, se creen las restricciones). Si no se especifican los parámetros **tablespace** y **storage** para estas restricciones durante la creación de la tabla, la base de datos creará automáticamente estos índices en el mismo espacio de tablas en el que se ubica la tabla, utilizando los parámetros de almacenamiento predeterminados de ese espacio de tablas. A fin de evitar este problema, se puede utilizar la cláusula **using index** del comando **create table** para separar los índices de restricciones de la tabla, como se muestra en el listado siguiente:

```
Create table JOB
(Job_Code       NUMBER,
Description     VARCHAR2(35),
constraint JOB_PK primary key (Job_Code)
using index tablespace INDEXES
storage (initial 2M next 2M pctincrease 0))
tablespace DATA
storage (initial 5M next 5M pctincrease 0) ;
```

La tabla JOB se creará en el espacio de tablas DATA, pero su índice de clave primaria, JOB_PK, se creará en el espacio de tablas INDEXES. Observe que la tabla JOB y su índice de clave primaria tienen cláusulas **storage y tablespace** diferentes.

3.2.6 Separación de los segmentos de índices de poco uso: INDEXES_2

Los índices de tablas estáticas de poco uso suelen ser también de poco uso y estáticos. Para simplificar las labores administrativas que requiere el espacio de tablas INDEXES, conviene colocar los índices de las tablas estáticas en un espacio de tablas INDEXES_2 independiente. El hecho de separarlos también contribuye a mejorar el rendimiento de las opciones de optimización, ya que las operaciones de E/S concurrentes entre índices pueden ahora dividirse entre varias unidades de disco.

Si los índices de poco uso ya se han creado en el espacio de tablas INDEXES, entonces hay que eliminarlos y volver a crearlos en INDEXES_2. Esto suele llevarse a cabo a la vez que se realiza el traslado de las tablas de poco uso a DATA_2.

En el siguiente listado se muestra un ejemplo de especificación de un espacio de tablas para un índice creado automáticamente. En este ejemplo, se crea una restricción unívoca (UNIQUE) sobre la columna Description de una tabla estática denominada EMPLOYEEJTYPE. El índice unívoco que creará la base de datos para esta restricción se almacenará en el espacio de tablas INDEXES_2.

```
alter table EMPLOYEE_TYPE
add constraint UNIQ_DESCR  unique(DESCRIPTION)
using index tablespace INDEXES_2;
```

En el caso de que el índice ya exista, se puede trasladar de su espacio de tablas actual a uno nuevo mediante la cláusula **rebuild** del comando **alter index.**

Si crea espacios de tablas DATA adicionales para vistas materializadas, tablas duplicadas, tablas temporales de trabajo y particiones, debería crear espacios de tablas INDEX adicionales para los índices de estas tablas.

3.2.7 Separación de los segmentos de herramientas: TOOLS

A pesar de las advertencias que se hacen en las secciones anteriores de no almacenar segmentos de datos en el espacio de tablas SYSTEM, hay muchas herramientas que hacen precisamente eso. No lo hacen porque necesiten que sus objetos se almacenen en el espacio de tablas SYSTEM, sino porque los almacenan en la cuenta SYSTEM de la base de datos, que normalmente tiene asignado el espacio de tablas SYSTEM como área predeterminada para almacenar objetos. Para evitarlo, basta con cambiar el espacio de tablas predeterminado de la cuenta SYSTEM por el espacio de tablas TOOLS.

Muchas herramientas de Oracle y de otros fabricantes crean tablas propiedad de SYSTEM. Si ya se han creado estas tablas en la base de datos, sus objetos pueden cambiarse de sitio utilizando la opción **move** del comando **alter table.**

El listado siguiente muestra una parte de este proceso, por el que se concede una cuota **(quota)** sobre el espacio de tablas TOOLS al usuario APP_OWNER:

```
alter user APP_OWNER quota 50M on TOOLS;
```

3.2.8 Separación de los índices de herramientas: TOOLS_I

Si observa demasiada actividad en el espacio de tablas TOOLS, puede trasladar los índices de las tablas de herramientas a otro espacio de tablas. Puede utilizar la cláusula **rebuild** del comando **alter index** para cambiar un índice existente a un espacio de tablas diferente mientras se vuelve a crear. En el ejemplo siguiente, se traslada el índice TOOLTAB_PK al espacio de tablas TOOLS_I y se le asignan nuevos valores de almacenamiento **(storage):**

```
alter index TOOLTAB_PK rebuild
tablespace TOOLS_I
storage (initial 2M next 2M pctincrease 0);
```

3.2.9 Separación de los segmentos de anulación: RBS

Los *segmentos de anulación* mantienen la coherencia de lectura tanto a nivel de instrucción, como a nivel de transacción en la base de datos. Para aislar los segmentos de anulación (que realizan operaciones de E/S para las transacciones de la base de datos) del diccionario de datos, hay que crear un espacio de tablas de segmentos de anulación que sólo contenga segmentos de anulación. Este modo de separarlos también simplifica enormemente su gestión.

Una vez que se ha creado el espacio de tablas RBS y que se ha activado un segmento de anulación dentro del mismo, se puede eliminar el segundo segmento de anulación del espacio de tablas SYSTEM. Tal vez prefiera mantener inactivo este segmento de anulación en SYSTEM en lugar de deshacerse de él, por si ocurriera algún problema con el espacio de tablas RBS.

Los segmentos de anulación se amplían dinámicamente hasta adquirir el tamaño de la transacción de mayor tamaño, y se contraen hasta adquirir un tamaño óptimo especificado. Las operaciones de E/S que afectan a los segmentos de anulación suelen ser concurrentes con las operaciones de E/S que afectan a los espacios de tablas DATA e INDEXES. Su separación ayuda a evitar la contienda de E/S y los hace más fáciles de administrar.

3.2.10 Separación de los segmentos de anulación especializados: RBS_2

Si no está utilizando un espacio de tablas de tipo deshacer cambios, los segmentos de anulación del espacio de tablas RBS han de ser del tamaño y del número adecuados para que permitan el uso de la aplicación en el entorno de la aplicación. Casi siempre habrá una transacción (normalmente, una transacción por lotes) de un tamaño que la configuración del segmento de anulación de producción no admita. Cuando se ejecute dicha transacción, ocupará uno de los segmentos de anulación de producción y lo ampliará en gran medida, puesto que la transacción ocupa una gran cantidad de espacio libre hasta que o bien se complete, o bien falle.

Los segmentos de anulación de producción deberían ser de uso exclusivo de los usuarios del entorno de producción. Los requisitos especiales de transacción (como grandes cargas de datos, agregaciones o eliminaciones) se gestionan mejor con un segmento de anulación independiente. Para especificar un segmento de anulación determinado, el código de la aplicación debe utilizar el comando

```
set transaction use rollback segment SEGMENT_NAME
```

antes de ejecutar la transacción. La necesidad de configurar parámetros tan específicos en el código de una aplicación puede requerir una mayor participación de los administradores de bases de datos en el desarrollo de aplicaciones. Sin embargo, el comando **set transaction** sólo resuelve el problema en parte porque el segmento de anulación seleccionado sigue ocupando espacio en el espacio de tablas RBS de un entorno de producción.

Puede crear un espacio de tablas de segmento de anulación independiente que exista únicamente para dar soporte a las transacciones de gran tamaño. Una vez que este segmento de anulación se encuentre en línea y listo para ser utilizado, todas las transacciones activas

lo podrán utilizar y lo utilizarán. El comando **set transaction use rollback segment** no reserva el segmento de anulación especificado para esa transacción, sino que cuando la transacción se complete, el segmento de anulación se colocará fuera de línea hasta que otra transacción lo necesite.

3.2.11 Uso de un espacio de tablas de tipo deshacer cambios

En Oracle9i se puede utilizar un *espacio de tablas de tipo deshacer cambios* para guardar toda la información de datos de tipo deshacer cambios en un solo espacio de tablas. Cuando se crea un espacio de tablas para deshacer cambios, Oracle gestiona el almacenamiento, la conservación y la utilización del espacio para los datos de anulación mediante operaciones de tipo deshacer cambios gestionadas por el sistema (sys-tem managed undo, SMU). En el espacio de tablas de tipo deshacer cambios no se almacena ningún objeto permanente.

3.2.12 Separación de los segmentos temporales: TEMP

Los *segmentos temporales* son objetos que se crean dinámicamente en la base de datos y que almacenan datos durante las operaciones de ordenación de gran tamaño (como **select distinct, union y create index).** Dada su naturaleza dinámica, no conviene almacenar los segmentos temporales junto con ningún otro tipo de segmentos.

Se puede definir un espacio de tablas como espacio de tablas <<temporal>> utilizando los comandos **create tablespace y alter tablespace.** Si se define un espacio de tablas como espacio de tablas temporal, no se podrán crear segmentos permanentes, como tablas e índices, en ese espacio de tablas. Además, el segmento temporal de ese espacio de tablas no se eliminará cuando se complete el comando relacionado, reduciendo la carga de la gestión de espacio que se lleve a cabo para ese espacio de tablas.

A menos que defina un espacio de tablas como espacio de tablas temporal, los segmentos temporales se eliminan una vez completado el comando al que dan soporte. Al crear un espacio de tablas temporal, éste crea un *segmento de ordenación,* que se amplía tanto como sea necesario para albergar cualquier tipo de ordenación que se lleve a cabo. El segmento de ordenación existe hasta que se cierre y reinicie la base de datos. Separando los segmentos temporales de SYSTEM, se elimina un problema potencial del área del diccionario de datos y se crea un espacio de tablas sencillo de gestionar.

Un espacio de tablas se puede definir como espacio de tablas temporal en el momento de la creación, en cuyo caso no se podrán crear objetos permanentes en él. Se puede consultar el tipo de contenido de un espacio de tablas ('PERMANENT', permanente, o 'TEMPORARY', temporal) desde la vista de diccionario DBA_TABLE_SPACES. Un espacio de tablas permanente que ya existe se puede convertir en un espacio de tablas temporal mediante el comando ***alter tablespace nombre_espacio_de_tablas*** **temporary.**

Para especificar un espacio de tablas temporal distinto de SYSTEM, se puede emplear el comando **create user,** tal y como se muestra en el siguiente listado:

```
createuser USERNAME identified by PASSWORD
default tablespace DATA temporary tablespace TEMP;
```

Si la cuenta ya ha sido creada, puede utilizarse el comando para volver a asignar un espacio de tablas temporal. Con el comando **alter user,**

```
alter user USERNAME temporary tablespace TEMP;
```

todos los segmentos temporales que se creen de aquí en adelante para la cuenta de ese usuario se crearán en TEMP.

Generalmente, conviene cambiar la configuración del espacio de tablas temporal de los usuarios SYSTEM y SYS a un espacio de tablas que no sea SYSTEM.

A partir de la versión Oracle9i es recomendable definir un espacio de tablas temporal predeterminado distinto a SYSTEM cuando se crea una base de datos utilizando la cláusula **default temporary tablespace** del comando **create database.** De lo contrario, Oracle utilizará el espacio de tablas SYSTEM como espacio de tablas temporal predeterminado para todos los usuarios y escribirá un mensaje de advertencia en el registro de alertas recordando que se recomienda el uso de un espacio de tablas temporal predeterminado que no sea SYSTEM.

Si utiliza Oracle9i y ya ha creado su base de datos, puede utilizar el comando **alter database** ***nombre_base_de_datos default temporary tablespace nombre jespacio_de_tablas*** para cambiar el espacio de tablas temporal predeterminado.

3.2.13 Separación de los segmentos temporales específicos del usuario:TEMP_USUARIO

Determinados usuarios, como GL (el esquema General Ledger, Libro mayor general) de una base de datos de Oracle Applications, pueden requerir segmentos temporales mucho mayores que el resto de los usuarios de la aplicación. En tal caso, conviene separar dichos segmentos temporales del espacio de tablas TEMP estándar. Esta separación facilita la administración, ya que se puede realizar el diseño pensando en el uso habitual del sistema y reservar segmentos temporales de tipo *TEMP_USUARIO* para las situaciones excepcionales. En la práctica, conviene nombrar estos espacios de tablas de acuerdo con el nombre del usuario, como TEMP_GL o TEMP_SCOTT.

Para especificar un espacio de tablas temporal para un usuario, puede utilizarse el comando **create user:**

```
Create user NOMBREUSUARIO identified by CONTRASEÑA
default  tablespace  NOMBRE_ESPACIOTABLAS
temporary tablespace TEMP_USUARIO;
```

En caso de que ya se haya creado el usuario, para cambiar la configuración del espacio de tablas temporal puede usarse el siguiente comando:

```
alter user NOMBREUSUARIO temporary tablespace
TEMP_USUARIO;
```

Al utilizar el comando **alter user,** todos los segmentos temporales que se creen en adelante para la cuenta de ese usuario, se crearán en el espacio de tablas *TEMP_USUARIO* personalizado del usuario.

Si las aplicaciones utilizan un mismo identificador de inicio de sesión en la base de datos para todos los usuarios, las opciones serán más limitadas, cualquier cambio que se introduzca en el tamaño o en la asignación del espacio de tablas temporal afectara a todos los usuarios. En este tipo de configuración, caben dos posibilidades principales:

- Dar al espacio de tablas temporal TEMP un tamaño adecuado para admitir transacciones extremadamente grandes, además de transacciones más pequeñas. No cree un segundo espacio de tablas temporal.

- Crear un espacio de tablas *TEMP_USUARIO* además del espacio de tablas TEMP. Programe las transacciones grandes para que se ejecuten fuera del horario de la jornada laboral. Antes de ejecutar las transacciones grandes, cambie la configuración del espacio de tablas temporal del usuario para que apunte a TEMP *USUARIO.*

3.2.14 Separación de los usuarios: USERS

Aunque los usuarios no suelen tener privilegios de creación de objetos en las bases de datos de producción, sí pueden tenerlos en las bases de datos de desarrollo. Los objetos de usuario, al contrario que los objetos que pertenecen a áreas funcionales, como el área de contabilidad, suelen ser de naturaleza transitoria y no suelen estar bien dimen-sionados. Por todo ello, conviene separar estos objetos del resto de la base de datos. De esta forma se contribuye a minimizar la influencia de los experimentos de los usuarios en el funcionamiento de la base de datos.

Si desea separar los objetos de los usuarios, cambie sus espacios de tablas predeterminados por el espacio de tablas USERS. Para especificar un espacio de tablas predeterminado alternativo puede utilizar el comando **create user,** tal y como se muestra en este ejemplo:

```
createuser NOMBREUSUARIO identified by CONTRASEÑA
default tablespace USERS temporary tablespace TEMP;
```

Si ya se ha creado una cuenta de usuario, puede emplear los comandos:

```
alter user NOMBREUSUARIO default tablespace USERS;
alter user NOMBREUSUARIO quota nnnn on USERS;
```

para asignar un nuevo espacio de tablas predeterminado. Al asignar un nuevo espacio de tablas predeterminado de esta forma, los objetos que se hayan creado sin una cláusula **tablespace** se almacenarán en USERS, pero no se moverá ningún objeto que ya estuviera creado en el espacio de tablas anterior del usuario al espacio de tablas recién creado.

3.2.15 Otros tipos de espacios de tablas

Dependiendo de la aplicación que se utilice, pueden existir otros tipos de objetos en la base de datos. Cada tipo diferente de objeto debería estar almacenado en su propio espacio de tablas a fin de minimizar la influencia que ejercen los objetos entre sí. En la tabla siguiente se enumeran los tipos.

Tabla 3.1 Otros tipos de espacios de tablas.

Tipo	Descripciòn
MVIEWS	Para vistas materializadas, ya sean agregaciones, combinaciones o duplicaciones. Las tablas e índices subyacentes de las vistas materializadas se gestionan de forma diferente a la mayor parte de los objetos de la base de datos.
MVIEWS_I	Para los índices en sus vistas materializadas.
PARTITIONn	Para las particiones. Si ha instalado la opción de particionamiento, puede utilizar las particiones para distribuir la carga de E/S y para facilitar la gestión de las tablas de muy gran tamaño. Conviene determinar qué particiones se utilizan con más frecuencia y gestionarlas como si se tratara de tablas independientes.
PARTITIONn_I	Para los índices locales y globales asociados a las particiones.
TEMP_WORK	Para utilizarlos durante las grandes cargas de datos. Los segmentos de trabajo temporales se caracterizan por grandes cargas de datos por lotes, seguidas de la eliminación de los datos o el truncado de la tabla. Es probable que necesite también un espacio de tablas diferente para los índices de las tablas temporales de trabajo.

En caso de no utilizar la funcionalidad de duplicación, vistas materializadas, particiones o tablas temporales de trabajo para las cargas de datos, ninguno de estos espacios de tablas será necesario. Si se emplean particiones, pueden trasladarse las particiones existentes a espacios de tablas distintos. Si se utilizan índices locales de partición, conviene separar los índices locales de sus respectivas particiones de tabla.

3.2.16 Tipos avanzados de espacios de tablas

Además de los tipos comunes de espacios de tablas, se pueden crear espacios de tablas para tareas más avanzadas, como se describe en las secciones siguientes.

3.2.16.1 Espacios de tablas para tablas temporales globales

Se puede utilizar el comando **create temporary tablespace** para crear un espacio de tablas temporal para los objetos de esquemas cuya existencia se reduce a una sesión. Al ejecutar el comando, se especifica el nombre y el tamaño de los archivos del espacio de tablas mediante la cláusula **tempfile.** En Oracle9i, si se omite la cláusula **tempfile** y se ha asignado el nombre de un directorio al parámetro DB_CREATE_FILE_DEST, Oracle creará un archivo temporal de 100 MB gestionado por Oracle en el directorio de destino predeterminado para estos archivos que se naya especificado en dicho parámetro. Se puede modificar la configuración del parámetro DB_CREATE_FILE_DEST utilizando el comando **alter system.**

La información sobre archivos temporales se muestra en DBA_TEMP_FILES y V$TEMPFILE, mientras que la información sobre el resto de los archivos de datos está contenida en DBA_DATA_FILES y V$DATAFILE.

3.2.16.2 Espacios de tablas de sólo lectura

Los espacios de tablas pueden ser de tipo lectura-escritura o de sólo lectura. Si el espacio de tablas contiene datos que no van a cambiar, se podría colocar en modo de sólo lectura. Aquellos espacios de tablas que se vayan a trasladar de una base de datos a otra deben estar en modo de sólo lectura en el momento en que se realice la operación. El siguiente listado muestra el comando que cambia el modo de funcionamiento del espacio de tablas TEMP_WORK al modo de sólo lectura:

```
alter tablespace TEMP_WORK read only;
```

3.2.16.3 Espacios de tablas de tamaño de bloque variable

A partir de la versión Oracle9i se puede especificar un tamaño de bloque de base de datos al nivel del espacio de tablas. Para seleccionar un tamaño de bloque, se utiliza la opción **blocksize** del comando **create tablespace.** Para ello, es necesario definir los valores de los parámetros incluidos en init.ora, DB_ CACHE_SIZE y DB_nK_CACHE_SIZE. La configuración que se establezca debe corresponderse con la configuración de uno de los parámetros DB_nK_CACHE_SIZE que se hayan definido. Se admiten hasta cinco tamaños de bloque diferentes en una base de datos. El tamaño del bloque del espacio de tablas SYSTEM, el espacio de tablas de tipo des- ;| hacer cambios y el espacio de tablas temporal deben coincidir con el tamaño de bloque predeterminado de la base de datos que se haya especificado en el parámetro DB BLOCK SIZE.

3.2.16.4 Espacios de tablas de tipo deshacer cambios

En Oracle9i se puede utilizar un espacio de tablas de tipo deshacer cambios en lugar de los segmentos de anulación. Se puede asignar un espacio de tablas de tipo deshacer cambios y dejar que Oracle gestione la contienda de bloques, la coherencia de lectura y la administración del espacio.

3.2.16.5 Espacios de tablas fuera de línea y sin registro (Nologging)

La mayoría de los tipos de espacios de tablas se pueden poner fuera de línea (durante las operaciones de mantenimiento) utilizando la cláusula **offline** del comando **alter tablespace.** Cuando se crea un espacio de tablas, se pone en modo de funcionamiento en línea. Los espacios de tablas que se crean con la opción **nologging** (sin registro) no generan entradas de reconstrucción para las transacciones en el nivel de bloque (como, por ejemplo, las inserciones que utilizan la indicación APPEND) pero sí las generan para las instrucciones de inserción **(insert),** actualización **(update)** y eliminación **(delete)** normales.

3.3 Diseños lógicos de sentido común

El diseño lógico resultante de la base de datos debe cumplir los siguientes criterios:

- Los tipos de segmentos que se utilicen de la misma forma deben almacenarse juntos.
- El sistema debe diseñarse para la utilización más habitual (tamaño de transacciones, número de usuarios, número de transacciones, etc.)
- Deben existir áreas independientes para las excepciones de tamaño de transacciones, número de usuarios y número de transacciones.
- Debe minimizarse la contienda entre los espacios de tablas.
- El diccionario de datos ha de estar aislado.

Recuerde que, para que se cumplan estos criterios, el DBA ha de conocer la aplicación que se esté implementando: qué herramientas utilizará, cuáles serán las tablas más activas, cuándo tendrán lugar las cargas de datos, qué usuarios van a necesitar unos recursos excepcionales y cómo se comportan las transacciones estándar. Para adquirir estos conocimientos, el DBA ha de estar muy implicado en el proceso de desarrollo. Las aplicaciones suministradas por terceros, podrían requerir una atención especial por parte del DBA.

Si se cumplen estos criterios, el resultado es un sistema cuyos tipos de segmentos, por variados que sean, no interfieren con los requisitos de recursos del resto de segmentos. La separación efectiva simplifica mucho la administración de la base de datos y el aislamiento y resolución de los problemas de rendimiento. Si la base de datos se diseña de esta forma,

cuando se produzca una fragmentación de segmentos o del espacio libre la solución será mucho más sencilla.

En esta configuración, el único espacio de tablas potencial distinto a SYSTEM en el que pueden existir varios tipos de segmentos es el espacio de tablas USERS. Si el entorno de desarrollo también se utiliza como un entorno de prueba, entonces puede ser conveniente separar los índices de los usuarios en un espacio de tablas denominado USERS_1.

La combinación de un buen diseño lógico de la base de datos con un diseño físico eficiente produce unos sistemas que requieren muy poca optimización después de la primera comprobación de postproducción. Las labores de planificación previa tienen su recompensa en la flexibilidad y el buen rendimiento de la base de datos. El coste de implementar este diseño desde el principio es mínimo; puede incorporarse en todos los scripts de creación de la base de datos de forma automática y constituye ya una parte de los scripts de creación que genera automáticamente el proceso de instalación del software de Oracle. El diseño final de todo el sistema ha de ser una combinación adecuada de las divisiones lógicas que se muestran en la Tabla 3.1.

Una vez que haya establecido la configuración estándar, aplíquela siempre que cree una nueva base de datos. De esta forma simplificará la administración, puesto que todas las bases de datos seguirán las mismas reglas de uso de los espacios de tablas. Estas reglas, que separan los segmentos por tipo y características, aislan la mayor parte de los problemas y simplifican su solución.

Tabla 3.2 Distribución lógica de los espacios de tablas

Espacio de tablas	Uso
SYSTEM	Diccionario de datos.
DATA	Tablas de operación normal.
DATA_2	Tablas estáticas que se utilizan durante la operación normal.
INDEXES	Índices para las tablas de operación normal.
INDEXES_2	Índices para las tablas estáticas.
RBS	Segmentos de anulación para la operación normal.
RBS_2	Segmentos de anulación especiales para las cargas de datos.
TEMP	Segmentos temporales para la operación normal.
TEMP_USUARIO	Segmentos temporales creados para un usuario específico.

Espacio de tablas	Uso
TOOLS	Tablas de herramientas del RDBMS.
TOOLS_I	Índices para tablas de herramientas del RDBMS con mucha actividad.
USERS	Objetos de usuarios, en bases de datos de entorno en desarrollo.
USERS_I	Índices de usuarios, en bases de datos de prueba.
MVIEWS	Vistas materializadas.
MVIEWS_I	Índices sobre las vistas materializadas.
PARTITIONn	Particiones de segmentos de índices o tablas; cree múltiples espacios de tablas para estos elementos.
PARTITIONn_I	Índices globales o locales de particiones.
TEMP WORK	Tablas temporales utilizadas durante el procesamiento de una carga de datos.

3.4 Soluciones

Puede utilizar los tipos de espacios de tablas sugeridos en la Tabla 3.2 para diseñar la configuración que mejor se adapte a sus necesidades. En la Tabla 3.3, verá las configuraciones que se encuentran habitualmente en las aplicaciones de bases de datos.

Tabla 3.3 Diseños comunes de espacios de tablas

Tipo de base de datos	Espacios de tablas
Base de datos de desarrollo pequeña	SYSTEM DATA INDEXES RBS TEMP USERS TOOLS
Base de datos OLTP de producción	SYSTEM DATA DATA_2 INDEXES INDEXES_2 RBS RBS_2 TEMP TEMP_USUARIO TOOLS
OLTP de producción con datos históricos	SYSTEM DATA DATA_2 DATA_ARCHIVE INDEXES INDEXES_2 INDEXES_ARCHIVE RBS RBS_2 TEMP TEMP_USUARIO TOOLS

Tipo de base de datos	Espacios de tablas
Almacén de datos	SYSTEM DATA DATA_2 INDEXES INDEXES_2 RBS RBS_2 TEMP TEMP_USUARIO TOOLS PARTITIONn PARTITIONn_I MVIEWS MVIEWS_I TEMP_WORK TEMP WORK_I

CAPÍTULO 4
DISEÑO FÌSICO DE LA BASE DE DATOS

4.1 Disposición de los archivos de la base de datos

El establecimiento de unos objetivos claros para el diseño de la distribución de archivos y la comprensión de la naturaleza de la base de datos (por ejemplo, de las bases de Datos orientadas a transacciones frente a las bases de datos de lectura intensiva), permite determinar cuál es el diseño más adecuado para distribuir los archivos entre cualquier número de dispositivos. En este capítulo podrá ver diseños físicos para las configuraciones más habituales, así como directrices para aplicarlos a cualquier situación de la que no se hable aquí directamente. Para el proceso de diseño hay que conocer:

1. Los orígenes de la contienda de E/S entre los archivos de datos.
2. Los cuellos de botella de E/S entre todos los archivos de la base de datos.
3. Operaciones de E/S concurrentes entre procesos en segundo plano.
4. Los objetivos de seguridad y de rendimiento de la base de datos.
5. El hardware del sistema y la arquitectura de duplicación en espejo disponibles.
6. El resto de aplicaciones que utilizan los mismos recursos.

En la mayoría de los casos, antes de crear la base de datos sólo se llevan a cabo las tareas relacionadas con la contienda de archivos de datos, duplicación en espejo del hardware y acceso a discos (las tareas 1, 5 y 6), por lo que sólo se está teniendo en cuenta la posibilidad de contienda dentro del diseño del sistema, y no otros aspectos. Si lleva a cabo todos los pasos que acabamos de enumerar, el producto final ha de ser un diseño físico de base de datos adaptado a todas sus necesidades.

4.1.1 Contienda de E/S entre archivos de datos

Al diseñar una base de datos lógica, siga los procedimientos de diseño que le proporcionamos en el Capítulo 3. Si lo hace, obtendrá una base de datos que contendrá alguna combinación de los espacios de tablas mostrados en la Tabla 4.1.

Cada uno de estos espacios de tablas necesita un archivo de datos diferente. Puede supervisar las operaciones de E/S de los archivos de datos después de haber creado la base de datos; esta característica sólo será útil durante las etapas de planificación si se dispone de una base de datos análoga como referencia. Si no se dispone de una base de datos así, entonces el DBA deberá estimar la carga de E/S para cada archivo de datos.

Comience el proceso de planificación del diseño físico calculando la actividad de E/S relativa entre los archivos de datos. Generalmente, los archivos de datos de los espacios de tablas que contienen tablas de aplicación serán muy activos, como lo serán los espacios de tablas de índices, el espacio de tablas SYSTEM y, en el caso de aplicaciones en las que se registren un número elevado de transacciones, el espacio de tablas de tipo deshacer cambios. En aplicaciones de procesamiento de transacciones (OLTP) donde muchos usuarios generen transacciones pequeñas, las tablas y los índices de las aplicaciones se dividirán o particionarán en muchos espacios de tablas con el fin de reducir las operaciones

de E/S registradas en un mismo archivo de datos. En las bases de datos OLTP, el espacio de tablas SYSTEM registra, normalmente, la mitad de operaciones de E/S que los espacios de tablas de datos. En el caso de las aplicaciones de almacenes de datos (las cuales registran menor número de transacciones y consultas pero de mayor duración cada una), el espacio de tablas SYSTEM suele registrar un tercio de la actividad de E/S de los espacios de tablas de datos. La actividad de E/S que se registre en un espacio de tablas para deshacer cambios o en los segmentos de anulación depende del volumen de las transacciones que se lleven a cabo en la base de datos.

Las operaciones de E/S de los espacios de tablas de índice pueden variar notablemente y, a menudo, superan a las operaciones de E/S de los espacios de tablas de datos. Las consultas que se resuelven sin necesidad de acceder a las tablas generan E/S sólo en los espacios de tablas de índices y en el espacio de tablas SYSTEM (cuando se comprueban los permisos de usuario, etc.).

Tabla 4.1. Distribución lógica de los espacios de tablas Espacios de tablas Uso

Espacios de tablas	Uso
SYSTEM	Diccionario de datos
DATA	Tablas de operación normal
DATA_2	Tablas estáticas utilizadas durante la operación normal
INDEXES	índices de las tablas para la operación normal
INDEXES_2	índices para las tablas estáticas
RBS o deshacer cambios (undo)	Segmentos de anulación para la operación normal u operaciones para deshacer cambios gestionados por el sistema
RBS_2	Segmentos de anulación especiales utilizados para cargas de datos
TEMP	Segmentos temporales para la operación normal
TEMP_USUARIO	Segmentos temporales creados para un usuario concreto
TOOLS	Tablas de herramientas del RDBMS
TOOLS_I	índices para tablas de herramientas del RDBMS con mucha actividad
USERS	Objetos de usuario, en las bases de datos de desarrollo

Espacios de tablas	Uso
USERS_I	índices de usuario, en las bases de datos de prueba
MVIEWS	Vistas materializadas
MVIEWS_I	índices sobre las vistas materializadas
PARTITIONn	Particiones de los segmentos de tablas o índices; cree múltiples espacios de tablas para estos elementos
PARTITIONn_I	índices locales o globales de particiones
TEMP WORK	Tablas temporales utilizadas durante el procesamiento de una carga de datos

Durante las transacciones, las tablas muy indexadas suelen registrar más E/S en sus índices que en las tablas base, especialmente si los índices tienen un valor de nivel B de 2 o más.

L os espacios de tablas TOOLS y USERS deberían experimentar muy poca actividad de E/S en un entorno de producción. Los espacios de tablas TEMP y TEMP_WORK, en un entorno de producción, sólo se utilizarán para ordenaciones de gran tamaño; estas operaciones de ordenación deberían llevarse a cabo principalmente utilizando procesos por lotes fuera del horario de trabajo. El tráfico de E/S de los espacios de tablas MVIEWS será intenso durante la creación y la actualización de las vistas materializadas, en cualquier otro momento, se limitará a las consultas de las vistas materializadas.

En las bases de datos de un entorno de producción, al menos un 90 por 100 del flujo de E/S de la base de datos se concentrará en la combinación de los espacios de tablas SYSTEM, DATA, INDEX y RBS/deshacer cambios. Se recomienda tener, como mínimo, un disco diferente para cada uno de estos espacios de tablas. Si las tablas y los índices de las aplicaciones están particionados, se necesitarán discos adicionales para distribuir las particiones. Al distribuir los archivos de datos por los discos, procure no agregar nuevos espacios de tablas a los dispositivos dedicados a estos cuatro espacios de tablas.

En la mayor parte de los sistemas de producción de empresas, las tecnologías RAID se utilizan para distribuir la actividad de E/S de los archivos entre varios discos. Por ejemplo, los discos RAID-5 distribuyen tanto los archivos a los que dan soporte como los bloques de datos de paridad que utilizan para recuperarse de posibles errores. En este tipo de entorno hay que supervisar la E/S a nivel de la matriz de discos RAID para asegurarse de que la matriz no presenta problemas de acumulación de operaciones de E/S pendientes. Para un entorno de UNIX, se puede supervisar el tráfico de E/S de un dispositivo utilizando el comando **sar -d**. Por ejemplo, el siguiente fragmento de la salida de un comando **sar -d** muestra el nombre del dispositivo, el porcentaje de ocupación y la media del tamaño de la cola para índices de actividad de E/S distintos de cero, en diferentes dispositivos utilizados para los archivos de datos de un sistema de producción:

```
device    %busy    avque

ssd1        99      16.6
ssd2        74       4.9
Ssd7         1       0.0
ssd8        63       1.1
sd35        30       0.3
```

El listado anterior muestra que cuatro dispositivos operan a un 30 por 100 o más de su capacidad de E/S, mientras que uno apenas se utiliza. Cada uno de los cuatro dispositivos que más se utilizan tiene una cola de trabajos que están en espera para utilizar sus recursos de E/S. A medida que aumenta el porcentaje de recursos ocupados, aumenta la media de trabajos en cola de espera.

Desde el punto de vista de la optimización, una salida como la siguiente no significa que algo vaya mal:

```
device    %busy    avque

ssd1        99       0.0
ssd2        74       0.0
Ssd7         1       0.0
ssd8        63       0.0
sd35        30       0.0
```

En este listado modificado no hay ningún trabajo a la espera de acceder a los dispositivos de E/S. El sistema puede dar soporte a todas las solicitudes de datos que se presentan a la aplicación, aunque esto significa que un disco está ocupado el 99 por 100 del tiempo. Si bien esto no supone ningún problema a corto plazo, sí lo será a largo plazo, puesto que es muy probable que el tráfico de E/S de todos los espacios de tablas aumente a medida que aumente el tamaño de la base de datos y que las solicitudes de E/S excedan la capacidad del disco. Normalmente, mientras el valor medio de la cola de espera sea inferior al número de discos de la matriz RAID, el sistema debería ser *capaz* de responder a las solicitudes de E/S sin que se produzcan esperas importantes. Si se utiliza un 99 o 100 por 100 del disco y el número de trabajos en espera aumenta con el tiempo, habrá que modificar la configuración del sistema o modificar la distribución de los datos entre los archivos de datos.

Estratégicamente, conviene intentar mantener el tráfico de E/S por debajo del 75 por 100 en cada uno de los discos con el fin de estar preparado para aumentos inesperados de la actividad de E/S.

4.1.2 Cuellos de botella de E/S entre todos los archivos de base de datos

Una vez que se haya estimado la actividad de E/S de los archivos de datos, se puede decidir la ubicación de cada uno relativa al resto de archivos. Para el diseño, también hay

que tener en cuenta el resto de tipos de archivo disponibles en la base de datos, como se describe en las secciones siguientes.

4.1.2.1 Archivos de registro de reconstrucción en línea

Los *archivos de registro de reconstrucción en línea* almacenan los registros de cada transacción realizada en la base de datos con la excepción de las transacciones **nologging** realizadas a nivel de bloque. Cada base de datos debe disponer de, al menos, tres archivos de registro de reconstrucción en línea. Oracle escribe secuencialmente en un archivo de registro hasta que está lleno y luego comienza a escribir el segundo archivo de registro de reconstrucción. Una vez que ha llenado el último archivo de registro de reconstrucción en línea, comienza a sobreescribir el contenido del primer registro de reconstrucción con nuevas transacciones.

Los DBA tienen que asegurarse de que los archivos de registro de reconstrucción en línea se dupliquen en espejo de alguna manera. Se pueden utilizar *grupos de registro de reconstrucción* para permitir que la base de datos mantenga dinámicamente varios conjuntos de archivos de registro de reconstrucción en línea. Los grupos de registro de reconstrucción utilizan la base de datos para duplicar en espejo los archivos de registro de reconstrucción en línea, minimizando así los problemas de recuperación que pueda provocar el fallo de un solo disco. También se puede utilizar el sistema operativo para este mismo fin. Es preferible la duplicación en espejo de Oracle porque envía dos o más comandos diferentes a dos o más dispositivos, mientras que con la duplicación en espejo proporcionada por el sistema operativo, si la escritura es incorrecta en el dispositivo original, el sistema operativo copiará los datos incorrectos al resto de dispositivos.

En las bases de datos OLTP, es conveniente separar los archivos de registro de reconstrucción en línea de los archivos de datos a causa de los posibles conflictos de E/S. Toda transacción que no se ejecute con el parámetro **nologging** se guarda en los archivos de registro de reconstrucción. El proceso en segundo plano *LGWR (Log Writer, escritor de registros)* es el que escribe las transacciones en los archivos de registro de reconstrucción en línea. Los datos de las transacciones se escriben concurrentemente en varios espacios de tablas (por ejemplo, en el espacio de tablas de segmentos de anulación RBS y en el espacio de tablas DATA). Las escrituras en los espacios de tablas se realizan por medio del proceso en segundo plano *DBWR (Database Writer, escritor de la base de datos)*. Por tanto, aunque la E/S del archivo de datos pueda distribuirse correctamente, se producirá una contienda entre los procesos en segundo plano DBWR y LGWR si un archivo de datos se almacena en el mismo disco que un archivo de registro de reconstrucción.

Las contiendas que se originen entre procesos en segundo plano sólo serán significativas si provocan colas de espera de E/S, como se ha señalado en la sección anterior.

Si tiene que guardar un archivo de datos en el mismo disco en el que están los archivos de registro de reconstrucción, ese archivo de datos no debería pertenecer al espacio de tablas SYSTEM, RBS o a un espacio de tablas DATA o INDEX muy activo. Todos estos espacios de tablas entrarán en conflicto directamente con los archivos de registro de reconstrucción en línea y aumentarán la probabilidad de que las escrituras en el registro se vean afectadas por las lecturas de la base de datos.

4.1.2.2 Archivos de control

Oracle puede duplicar en espejo los *archivos de control.* El número y nombre de los archivos de control se puede especificar por medio del parámetro CONTROL_FILES del archivo de parámetros de la base de datos. Si los nombres de los archivos de control se especifican por medio de este parámetro durante la creación de la base de datos, los archivos se crearán automáticamente durante dicho proceso de creación. A partir de ese momento, la base de datos mantendrá los archivos de control como copias idénticas unos de otros.

Cada base de datos debería tener un mínimo de tres copias de sus archivos de control, ubicadas en tres dispositivos físicos diferentes para mitigar el impacto que pueda tener un fallo del soporte físico. A pesar de que no hay demasiados datos escritos, a partir de Oracle8, el proceso CKPT escribe información sobre el progreso de los puntos de comprobación en el archivo de control con una frecuencia de tres segundos.

4.1.2.3 Archivos de registro de reconstrucción archivados

Cuando Oracle se ejecuta en modo ARCHIVELOG, la base de datos realiza una copia de cada archivo de registro de reconstrucción en línea una vez que lo ha llenado.

Estos archivos de registro de reconstrucción archivados se suelen escribir en un dispositivo de disco; también se pueden escribir directamente en un dispositivo de cinta, aunque esta opción suele dar mucho trabajo al operador.

El proceso en segundo plano ARCH es el que lleva a cabo la función de archivado. Las bases de datos que utilicen la opción ARCHIVELOG se encontrarán con problemas de contiendas en su disco de registro de reconstrucción en línea en aquellos momentos en los que el volumen de transacciones de datos sea muy alto, ya que LGWR estará intentando escribir en un archivo de registro de reconstrucción mientras que ARCH estará intentando leer de otro. Para evitar esta contienda, distribuya los archivos de registro de reconstrucción en línea entre varios discos. Si está utilizando el modo ARCHIVELOG en una base de datos que lleve a cabo un elevado número de transacciones, evite la contienda entre LGWR y ARCH distribuyendo los archivos de registro de reconstrucción en línea entre varios dispositivos.

Para mejorar aún más el rendimiento del proceso de archivado, cree grupos de archivos de registro de reconstrucción en línea con múltiples miembros. El proceso de archivado será más rápido si es posible leer de un grupo de varios archivos de registro de reconstrucción mientras los está archivando. Si ha activado la característica de destinos múltiples para los archivos de registro de reconstrucción archivados, tendrá que contar con que se escribirá la E/S en cada uno de estos dispositivos de destino.

Los dispositivos de archivos de registro de reconstrucción archivados, por su propia naturaleza, tendrán la misma cantidad de E/S que el dispositivo de los registros de reconstrucción en línea. Los archivos de registro de reconstrucción archivados no se deben guardar en el mismo dispositivo que los espacios de tablas SYSTEM, RBS, DATA o INDEXES, ni en el mismo dispositivo que cualquiera de los archivos de registro de reconstrucción en línea.

4.1.2.4 El software de Oracle

Los archivos del software de Oracle a los que se accede durante el funcionamiento normal de la base de datos varían de acuerdo con los paquetes de los que se tenga licencia en el host en el que reside el servidor. Las operaciones de E/S en estos archivos no se registran en la base de datos, pero se pueden ver mediante utilidades como **sar** en las plataformas UNIX.

En el caso de que surgieran numerosos errores relacionados con los datos durante el uso de la aplicación, vería un aumento significativo de la actividad del archivo de mensajes de ayuda.

Para minimizar la contienda entre los archivos de la base de datos y el código de la misma, intente no incluir los archivos de base de datos en el mismo dispositivo de disco en el que estén los archivos de código. Si hay que colocar archivos de datos en ese dispositivo de disco, se deberían guardar allí los archivos de datos que se utilicen con menos frecuencia.

4.1.3 E/S concurrente entre procesos de segundo plano

A la hora de evaluar las contiendas entre diversos procesos, es importante identificar el tipo de operación de E/S que se está llevando a cabo y su temporización. Los archivos inician una contienda entre sí cuando la E/S de un archivo interfiere con la de un segundo, por lo que es posible guardar en el mismo dispositivo dos archivos a los que no se suela acceder nunca a la vez.

En los entornos RAID en los que se distribuye un único archivo entre varios discos, la probabilidad de que tengan lugar solicitudes concurrentes de E/S en un solo disco es mucho menor. Con los entornos RAID, nos centraremos en analizar separadamente las matrices de discos (como matrices de datos o de índices), en lugar de intentar aislar discos individuales dentro de las matrices.

Los procesos en segundo plano que leen y escriben datos en archivos de datos suelen operar de forma aleatoria. Suele haber <<zonas de alta actividad>>, pero es difícil predecir dónde se localizarán estas zonas y dónde aparecerán la siguiente vez. Los procesos en segundo plano que trabajan con archivos de registro, LGWR y ARCH, llevan a cabo lecturas y escrituras secuenciales.

LGWR escribe en todos los miembros del mismo grupo de registros de reconstrucción al mismo tiempo y el proceso ARCH puede escribir en múltiples destinos a la vez. DBWR también podría intentar escribir en múltiples archivos a un mismo tiempo. Por tanto, existe la posibilidad de que DBWR provoque una contienda consigo mismo al escribir bloques modificados de varias tablas. Para abordar este problema, la mayoría de los sistemas operativos pueden crear varios procesos DBWR para cada instancia. El número de estos procesos se especifica mediante el parámetro de inicialización DBWR_PROCESSES; también se pueden iniciar varios procesos esclavos de E/S asociados a un único proceso principal DBWR mediante el parámetro DBWR_IO_SLAVES; Oracle recomienda asignar un valor comprendido entre n y 2n para este parámetro, donde n representa el número de discos. También se pueden iniciar varios procesos esclavos de E/S asociados a LGWR (por medio de LGWR_IO_SLAVES) y varios procesos esclavos de E/S asociados a ARCH (con

ARCH_IO_SLAVES). A partir de la versión 8.1, Oracle utiliza el parámetro DBWR_IO_SLAVES para determinar cuántos procesos esclavos de E/S asociados a LGWR y a ARCH debe iniciar; asignando a DBWR_IO_ SLAVES un valor mayor que O, los parámetros LGWR_IO_SLAVES y ARCH_IO_SLAVES correspondientes quedan configurados con un valor de 4. Si no puede activar varios procesos DBWR, quizá pueda utilizar la E/S asincrona para reducir la contienda interna de DBWR. Con la E/S asincrona, sólo se inicia un proceso DBWR ya que el procesamiento de E/S se lleva a cabo de forma asincrona.

Compruebe periódicamente la vista V$WAITSTAT para obtener datos sobre la espera de bloques de datos. El hecho de que las esperas de bloques de datos sean importantes podría indicar la necesidad de más procesos DBWR. STATSPACK dedica varias secciones de su informe de salida a los sucesos de espera de la base de datos y de espera en segundo plano.

La naturaleza de la contienda que tendrá lugar entre los procesos de segundo plano es, por tanto, una función del esquema de copia de seguridad que se utilice (ARCH), la carga de transacciones en el sistema (LGWR) y el sistema operativo del host (DBWR). Para diseñar un esquema que elimine las contiendas entre los archivos y procesos, hay que entender claramente las formas en que van a interactuar esos archivos y procesos en el sistema de producción.

4.1.4 Definición de los objetivos de recuperabilidad y de rendimiento del sistema

Antes de diseñar el esquema de distribución física del espacio de almacenamiento de los discos de la base de datos, hay que definir claramente cuáles son los objetivos de recuperabilidad y de rendimiento que buscamos. En los objetivos de recuperabilidad se han de tener en cuenta todos los procesos que afectan a los discos, entre los que debemos incluir, el área de almacenamiento destinada a los archivos de registro de reconstrucción y el área de almacenamiento destinada a las copias de seguridad (en el caso de que se hagan copias de disco a disco). Como parte de la definición de los requisitos de recuperabilidad, deben establecerse los acuerdos relativos al nivel del servicio con los usuarios de las aplicaciones de negocio, definiendo los períodos de tiempo en los que es aceptable que el sistema esté parado y la pérdida de datos que puede ser asumida en el caso de que se produzcan fallos del sistema y desastres importantes.

La recuperabilidad de la base de datos tiene que ser siempre una preocupación principal. La arquitectura destinada a garantizar la recuperabilidad debería complementar a la arquitectura diseñada para la optimización del rendimiento. Si el diseño para la optimización del rendimiento entra en conflicto con el diseño para la recuperabilidad, este último deberá prevalecer.

Puesto que los sistemas heterogéneos podrían tener unos discos más rápidos que otros, para el diseño se ha de tener en cuenta la velocidad de acceso relativa de los discos disponibles.

En cuanto a los objetivos de rendimiento, se han de definir los objetivos para los procesos por lotes, los procesos administrativos, el tiempo de respuesta de las consultas y las operaciones de carga de datos. En cada caso, hay que definir el objetivo principal y el

máximo. Durante la prueba de rendimiento de la aplicación, intente alcanzar el máximo objetivo sin incurrir en esperas de E/S en los discos.

4.1.5 Definición del hardware del sistema y de la arquitectura de duplicación en espejo

Dado que los responsables de administración de sistemas son los que tienen que asignar y administrar el conjunto de discos del servidor, los DBA tienen que trabajar con dichos responsables para administrar el hardware del sistema y la arquitectura de duplicación en espejo. La arquitectura del sistema incluye especificar lo siguiente:

- El número de discos que se necesitan.
- Los modelos de discos que se necesitan (con más rendimiento o con mayor tamaño).
- La estrategia de duplicación en espejo adecuada para las matrices de disco.

El número de matrices de discos necesarias vendrá determinado por el tamaño de la base de datos y el volumen de la actividad de E/S de la base de datos. Siempre que sea posible, esos discos deberían dedicarse en exclusiva a archivos de Oracle para evitar contiendas con archivos que no pertenezcan a Oracle. Si el conjunto de discos es heterogéneo, a la hora de determinar qué unidades de discos se van a dedicar a los archivos de Oracle, hay que tener en cuenta el tamaño y la velocidad de las unidades de disco disponibles.

La duplicación en espejo de discos se emplea para proporcionar tolerancia a fallos, con respecto a los fallos del soporte físico. Este tipo de duplicación se realiza de dos maneras: manteniendo un duplicado en línea de cada disco (lo que se conoce como RAID-1 o *duplicación de volumen),* o bien empleando un sistema de *comprobación de paridad* entre un grupo de discos (normalmente RAID-3 o RAID-5). Los sistemas de comprobación de paridad distribuyen implícitamente los archivos en fragmentos distribuidos entre los discos. En RAID-5, por ejemplo, cada archivo se distribuye por bloques entre los discos del grupo de duplicación en espejo. A continuación, se escribe un dato de comprobación de paridad en otro disco del conjunto, de forma que, si se elimina un disco, su contenido se puede volver a generar si se conoce la información de comprobación de paridad y los contenidos del resto de discos del conjunto de duplicación en espejo.

Así pues, la arquitectura de duplicación en espejo del sistema afecta a la distribución de los archivos de base de datos entre dichos discos. Los discos que se duplican en espejo uno a uno (RAID-1) se pueden tratar como discos independientes. Los discos que forman parte de un sistema de comprobación de paridad (como RAID-3 o RAID-5) se deben considerar como un conjunto único y pueden beneficiarse de la funcionalidad de distribución en bandas que gestionan de forma implícita.

Generalmente, es preferible la distribución en bandas entre gran número de discos pequeños que la distribución en bandas entre un número pequeño de discos grandes.

4.1.6 Identificación de los discos que pueden dedicarse a la base de datos

Con independencia de la arquitectura de duplicación en espejo que se utilice, es importante que los discos que se hayan elegido estén dedicados en exclusiva a la base de datos. De lo contrario, la carga correspondiente a operaciones no relacionadas con la base de datos que se realice sobre esos discos puede afectar a la base de datos y suele ser imposible predecir correctamente cuál va a ser ese efecto. Las áreas de directorios de usuario, por ejemplo, podrían sufrir un incremento repentino de tamaño y, de esta forma, nos quedaríamos sin el espacio reservado para los registros de reconstrucción archivados, lo que provocaría la parada total de la base de datos. Es posible que haya otros archivos que tengan requisitos de E/S muy estrictos, que no se hayan tenido en cuenta al establecer las métricas de utilización de E/S de la base de datos calculadas previamente.

4.1.6.1 Soporte a archivos externos

A partir de Oracle9i se puede hacer referencia a tablas externas utilizando instrucciones SQL. Si los datos se almacenan en un formato coherente, se puede ejecutar el comando **create table** combinado con comandos de tipo SQL*Loader para permitir a Oracle que lleve a cabo una selección en los datos externos. No se pueden ejecutar comandos DML en la tabla externa y no se puede crear un índice Oracle en este tipo de tablas.

Las tablas externas deben almacenarse en un directorio físico previamente identificado mediante el comando **create directory.** Los directorios creados con comandos **create directory** deben tratarse como parte de la base de datos y deben estar en dispositivos de disco aislados.

Los procedimientos de realización de copias de seguridad y de recuperación también deben tener en cuenta los datos de las tablas externas.

4.1.7 Selección del diseño correcto

Para que el diseño de discos sea apropiado:

- La base de datos tiene que ser recuperable.
- Hay que duplicar en espejo los archivos de registro de reconstrucción en línea, utilizando las posibilidades que ofrezca el sistema o la propia base de datos.
- La actividad de E/S de los archivos de la base de datos no debe exceder la capacidad de tasa de transferencia de E/S de un disco individual.
- Hay que minimizar la contienda entre los procesos en segundo plano de la base de datos.
- El rendimiento del disco no debe obstaculizar la consecución de los objetivos de rendimiento de la base de datos.
- El hardware del sistema y las opciones de duplicación en espejo deben reunir los requisitos de rendimiento y recuperabilidad.
- Los discos de la base de datos deben estar dedicados en exclusiva a los archivos de la base de datos.

Además de tener presentes estos objetivos, hay que tener en cuenta el futuro: ¿cómo se espera que cambien con el tiempo las tasas de E/S de los diferentes archivos? Cuando se añadan nuevas tablas, índices y vistas materializadas a la base de datos, ¿se almacenarán en los espacios de tablas existentes o en otros nuevos? ¿En archivos de datos existentes o en otros nuevos? ¿Cómo serán de activos estos objetos nuevos y dónde deberían ubicarse sus archivos?

Dado que es muy probable que la estructura de la base de datos cambie con el tiempo, deberíamos elegir un diseño que permita el crecimiento, tanto en términos de espacio, como en términos de tasa de transferencia de E/S. En las secciones siguientes, se ofrecen algunas directrices y consejos que le servirán de ayuda a la hora de tomar decisiones.

4.1.7.1 Cómo minimizar el impacto de E/S de tablas estáticas

Antes de tomar ninguna decisión sobre la ubicación de los archivos, hay que intentar eliminar tantas variables como sea posible. Si los principales espacios de tablas de datos, espacios de tablas de índices, espacios de tablas de tipo deshacer cambios y la cuenta SYSTEM suponen un 90 por 100 de la actividad de E/S de la base de datos, asegúrese de que el 10 por 100 restante de la actividad está controlado. Para las tablas estáticas pequeñas configure un área diferente en la caché de buffers de bloques de datos. El área de preservación (KEEP) de la caché de buffers de bloques de datos es independiente de la caché de buffers de bloques de datos principal. El parámetro de inicialización que establece el tamaño de la caché de preservación es DB_KEEP_CACHE_SIZE, como se muestra en la siguiente línea:

```
DB_KEEP_CACHE_SIZE=16M
```

Se asignan las tablas estáticas y los índices a la caché de preservación. Considere una tabla estática denominada COUNTRY:

```
Alter table COUNTRY cache storage (buffer_pool  KEEP);
```

La tabla COUNTRY se almacenará en la caché cuando sea leída y se almacenará en el área de preservación, separada de las tablas de aplicación principales. Como resultado, los accesos a la tabla COUNTRY no deberían tener un impacto significativo en la carga de E/S de los discos. Puede centrarse en buscar una solución equilibrada para las cuatro áreas principales (DATA, INDEX, RBS/deshacer cambios y SYSTEM) sin preocuparse de otras tablas, hasta que se añadan objetos nuevos a la base de datos.

4.1.7.2 Cómo minimizar el impacto de los procesos administrativos en la E/S

La actividad de E/S de los procesos de copias de seguridad y de las operaciones de administración (por ejemplo, la recopilación de datos estadísticos) no se puede alterar fácilmente. En cuanto a las copias de seguridad, se debería considerar la posibilidad de utilizar métodos de copias increméntales como los disponibles mediante el administrador

de recuperación, RMAN. En las operaciones de administración, se podrían utilizar particiones para aislar los datos con los que se está trabajando. Siempre que sea posible, conviene programar las operaciones de administración para que se lleven a cabo durante los períodos de escasa actividad por parte de los usuarios a fin de minimizar el impacto que éstas puedan tener en el rendimiento de E/S del sistema.

Conviene planificar cualquier operación por lotes que requiera un movimiento importante de datos (por ejemplo, la duplicación) o consultas (como la elaboración de informes periódicos) para que se lleve a cabo en los períodos de tiempo en los que el grado de uso del sistema sea menor. Si la aplicación de negocio requiere que las operaciones por lotes se lleven a cabo con más frecuencia, necesitará tener en cuenta las necesidades de E/S de los procesos por lotes en los cálculos totales de contienda y de carga del sistema.

4.1.7.3 Establecimiento de compromisos

A menos que disponga de un número infinito de discos y controladores, el diseño físico para un sistema de producción empresarial implicará comprometer unas características en favor de otras. No será posible mantener separadas las tablas de los índices, ni separar los archivos de registro de reconstrucción en línea de los espacios de tablas de tipo deshacer cambios. El desarrollo de un diseño efectivo es un proceso iterativo basado en un conjunto de reglas:

- No comprometa nunca la recuperabilidad de la base de datos. Si tuviera que elegir entre 10 discos llenos de datos y duplicados en espejo o 20 discos medio llenos que no estén duplicados, elija la primera solución. Los sistemas operativos y los controladores de discos de sistemas duplicados en espejo deben ser capaces de distribuir las solicitudes de E/S entre los duplicados, de forma que el impacto que se produzca en el rendimiento de la opción de los 10 discos no será tan importante como se podría pensar en un principio.
- Separe los archivos de datos de espacios de tablas DATA, INDEX, SYSTEM y RBS/deshacer cambios más activos.
- Separe las tablas más activas de las tablas más inactivas. En lugar de limitarse a separar las tablas activas de las tablas de códigos estáticos, debería observar la aplicación con el objetivo de determinar qué tablas almacenan el grueso de los datos. A menudo, menos del 10 por 100 de las tablas de una aplicación registran el 80 por 100 del tráfico de E/S. Aisle las tablas más activas para conseguir más alternativas de ajuste del rendimiento cuando la base de datos crezca en tamaño y en uso.
- Divida en particiones las tablas más activas. Dentro de cada tabla, suele haber unas áreas más activas que otras. En los sistemas OLTP, la mayor parte de las consultas se realizan generalmente en los datos introducidos más recientemente. En un sistema de pruebas de una empresa grande, la mitad de las consultas se realizan en los datos introducidos en los últimos siete días. Dividiendo en particiones la tabla de modo que aisle los datos a los que se accede con más frecuencia, se mejoraría la capacidad para responder a los requisitos de rendimiento de E/S cuando la tabla crezca. Cuando se opta por el particionamiento, la ubicación de la carga de E/S se

desplaza, pero se sigue teniendo el control sobre el punto en el que se produce el problema y cómo solucionarlo.

- Deje espacio para que crezca la tabla, tanto en términos de tasa de transferencia de E/S, como de espacio.
- Supervise las operaciones de E/S que se produzcan en el nivel del sistema operativo y en la base de datos.
- Repita el proceso de diseño físico a intervalos regulares y durante la planificación de cambios importantes en las aplicaciones. Vuelva a comprobar las decisiones tomadas anteriormente en función de los datos de supervisión del sistema disponibles.

En la sección siguiente, se ofrecen algunas directrices relacionadas con la supervisión de la actividad de E/S de los sistemas existentes. A la hora de diseñar nuevos sistemas, es conveniente tener en cuenta los datos obtenidos de la supervisión de los sistemas existentes, ya que ofrecen, como mínimo, una idea aproximada de cómo gestionan los servidores la actividad de E/S y cómo se distribuyen las operaciones de E/S entre las bases de datos. Una vez que se haya sometido a la aplicación a la prueba de rendimiento, se podrán afinar más los cálculos y tomar decisiones sobre las distintas posibilidades de configuración de los discos.

4.2 Verificación de las estimaciones de E/S

Las tablas de estadísticas que hay en el diccionario de datos registran la cantidad de operaciones de E/S que se llevan a cabo sobre cada archivo de datos. Se pueden consultar las tablas de estadísticas internas para verificar las suposiciones que se hicieron durante el proceso de diseño.

La vista V$FILESTAT registra todas las operaciones de E/S que se han realizado en la base de datos desde la última vez que se inició. La consulta siguiente genera un informe sobre la E/S por cada archivo de datos, agrupada por dispositivo de disco.

El script da por hecho que los dispositivos poseen un nombre coherente y que el nombre de dispositivo consta de cinco caracteres (fíjese en SUBSTR en la columna DF.Name). Si los nombres de dispositivos son más largos, tendrá que modificar el script para ajustarlo a sus especificaciones.

```
clear breaks
clear computes
break on Drive skip 1 on report
compute sum of Blocks_Read on Drive
compute sum of Blocks_Written on Drive
compute sum of Total_IOs on Drive
compute sum of Blocks_Read on Report
compute sum of Blocks_Written on Report
compute sum of Total_IOs on Report
ttitle skip center <<Database File I/O by Orive<< skip 2
```

```
select substr(DF.Name,1, 5) Drive,
       DF.Name File_Name,
       FS.Phyblkrd+FS.Phyblkwrt Total_IOs,
       FS.Phyblkrd Blocks_Read,
       FS.Phyblkwrt Blocks_Written
  from V$FILESTAT FS, V$DATAFILE DF
  where DF.File#=FS.File#
  order by Drive, File_Name desc;
```

El siguiente listado muestra el resultado de esta consulta de ejemplo.

DRIVE	FILE_NAME	TOTAL_IOS	BLOCKS_READ	BLOCKS_WRITTEN
/db01	/db01/oracle/CC1/sys.dbf	29551	27708	1843
	/db01/oracle/CC1/temp.dbf	4389	4	4385
*********				
sum		33940	27712	6228
/db02	/db02/oracle/CC1/rbs01.dbf	1134	3	1131
	/db02/oracle/CC1/rbs02.dbf	349		349
	/db02/oracle/CC1/rbs03.dbf	415	7	408
*********				
sum		1898	10	1888
/db03	/db03/oracle/CC1/cc.dbf	57217	56820	397
*********				
sum		57217	56820	397
/db04	/db04/oracle/CC1/ccindx.dbf	15759	14728	1031
	/db04/oracle/CC1/tests1 .dbf			
*********				
sum		15759	14728	1031
*********				
Sum		108814	99270	9544

Los datos del listado anterior muestran el formato de salida de la consulta. El listado de salida muestra que el dispositivo denominado /db03 es el más utilizado durante el funcionamiento de la base de datos. Sólo hay un archivo de datos de esta base de datos en el dispositivo /db03. El informe muestra también que los accesos al archivo de /db03 se corresponden con una utilización intensiva de lectura, con una diferencia abrumadora.

El segundo dispositivo más activo es /db01, el cual contiene dos archivos de la base de datos. La mayor parte de la actividad de ese disco se produce en el archivo del espacio de tablas SYSTEM, frente al espacio de tablas TEMP que requiere mucha menos actividad de E/S.

Los datos de ejemplo mostrados en el listado anterior deben compararse con los datos generados mediante los comandos de supervisión del sistema operativo, como los listados sar mostrados anteriormente en este mismo capítulo. Si la carga de E/S de la base de datos no está originando esperas y deja margen para que la aplicación crezca, el diseño será satisfactorio. Si el sistema operativo muestra que se están produciendo esperas, se puede utilizar la consulta de V$FILESTAT para determinar a qué archivos se accede con más frecuencia.

¿A qué tablas se accede con más frecuencia? El modo más sencillo de responder a esta pregunta consiste en consultar la vista V$SQL para ver las consultas que se están ejecutando. Examine las consultas que tengan los valores más altos de número de ejecuciones (executions), de lecturas de disco (disk reads) y de extracciones de buffer (buffer gets) para determinar cuáles son las tablas que, con mayor probabilidad, estén originando la actividad de E/S que se observa.

```
select Buffer_Gets,
       Disk_Reads,
       Executions,
       Buffer_Gets/Executions  B_E,
       SQL_Text
  from V$SQL
  order by Disk_Reads  desc;
```

Para determinar las estadísticas de E/S de un archivo durante un período de tiempo concreto, puede servirse de la utilidad STATSPACK. El informe estándar STATSPACK genera un listado de las operaciones SQL que han causado más actividad de E/S durante ese período de tiempo. En Oracle9i el informe sobre SQL costoso (expensive SQL) proporciona información sobre el número de extracciones de buffer, de ejecuciones, el número de extracciones de buffer por ejecución y el tiempo de procesamiento de cada instrucción SQL incluida en dicho informe. Este informe muestra también el porcentaje de todas las extracciones de buffer de la base de datos que se atribuyen a cada instrucción SQL. Los datos del informe STATSPACK se pueden utilizar para identificar rápidamente aquellas consultas que generan cargas de E/S significativas en la base de datos.

El informe STATSPACK muestra también la actividad de E/S por espacio de tablas registrada durante el período de supervisión. Los datos se presentan ordenados en función de la actividad del espacio de tablas y ordenados por espacio de tablas y archivo. Puede

utilizar esta parte del informe para supervisar la E/S de los archivos durante las ejecuciones por lotes, las acciones de administración o cualquier otro tipo de actividad planificada.

4.3 Soluciones

Puede utilizar las directrices presentadas en este capítulo para diseñar una configuración física para la configuración lógica que ha diseñado previamente en el Capítulo 3. En la sección <<Soluciones>> del Capítulo 3, se han descrito cuatro configuraciones lógicas comunes: una base de datos de desarrollo pequeña, una base de datos OLTP de producción, una base de datos OLTP de producción con datos históricos y un almacén de datos. En esta sección, se presentan los diseños físicos comunes para estos cuatros tipos de bases de datos.

4.3.1 Diseño de una base de datos de desarrollo pequeña

En una base de datos pequeña, habrá, normalmente, los siguientes espacios de tablas: SYSTEM, DATA, INDEXES, RBS, TEMP, USERS y TOOLS (véase la Tabla 3.2). Para guardarlos en dispositivos separados (siempre y cuando el espacio de tablas DATA sea lo suficientemente pequeño), sólo se necesitan siete discos. Si dispone de menos de siete discos, siga las recomendaciones expuestas en la sección <<Recomendaciones>> de este capítulo. Aisle, tanto como sea posible, los archivos de datos DATA, INDEXES, SYSTEM y RBS/deshacer cambios del resto de archivos. Por ejemplo, en un sistema que conste de cinco discos, se podrían colocar todos los archivos de datos que no sean DATA, INDEXES, SYSTEM y RBS en un disco. En interés de la recuperabilidad, todavía debería distribuir los archivos de control entre varios discos.

Si el sistema utiliza tecnología RAID, como, por ejemplo, RAID-5, una base de datos de desarrollo pequeña podría tener la totalidad de sus archivos en la misma matriz de discos RAID. La base de datos se encontrará disponible a menos que se produzcan simultáneamente fallos en varios discos de la matriz de discos.

4.3.2 Diseño de base de datos OLTP de producción

Una base de datos OLTP de producción admite muchas transacciones pequeñas. Una base de datos OLTP suele contener los siguientes espacios de tablas: SYSTEM, DATA, DATA_2, INDEXES, INDEXES_2, RBS (o un espacio de tablas de tipo deshacer cambios), TEMP (y, potencialmente, otros espacios de tablas temporales) y TOOLS (consulte la Tabla 3.2).

En una base de datos OLTP de producción, los espacios de tablas DATA e INDEXES suelen crecer rápidamente, sobrepasando, generalmente, el tamaño de un disco cada uno de ellos; mientras que el resto de espacios de tablas crecen a un ritmo muy inferior. Las tablas y los índices de las aplicaciones suelen dividirse en particiones a fin de facilitar la gestión de los espacios de tablas cuando el volumen de los datos aumenta. Si no se accede simultáneamente a las particiones con frecuencia, no se originarán contiendas al almacenar las particiones del espacio DATA en el mismo dispositivo de disco o en la misma matriz de discos RAID.

En un sistema OLTP de producción, la configuración se diseña persiguiendo la optimización del rendimiento de las transacciones y las consultas pequeñas. Los espacios de tablas TEMP y TOOLS rara vez se utilizan. El diseño debería poner énfasis en dar soporte a las zonas de mayor actividad de los espacios de tablas DATA e INDEXES, utilizando particiones para aislar las partes más activas de las tablas más activas. También se debería aislar el espacio de tablas SYSTEM, ya que todos los usuarios y comandos de la base de datos generarán consultas del diccionario de datos. La actividad del espacio de tablas RBS/deshacer cambios suele ser inferior a la actividad de SYSTEM. Si se utilizan segmentos de anulación en lugar de un espacio de tablas para deshacer cambios, se han de crear segmentos de anulación suficientes para evitar la contienda por cabeceras de segmentos de anulación (como se puede ver en la vista V$WAITSTAT y mediante el informe STATS-PACK).

Si no dispone de discos suficientes para que cada tipo principal de espacios de tablas posea el suyo,puede almacenar los espacios de tablas más pequeños juntos. Si guarda en caché las tablas estáticas pequeñas, podría incluso agrupar DATA_2 e INDEXES_2 .

Si utiliza matrices de discos RAID, pruebe a aislar los espacios de tablas DATA e INDEXES en matrices diferentes. En un sistema de tres matrices, se pueden colocar los espacios de tablas DATA en una matriz, los INDEXES en la segunda y el resto, en la tercera. Asegúrese de que distribuye los archivos de control entre las distintas matrices.

4.3.3 Base de datos OLTP de producción con datos históricos

Además de los datos actuales del entorno de producción, es probable que tenga que mantener datos históricos. Dado que los modelos de acceso y las pautas de crecimiento de los datos históricos pueden ser diferentes a los de los datos de las transacciones actuales, es conveniente almacenar los datos históricos independientemente del espacio de tablas DATA actual. Como se puede ver en la Tabla 3.2, una base de datos OLTP de producción que incluya datos históricos contará con los mismos espacios de tablas que una base de datos OLTP de producción, además de dos espacios de tablas nuevos: DATA_ARCHIVE e INDEXES_ARCHIVE.

Las áreas de archivado se pueden comparar con particiones separadas de los espacios de tablas DATA e INDEXES, por lo que se pueden seguir las directrices de diseño proporcionadas en la sección anterior.

Cuando se añaden datos históricos a la configuración, es necesario conocer cómo se utilizarán dichos datos históricos. Si se va a hacer referencia constante a los datos antiguos, habrá que separarlos de los espacios de tablas DATA *e* INDEXES actuales. En el caso de que se utilicen activamente los dos espacios de tablas DATA, el antiguo y el actual, se producirían contiendas de E/S si se almacenaran en el mismo dispositivo físico.

Si los datos históricos no se utilizan activamente, se podrán almacenar en los mismos dispositivos que los datos actuales sin que se originen contiendas de E/S. Por ejemplo, si los datos históricos sólo se utilizan para informes o procesos por lotes que rara vez se ejecutan, las operaciones de E/S serán escasas, como también lo serán las contiendas de E/S que se produzcan entre los espacios de tablas DATA histórico y actual.

4.3.4 Diseño de almacén de datos

El volumen de datos de un almacén de datos puede ser notablemente mayor que el de cualquiera de las bases de datos OLTP. Como muestra la Tabla 3.2, es muy probable que un almacén de datos contenga todos los tipos de segmentos utilizados en Oracle: tablas del diccionario de datos (espacio de tablas SYSTEM), tablas (DATA y DATA_2), índices (INDEXES e INDEXES_2), segmentos de anulación grandes y pequeños (RBS y RBS__2 o un espacio de tablas de tipo deshacer cambios), segmentos temporales (TEMP y TEMP_USUARIO) y tablas de herramientas (TOOLS). Además de todos los segmentos, un almacén de datos también incluirá, por lo general, particiones (espacios de tablas PARTITIONS y PARTITIONS_I), datos agregados (AGG_DATA y AGG_DATA_I), vistas materializadas (MVIEWS y MVIEWS_I) y tablas de trabajo que se utilizan únicamente durante el procesamiento por lotes (TEMP_WORK y TEMP_WORK_I). Para elaborar un diseño físico para todos estos espacios de tablas, es necesario tener en cuenta dos formas completamente diferentes de utilizar los almacenes de datos: la carga y la recuperación de datos.

En una base de datos OLTP, un gran número de usuarios llevan a cabo el proceso efectivo de carga de datos ejecutando pequeñas transacciones de inserción (**insert**) y actualización (**update**). En un almacén de datos, el proceso de carga de datos consiste normalmente en un conjunto de operaciones por lotes de gran tamaño que puede necesitar días o semanas para completarse durante cada ciclo de carga de datos. Por tanto, hay que ajustar la base de datos para optimizar el proceso de carga de datos por lotes. Al mismo tiempo, hay que pensar en cómo recuperarán los usuarios los datos una vez que hayan sido cargados.

Como ocurre con una base de datos OLTP, los usuarios del almacén de datos realizan numerosas consultas pequeñas en un gran número de tablas. A pesar de que las tablas base de un almacén de datos son grandes, es probable que los usuarios finales no las consulten directamente, sino que, en su lugar, consulten tablas de agregación muy indexadas. Es recomendable desnormalizar los datos del almacén de datos con el fin de facilitar las rutas de acceso que los usuarios finales sigan habitualmente.

Al planificar el diseño lógico del almacén de datos, hay que pensar en los procesos de carga de datos por lotes y en los procesos de recuperación de datos por separado. En el caso de un proceso de carga por lotes para el que no se puedan utilizar espacios de tablas transportables o tablas externas, los principales espacios de tablas involucrados serán los siguientes:

SYSTEM	Tablas del diccionario de datos
TEMP_WORK	Tablas temporales utilizadas durante el proceso de carga de datos
TEMP_WORK_I	índices para tablas temporales de trabajo
TEMP_USUARIO	Segmentos temporales grandes para las ordenaciones por lotes

espacio de tablas TEMP para los segmentos temporales asociados durante la creación del índice.

6. Se crean tablas de agregación en AGG_DATA y se indexan estas tablas en AGG_ DATA_I utilizando el espacio de tablas RBS_2 para segmentos de anulación; se utiliza el espacio de tablas TEMP para los segmentos temporales asociados durante la creación del índice.

Antes de ubicar dos o más espacios de tablas en el mismo dispositivo físico, hay que considerar los requisitos de acceso a los datos de los usuarios finales. En lugar de limitarse a buscar en los espacios de tablas DATA e INDEXES durante las consultas, es probable que los usuarios también accedan a AGG_DATA, AGG_ DATA_I, PARTITIONS, PARTITIONS_I, MVIEWS y MVIEWS_I. Si las solicitudes de E/S exceden la capacidad de los dispositivos, habrá que separar todos estos espacios de tablas.

Como se ha mencionado anteriormente, dar soporte a un almacén de datos requiere el uso de muchos tipos de espacios de tablas, si bien es cierto que no todos se utilizan simultáneamente. Por ejemplo, los espacios de tablas TEMP_WORK y TEMP_WORK_I sólo se utilizan durante el proceso de carga de datos. Por tanto, podrá ubicarlos con otros espacios de tablas (como pueden ser MVIEWS y MVIEWS_I) para reducir el número de discos utilizados. Además, se puede optar por dividir en particiones todas las tablas que no sean estáticas, con lo que se elimina efectivamente la necesidad de los espacios de tablas DATA e INDEXES. El diseño final debería reflejar las características de uso de los archivos de la aplicación. En un almacén de datos ideal, los espacios de tablas que se utilizan con más frecuencia son AGG_DATA_I, MVIEWS_I y PARTITIONS_I, mientras que rara vez se accede a las tablas asociadas. Si se lograra indexar las tablas del almacén de datos de forma tan eficaz que apenas fuera necesario acceder a ellas, se podrían colocar los espacios de tablas AGG_DATA y PARTITIONS en la misma ubicación sin que se produzcan contiendas de E/S.

4.4 Ubicación de los archivos

Para simplificar la administración de la base de datos, sería aconsejable guardar los archivos asociados a ella en directorios que se hayan creado específicamente para la base de datos. No es aconsejable almacenar juntos archivos de bases de datos diferentes.

Además, se deberían separar los archivos de datos de la base de datos del software utilizado para acceder a la base de datos (a pesar de que ésta sea la opción predeterminada en algunos programas de instalación). No se debe permitir que las versiones activas del software de Oracle provoquen contiendas con los archivos de datos.

Al guardar los archivos de datos en el mismo nivel de una jerarquía de directorios simplificamos los procedimientos de administración. De esta forma también se evita tener que incluir el identificador de instancia en el nombre de archivo, usándolo, en lugar de eso, como parte de la ruta de directorio. Esta separación permite emplear caracteres comodín o listas de búsqueda cuando se hace referencia a ellos (si los discos se han denominado de manera coherente). Por ejemplo, en entornos UNIX, todos los archivos que pertenecieran a una instancia concreta se podrían copiar en un dispositivo de cinta con un solo comando, de esta forma:

```
> tar /dev/rmt/lhc /db0[1-8]/oracle/CASE
```

En este ejemplo, el sistema escribirá en la unidad de cinta (/dev/rmt/lhc) el contenido del subdirectorio /oracle/CASE de los dispositivos denominados /db0l hasta /db08.

4.5 Panorámica del uso del espacio en la base de datos

Para comprender cómo se debería asignar el espacio dentro de la base de datos, antes de nada hay que saber cómo se utiliza el espacio dentro de la propia base de datos. En esta sección, vamos a ofrecer una panorámica sobre el uso del espacio en una base de datos Oracle.

Al crear una base de datos, ésta se divide en numerosas secciones lógicas denominadas *espacios de tablas.* El espacio de tablas SYSTEM es el primero que se crea y, a continuación, se crean otros en los que se guardan los diferentes tipos de datos, tal y como ya explicamos en el Capítulo 3.

Al crear un espacio de tablas, también se crean *archivos de datos* para guardar los datos correspondientes. Estos archivos asignan inmediatamente el espacio que se ha especificado durante su creación. Por tanto, existe una relación uno a muchos entre las bases de datos y los espacios de tablas, y también hay una relación uno a muchos entre los espacios de tablas y los archivos de datos. Se pueden añadir nuevos archivos de datos a un espacio de tablas y los archivos de datos existentes se pueden ampliar.

Cada base de datos puede tener numerosos usuarios, cada uno de los cuales posee un *esquema.* Cada esquema de usuario es una colección de objetos lógicos de la base de datos, como tablas e índices. Estos objetos hacen referencia a estructuras físicas de datos (las tablas y los índices físicos) que se almacenan en espacios de tablas. Los objetos del esquema de un usuario pueden guardarse en múltiples espacios de tablas y un solo espacio de tablas puede contener objetos de múltiples esquemas.

Cuando se crea un objeto de la base de datos (por ejemplo, una tabla o un índice), se asigna a un espacio de tablas mediante las opciones predeterminadas del usuario o mediante instrucciones específicas. En ese espacio de tablas se crea un *segmento,* que es el que contiene los datos asociados a dicho objeto. El espacio que se asigna al segmento no se libera hasta que el segmento se elimina, se contrae o se trunca **(truncate).** En la sección <<Cómo desasignar espacio de los segmentos>>, incluida más adelante, encontrará información más detallada sobre cómo reducir manualmente el espacio asignado a las tablas, los índices y los clusters.

Cada segmento consta de una serie de secciones denominadas *extensiones,* es decir, conjuntos de bloques de Oracle contiguos. Cuando las extensiones existentes ya no pueden contener nuevos datos, el segmento tendrá que obtener una nueva extensión para dar soporte a nuevas operaciones de inserción **(insert)** de datos en el objeto. Este proceso de ampliación continúa hasta que no hay más espacio disponible en los archivos de datos del espacio de tablas o hasta que se alcance el número máximo interno de extensiones por segmento. Cuando se llena un archivo de datos, éste se puede extender ciñéndose a las reglas de almacenamiento que se hayan definido para él.

Entre los tipos de segmentos disponibles en Oracle se encuentran los siguientes:

- TABLE
- INDEX
- ROLLBACK
- TEMPORARY
- PARTITION
- CLUSTER

La administración del espacio que utiliza cada uno de estos segmentos es una de las funciones básicas del DBA. El propósito de este tema es el de ayudar a planificar el almacenamiento físico.

4.5.1 Implicaciones de la cláusula storage

La cantidad de espacio que utiliza un segmento está determinada por sus parámetros de almacenamiento. Estos parámetros los determina la base de datos en el momento de crear el segmento y se pueden modificar posteriormente; si no se especifica ningún parámetro de almacenamiento (storage) en el comando **create table, create index, create cluster o create rollback segment,** la base de datos utilizará los parámetros de almacenamiento predeterminados del espacio de tablas en el cual se va a guardar. Los parámetros de almacenamiento especifican el tamaño inicial de la extensión **(initial),** el tamaño de la siguiente extensión (next), el valor **pctincrease** (un factor por el cual cada extensión subsiguiente aumentará el tamaño del segmento de forma geométrica), el valor **maxextents** (el número máximo de extensiones) y el valor **minextents** (el número mínimo de extensiones). Una vez creado el segmento, no se pueden modificar los valores **initial y minextents.** Los valores predeterminados de los parámetros de almacenamiento de cada espacio de tablas se encuentran en las vistas DBA_TABLESPACES y USER_TABLESPACES.

Al crear un segmento, éste adquirirá al menos, una extensión. La extensión inicial se utilizará para almacenar los datos hasta que ya no haya más espacio libre disponible. Se puede utilizar la cláusula **pctfree** para reservar, en el interior de cada bloque de cada extensión, un porcentaje de espacio, que se mantendrá disponible para las actualizaciones **(updates)** de las filas existentes. Cuando se añadan datos adicionales al segmento, éste se ampliará obteniendo una segunda extensión del tamaño que se especifique en el parámetro **next.**

El parámetro **pctincrease** está diseñado para minimizar el número de extensiones de las tablas en crecimiento. Si el valor de este parámetro fuera diferente de cero, el tamaño de cada extensión subsiguiente aumentaría geométricamente el tamaño del segmento en función del factor **pctincrease** especificado. Cuando se introdujo **pctincrease,** este factor ayudaba a reducir el número de extensiones por segmento. Cuando se tiene un número ilimitado de extensiones por segmento, ya no tiene sentido utilizar ningún valor de **pctincrease** que no sea 0, a menos que el volumen de datos de la tabla sea muy diferente del volumen para el que se pensó al diseñarla.

El factor pctincrease no se puede utilizar con espacios de tablas gestionados manualmente.

4.5.2 Espacios de tablas gestionados localmente

Existen dos métodos para determinar el espacio libre y el espacio utilizado de un espacio de tablas. El primer método, disponible desde que se introdujeron los espacios de tablas, consiste en gestionar las extensiones mediante el diccionario de datos (es decir, espacios de tablas gestionados por el diccionario). En este tipo de espacios de tablas, cada vez que se asigna una extensión o se libera para volver a utilizarla en un espacio de tablas, se actualiza **(update)** la entrada correspondiente en la tabla del diccionario de datos.

El segundo método, disponible a partir de la versión Oracle8i, gestiona las extensiones dentro de los propios espacios de tablas (espacios de tablas gestionados localmente). En estos espacios de tablas, el espacio de tablas gestiona su propio espacio manteniendo un mapa de bits en cada archivo de datos de los bloques o conjuntos de bloques liberados o usados del archivo de datos. Cada vez que se asigna una extensión o se libera para volver a utilizarla, Oracle actualiza **(update)** el mapa de bits para reflejar el nuevo estado. En un espacio de tablas gestionado localmente, se puede especificar el mismo tamaño para todas las extensiones o se puede hacer que el sistema determine automáticamente el tamaño de las extensiones.

Para crear un espacio de tablas gestionado localmente, especifique la opción **local** para la cláusula **extent management** del comando **create tablespace.** A continuación, mostramos un ejemplo de cómo se declara un espacio de tablas gestionado localmente con el comando **create tablespace:**

```
create tablespace CODES_TABLES
datafile '/u01/oracle/VLDB/codes_tables.dbf'
size 10M
extent management local uniform size 256K;
```

Suponiendo que el tamaño del bloque para la base de datos en la que se crea este espacio de tablas es de 4 KB, en este ejemplo, se define la opción **local** para el tipo de gestión de extensiones y se determina un tamaño uniforme de 256 KB para el espacio de tablas. Cada bit del mapa de bits describe 64 bloques (256/4). En caso de que se omita la cláusula **uniform size,** la opción predeterminada será **autoallocate.** El tamaño (size) predeterminado para la opción uniforme **(uniform)** es de 1MB.

Si se especifica la opción **local** en el comando **create tablespace,** no se puede especificar la cláusula **default storage, minextents, ni temporary.** Si se utiliza el comando **create temporary tablespace** para crear el espacio de tablas, se puede especificar **extent management local.**

Los espacios de tablas gestionados localmente pueden ocuparse de parte de las tareas de gestión del espacio que llevan a cabo los DBA en los espacios de tablas gestionados por el diccionario. Gracias a su arquitectura, la fragmentación de los espacios de tablas es menos probable, como es también poco probable que aparezcan problemas relacionados con el espacio ocupado por sus objetos.

En Oracle8i o en Oracle9i, cuando se crea la base de datos, el espacio de tablas SYSTEM no se puede definir como un espacio de tablas gestionado localmente, ni tampoco se puede convertir posteriormente.

Se puede crear un conjunto de espacios de tablas gestionados localmente con un número reducido de parámetros de almacenamiento y solventar la mayor parte de las solicitudes de espacio de la base de datos. Por ejemplo, se podrían crear tres espacios de tablas DATA como se muestra a continuación:

```
createtablespace DATA_SMALL
datafile '/u01/oracle/VLDB/codes_tables.dbf'
size 10M
extent management local uniform size 1M;

createtablespace DATA_MEDIUM
datafile '/u01/oracle/VLDB/codes_tables.dbf'
size 100M
extent management local uniform size 4M;

createtablespace DATA_LARGE
datafile '/u01/oracle/VLDB/codes_tables.dbf'
size 1000M
extent management local uniform size 16M;
```

En este ejemplo, el espacio de tablas DATA_SMALL crea objetos con extensiones de 1 MB; DATA_MEDIUM utiliza extensiones de 4 MB y DATA_LARGE usa extensiones de 16 MB. Una tabla pequeña se ubicaría en el espacio de tablas DATA_SMALL y en caso de que creciera notablemente podríamos trasladarla al espacio de tablas DATA_MEDIUM o a DATA_LARGE.

Todos los parámetros de cada uno de estos espacios de tablas de ejemplo poseen los mismos parámetros de almacenamiento, con lo que se simplifica cualquier actividad de mantenimiento que se lleve a cabo. Utilizando tamaños coherentes de extensión también se mejora la reutilización del espacio después de haber eliminado o cambiado de lugar un objeto del espacio de tablas.

En las secciones siguientes se describe cómo se administra el espacio de cada uno de los principales tipos de objetos. Observe que cuando se utilizan los espacios de tablas gestionados localmente, muchos de los aspectos de la cláusula de almacenamiento **(storage)** no se pueden emplear. Se utiliza el espacio de tablas para gestionar el espacio en función de los parámetros que se hayan definido en el nivel del espacio de tablas. A pesar de que esto puede ocasionar que la asignación de espacio sea excesiva, simplificará enormemente las operaciones de gestión del espacio en la base de datos.

4.5.3 Segmentos de tablas

Los *segmentos de tabla,* también denominados *segmentos de datos,* son los que guardan las filas de datos asociadas a las tablas o clusters. Cada segmento de datos contiene un bloque de cabecera, que sirve como directorio del espacio ocupado por el segmento.

Una vez que el segmento de datos ha adquirido una extensión, la mantiene hasta que dicho segmento se elimina o se trunca **(truncate).** Eliminar filas de una tabla mediante el comando **delete** no tiene ningún efecto sobre la cantidad de espacio que se ha asignado a esa tabla. El número de extensiones aumentará hasta que (1) se alcance el valor **maxextents,** si se ha definido, (2) se alcance la cuota del usuario en el espacio de tablas o (3) el espacio de tablas se quede sin espacio en el caso de que los archivos de datos no se puedan ampliar automáticamente. Si desea permitir que un archivo de datos se amplíe automáticamente, consulte la sección <<Automatización de las ampliaciones de los archivos de datos>> .

Para minimizar la cantidad de espacio que se desperdicia en un segmento de datos, ajuste el valor del parámetro **pctfree.** Este parámetro especifica la cantidad de espacio que se mantendrá libre en cada bloque de datos; dicho espacio libre se puede utilizar cuando las columnas que contienen valores NULL se actualizan con valores no nulos, o cuando las actualizaciones de otros valores de una fila hagan que ésta tenga que crecer. El valor correcto de **pctfree** es específico de cada aplicación, ya que depende de la naturaleza de las actualizaciones que se estén llevando a cabo. Si quiere obtener más información sobre la configuración de éste y otros parámetros de almacenamiento de tablas e índices.

Se puede utilizar el comando **alter table** para modificar la mayor parte de los parámetros de almacenamiento de una tabla existente. La opción **move** del comando **alter table** sirve para cambiar el espacio de tablas al que una tabla está asignada.

Al igual que los segmentos de tabla, los *segmentos de índice* conservan el espacio que se les ha asignado hasta que se eliminan; no obstante, también se pueden eliminar indirectamente al eliminar las tablas o los clusters a los que indexan. Para minimizar las contiendas, sería aconsejable guardar los índices en un espacio de tablas que estuviera separado de las tablas asociadas a ellos.

Los segmentos de índice de los espacios de tablas gestionados por el diccionario disponen de cláusulas storage, que especifican sus valores **initial, next, minextents, maxextents y pctincrease** y tienen la misma probabilidad de fragmentarse que las tablas.

Se puede utilizar la opción **rebuild** del comando **alter index** para modificar la configuración de **storage y tablespace** de un índice. Por ejemplo, si se crea un índice con una extensión inicial **(initial)** muy grande, se podría recuperar el espacio de esa extensión reconstruyendo el índice y especificando un nuevo valor para **initial,** como muestra el siguiente ejemplo. En este listado, se vuelve a crear el índice JOB_PK asignando esta vez un valor **initial** de 10 MB para la extensión.

```
alter index JOB_PK rebuild
tablespace INDEXES
storage (initial 10M next 10M pctincrease 0);
```

Durante el proceso de reconstrucción **(rebuild),** ambos índices, el nuevo y el antiguo, se ubicarán en la base de datos. Por tanto, se necesitará espacio disponible suficiente para almacenar ambos índices antes de ejecutar el comando **alter index rebuild.**

Mientras se crea el índice, el optimizador puede recopilar datos estadísticos sobre el contenido del índice. Para ello, se utiliza la cláusula **compute statistics** del comando **create index y alter index rebuild.** Combinar estas dos operaciones lleva menos tiempo que separar los comandos de creación y de recopilación de datos.

4.5.5 Segmentos de anulación

Los principios que rigen el diseño adecuado de las tablas también se aplican a los segmentos de anulación. Los segmentos de anulación tendrán múltiples extensiones del mismo tamaño, que se irán añadiendo hasta completar su tamaño total óptimo (al crearse, tendrán un mínimo de dos extensiones).

El tamaño de los segmentos de anulación se puede reducir dinámicamente hasta un tamaño específico, o se puede reducir manualmente hasta el tamaño que se elija. La cláusula **optimal** permite reducir el tamaño de los segmentos de anulación hasta un tamaño óptimo después de haber sido ampliados. Esto sirve para dar un soporte provisional a los sistemas que no se han implementado adecuadamente para la forma en que se están utilizando. Si las reducciones de tamaño son frecuentes, eso significa que hay que volver a diseñar los segmentos de anulación.

En el caso de las operaciones de deshacer cambios gestionadas por el sistema, Oracle asigna y desasigna automáticamente segmentos de tipo de deshacer cambios según convenga

4.5.6 Segmentos temporales

Los *segmentos temporales* sirven para guardar los datos temporales durante las operaciones de ordenación (por ejemplo, las consultas de gran tamaño, las creaciones de índices y las operaciones de **unión).** Cada usuario tiene un espacio de tablas temporal, que se especifica al crear la cuenta mediante el comando **create user** o se modifica por medio del comando **alter user.** El espacio de tablas temporal del usuario debe ser un espacio de tablas que haya sido designado como espacio de tablas temporal; a partir de la versión Oracle9i los espacios de tablas permanentes no se pueden utilizar para este fin.

Cuando se crea una base de datos, se puede especificar un espacio de tablas temporal predeterminado para todos los usuarios. En una base de datos existente, se puede cambiar el espacio de tablas predeterminado con el comando **alter database,** como se puede observar en el listado siguiente:

```
alter database default temporary tablespace TEMP;
```

Cuando se modifica la asignación del espacio de tablas predeterminado de una base de datos, a los usuarios que hayan estado utilizando el espacio de tablas temporal predeterminado antiguo se les redirige hacia la nueva ubicación predeterminada. Para ver el espacio de tablas temporal predeterminado actual, ejecute la siguiente consulta:

```
select Property_Value
from DATABASE_PROPERTIES
where Property_Name = 'DEFAULT_TEMP_TABLESPACE';
```

Puede especificar un espacio de tablas como espacio de tablas <<temporal>> utilizando el comando **create temporary tablespace.** Un espacio de tablas temporal no se puede utilizar para guardar ningún segmento permanente, sino sólo los segmentos temporales que se crean durante las consultas. La primera operación de ordenación que utilice el espacio de tablas temporal reservará un segmento temporal del espacio de tablas temporal; una vez que la consulta concluya, el espacio que ha utilizado el segmento temporal no se elimina. En su lugar, el espacio que ha utilizado queda disponible para que lo utilicen otras consultas; gracias a esto, la operación de ordenación puede evitar los costes derivados de la reserva y liberación de espacio de los segmentos temporales. Si su aplicación utiliza con frecuencia segmentos temporales para las operaciones de ordenación, el proceso de ordenación debería tener un mejor rendimiento si se utilizase un espacio de tablas temporal dedicado.

Para convertir un espacio de tablas en un espacio dedicado para segmentos temporales, hay que especificar la cláusula **temporary** en los comandos **create tablespace o alter tablespace,** como muestra el siguiente listado:

```
alter tablespace TEMP temporary;
```

Para hacer que el espacio de tablas TEMP almacene objetos permanentes (es decir, no temporales) utilice la cláusula **permanent** de los comandos **create tablespace o alter tablespace,** como muestra el siguiente listado:

```
alter tablespace TEMP permanent;
```

La columna Content de la vista del diccionario de datos DBA_TABLESPACES muestra el estado del espacio de tablas como 'TEMPORARY', temporal, o 'PERMANENT', permanente.

4.5.7 Espacio libre

Una *extensión libre* de un espacio de tablas es un conjunto de bloques libres contiguos en dicho espacio de tablas. Un espacio de tablas puede contener numerosas extensiones de datos y extensiones libres (véase la Figura 4.1a). Cuando se elimina un segmento, sus extensiones se desasignan y se marcan como libres. En los espacios de tablas gestionados por el diccionario, estas extensiones libres no siempre se vuelven a combinar con otras extensiones libres vecinas, porque puede que las barreras entre dichas extensiones libres se mantengan (véase la Figura 4. Ib). El proceso de segundo plano SMON agrupa de forma periódica las extensiones libres vecinas (véase la Figura 4. le), siempre y cuando el valor predeterminado de **pctincrease** para el espacio de tablas sea distinto de cero.

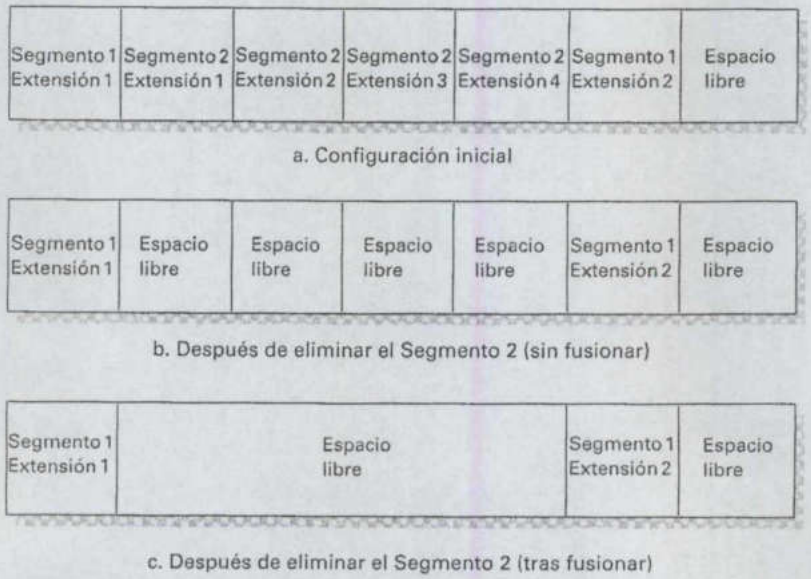


Figura 4.1. Gestión de las extensiones libres en espacios de tablas gestionados por el diccionario.

El proceso en segundo plano SMON sólo agrupa los espacios de tablas cuyo valor predeterminado para **pctincrease** es distinto de cero. Asignando a **pctincrease** el valor 1, obligamos a SMON a agrupar el espacio libre adyacente de un espacio de tablas, pero únicamente agrupará ocho áreas de extensiones adyacentes cada vez. Cada una de las ocho áreas agrupará dos o más extensiones creando así una extensión más grande. Para conseguir una mejor gestión de espacio, utilice los espacios de tablas gestionados localmente; de este modo nunca tendrá que preocuparse de las operaciones de agrupación. Si se ve obligado a utilizar espacios de tablas gestionados por el diccionario, mantenga el parámetro **pctincrease** en 0 y, de vez en cuando, agrupe manualmente las extensiones que haya eliminado.

Para hacer que el espacio de tablas agrupe el espacio libre, utilice la cláusula **coalesce** del comando **alter tablespace,** como se muestra a continuación:

```
alter tablespace DATA coalesce;
```

Con el comando anterior hará que las extensiones libres vecinas del espacio de tablas DATA se agrupen en extensiones libres de mayor tamaño. Al igual que SMON, este comando agrupa hasta ocho áreas separadas.

El comando **alter tablespace** no agrupa extensiones libres que estén separadas por extensiones de datos.

Los espacios de tablas gestionados localmente no requieren el mismo grado de gestión del espacio libre. Dado que este tipo de espacios de tablas se pueden configurar para que los tamaños de las extensiones sean iguales, las extensiones eliminadas se pueden volver a utilizar fácilmente.

Cuando se asigna una nueva extensión, Oracle no combina las extensiones libres contiguas a menos que no haya otra alternativa. Como resultado, en las tablas gestionadas por el diccionario, se suele utilizar la extensión libre de mayor tamaño, ubicada lo más cerca posible del final del espacio de tablas, mientras que las extensiones libres más pequeñas, que se encuentran al principio, quedan relativamente inutilizadas. Las extensiones libres más pequeñas se convierten en «huecos>> del espacio de tablas, porque no son, por sí mismas, del tamaño adecuado para ser de utilidad. A medida que este patrón de uso progresa, aumenta la cantidad de espacio desperdiciado en el espacio de tablas.

En una base de datos ideal, todos los objetos tendrían el tamaño adecuado y el espacio libre se guardaría siempre de forma contigua, lo que constituiría un área de recursos que estaría siempre esperando a ser utilizada. Si se puede evitar la asignación dinámica de espacio durante el uso de la aplicación, se suprime el impacto en el rendimiento y una fuente de posibles errores de la aplicación.

4.6 Redímensionamíento de los archivos de datos

Los archivos de datos existentes se pueden redimensionar utilizando los comandos **alter database y alter tablespace.** Se pueden especificar los valores de los parámetros de extensión de almacenamiento para cada archivo de datos que haya en una base de datos; Oracle utilizará esos valores cuando amplíe automáticamente el archivo de datos. Los archivos de datos también se pueden ampliar y reducir de tamaño manualmente. Para ampliar manualmente un archivo de datos, hay que utilizar el comando **alter database,** como se muestra en el siguiente ejemplo:

```
alter database
datafile '/db05/oracle/CC1/data01.dbf' resize 200M;
```

Tras ejecutar el comando **alter database** mostrado en el ejemplo anterior, el archivo especificado adquirirá un tamaño de 200 MB. Si el archivo ya tuviera un tamaño superior a 200 MB, se reduciría a 200 MB siempre que fuera posible. Cuando la base de datos reduce el tamaño del archivo, lo hace comenzando desde el final. Por tanto, si hubiera cualquier segmento almacenado en la parte final del archivo, la base de datos no podría reducir el archivo. Si hubiera alguna extensión almacenada en una parte posterior al punto especificado (en este ejemplo, 200 MB), el comando **alter database** del ejemplo anterior no podría ejecutarse.

4.6.1 Automatización de las ampliaciones de los archivos de datos

Cuando se crean archivos de datos, se pueden especificar parámetros que permitan a Oracle ampliarlos automáticamente. Con ello, los archivos de datos se podrían ampliar automáticamente siempre que se supere el tamaño que tengan asignado actualmente. En cada archivo de datos se pueden especificar tres parámetros de dimensionamiento:

autoextend	Es un indicador, que puede estar activado (ON) o desactivado (OFF), para señalar si se debe permitir que el archivo se amplíe

	automáticamente. Si su valor es OFF, el resto de los parámetros de dimensionamiento tomarán el valor cero
next *tamaño*	Es el tamaño, en bytes, del área de espacio de disco que se va a asignar al archivo de datos, cuando se necesite más espacio. El valor de *tamaño* se puede calificar con una 'K' o una 'M', para indicar kilobytes o megabytes, respectivamente
maxsize *tamaño*	Es el tamaño máximo, en bytes, hasta el cual se debería permitir ampliar el archivo de datos. En caso de que no se desee limitar el tamaño del archivo de datos en Oracle, se puede especificar el valor **unlimited.** El valor de *tamaño* se puede calificar con una 'K' o una 'M', para indicar kilobytes o megabytes, respectivamente

Si se especifica **maxsize unlimited,** el tamaño máximo del archivo de datos quedará limitado por el espacio disponible en el disco donde reside el archivo y por el tamaño máximo del archivo que permita el sistema operativo. El tamaño del archivo también podría limitarse mediante una cuota de tamaño de archivo definida en el nivel del sistema operativo para el identificador de usuario <<oracle>>.

Se pueden especificar los parámetros **autoextend, next y maxsize** para un archivo de datos utilizando los comandos **create database, create tablespace, alter database y alter tablespace.** En el ejemplo siguiente, el comando **create tablespace** crea un archivo de datos que se ampliará automáticamente en función de las necesidades:

```
create tablespace DATA
datafile '/db05/oracle/CCl/data01.dbf' size 200M
autoextend ON
next 10M
maxsize 250M;
```

El espacio de tablas que se ha creado en este ejemplo contendrá un solo archivo de datos con una extensión inicial de 200 MB. Cuando dicho archivo de datos se llene, y los objetos contenidos en él requieran espacio adicional, el archivo de datos se ampliará automáticamente en incrementos de 10 MB. El proceso de ampliación continuará según sea preciso hasta que el archivo llegue hasta los 250 MB, en cuyo momento habrá alcanzado su tamaño máximo.

Se puede añadir un nuevo archivo de datos, utilizando el comando **alter tablespace o alter database,** para activar las capacidades **autoextend** del espacio de tablas. El comando del siguiente listado se encarga de añadir un nuevo archivo de datos al espacio de tablas DATA, especificando el valor **autoextend on y maxsize** 300 MB.

```
alter tablespace DATA
add datafile  '/db05/oracle/CCl/data02.dbf'
size  50M
autoextend ON
maxsize 300M;
```

4.7 Como cambiar de ubicación a los archivos de la base de datos

Una vez creado un archivo en una base de datos, puede que sea necesario trasladarlo a otra ubicación para gestionar mejor su tamaño o sus requisitos de E/S. En las siguientes secciones veremos los procedimientos que hay que seguir para cambiar de ubicación a los archivos de datos, a los archivos de registro de reconstrucción en línea y a los archivos de control. En todos estos procedimientos se utilizan comandos del sistema operativo para cambiar de ubicación a los archivos; los comandos de Oracle sirven principalmente para restablecer los punteros a esos archivos.

4.7.1 Cambio de ubicación de archivos de datos

Existen dos métodos para cambiar de ubicación a los archivos de datos: el comando **alter database** y el comando **alter tablespace.** El primero de estos comandos solamente se aplica a los archivos de datos cuyos espacios de tablas no incluyan el espacio de tablas SYSTEM, segmentos de anulación o segmentos temporales. En cambio, el comando **alter database** funciona para todos los archivos de datos.

4.7.1.1 El método alter database

Al utilizar el método **alter database** para cambiar de ubicación a los archivos de datos, el archivo de datos cambia de ubicación después de cerrar la instancia. Los pasos que hay que seguir son los siguientes:

1. Cierre la instancia.
2. Utilice comandos del sistema operativo para desplazar el archivo de datos.
3. Monte la base de datos (pero no la abra) y utilice el comando **alter database** para cambiar el nombre al archivo en la base de datos.
4. Inicie la instancia.
5. Después de verificar que se ha aplicado el cambio, haga una copia de seguridad del sistema de archivos de la base de datos.

Por ejemplo, después de cerrar la base de datos, podría haber trasladado un archivo de datos a un nuevo disco:

```
> mv/db01/oracle/CC1/data01.dbf    /db02/oracle/CC1
```

Para que la base de datos reconozca la nueva ubicación del archivo, hay que montar la base de datos (en este ejemplo, se llama CC1) e indicarle la nueva ubicación. El comando **alter database** mostrado en el listado siguiente no cambia el nombre al archivo, sino que ya se deberá haber cambiado el nombre o la ubicación del archivo.

```
SQL>  connect    /    as  sysdba;
SQL>  startup mount  CC1;
SQL>  alter database  rename  file
```

```
2>    '/db01/oracle/CCI/data01.dbf' to
3>    '/db02/oracle/CC1/data01.dbf' ;
```

En este momento, se puede abrir la base de datos, comprobar que la operación se ha llevado a cabo correctamente y, a continuación, llevar a cabo una copia de seguridad del sistema de archivos de la base de datos.

Al ejecutar el comando **alter database,** Oracle comprueba si el nombre que se está asignando al archivo ya existe. Si este paso falla, compruebe si el nombre de archivo de destino es correcto.

4.7.1.2 El método alter tablespace

Cuando se utiliza el método **alter tablespace** para trasladar archivos de datos, dichos archivos de datos cambian de ubicación mientras se está ejecutando la instancia. Los pasos que debe seguir y que vamos a detallar en las siguientes secciones, son los siguientes:

1. Ponga fuera de línea el espacio de tablas.
2. Utilice los comandos del sistema operativo para cambiar de ubicación al archivo.
3. Utilice el comando **alter tablespace** para cambiar el nombre al archivo en la base de datos.
4. Ponga de nuevo en línea el espacio de tablas.

Este método sólo puede utilizarse para los espacios de tablas distintos de SYSTEM. Tampoco se puede utilizar en los espacios de tablas que contengan segmentos temporales o segmentos de anulación activos.

Paso 1. Ponga fuera de línea el espacio de tablas. Utilice el comando **alter tablespace** en SQL*Plus u OEM para activar el estado **offline** (fuera de línea) del espacio de tablas, tal y como se muestra en el siguiente ejemplo. Este comando se ejecuta mientras la instancia está en ejecución.

```
SQL> alter tablespace DATA offline;
```

Paso *2. Utilice los comandos del sistema operativo para cambiar de ubicación al archivo.* En UNIX, el comando mv sirve para desplazar archivos a nuevas ubicaciones. El siguiente ejemplo muestra cómo se traslada al archivo 'data01.dbf' desde el dispositivo denominado '/db01' a uno denominado '/db02':

```
> mv /db01/oracle/CC1/data01.dbf /db02/oracle/CC1
```

Debe especificar el nombre completo del archivo, empleando los convenios de denominación propios del sistema operativo.

Paso *3. Utilice el comando alter tablespace para cambiar el nombre al archivo en la base de datos.* En el siguiente ejemplo, se cambia el nombre al archivo de datos

'data01.dbf', que hemos cambiado de ubicación en el Paso 2, dentro de la base de datos. De esta manera, la base de datos podrá acceder a ese archivo. El comando **alter tablespace** que se muestra aquí no cambia el nombre al archivo físico, sino que se habrá debido cambiar previamente el nombre o la ubicación de dicho archivo.

```
SQL> alter tablespace DATA rename datafile
2> '/db01/oracle/CC1/data01.dbf' to
3> '/db02/oracle/CC1/data01.dbf';
```

No desconecte la base de datos después de completado este paso; permanezca conectado a ella y luego continúe con el Paso 4.

Al ejecutar el comando **alter tablespace,** Oracle comprueba si el nombre que se está asignando al archivo ya existe. Si este paso falla, compruebe si el nombre del archivo de destino es correcto.

Paso 4. Ponga de nuevo en línea el espacio de tablas. Utilice el comando **alter tablespace** para volver a poner en línea el espacio de tablas.

```
SQL> alter tablespace DATA online;
```

De este modo, se pondrá en línea el espacio de tablas DATA, y pasará a utilizarse la nueva ubicación para el archivo de datos.

4.7.2 Cambio de ubicación de un archivo de datos con Oracle Enterprise Manager

A pesar de que existen dos métodos para cambiar de ubicación a un archivo de datos de forma interactiva desde el nivel del sistema, sólo hay una forma de llevar a cabo la misma tarea desde Oracle Enterprise Manager (OEM). En la sección siguiente, se proporcionan los pasos que tendrá que seguir en OEM para cambiar un archivo de datos de una ubicación a otra. En las figuras de esta sección, se traslada el archivo de datos del espacio de tablas EXAMPLE del directorio D:\Oracle\Ora90\oradata\mydb9\ al directorio G:\Oracle\oradata\mydb9\.

4.7.2.1 Pasos para cambiar de ubicación un archivo de datos empleando OEM

Para desplazar un archivo de datos de un directorio a otro utilizando OEM, hay que seguir los mismos pasos que se describen en la sección <<El método **alter tablespace>>.** En el ejemplo siguiente, se utilizan las herramientas de OEM en lugar de utilizar SQL*Plus y las operaciones se pueden llevar a cabo desde una consola remota.

4.7.2.2 El método alter tablespace desde OEM

Cuando se emplea el método **alter tablespace** para cambiar de ubicación a un archivo de datos, la operación se lleva a cabo mientras la instancia se encuentra en ejecución. Los pasos necesarios, descritos en las secciones siguientes, son los siguientes:

1. Ponga fuera de línea el espacio de tablas.
2. Desde la herramienta Microsoft Explorer de Windows NT, traslade el archivo de datos desde la carpeta antigua a la nueva.
3. Utilice la pantalla de detalles del archivo de datos para modificarle el nombre.
4. Ponga de nuevo en línea el espacio de tablas.

Este método sólo se puede utilizar con espacios de tablas distintos de SYSTEM. No se puede utilizar para espacios de tablas que contengan segmentos de anulación o segmentos temporales activos.

Paso 1. Ponga fuera de línea el espacio de tablas. Desde la opción Oracle Storage Manager (administrador del almacenamiento de Oracle) de OEM, seleccione el espacio de tablas cuyo archivo de datos desee cambiar de ubicación. Tras seleccionar el espacio de tablas, utilice el menú desplegable, o haga clic con el botón secundario del ratón, y seleccione la opción Take Offline I Normal (poner fuera de línea/normal) para colocar el espacio de tablas en estado **offline,** como puede ver en la Figura 4.2.

El espacio de tablas se cambia al estado **offline** (mientras la instancia todavía se encuentra en ejecución) y se ponen a cero los valores de las columnas Size (tamaño) y Used (utilizado). Una vez seleccionada la opción oportuna para poner fuera de línea el espacio de tablas, aparece una pantalla de confirmación mostrando la pregunta «Are you sure you want to take this Tablaspace Offline?>> (¿Está seguro de que desea poner fuera de línea este espacio de tablas?)

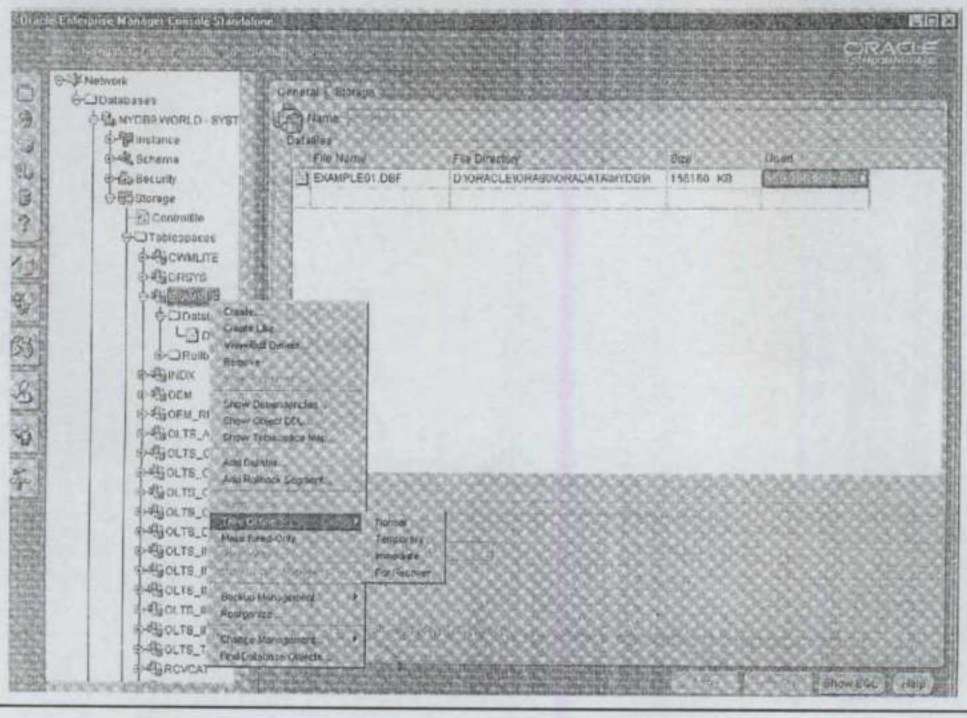


Fig 4.2 Haga clic con el botón secundario del ratón para acceder a la opción Take Offline.

Paso 2. Utilice Microsoft Explorer para cambiar de ubicación al archivo. En Windows NT, puede utilizar la herramienta Microsoft Explorer para cambiar de ubicación al archivo de datos, mientras que en UNIX, puede utilizar el comando mv para trasladar los archivos a las nuevas ubicaciones.

En la Figura 4.3, se muestra el archivo de datos en el directorio D:\Oracle\ Ora90\oradata\mydb9\. Se utiliza el cursor para hacer clic en el archivo de datos y, con el botón principal del ratón pulsado, se arrastra el archivo desde el directorio actual al nuevo, el directorio G:\Oracle\oradata\mydb9\. En la Figura 4.4, se muestra el archivo de datos en el directorio G:\Oracle\oradata\mydb9\.

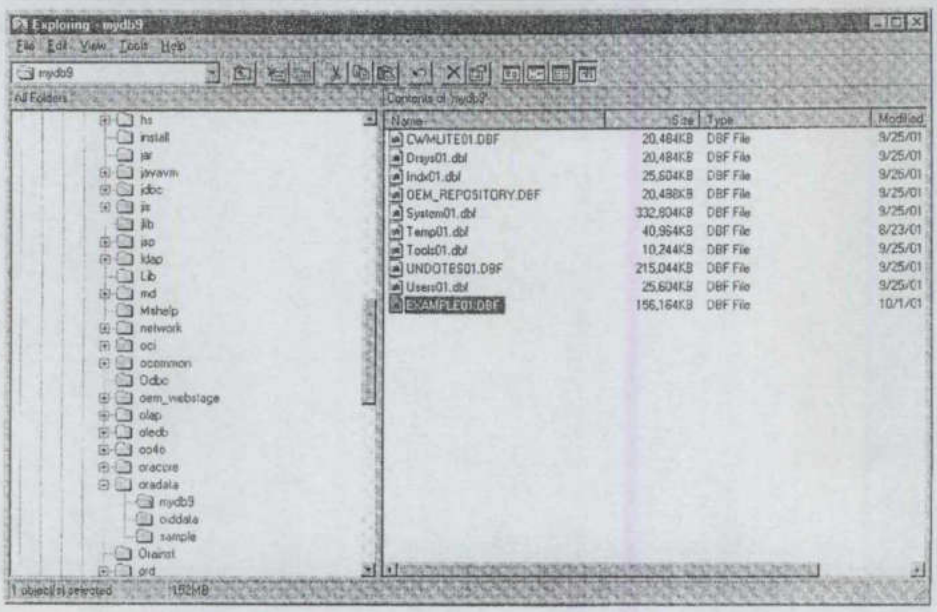


Fig. 4.3 Ubicación original de los archivos en Windows NT.

Paso ***3. Utilice la pantalla de detalles del archivo de datos para modificar el nombre.*** En este ejemplo, se cambia, dentro de la base de datos, el nombre del archivo de datos ExampleOl.dbf, que hemos cambiado de ubicación en el Paso 2. De este modo, la base de datos podrá acceder a ese archivo. Al cambiar el nombre al directorio del archivo de datos, no se cambia el nombre al archivo, sino que se le deberá haber cambiado de nombre o de ubicación previamente. En las Figuras 4.5, 4.6, 4.7 y 4.8 se muestran los sucesivos pasos que se han de dar para modificar la ubicación de los archivos de datos y ver la modificación desde la consola OEM.

En la Figura 4.5 se muestra la pantalla original antes de que se modifique la ubicación del directorio de archivos. También se muestra el menú desplegable Object (objeto) con la opción View/Edit Details (ver o modificar detalles) seleccionada.

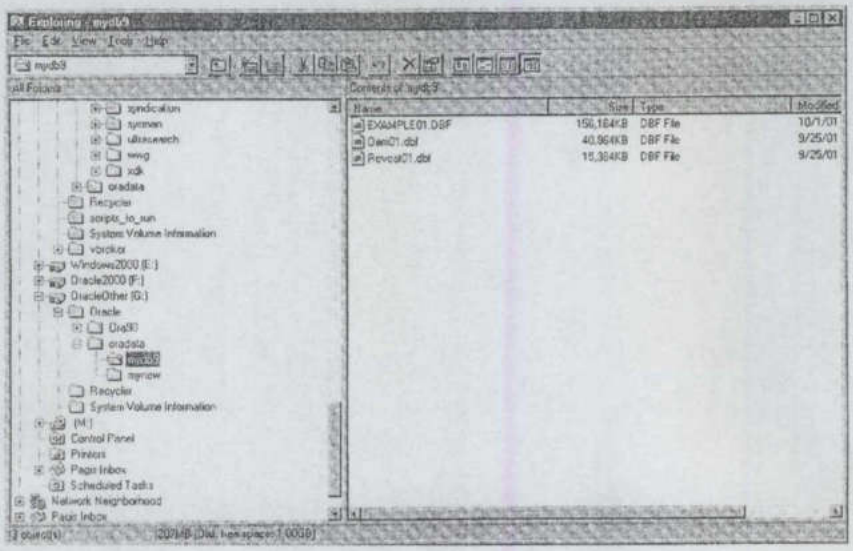


Fig. 4.4 Nueva ubicación de los archivos de Windows NT.

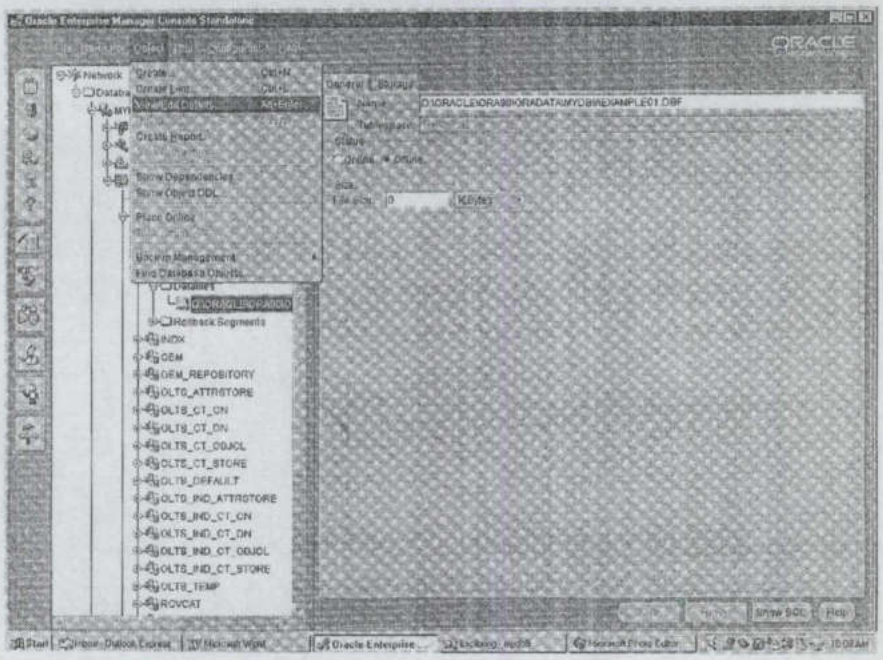


Fig 4.5 Consola de administración de OEM con la opción Object | View/Edit Details seleccionada.

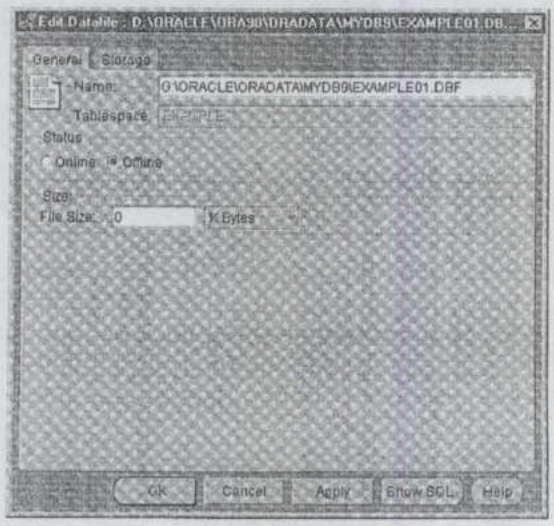


Fig. 4.6 Ficha general de View/Edit Datafile.

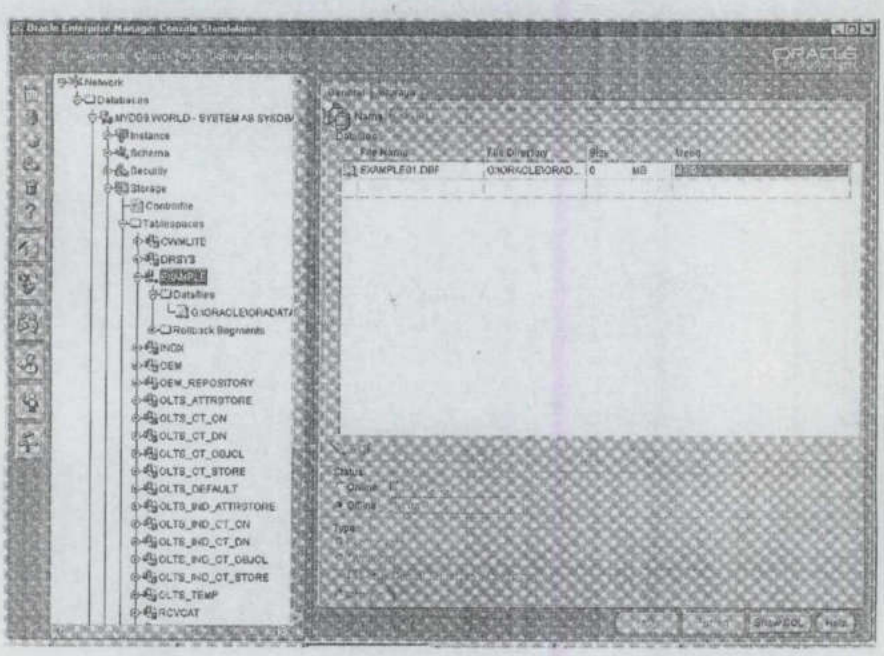


Fig. 4.7 Pantalla de la consola de OEM tras la modificación del archivo de datos.

En la Figura 4.6 se muestran los detalles del archivo de datos, una vez cambiada la ubicación del directorio de la unidad "D:" a la unidad "G:". Observe que el botón Apply (aplicar) que aparece en la parte inferior de la pantalla de detalles se encuentra activo. Al hacer clic en este botón, Oracle comprueba si el nombre que se está asignando al archivo existe; en caso de que este paso falle, compruebe si el nombre del archivo de destino es correcto.

Para actualizar la pantalla izquierda de OEM y visualizar el cambio, utilice la opción Refresh (actualizar), que encontrará debajo de la opción Navigator (navegador i del menú OEM.

En la Figura 4.7, se ha actualizado la pantalla y tanto la parte izquierda como la pantalla de detalles muestran ahora la información correcta. También se reflejan los cambios al salir de la herramienta OEM y volver a abrirla.

No cierre OEM al terminar este paso; permanezca en esta herramienta y continúe con el Paso 4.

Paso *4. Ponga de nuevo en línea el espacio de tablas.* Utilice el menú des-plegable o haga clic con el botón secundario del ratón para volver a poner en línea el espacio de tablas desde la opción Oracle Storage (almacenamiento de Oracle) de OEM. Seleccione la opción Place Online para devolver al espacio de tablas al estado **online** (en línea), como se muestra en la Figura 4.8.

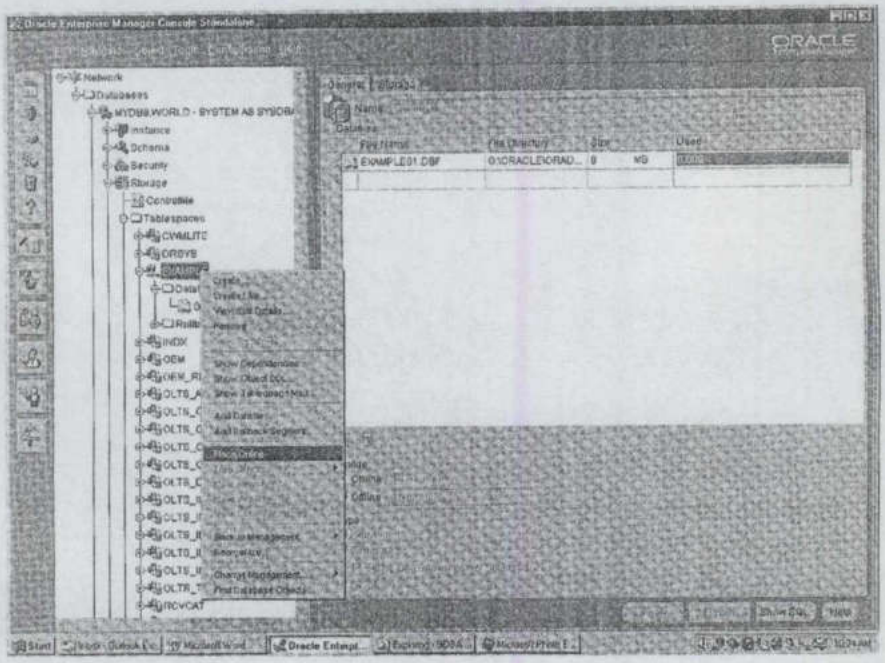


Fig. 4.8 Haga clic con el botón secundario del ratón para acceder a la opción Place Online.

Tras seleccionar la opción apropiada para poner en línea el espacio de tablas, aparece una pantalla de confirmación mostrando la pregunta «Are you sure you want to bring this Tablespace Online?>> (¿Está seguro de que desea poner en línea este espacio de tablas?) Confirme su selección haciendo clic en la opción Yes, o cancele la operación seleccionando No. Al hacer clic en Yes, el espacio de tablas EXAMPLE vuelve a estar en línea, utilizando ya la ubicación nueva para el archivo de datos.

4.7.3 Cambio de ubicación de los archivos de registro de reconstrucción en línea

Los archivos de registro de reconstrucción en línea se pueden cambiar de ubicación mientras la base de datos se encuentra cerrada y se les puede cambiar el nombre, en la base de datos, utilizando el comando **alter database.** Los procedimientos que hay que seguir para cambiar de ubicación a los archivos de registro de reconstrucción en línea son muy parecidos a los que se utilizan para cambiar de ubicación a los archivos de datos mediante el comando **alter database.**

En primer lugar, se cierra la base de datos y se cambia de ubicación al archivo de registro de reconstrucción en línea. A continuación, se monta la base de datos y se utiliza el comando **alter database** para indicar a la base de datos cuál es la nueva ubicación del archivo de registro de reconstrucción en línea. Entonces, ya se puede abrir la instancia, utilizando el archivo de registro de reconstrucción en línea en su nueva ubicación.

El ejemplo siguiente muestra el uso del comando **mv** de UNIX para cambiar de ubicación al archivo redo0lCCl.dbf mientras la base de datos se encuentra cerrada. En este caso, el archivo se traslada del dispositivo /db05 a otro denominado /db02:

```
> mv /db05/oracle/CCl/redo01CCl.dbf  /db02/oracle/CCl
```

Hay que especificar el nombre completo de archivo utilizando los convenios de denominación del sistema operativo.

Para que la base de datos reconozca la nueva ubicación del archivo, es necesario montar la base de datos (en este ejemplo, se llama CC1) y hacer que apunte hacia la nueva ubicación. El comando **alter database** mostrado en el siguiente listado no cambia el nombre del archivo, sino que se habrá tenido que cambiar previamente el nombre o la ubicación de éste.

```
SQL>  connect  /  as  sysdba;
SQL>  startup mount CC1;
SQL> alter database  rename  file
     2>   '/db05/oracle/CCl/redo0lCCl.dbf'   to
     3>   '/db02/oracle/CCl/redo01CCl.dbf';
```

Ahora podrá abrir la base de datos utilizando el comando **alter database open.** Verifique que el archivo ha cambiado de ubicación correctamente y haga una copia de seguridad del sistema de archivos de la base de datos.

4.7.4 Cambio de ubicación de los archivos de control

La ubicación de los archivos de control se especifica en el archivo de parámetros de inicialización de la instancia. Para cambiar de ubicación a un archivo de control, primero hay que cerrar la instancia, cambiar de ubicación a dicho archivo, modificar el archivo de inicialización y luego volver a iniciar la instancia.

El listado siguiente muestra el uso del comando **mv** de UNIX para trasladar un archivo de control a una nueva ubicación. En el ejemplo siguiente se muestra cómo se traslada el archivo ctrllCCl.ctl del dispositivo /db05 a otro denominado /db02.

```
> mv /db05/oracle/CCl/ctrllCCl.ctl /db02/oracle/CCl
```

Hay que especificar el nombre completo de archivo, empleando los convenios de denominación del sistema operativo.

En el archivo de parámetros de la base de datos, modifique la instrucción del parámetro *controlarles:*

```
control_files     =  (/db01/oracle/CCl/ctrllCCl.ctl,
                      /db03/oracle/CCl/ctrllCCl.ctl,
                      /db05/oracle/CCl/ctrllCCl.ctl)
```

Modifique esta instrucción de modo que refleje los cambios introducidos en la ubicación del archivo:

```
control_files     =  (/db01/oracle/CCl/ctrllCCl.ctl,
                      /db03/oracle/CCl/ctrllCCl.ctl,
                      /db02/oracle/CCl/ctrllCCl.ctl)
```

Ya puede reiniciar la instancia utilizando la nueva ubicación del archivo de control.

4.8 Cómo desasignar espacio de los segmentos

A partir de la versión 7.2 de Oracle, es posible reclamar el espacio que no se está utilizando de los archivos de datos existentes. A partir de la versión 7.3 de Oracle, se puede reclamar espacio de tablas, índices y clusters. En las siguientes secciones, veremos ejemplos de archivos de datos, tablas e índices que se <<contraen>> para permitir que se reclame el espacio asignado previamente.

4.8.1 Reducción de tamaño de los archivos de datos

Se puede utilizar el comando **alter database** para reclamar el espacio no utilizado que tengan los archivos de datos. Es imposible redimensionar un archivo de datos si el espacio que se está intentando reclamar está asignado actualmente a un objeto de base de datos.

Por ejemplo, si el archivo de datos tiene 100 MB y, de ellos, actualmente se están utilizando 70 MB, será necesario dejar 70 MB, como mínimo, en dicho archivo de datos. La cláusula **resize** del comando **alter database** se utiliza para reclamar espacio, tal y como vemos en el siguiente ejemplo:

```
alter database datafile '/db05/oracle/CCl/data01.dbf'
resize  80M;
```

Como muestra este listado, se especifica el nombre del archivo cuyo tamaño se va a reducir y su nuevo tamaño. Si no hay ningún objeto de base de datos tras los primeros 80 MB del archivo de datos especificado, éste reducirá su tamaño hasta los 80 MB.

En cambio, si se está utilizando espacio en el interior del archivo de datos pasados los primeros 80 MB, se devolverá un error. Tal y como muestra el siguiente listado, este error informará de la cantidad de espacio que se está utilizando en el archivo de datos por encima del valor **resize** que se haya especificado:

```
alter database datafile '/db05/oracle/CCl/data01.dbf'
resize 80M;
*
ERROR at line 1:
ORA-03297: file contains 507 blocks of data beyond
requested RESIZE value
```

Si el tamaño de bloque de la base de datos es de 4 K, 507 bloques de la base de datos equivaldrían a 1,98 MB. Si incrementamos el valor de **resize** especificado hasta 82 MB, es posible redimensionar el archivo de datos.

Para minimizar la probabilidad de encontrar un error durante el proceso de reclamación de espacio libre de los archivos de datos, se puede averiguar el espacio libre de un archivo de datos. Si el espacio libre está fragmentado, Oracle podría no ser capaz de reclamarlo por entero durante la operación de redimensionamiento de un archivo de datos.

4.8.2 Reducción de tamaño de tablas, clusters e índices

Cuando Oracle escribe datos en un segmento, actualiza a *marca de límite superior (high-water mark)* de dicho segmento. Internamente, la marca de límite superior aumenta a medida que se colocan bloques en la lista de bloques libres, normalmente, de cinco en cinco bloques. La marca de límite superior apunta al bloque inmediatamente después del último bloque de la lista de bloques libres. Si se insertan **(insert)** miles de filas en la tabla, la marca de límite superior aumenta; si, por el contrario, se borran **(delete)** registros, dicha marca de límite superior no disminuye. Si exceptuamos las operaciones de eliminar y volver a crear la tabla, la marca de límite superior de un segmento sólo se restablece cuando se ejecuta un comando **truncate** (truncar) o cuando se elimina y se vuelve a crear el segmento.

A partir de la versión 7.3 de Oracle, se puede reclamar el espacio no utilizado de un segmento que se encuentre por encima de esta marca de límite superior. Si hemos sobrestimado los requisitos de almacenamiento de un objeto, puede que deseemos reclamar el espacio que se ha asignado innecesariamente. Se puede reclamar espacio del segmento sin necesidad de tenerlo que eliminar y volverlo a crear, pero con una condición: sólo se puede reclamar el espacio que se encuentra por encima de la marca de límite superior de la tabla.

Antes de poder reclamar espacio de una tabla, por tanto, se debería determinar cuál es su marca de límite superior. En el siguiente listado, se utiliza el procedimiento

UNUSED_SPACE del paquete DBMS_SPACE para determinar el uso de espacio de la tabla denominada SPACES, que es propiedad del usuario OPS$CC1:

```
declare
  VAR1 number;
  VAR2 number;
  VAR3 number;
  VAR4 number;
  VAR5 number;
  VAR6 number;
  VAR7 number;
begin
     dbms_space .unused_space('OPS$CC1','SPACES','TABLE',
                         VAR1,VAR2,VAR3,VAR4,VAR5,VAR6,VAR7);
    dbms_output.put_line( 'OBJECT_NAME  = SPACES ' ) ;
    dbms_output.put_line( '----------------------------' ) ;
    dbms_output.put_line( 'TOTAL_BLOCKS ='  ||VAR1) ;
    dbms_output.put_line( 'TOTAL_BYTES  ='  ||VAR2) ;
    dbms_output.put_line( 'UNUSED_BLOCKS='  ||VAR3) ;
    dbms_output.put_line( 'UNUSED_BYTES ='  ||VAR4 ) ;
    dbms_output.put_line( 'LAST_USED_EXTENT_FILE_ID  =
    'VAR5);
    dbms_output.put_line( ' LAST_USED_EXTENT_BLOCK_ID =
    'VAR6);
    dbms_output.put_line( ' LAST_USED_BLOCK   = ' ||VAR7 );
    end;
    /
```

A continuación, puede ver un ejemplo de la salida del script anterior para una tabla de 2 MB denominada SPACES:

```
OBJECT_NAME       =    SPACES

------------------------------
TOTAL_BLOCKS      =    500
TOTAL_BYTES       =    2048000
UNUSED_BLOCKS     =    200
UNUSED BYTES      =    819200
```

La marca de límite superior de la tabla (en bytes) representa la diferencia entre el valor TOTAL_BYTES y el valor UNUSED_BYTES que ha devuelto esta llamada de procedimiento. El valor UNUSED_BLOCKS representa el número de bloques que hay por encima de la marca de límite superior; por su parte, el valor TOTAL_BLOCKS refleja el número total de bloques asignados a la tabla.

Si quiere reclamar espacio de la tabla, y su valor UNUSED_BLOCKS es distinto de cero, puede utilizar el comando **alter table** para reclamar el espacio que se encuentra por

encima de la marca de límite superior. En el ejemplo anterior, el valor TOTAL_BLOCKS es de 500 y el valor de UNUSED_BLOCKS es de 200, con un bloque de base de datos de 4 KB. En este caso, se podrían reclamar 200 bloques no utilizados; es decir, 800 KB.

Si quiere dejar 20 bloques en la tabla como espacio no utilizado por encima de la marca de límite superior, puede modificar la tabla, especificando que la base de datos mantenga 20 bloques; es decir, 80 K.

Para reclamar espacio de una tabla, utilice el comando **alter table,** tal y como muestra el siguiente listado. El parámetro **keep** especifica la cantidad de espacio libre que hay que mantener.

```
alter table SPACES deallocate unused keep 80K;
```

Si no se hubiera especificado la cláusula **keep,** se habrían conservado los valores de almacenamiento **minextents e initial** de la tabla. En cambio, al utilizar la cláusula **keep,** se puede eliminar el espacio libre de cualquier extensión, incluso de la extensión inicial, si no hay datos en ninguna otra.

Para desasignar espacio que esté siendo usado por clusters, se puede utilizar la cláusula **deallocate unused** del comando **alter cluster.** Tras haber desasignado espacio de un segmento, debería ejecutar los procedimientos del paquete DBMS_SPACE para ver los nuevos valores de bloques totales y no utilizados asignados al segmento.

Para desasignar espacio que esté siendo usado por índices, se utiliza la cláusula **deallocate unused** del comando **alter index.** No obstante, existe otra opción para los índices, que proporciona aún más flexibilidad a la hora de gestionar su uso del espacio; esta opción es la cláusula **alter index rebuild,** de la que vamos a hablar en la siguiente sección.

4.8.3 Cómo reconstruir los índices

A partir de la versión 7.3 de Oracle, se puede utilizar la cláusula **alter index rebuild** para cambiar rápidamente los parámetros **storage** y **tablespace** de un índice existente, sin tener que eliminar el índice original.

Al utilizar el comando **alter index rebuild,** el índice existente se usa como origen de datos del nuevo índice (en lugar de la tabla), mejorando con ello el rendimiento de su proceso de creación. Durante la reconstrucción del índice, se pueden cambiar sus parámetros **storage y tablespace.**

En el siguiente ejemplo, el índice IU_SPACES$DB_TS_CD se reconstruye por medio del comando **alter índex rebuild.** Asimismo, se cambian sus parámetros de almacenamiento (storage), con el fin de utilizar una extensión inicial **(initial)** de 16 MB y un tamaño de extensión **next** de 16 MB en el espacio de tablas INDX_1.

alter Índex IU_SPACES$DB_TS_CD rebuild storage (initial 16M next 16M pctincrease 0) tablespace INDX_1;

Mientras se está construyendo el nuevo índice IU_SPACES$DB_TS_DB, éste coexistirá en la base de datos con el índice antiguo. Por tanto, si se quiere utilizar el comando **alter index rebuild,** ha de haber suficiente espacio disponible en la base de datos para guardar tanto el índice antiguo como el nuevo. Una vez que este comando se haya completado y el nuevo índice ya esté disponible, el antiguo se eliminará automáticamente y

el espacio que deje libre podrá ser reclamado, pero ese espacio tiene que estar disponible durante la ejecución del comando, ya que, si no, el proceso de creación del nuevo índice fallará.

El comando **alter índex rebuild** se puede utilizar para reconstruir rápidamente índices a la vez que son trasladados a otros espacios de tablas. Gracias a las reconstrucciones de índices, se puede diseñar un programa de mantenimiento muy sencillo para los índices que más se utilicen en la base de datos. Si los registros de una tabla se suelen insertar **(insert)** y borrar **(delete)** con frecuencia, el espacio que utilizan sus índices continuará aumentando, aun cuando el número total de registros no cambie.

Para reclamar el espacio que no se utilice en un índice, se puede utilizar el comando **alter índex rebuild.** Planifique que se ejecute periódicamente un trabajo por lotes para reconstruir los índices en las tablas más activas. Ejecute esta tarea en horas fuera de la jornada de trabajo, para evitar conflictos con las operaciones normales de los usuarios. Si utiliza un plan de mantenimiento para los índices, podrá reclamar el espacio no utilizado rápidamente. Durante la ejecución de cada comando **alter index rebuild,** especifique tanto el parámetro **tablespace** como **storage** para el nuevo índice.

4.8.4 Reconstrucciones de índices en línea

Se pueden reconstruir índices en línea mientras se llevan a cabo operaciones de manipulación de datos (DML) en la tabla o partición. El comando **alter index rebuild online** no admite operaciones DML paralelas, índices de mapa de bits, índices de clusters, ni los índices utilizados para aplicar restricciones de integridad referencial. Para el resto de tipos de índices, el comando **rebuild online** constituye una potente herramienta para las bases de datos de alta disponibilidad.

El comando siguiente reconstruye el índice EMP$DEPT_NO en línea:

```
alter index EMP$DEPT_NO rebuild online
storage (initial 16M next 16M pctincrease 0)
tablespace INDX_1;
```

Es necesario disponer de espacio suficiente para que ambos índices, el nuevo y el antiguo, puedan estar en línea simultáneamente. Durante la operación de reconstrucción, se pueden realizar transacciones de usuario contra la tabla EMP.

4.9 Uso de OMF (Oracle Managed Files)

En Oracle9i se puede utilizar OMF (Oracle Managed Files, archivos administrados por Oracle) con el fin de simplificar la administración de los archivos de base de datos. Cuando se utiliza este tipo de archivos, Oracle utiliza sus propias interfaces de sistema de archivos para crear y eliminar archivos de datos, archivos de control y archivos de registro de reconstrucción en línea en función de las necesidades de los comandos de definición de datos (DDL). Al eliminar un espacio de tablas utilizando OMF, por ejemplo, Oracle elimina los archivos de datos asociados al espacio de tablas, en lugar de dejarlos en el sistema de archivos.

4.9.1 Configuración del entorno

Para permitir el uso de OMF, es necesario definir los valores de los siguientes parámetros de inicialización:

DB_CREATE_FILE_DEST. Directorio predeterminado de todos los archivos de datos y archivos temporales cuando no se proporcionan especificaciones de archivo. También se utiliza para los archivos de registro de reconstrucción en línea y los archivos de control si no se especifica DB_CREATE_ONLINE_LOG_DEST_n.

DB_*CREATE_ONLINE_LOG_DEST_n.* Directorio predeterminado para los archivos de registro de reconstrucción en línea y los archivos de control cuando no se proporciona ninguna especificación. *N* puede ser cualquier número entre 1 y 5, por cada archivo de control o archivo de registro de reconstrucción en línea multiplexado.

El parámetro DB_CREATE_FILE_DEST indica a Oracle dónde puede crear y encontrar archivos de datos; es necesario que el directorio exista ya. Cuando se crean archivos de datos utilizando OMF, se puede especificar un directorio o utilizar el valor del parámetro DB_CREATE_FILE_DEST de manera predeterminada.

Puede cambiar dinámicamente los valores de DB_CREATE_FILE_DEST y DB_CREATE_ONLINE_LOG_DEST_n empleando los comandos **alter system y altar session.**

Al crear un archivo OMF, el nombre de archivo se escribe en el registro de alertas con el fin de que puedan ser consultados posteriormente. Asegúrese de que registra los nombres de archivo en el caso de que necesite llevar a cabo una recuperación o cualquier otra operación de mantenimiento de archivos. El nombre de archivo incluirá una cadena unívoca generada por Oracle para evitar que se sobreescriban archivos existentes del sistema operativo.

4.9.2 Creación de archivos OMF

Si se han configurado los parámetros de entorno para permitir el uso de OMF, Oracle administrará los archivos cuando se omitan las cláusulas **datafile o logfile** durante la creación de archivos de datos y archivos de registro. Por ejemplo, se podrían especificar valores de parámetros como:

```
DB_CREATE_FILE_DEST ='/u01/oracle/CC1'
DB_CREATE_ONLINE_LOG_DEST_1 = '/u02/oracle/CC1'
DB_CREATE_ONLINE_LOG_DEST_2 = '/u03/oracle/CC1'
```

Si se ejecuta ahora el comando **create database,** Oracle creará archivos OMF según corresponda. El comando siguiente crea la base de datos CC1:

```
create database cc1;
```

Si se utilizan archivos OMF, este comando generará un archivo de datos de espacio de tablas SYSTEM de 100 MB capaz de extenderse automáticamente en la ubicación /u01/oracle/CCl, además de dos grupos de registros en línea en cada una de las áreas de destino de los registros.

De manera predeterminada, los archivos de registro tendrán 100 MB cada uno. Oracle también creará automáticamente un espacio de tablas de deshacer cambios con un archivo de datos de 10 MB capaz de ampliarse automáticamente en la ubicación /u01/oracle/CCl. Si no se ha definido también un valor para el parámetro de inicialización CONTROL_FILES, Oracle creará archivos de control en las áreas de destino de los registros de reconstrucción.

En una base de datos existente, se puede crear un archivo OMF mientras se crea un espacio de tablas, siempre y cuando se haya configurado el parámetro de inicialización DB_CREATE_FILE_DEST. El comando siguiente crea un archivo OMF para un espacio de tablas USERS_2 nuevo:

```
create tablespace USERS_2;
```

También se puede especificar el lugar de almacenamiento y otros parámetros para los espacios de tablas. Por ejemplo, se pueden crear dos archivos OMF para un espacio de tablas con un único comando:

```
create tablespace USERS_3 datafile size 500M, size 500M;
```

En este ejemplo, se crean dos archivos de datos de 500 MB cada uno para el espacio de tablas USERS_3. Estos archivos podrán ampliarse automáticamente a menos que también se especifique el valor **autoextend off.**

Para añadir un archivo de datos OMF a un espacio de tablas existente, se utiliza el comando **alter tablespace,** como se muestra en el ejemplo siguiente:

```
alter tablespace USERS_3 add datafile size 300M;
```

Se pueden añadir archivos de registro de reconstrucción en línea utilizando el comando **alter database:**

```
alter database add logfile;
```

El comando anterior crea un nuevo archivo de registro con un miembro en cada una de las ubicaciones DB_CREATE_ONLINE _LOG _DEST _*n*.

4.9.3 Mantenimiento de los archivos OMF

Los archivos OMF se pueden gestionar del mismo modo que los archivos gestionados por el usuario. Por ejemplo, si Oracle crea un archivo OMF denominado ORA_USERS_3_ERJ42201.dbf para el espacio de tablas USERS_3, se puede cambiar de ubicación a ese archivo empleando los comandos **alter tablespace o alter database,** como

se ha mostrado anteriormente en este capítulo. El comando siguiente redirige el puntero interno de Oracle para que utilice el directorio /u04/oracle/CCl para este archivo:

```
alter database rename file
'/u01/oracle/CCl/ora_users_3_erj42201.dbf' to
'u04/oracle/CCl/ora_users_3_erj42201.dbf' ;
```

Si se desea, se puede cambiar el nombre del archivo para eliminar la cadena unívoca de ocho caracteres.

El uso de archivos OMF no interfiere en la capacidad para hacer copias de seguridad y recuperar la base de datos mediante exportaciones, copias de seguridad del sistema de archivos o RMAN.

Al eliminar un archivo de datos que utiliza archivos OMF, Oracle elimina automáticamente los archivos de datos. El siguiente comando elimina el espacio de tablas USERS_3 y borra sus archivos en el nivel del sistema operativo:

```
drop tablespace USERS_3;
```

En el caso de que se utilicen archivos administrados por el usuario, los archivos de datos del espacio de tablas USERS_3 continuarán en el sistema operativo, incluso después de que se haya eliminado el espacio de tablas.

4.10 Diseño físico bien ajustado

Hay que planificar la asignación de archivos de las bases de datos (empleando las estimaciones de E/S) y hay que verificar la reserva de espacio cuando el sistema entra en la etapa de producción. Luego, se puede modificar la disposición de los archivos para equilibrar mejor los requisitos de E/S de los mismos. Como resultado, obtendrá una base de datos que cumple sus objetivos de rendimiento sin sacrificar la recuperabilidad y sus objetivos de recuperabilidad sin sacrificar el rendimiento.

También hay que dimensionar correctamente todas las facetas de la base de datos; es decir, las tablas, los índices, los segmentos de anulación y los segmentos temporales. Para ello, es necesario saber el modo en que se van a introducir los datos, cómo se van a guardar y los procesos que se van a realizar sobre ellos cuando ya estén allí. Los costes derivados de la planificación de los parámetros de almacenamiento son mínimos, si los comparamos con los costes de manipular el sistema cuando ya está en la fase de producción. La optimización de postproducción debería ser un paso final de menor importancia en el proceso de planificación del diseño físico de la base de datos.

CONCLUSIÓN

Todo el mundo puede conducir un automóvil sin necesidad de conocer cómo funciona un motor de combustión interna y todos los subsistemas asociados a él. Pero entonces ciertos conceptos como aprovechamiento de la potencia, compresión, endurecimiento de la suspensión, motricidad, etc., le serán ajenos y nunca podrá sacar lo mejor del automóvil. Y si tiene algún problema se quedará tirado en la carretera.

De la misma manera, no podremos aspirar a que nuestras aplicaciones de BD funcionen bien si no conocemos la arquitectura del *motor* de la BD, el servidor. Es indispensable conocer los factores y parámetros que influyen en el funcionamiento de nuestro SGBD para poder solucionar los problemas que se pueden plantear en cuanto nos salgamos de las aplicaciones estándares y básicas de BD, o en cuanto tengamos algún problema.

Entre las tareas de un administrador de base de datos se encuentra la Gestiòn de Recursos. Esto es asignaciòn de tablespaces y espacio en disco duro. Asì como actualizaciòn de paràmetros del sistema y modificaciòn de objetos de la base de datos. Todo esto implica que se disponga de un conocimiento previo de la arquitectura de Oracle para su instalaciòn.

Para poner a punto una base de datos Oracle, es necesario conocer como interactúan sus distintos componentes, dónde encajan dentro del conjunto y cómo personalizar el sistema en función de las necesidades propias.

La arquitectura de Oracle 9i anuncia "una tercera era de la informática" en el manejo de la información con un tercer modelo de agrupamiento de bases de datos en sistemas de hardware (cluster) menos costoso -que se contrapone a los modelos mainframe y cliente/servidor- para aumentar el desempeño, escalabilidad y disponibilidad.

Entre las características de Oracle destacamos su escalabilidad y alta disponibilidad, aportando un sistema de administración completo para gestionar todas las situaciones críticas de una Base de Datos de estas características: Sistema de seguridad basados en usuarios, grupos y roles, alertas, backups y restauración de datos, auditorías, etc. Oracle, desde su versión Oracle 9i, hace especial hincapié en la seguridad y robustez de su base de datos, óptimo rendimiento y mejoras en sus mecanismos de concurrencia (número elevado de usuarios accediendo a la base de datos).

Algunas tablas y vistas del diccionario de datos

Significado	dba	user	all
Información sobre todos los objetos: tablas, vistas, funciones, procedimientos, índices, triggers, etc.	dba_objects	user_objects	all_objects
Código de funciones y procedimientos	dba_sourc	user_source	all_source
Usuarios	dba_users	user_users	all_users
Roles	dba_roles	-	-
Roles asignados a roles o usuarios	dba_role_privs	user_role_privs	-
Privilegios asignados a roles o usuarios	dba_sys_privs	-	-
Permisos sobre tablas asignados a roles o usuarios	dba_tab_privs	-	-
Límites de recursos	-	user_resource_limits	-
Perfiles y sus límites de recursos asociados	dba_profiles	-	-
Límites de recursos en cuanto a restricciones en claves	-	user_password_limits	-
Límites de recursos en cuanto a espacio máximo en tablespaces	dba_ts_quotas	user_ts_quotas	-

Tablespaces	dba_tablespace	user_tablespaces	-
Ficheros que componen los datafiles	dba_data_files	-	-
Segmentos	dba_segments	user_segment	all_segments
Segmentos de Rollback	dba_rollback_segs	-	-
Extensiones que forman los segmentos	dba_extents	user_extents	-
Bloques libres	dba_free_spac	user_free_spac	-
Bloques libres que podrían unirse	dba_free_space_coalesce	-	-
Secuencias	dba_sequence	user_sequence	all_sequences
Tablas, vistas, sinónimos y secuencias	dba_catalog	user_catalog	all_catalog
Tablas	dba_tables	user_tables	all_tables
Campos de tablas	dba_cons_colum	user_cons_columr	all_cons_columns
Columnas de las tablas	dba_tab_column	user_tab_column	all_tab_columns
Vistas	dba_view	user_view	all_views
Sinónimos	dba_synonym	user_synonyn	all_synonyms
Restricciones de clave primaria, externa, not null, integridad referencial	dba_constraints	user_constraints	all_constraints

Índices	dba_indexes	user_indexes	all_indexes
Columnas de los índices	dba_ind_column	user_ind_column	all_ind_columns

Significado	Vista
Roles asignados a roles	role_role_privs
Privilegios de cada rol	role_sys_privs
Propiedades de tablas en cuanto a gestión de espacio	tabquotas
Espacio total (libre + usado) en los tablespaces	sm$ts_avail
Espacio libre en los tablespaces	sm$ts_free
Espacio usado en los tablespaces	sm$ts_use

Significado	Vista
Objetos que existen en la caché de la sga	v$db_object_cach
Número de lecturas y escrituras por fichero	v$filestat
Estadísticas de rendimiento global de la cache de la sga	v$librarycache
Estadísticas de los segmentos de anulación en línea	v$rollstat
Rendimiento de la caché del diccionario de datos	v$rollcache
Sesiones que esperan a otras	v$sessionwai
Sesiones activas	v$session

Procesos	v$process
Estadísticas de las sesiones activas	v$sesstat
Operaciones de e/s lógicas y físicas por cada sesión activa	v$sess_id
Estadísticas de sga global	v$sgastat
Estadísticas de caché de cursor	v$sqlarea
Estadísticas del v$sesstat	v$statname
Conflictos entre bloques. sólo está activa si timed_statistics es true	v$waitstat
Segmentos de Rollback	v$rollstat
Datafiles	v$datafile
Tablespaces	v$tablespace
Base de datos	v$database

Estas son sólo algunas de las tablas y vistas. Se puede obtener un listado de todas con

```
select OBJECT_NAME from
```

Para seleccionar sólo un cierto número de registros:

```
SELECT * FROM EMP WHERE ROWNUM <= 10;
```

REFERENCIAS BIBLIOGRÁFICAS

Bases de datos en castellano. Estructuras de Oracle
http://www.programacion.com/bbdd/tutorial/oracle/
6/05/2004

Administración y Optimización de Bases de Datos Oracle
http://www.redcientifica.com/oracle/c0002p0001.html
7/30/2004

Solocursos.net
http://www.solocursos.net/oracle-slctema118.htm
7/28/2004

Introducción a la Arquitectura del RDBMS ORACLE
http://www.bd.cesma.usb.ve/ci5313/em04/docs/clase1-spdec01.pdf
09/5/2004

Sección de Oracle
http://www.lawebdejm.com/prog/oracle/index.html
10/08/2004

Loney, Kevin; Theriault, Marlene, TUSC; ORACLE 9i Manual del Administrador, Edición Oracle Press Oficial, McGrawHill, 2002.