UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

FACULTAD DE INFORMÁTICA

ADMINISTRACIÓN DE BASES DE DATOS

TESINA

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN INFORMÁTICA

PRESENTA

ALFREDO IVAN SAFONT BEDOLLA

DIRIGIDA POR:

ISC. JABEL RESÉNDIZ GONZÁLEZ

SANTIAGO DE QUERÉTARO, QRO., NOVIEMBRE DE 2002.

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 **ALFREDO IVAN SAFONT BEDOLLA** con el tema denominado "*Administración de Base de Datos*".

Este trabajo fue desarrollado como una investigación derivada del curso de titulación **"ADMINISTRACIÓN DE BASE DE DATOS",** 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 ocho días del mes de noviembre del dos mil dos.

ATENTAMENTE

ING. JABEL RESENDIZ GONZÁLEZ
PROFR. CURSO DE TITULACIÓN

INDICE

1 CONCEPTOS BÁSICOS DE BASES DE DATOS

1.1. Introducción

En la actualidad, el manejo de la información a través de las Bases de Datos, se ha convertido en la forma más segura, eficiente, rápida y conveniente para la mayoría de las organizaciones; dado que los Sistemas de Bases de Datos permiten un almacenamiento de información coherente relacionado e integral.

Las Bases de Datos son la estructura por excelencia que han adoptado los expertos en sistemas de información y que han implementado en las empresas para que la toma de decisiones sea más rápida y precisa. Esta es por supuesto, una gran ventaja para las diferentes organizaciones, ya que confían plenamente en los reportes arrojados por el Sistema de Gestión de Bases de Datos (SGBD) de su empresa, puesto que se supone que éste sistema cuenta con la precisión y confiabilidad necesaria para su objetivo.

Por lo anterior, es necesario que el Sistema de Bases de Datos cuente con una planificación, desarrollo y administración que permitan garantizar a los altos directivos la integridad de la información que de éste tomarán para el crecimiento, mejoras y detección de fallas en los diferentes procesos de la organización y que sirven para llegar a las metas.

La Administración de las Bases de Datos es una tarea muy importante, no podemos prescindir de un buen administrador de las Bases de Datos que nos guíe y que proteja toda esa información que estamos depositando en el sistema. La administración de Bases de Datos incluye varias tareas, desde la planificación y el diseño, hasta el mantenimiento. No es una tarea sencilla, sin embargo una persona capacitada y que se involucre en el proyecto desde el principio podrá sentirse apto para llevar a cabo las responsabilidades del Administrador.

Este trabajo está dedicado a la Administración de las Bases de Datos, aquí trato de resaltar la importancia de de esta área y además se incluyen una serie de ejemplo y ejercicios tomando como base ORACLE, ya que es una de las más utilizadas actualmente. Asimismo veremos algunas sentencias SQL y PLSQL que nos permitirán observar cómo se ejecutan para obtener determinados resultados que podrían ser de utilidad no sólo para los administradores sino para cualquier persona que se interese en el manejo de las Bases de Datos.

1.2. Planificación, Diseño y Administración de las Bases de Datos

Algunas de las personas que trabajan con SGBD (Sistemas de Gestión de Bases de Datos) parecen preguntarse porqué deberían preocuparse del diseño de las bases de datos que utilizan. Después de todo, la mayoría de los SGBD vienen con bases de datos de ejemplo que se pueden copiar y después modificar, para que se adecuen a cada caso particular, e incluso las tablas de esas bases de datos de ejemplo se pueden cortar y pegar en una nueva base de datos. Algunos SGBD tienen ``asistentes", herramientas que guían al usuario a través del proceso de definición y creación de tablas. Sin embargo, esas herramientas no sirven para diseñar una base de datos, tan solo ayudan a crear las tablas físicas que se incluirán en la base de datos.

Lo que la mayoría de la gente no parece entender es que esas herramientas se deben utilizar después de que se haya realizado el diseño lógico de la base de datos. Los asistentes y las bases de datos de ejemplo se suministran para minimizar el tiempo que lleva implementar la estructura física de la base de datos. La idea es que si se ahorra tiempo en la implementación de la estructura de la base de datos una vez se ha realizado el diseño lógico, habrá más tiempo para centrarse en la creación y construcción de las aplicaciones que se utilizarán para trabajar con los datos de la base de datos.

Por lo tanto, la razón para preocuparse por el diseño de las bases de datos es que es crucial para la consistencia, integridad y precisión de los datos. Si una base de datos está mal diseñada, los usuarios tendrán dificultades a la hora de acceder a ciertos tipos de información y existe el riesgo añadido de que ciertas búsquedas puedan producir información errónea. La información errónea es, probablemente, el peor de los resultados de un mal diseño de la base de datos. Puede repercutir muy negativamente a la empresa u organización propietaria de los datos. De hecho, si los datos de una base de datos van a influir en la gestión del negocio, si van a servir para la toma de decisiones de la empresa, la base de datos debe ser una preocupación.

Viéndolo desde una perspectiva diferente, la base de datos es como una casa que queremos que nos construyan. ¿Qué es lo primero que hay que hacer? Desde luego, lo que no vamos a hacer es buscar a un constructor que haga la casa sobre la marcha y como él quiera. Seguramente, buscaremos primero a un arquitecto que diseñe nuestra nueva casa y después haremos que el constructor la edifique. El arquitecto expresará nuestras necesidades en una serie de planos, anotando todos los requisitos de los diversos sistemas (estructural, mecánico y eléctrico). Después, el constructor pondrá los materiales necesarios, tal y como se indica en los planos y en las especificaciones.

Volviendo a la perspectiva de las bases de datos, el diseño lógico corresponde con la fase de elaboración de los planos arquitectónicos, y la implementación física de la base de datos es la casa ya construida. El diseño lógico describe el tamaño, la forma y los sistemas necesarios para la base de datos: contiene las necesidades en cuanto a información y modo de operación del negocio. Después, se construye la implementación física del diseño lógico de la base de

datos mediante el SGBD. Si pensamos en un sistema relacional, una vez creadas las tablas, establecidas las relaciones y los niveles de integridad necesarios, la base de datos está finalizada. Ahora ya se pueden crear las aplicaciones que permiten interactuar con los datos de la base de datos, y podemos estar seguros de que estas aplicaciones proporcionarán la información oportuna y, sobre todo, la información correcta.

Se pueden hacer malos diseños, pero una base de datos bien diseñada contendrá información correcta, almacenará los datos más eficientemente y será más fácil de gestionar y de mantener.

1.2.1. Ciclo de vida de los sistemas de información

Un sistema de información es el conjunto de recursos que permiten recoger, gestionar, controlar y difundir la información de toda una empresa u organización.

Desde los años setenta, los sistemas de bases de datos han ido reemplazando a los sistemas de ficheros en los sistemas de información de las empresas. Al mismo tiempo, se ha ido reconociendo la gran importancia que tienen los datos que éstas manejan, convirtiéndose en uno de sus recursos más importantes. Esto ha hecho que muchas empresas tengan departamentos que se encarguen de gestionar toda su información, que estará almacenada en una base de datos. Aparecen los papeles de administrador de datos y administrador de la base de datos, que son las personas encargadas de supervisar y controlar todas las actividades relacionadas con los datos de la empresa y con el ciclo de vida de las aplicaciones de bases de datos, respectivamente.

Un sistema de información está formado por los siguientes componentes:

- La base de datos.
- El SGBD.
- Los programas de aplicación.
- Los dispositivos fisicos (ordenadores, dispositivos de almacenamiento, etc.).
- El personal que utiliza y que desarrolla el sistema.
-

La *base de datos* es un componente fundamental de un sistema de información. El ciclo de vida de un sistema de información está ligado al ciclo de vida del sistema de base de datos sobre el que se apoya. Al ciclo de vida de los sistemas de información también se le denomina ciclo de vida de desarrollo del software. Las etapas típicas del ciclo de vida de desarrollo del software son: planificación, recolección y análisis de los requisitos, diseño (incluyendo el diseño de la base de datos), creación de prototipos, implementación, prueba, conversión y mantenimiento. Este ciclo de vida hace énfasis en la identificación de las funciones que realiza la empresa y en el desarrollo de las aplicaciones que lleven a cabo estas funciones. Se dice que el ciclo de vida de desarrollo del software sigue un enfoque orientado a funciones, ya que los sistemas se ven desde el punto de vista de las funciones que llevan a cabo. Por esta razón, el análisis estructurado hace énfasis en los diagramas de flujo de datos, siguiendo el movimiento de los datos a través de una secuencia de transformaciones, y refinando éstas a través de una serie de niveles. Lo mismo ocurre en el diseño estructurado, que ve a un sistema como una función que se descompone sucesivamente en niveles o subfunciones.

Concentrándose en las funciones se infravaloran los datos y, en especial, la estructura de los datos que son manipulados por las funciones. El resultado es que estos sistemas tienen valor durante poco tiempo en relación con las necesidades de los usuarios a largo plazo. Esto sucede debido a que al poco tiempo de haber instalado un sistema, las funciones implementadas son en realidad un subconjunto de las funciones que los usuarios realmente desean. Casi inmediatamente, los usuarios descubren una gran variedad de servicios adicionales que quisieran incorporar al sistema. Estas necesidades causan problemas a los sistemas obtenidos con un diseño orientado a funciones, puesto que este diseño puede requerir una revisión importante para acomodar las funciones adicionales.

En contraste, el enfoque orientado a datos centra el foco de atención en el análisis de los datos utilizados por las funciones. Esto tiene dos ventajas. La primera es que los datos son una parte considerablemente más estable que las funciones. La segunda ventaja es que la propia estructura de un esquema de base de datos requiere de un análisis sofisticado de los datos y de sus relaciones. Una vez que se haya construido un esquema para la base de datos que sea lógico, podrían diseñarse tantas funciones como fuera necesario para sacar provecho del mismo. Sin embargo, sin un esquema tal, la base de datos sólo podría ser útil para una única aplicación. Por lo tanto, el enfoque orientado a funciones puede ser bueno para el desarrollo a corto plazo, pero pierde su valor real a largo plazo. Usando un enfoque orientado a datos, los datos pasan a ser los cimientos sobre los cuales se puede construir una gran variedad de funciones diferentes.

1.2.2. Ciclo de vida de las aplicaciones de bases de datos

Las etapas del ciclo de vida de una aplicación de bases de datos son las siguientes:

1. Planificación del proyecto.
2. Definición del sistema.
3. Recolección y análisis de los requisitos.
4. Diseño de la base de datos.
5. Selección del SGBD.
6. Diseño de la aplicación.
7. Prototipado.
8. Implementación.
9. Conversión y carga de datos.
10. Prueba.
11. Mantenimiento.

Estas etapas no son estrictamente secuenciales. De hecho hay que repetir algunas de las etapas varias veces, haciendo lo que se conocen como ciclos de realimentación. Por ejemplo, los problemas que se encuentran en la etapa del diseño de la base de datos pueden requerir una recolección de requisitos adicional y su posterior análisis.
A continuación, se muestran las tareas más importantes que se realizan en cada etapa.

1. Planificación del proyecto
Esta etapa conlleva la planificación de cómo se pueden llevar a cabo las etapas del ciclo de vida de la manera más eficiente. Hay tres componentes principales: el trabajo que se ha de realizar, los recursos para llevarlo a cabo y el dinero para pagar por todo ello. Como apoyo a esta etapa, se necesitará un modelo de datos corporativo en donde se muestren las entidades principales de la empresa y sus relaciones, y en donde se identifiquen las principales áreas funcionales. Normalmente, este modelo de datos se representa mediante un diagrama entidad-relación. En este modelo se tiene que mostrar también qué datos comparten las distintas áreas funcionales de la empresa.

La planificación de la base de datos también incluye el desarrollo de estándares que especifiquen cómo realizar la recolección de datos, cómo especificar su formato, qué documentación será necesaria y cómo se va a llevar a cabo el diseño y la implementación. El desarrollo y el mantenimiento de los estándares puede llevar bastante tiempo, pero si están bien diseñados, son una base para el personal informático en formación y para medir la calidad, además, garantizan que el trabajo se ajusta a unos patrones, independientemente de las habilidades y la experiencia del diseñador. Por ejemplo, se pueden establecer reglas sobre cómo dar nombres a los datos, lo que evitará redundancias e inconsistencias. Se deben documentar todos los aspectos legales sobre los datos y los establecidos por la empresa como, por ejemplo, qué datos deben tratarse de modo confidencial.

2. Definición del sistema
En esta etapa se especifica el ámbito y los límites de la aplicación de bases de datos, así como con qué otros sistemas interactúa. También hay que determinar quienes son los usuarios y las áreas de aplicación.

3. Recolección y análisis de los requisitos
En esta etapa se recogen y analizan los requerimientos de los usuarios y de las áreas de aplicación. Esta información se puede recoger de varias formas:

- Entrevistando al personal de la empresa, concretamente, a aquellos que son considerados expertos en las áreas de interés.
- Observando el funcionamiento de la empresa.
- Examinando documentos, sobre todo aquellos que se utilizan para recoger o visualizar información.
- Utilizando cuestionarios para recoger información de grandes grupos de usuarios.
- Utilizando la experiencia adquirida en el diseño de sistemas similares.

La información recogida debe incluir las principales áreas de aplicación y los grupos de usuarios, la documentación utilizada o generada por estas áreas de aplicación o grupos de usuarios, las transacciones requeridas por cada área de aplicación o grupo de usuarios y una lista priorizada de los requerimientos de cada área de aplicación o grupo de usuarios.
Esta etapa tiene como resultado un conjunto de documentos con las especificaciones de requisitos de los usuarios, en donde se describen las operaciones que se realizan en la empresa desde distintos puntos de vista.

La información recogida se debe estructurar utilizando técnicas de especificación de requisitos, como por ejemplo técnicas de análisis y diseño estructurado y diagramas de flujo de datos. También las herramientas CASE (Computer-Aided Software Engineering) pueden proporcionar una asistencia automatizada que garantice que los requisitos son completos y consistentes.

4. Diseño de la base de datos

Esta etapa consta de tres fases: diseño conceptual, diseño lógico y diseño físico de la base de datos. La primera fase consiste en la producción de un esquema conceptual, que es independiente de todas las consideraciones físicas. Este modelo se refina después en un esquema lógico eliminando las construcciones que no se pueden representar en el modelo de base de datos escogido (relacional, orientado a objetos, etc.). En la tercera fase, el esquema lógico se traduce en un esquema físico para el SGBD escogido. La fase de diseño físico considera las estructuras de almacenamiento y los métodos de acceso necesarios para proporcionar un acceso eficiente a la base de datos en memoria secundaria. Más adelante se profundiza en estas fases.

Los objetivos del diseño de la base de datos son:

- Representar los datos que requieren las principales áreas de aplicación y los grupos de usuarios, y representar las relaciones entre dichos datos.
- Proporcionar un modelo de datos que soporte las transacciones que se vayan a realizar sobre los datos.
- Especificar un esquema que alcance las prestaciones requeridas para el sistema.

Hay varias estrategias a seguir para realizar el diseño: de abajo a arriba, de arriba a abajo, de dentro a fuera y la estrategia mixta. La estrategia de abajo a arriba parte de todos los atributos y los va agrupando en entidades y relaciones. Es apropiada cuando la base de datos es simple, con pocos atributos. La estrategia de arriba a abajo es más apropiada cuando se trata de bases de datos complejas. Se comienza con un esquema con entidades de alto nivel, que se van refinando para obtener entidades de bajo nivel, atributos y relaciones. La estrategia de dentro a fuera es similar a la estrategia de abajo a arriba, pero difiere en que se parte de los conceptos principales y se va extendiendo el esquema para considerar también otros conceptos, asociados con los que se han identificado en primer lugar. La estrategia mixta utiliza ambas estrategias, de abajo a arriba y de arriba a abajo, con un esquema de divide y vencerás. Se obtiene un esquema inicial de alto nivel, se divide en partes, y de cada parte se obtiene un subesquema. Estos subesquemas se integran después para obtener el modelo final.

5. Selección del SGBD

Si no se dispone de un SGBD, o el que hay se encuentra obsoleto, se debe escoger un SGBD que sea adecuado para el sistema de información. Esta elección se debe hacer en cualquier momento antes del diseño lógico.

6. Diseño de la aplicación

En esta etapa se diseñan los programas de aplicación que usarán y procesarán la base de datos. Esta etapa y el diseño de la base de datos, son paralelas. En la mayor parte de los casos no se

puede finalizar el diseño de las aplicaciones hasta que se ha terminado con el diseño de la base de datos. Por otro lado, la base de datos existe para dar soporte a las aplicaciones, por lo que habrá una realimentación desde el diseño de las aplicaciones al diseño de la base de datos.

En esta etapa hay que asegurarse de que toda la funcionalidad especificada en los requisitos de usuario se encuentra en el diseño de la aplicación. Habrá algunos programas que utilicen y procesen los datos de la base de datos.

Además, habrá que diseñar las interfaces de usuario, aspecto muy importante que se suele ignorar. El sistema debe ser fácil de aprender, fácil de usar, ser directo y estar ``dispuesto a perdonar''. Si la interface no tiene estas características, el sistema dará problemas, sin lugar a dudas.

7. Prototipado

Esta etapa, que es opcional, es para construir prototipos de la aplicación que permitan a los diseñadores y a los usuarios probar el sistema. Un prototipo es un modelo de trabajo de las aplicaciones del sistema. El prototipo no tiene toda la funcionalidad del sistema final, pero es suficiente para que los usuarios puedan utilizar el sistema e identificar qué aspectos están bien y cuáles no son adecuados, además de poder sugerir mejoras o la inclusión de nuevos elementos. Este proceso permite que quienes diseñan e implementan el sistema sepan si han interpretado correctamente los requisitos de los usuarios. Otra ventaja de los prototipos es que se construyen rápidamente.

Esta etapa es imprescindible cuando el sistema que se va a implementar tiene un gran coste, alto riesgo o utiliza nuevas tecnologías.

8. Implementación

En esta etapa se crean las definiciones de la base de datos a nivel conceptual, externo e interno, así como los programas de aplicación. La implementación de la base de datos se realiza mediante las sentencias del lenguaje de definición de datos (LDD) del SGBD escogido. Estas sentencias se encargan de crear el esquema de la base de datos, los ficheros en donde se almacenarán los datos y las vistas de los usuarios.

Los programas de aplicación se implementan utilizando lenguajes de tercera o cuarta generación. Partes de estas aplicaciones son transacciones sobre la base de datos, que se implementan mediante el lenguaje de manejo de datos (LMD) del SGBD. Las sentencias de este lenguaje se pueden embeber en un lenguaje de programación anfitrión como Visual Basic, Delphi, C, C++, Java, COBOL, Fortran, Ada o Pascal. En esta etapa, también se implementan los menús, los formularios para la introducción de datos y los informes de visualización de datos. Para ello, el SGBD puede disponer de lenguajes de cuarta generación que permiten el desarrollo rápido de aplicaciones mediante lenguajes de consultas no procedurales, generadores de informes, generadores de formularios, generadores de gráficos y generadores de aplicaciones.

También se implementan en esta etapa todos los controles de seguridad e integridad. Algunos de estos controles se pueden implementar mediante el LDD y otros puede que haya que implementarlos mediante utilidades del SGBD o mediante programas de aplicación.

9. Conversión y carga de datos

Esta etapa es necesaria cuando se está reemplazando un sistema antiguo por uno nuevo. Los datos se cargan desde el sistema viejo al nuevo directamente o, si es necesario, se convierten al formato que requiera el nuevo SGBD y luego se cargan. Si es posible, los programas de aplicación del sistema antiguo también se convierten para que se puedan utilizar en el sistema nuevo.

10. Prueba

En esta etapa se prueba y valida el sistema con los requisitos especificados por los usuarios. Para ello, se debe diseñar una batería de tests con datos reales, que se deben llevar a cabo de manera metódica y rigurosa. Es importante darse cuenta de que la fase de prueba no sirve para demostrar que no hay fallos, sirve para encontrarlos. Si la fase de prueba se lleva a cabo correctamente, descubrirá los errores en los programas de aplicación y en la estructura de la base de datos. Además, demostrará que los programas ``parecen" trabajar tal y como se especificaba en los requisitos y que las prestaciones deseadas ``parecen" obtenerse. Por último, en las pruebas se podrá hacer una medida de la fiabilidad y la calidad del software desarrollado.

11. Mantenimiento

Una vez que el sistema está completamente implementado y probado, se pone en marcha. El sistema está ahora en la fase de mantenimiento en la que se llevan a cabo las siguientes tareas:

1. Monitorización de las prestaciones del sistema. Si las prestaciones caen por debajo de un determinado nivel, puede ser necesario reorganizar la base de datos.
2. Mantenimiento y actualización del sistema. Cuando sea necesario, los nuevos requisitos que vayan surgiendo se incorporarán al sistema, siguiendo de nuevo las etapas del ciclo de vida que se acaban de presentar.

1.2.3. Diseño de bases de datos

En este apartado se describen con más detalle los objetivos de cada una de las etapas del diseño de bases de datos: diseño conceptual, diseño lógico y diseño físico.

Fig. 1.2.3.1. Proceso de Desarrollo de una Base de Datos

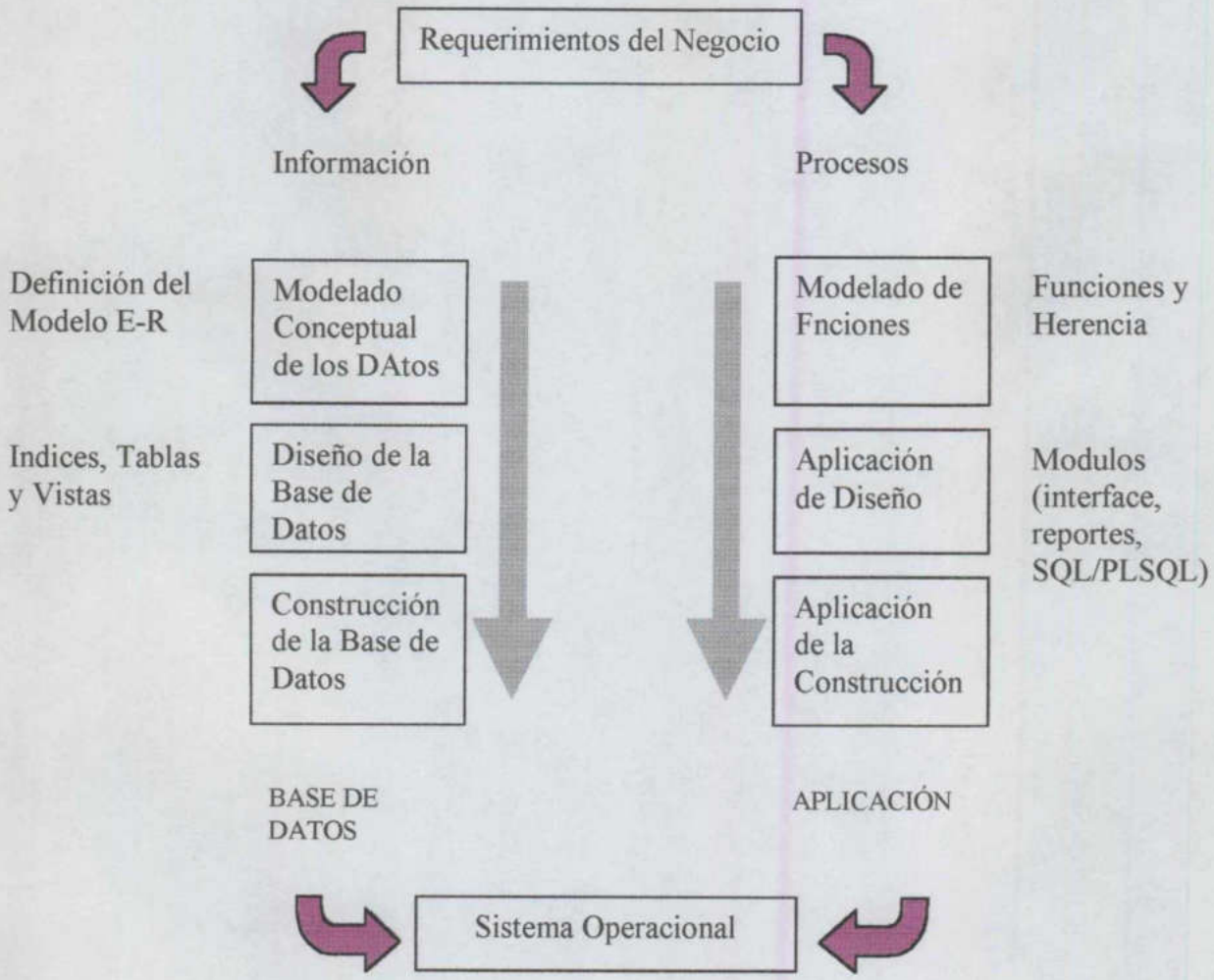
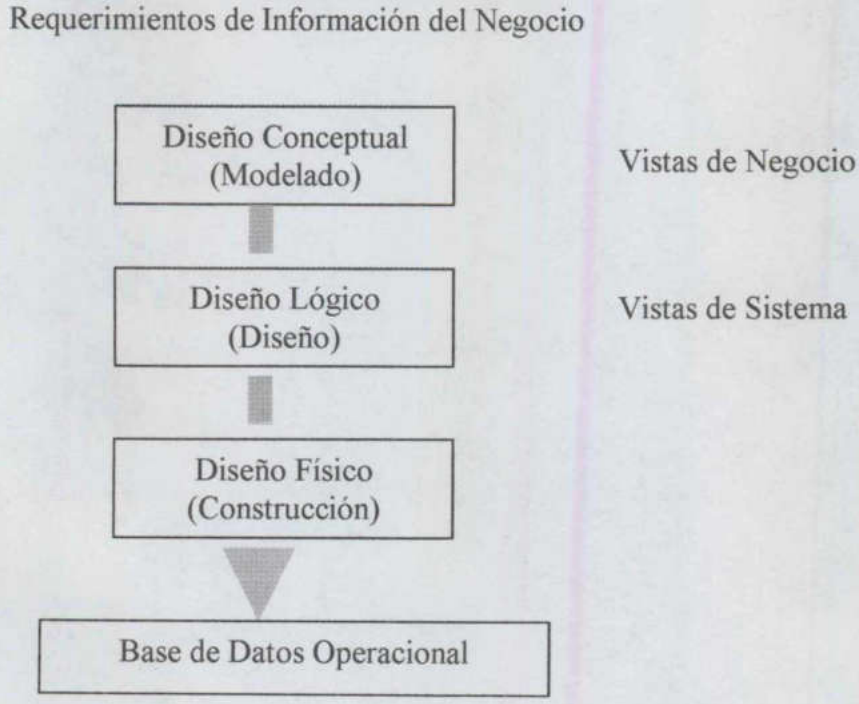


Fig. 1.2.3.2. Diseño de una Base de Datos



1.2.3.1. Diseño conceptual, Diseño lógico y Diseño físico

Diseño conceptual

En esta etapa se debe construir un esquema de la información que se usa en la empresa, independientemente de cualquier consideración física. A este esquema se le denomina esquema conceptual. Al construir el esquema, los diseñadores descubren la semántica (significado) de los datos de la empresa: encuentran entidades, atributos y relaciones. El objetivo es comprender:

1. La perspectiva que cada usuario tiene de los datos.
2. La naturaleza de los datos, independientemente de su representación física.
3. El uso de los datos a través de las áreas de aplicación.

El esquema conceptual se puede utilizar para que el diseñador transmita a la empresa lo que ha entendido sobre la información que ésta maneja. Para ello, ambas partes deben estar familiarizadas con la notación utilizada en el esquema. La más popular es la notación del modelo entidad-relación, que se describirá en el capítulo dedicado al diseño conceptual.

El esquema conceptual se construye utilizando la información que se encuentra en la especificación de los requisitos de usuario. El diseño conceptual es completamente independiente de los aspectos de implementación, como puede ser el SGBD que se vaya a usar, los programas de aplicación, los lenguajes de programación, el hardware disponible o

cualquier otra consideración física. Durante todo el proceso de desarrollo del esquema conceptual éste se prueba y se valida con los requisitos de los usuarios. El esquema conceptual es una fuente de información para el diseño lógico de la base de datos.

Diseño lógico

El diseño lógico es el proceso de construir un esquema de la información que utiliza la empresa, basándose en un modelo de base de datos específico, independiente del SGBD concreto que se vaya a utilizar y de cualquier otra consideración física.

En esta etapa, se transforma el esquema conceptual en un esquema lógico que utilizará las estructuras de datos del modelo de base de datos en el que se basa el SGBD que se vaya a utilizar, como puede ser el modelo relacional, el modelo de red, el modelo jerárquico o el modelo orientado a objetos. Conforme se va desarrollando el esquema lógico, éste se va probando y validando con los requisitos de usuario.

La normalización es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional, ya que asegura que las relaciones (tablas) obtenidas no tienen datos redundantes. Esta técnica se presenta en el capítulo dedicado al diseño lógico de bases de datos.

El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos.

Tanto el diseño conceptual, como el diseño lógico, son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. Ambos se deben ver como un proceso de aprendizaje en el que el diseñador va comprendiendo el funcionamiento de la empresa y el significado de los datos que maneja. El diseño conceptual y el diseño lógico son etapas clave para conseguir un sistema que funcione correctamente. Si el esquema no es una representación fiel de la empresa, será difícil, sino imposible, definir todas las vistas de usuario (esquemas externos), o mantener la integridad de la base de datos. También puede ser difícil definir la implementación física o el mantener unas prestaciones aceptables del sistema. Además, hay que tener en cuenta que la capacidad de ajustarse a futuros cambios es un sello que identifica a los buenos diseños de bases de datos. Por todo esto, es fundamental dedicar el tiempo y las energías necesarias para producir el mejor esquema que sea posible.

Diseño físico

El diseño físico es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos.

Para llevar a cabo esta etapa, se debe haber decidido cuál es el SGBD que se va a utilizar, ya que el esquema físico se adapta a él. Entre el diseño físico y el diseño lógico hay una realimentación, ya que algunas de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico.

En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior. Concretamente, en el modelo relacional, esto consiste en:

Obtener un conjunto de relaciones (tablas) y las restricciones que se deben cumplir sobre ellas.
Determinar las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir unas prestaciones óptimas.
Diseñar el modelo de seguridad del sistema.

1.2.4. Diseño de aplicaciones

En este apartado se examinan los dos aspectos del diseño de las aplicaciones: el diseño de las transacciones y el diseño de las interfaces de usuario.

1.2.4.1. Diseño de transacciones

Una transacción es un conjunto de acciones llevadas a cabo por un usuario o un programa de aplicación, que acceden o cambian el contenido de la base de datos. Las transacciones representan eventos del mundo real, como registrar un inmueble para ponerlo en alquiler, concertar una visita con un cliente a un inmueble, dar de alta un nuevo empleado o registrar un nuevo cliente. Estas transacciones se deben realizar sobre la base de datos para que ésta siga siendo un fiel reflejo de la realidad.

Una transacción puede estar compuesta por varias operaciones, como la transferencia de dinero de una cuenta bancaria a otra. Sin embargo, desde el punto de vista del usuario, estas operaciones conforman una sola tarea. Desde el punto de vista del SGBD, una transacción lleva a la base de datos de un estado consistente a otro estado consistente. El SGBD garantiza la consistencia de la base de datos incluso si se produce algún fallo, y también garantiza que una vez se ha finalizado una transacción, los cambios realizados por ésta quedan permanentemente en la base de datos, no se pueden perder ni deshacer (a menos que se realice otra transacción que compense el efecto de la primera). Si la transacción no se puede finalizar por cualquier motivo, el SGBD garantiza que los cambios realizados por esta transacción son deshechos. En el ejemplo de la transferencia de fondos entre dos cuentas bancarias, si el dinero se extrae de una cuenta y la transacción falla antes de que el dinero se ingrese en la otra cuenta, el SGBD deshará la extracción de fondos.

El objetivo del diseño de las transacciones es definir y documentar las características de alto nivel de las transacciones que requiere el sistema. Esta tarea se debe llevar a cabo al principio del proceso de diseño para garantizar que el esquema lógico es capaz de soportar todas las transacciones necesarias. Las características que se deben recoger de cada transacción son las siguientes:

- Datos que utiliza la transacción.
- Características funcionales de la transacción.
- Salida de la transacción.

- Importancia para los usuarios.
- Frecuencia de utilización.

Hay tres tipos de transacciones:

1. En las transacciones de recuperación se accede a los datos para visualizarlos en la pantalla a modo de informe.
2. En las transacciones de actualización se insertan, borran o actualizan datos de la base de datos.
3. En las transacciones mixtas se mezclan operaciones de recuperación de datos y de actualización.

El diseño de las transacciones utiliza la información dada en las especificaciones de requisitos de usuario.

1.2.4.2. Diseño de interfaces de usuario

Antes de implementar los formularios (manuales) y los informes, hay que diseñar su aspecto. Es conveniente tener en cuenta las siguientes recomendaciones:

- Utilizar títulos que sean significativos, que identifiquen sin ambigüedad el propósito del informe o formulario.
- Dar instrucciones breves y fáciles de comprender.
- Agrupar y secuenciar los campos de forma lógica.
- Hacer que el aspecto del informe o formulario sea atractivo a la vista.
- Utilizar nombres familiares para etiquetar los campos.
- Utilizar terminología y abreviaturas consistentes.
- Hacer un uso razonable y consistente de los colores.
- Dejar un espacio visible para los datos de entrada y delimitarlos.
- Permitir un uso sencillo y adecuado del cursor.
- Permitir la corrección carácter a carácter y de campos completos.
- Dar mensajes de error para los valores ``ilegales".
- Marcar los campos que sean opcionales.
- Dar mensajes a nivel de campo para explicar su significado.
- Dar una señal que indique cuándo el informe o formulario está completo.

1.2.5. Administración de datos y de la base de datos

El administrador de datos y el administrador de la base de datos son las personas o grupos de personas encargadas de gestionar y controlar todas las actividades que tienen que ver con los datos de la empresa y con la base de datos, respectivamente.

El administrador de datos es quien entiende los datos y las necesidades de la empresa con respecto a dichos datos. Su trabajo es decidir qué datos deben almacenarse en la base de datos y establecer políticas para mantener y gestionar los datos una vez hayan sido almacenados.

Un ejemplo de tal política sería una que estableciera quién puede realizar qué operaciones sobre qué datos y en qué circunstancias.

La persona (o personas) que se encarga de implementar las decisiones del administrador de datos es el administrador de la base de datos. Su trabajo es crear la base de datos e implementar los controles necesarios para que se respeten las políticas establecidas por el administrador de datos. El administrador de la base de datos es el responsable de garantizar que el sistema obtenga las prestaciones deseadas, además de prestar otros servicios técnicos.

El administrador de datos juega un papel más importante que el administrador de la base de datos en las siguientes etapas del ciclo de vida: planificación de la base de datos, definición del sistema, recolección y análisis de los requisitos, diseño conceptual y diseño lógico de la base de datos. En el resto de las etapas es donde el administrador de la base de datos tiene el papel más importante: selección del SGBD, diseño de las aplicaciones, diseño físico, prototipado, implementación, conversión y carga de datos, prueba y mantenimiento.

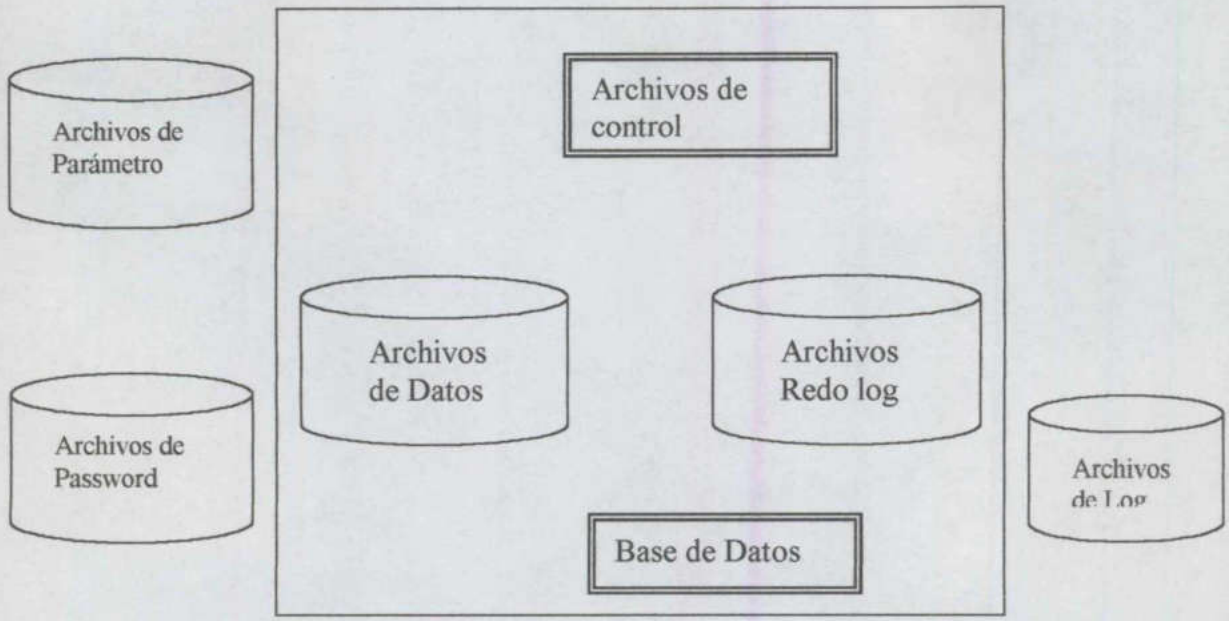
2 ESTRUCTURA DE UNA BASE DE DATOS ORACLE

2.1. Tablas y usuarios

2.1.1. Tablas

En este capítulo se van a tratar dos temas fundamentales en toda base de datos Oracle: los usuarios o esquemas y las tablas. Se explicará todo muy por encima ya que el objetivo de este material no es mostrar cómo consultar o crear datos en una base de datos Oracle, sino conocer los elementos que soportan una base de datos Oracle y que será lo que se explica con profundidad más adelante. Además, se parte de la idea de que se conoce SQL y la herramienta SQLPLUS, desde la cual se ejecutan todas las sentencias para la creación de usuarios, tablas, tablespaces, datafiles, etc.

Fig. 2.1.1.1. Estructura Básica de una Base de Datos Oracle



La función básica de una base de datos es la de almacenar información. Esta información se almacena en unas unidades lógicas llamadas tablas, tal como muestra la figura anterior *(Fig. 2.1.1.1.)*.

La forma en que se almacenan los datos en las tablas está estructurada de manera que resulte muy sencillo su acceso y en cada tabla guardaremos los datos que tienen relación o que definen una idea del mundo real. Por ejemplo, podemos crear una tabla para almacenar la información de los empleados de una empresa. A esta tabla la podemos llamar empleados y, por cada empleado podemos querer guardar información de su edad, sus años de antigüedad, su nombre y sus apellidos.

Cuando creamos una tabla de Oracle, hay que indicar su nombre, que en nuestro caso va a ser empleados, y qué información y de qué tipo vamos a guardar en ella, siendo en nuestro caso la edad que será un número, la antigüedad que será también un número, su nombre que será un dato carácter y su apellido que serán también caracteres.

Una sintaxis básica para crear esta tabla puede ser la siguiente:

```
Create table empeados (edad number, antigüedad number, nombre
varchar2(30),
apellido varchar2(30));
```

¿Cómo se almacena la información en una tabla?.

Cuando creamos una tabla, ésta se crea vacía, es decir, no contiene datos. Para crear los datos se deben ir ejecutando sentencias de manipulación de datos. La instrucción básica para crear un nuevo registro es la INSERT y con ella lo que hacemos es crear un nuevo empleado con sus datos en la tabla de empleados. Existen 3 operaciones básicas más que se pueden realizar sobre las tablas. La sentencia UPDATE se utiliza para modificar los valores de algún registro o fila ya existente, por ejemplo, si hemos insertado en la tabla de empleados a un empleado el año pasado, este año tendremos que aumentarle en uno su edad por lo que habrá que hacer una modificación a ese empleado, a esa fila.

Para borrar registos de una tabla porque, por ejemplo, el empleado se ha cambiado de empresa, se utiliza la sentencia DELETE.

Y finalmente, para consultar datos de nuestra tabla de empleados, la sentencia utilizada es la SELECT. Con ella podremos hacer todas las preguntas que se nos ocurran a nuestra base de datos sobre los empleados que tenemos. Con la definición de la tabla que hemos creados, podremos hacer preguntas como ¿qué nombre tiene el empleado de mayor edad?, ¿cual es la antigüedad del empleado de nombre 'X'?, etc.

Ahora que tenemos una idea sobre qué es una tabla, vamos a ver algunas operaciones que nos permiten modificar su estructura, que no es lo mismo que modificar sus datos.

Por ejemplo, imaginemos que nos damos cuenta de que queremos guardar de nuestros empleados también el departamento de la empresa al que pertenecen. Tal y como hemos creado la tabla, no hemos recogido esta información. Por lo tanto, hay que modificar su estructura para que podamos añadir este dato a las filas que ya existen, a los empleados que ya tenemos y para que a los próximos empleados que insertemos les podamos dar directamente ya su número de departamento.

En estos momentos nuestra tabla tiene cuatro campos o atributos, que son la edad, la antigüedad, el nombre y el apellido. Ahora, vamos a añadir un nuevo campo para que podamos almacenar también el número del departamento de cada empleado.

```
Alter table empleados (add departamento number(8));
```

Con esta sentencia hemos creado por cada registro que existía en nuestra tabla, es decir, por cada empleado que habíamos metido en nuestra tabla, un espacio vacío con el nombre de "departamento" para que pongamos en él su número de departamento. Insistimos, el espacio creado está vacío.

También podemos borrar la definición de una tabla junto con todos sus datos, que no es lo mismo que borrar simplemente todos sus datos. Para borrar esta definición, podemos usar la sentencia drop.

```
Drop table empleados;
```

2.1.2. Usuarios

La unidad básica de almacenamiento de una base de datos Oracle es la Tabla, sin embargo, para tener una mejor estructuración de la información dentro de la base de datos Oracle, las tablas se agrupan a su vez dentro de los Usuarios, llamados también Esquemas. Por lo tanto, un usuario puede tener cero o muchas tablas y se dice que es el propietario de dichas tablas. Además, una tabla pertenece a un solo usuario o esquema.

Cada vez que se crea una base de datos nueva, hay una serie de elementos que no pueden faltar en ella y siempre se crean. Los dos principales elementos que se crean son el usuario SYS y el usuario SYSTEM. Toda base de datos Oracle tiene siempre estos dos usuarios.

¿Por qué se crean estos dos usuarios automáticamente?
Para poder gestionar la base de datos recién creada, el sistema Oracle necesita tener información sobre las tablas que existen en la base de datos, los usuarios que existen, los índices que se van creando y borrando, la cantidad de datos que hay en cada tabla, etc. Por lo tanto, necesita unas tablas en las que ir almacenando toda esta información. A este conjunto de tablas se le llama diccionario de la base de datos y, como hemos dicho, toda tabla de una base de datos Oracle debe pertenercer a un usuario, por eso se crean siempre estos dos usuarios especiales, SYS y SYSTEM que son los propietarios de las tablas del diccionario de la base de datos y, por lo tanto, son lo más importante para que funcione correctamente la base de datos. Si por algún error se borraran tablas de alguno de estos usuarios especiales, se podría corromper toda la base de datos.

Lo normal en los proyectos informáticos es que, una vez que se crea una base de datos Oracle vacía, es decir, solamente existen estos dos usuarios con sus tablas, se crean nuevos usuarios y en cada uno de esos usuarios se van creando las tablas necesarias para cada proyecto.

Así, si en nuestra base de datos tenemos que crear dos aplicaciones totalmente distintas, una para llevar un registro de los empleados de nuestra empresa y con sus sueldos y primas, y otra con los artículos que vendemos, con los clientes que nos compran y con los proveedores que nos reaprovisionan, podemos crear dos usuarios distintos, a uno lo podemos llamar "contabilidad" y al otro "ventas", y dentro de cada uno ir creando las tablas que vamos necesitando para cada proyecto.

Para crear un nuevo usuario, se le debe indicar un nombre, un password o contraseña, un tablespace por defecto en el que se crearán todas las tablas de dicho usuario y un tablespace temporal en el que se ejecutarán las select que necesitan de ordenaciones. Estos conceptos se irán aclarando más adelante. La sentencia podría ser como la que sigue:

```
Create user nombre_de_usuario  identified by pasword_de_usuario
default tablespace
nombre_tablespace_default          temporary          tablespace
nombre_tablespace_temporal;
```

Si por cualquier motivo queremos borrar un usuario deberemos usar el comando drop, pero si ya hemos creado tablas en este usuario, Oracle no nos dejará, nos indicará este hecho y, si queremos borrar el usuario y todas sus tablas debemos añadir la coletilla "cascade" a la sentencia.

```
Drop user nombre_de_usuario;
```

2.2. Tablespaces

2.2.1. Introducción

En primer lugar vamos a dar a conocer muy por encima las unidades básicas que forman una base de datos. Estas unidades son los tablespaces y los datafiles.

Una base de datos está formada por una o varias unidades lógicas llamadas tablespaces. Además, cada una de estos tablespaces está formado por uno o varios ficheros físicos que son los datafiles. Un datafile solamente puede pertenecer a un tablespace. Por lo tanto, los datafiles de una base de datos son todos los datafiles que forman parte de todos los tablespaces de la base.

Cuando se crea una base de datos, hay que crear al menos un tablespace, por lo que durante el proceso de creación de la base de datos siempre se indica el tablespace principal de ésta, que se llama SYSTEM.

De igual manera, cuando se crea un tablespace que, como hemos dicho, es una unidad lógica, se debe indicar obligatoriamente también el nombre de al menos un datafile que formará parte de ese tablespace. El datafile es un fichero físico al que le tendremos que asignar un directorio, un nombre y un tamaño.

2.2.2. El Tablespace System

Cuando se crea una base de datos es obligatorio crear un tablespace inicial en el que se van a crear los usuarios SYS y SYSTEM automáticamente. Estos usuarios son los que tienen la información necesaria para que funcione nuestra base de datos y podamos hacer todo tipo de operaciones como, por ejemplo, crear nuevos usuarios o crear nuevos tablespaces y tablas en esos nuevos tablespaces.

Este tablespace inicial se llama por defecto SYSTEM. Es una pieza clave para un buen funcionamiento de la base de datos ya que en él residen todos los objetos de los usuarios SYS y SYSTEM.

Es muy recomendable crear al menos otro tablespace nuevo distinto al SYSTEM. Así, todos los nuevos usuarios que creemos en nuestra base de datos, junto con todas sus tablas e índices se almacenarán en un tablespace diferente a SYSTEM. Se realiza esta separación para evitar que se bloquee toda la base de datos si ocurre algo grave en el tablespace SYSTEM. Suele ser habitual que para nuestras aplicaciones creemos usuarios y tablas en las que introducimos información y que sin darnos cuenta se llene de información el tablespace en el que están estas tablas. Si no hemos sido previsores, podemos haber llenado el tablespace SYSTEM con lo que es posible que se paralice toda la base de datos.

2.2.3. Manipulando Tablespaces

Ahora que nos hemos hecho una idea acerca de qué es un tablespace, vamos a realizar sobre él las manipulaciones básicas.

Partimos de una base de datos creada y levantada. Nos conectaremos a la misma con el usuario SYSTEM y su contraseña. La contraseña del usuario SYSTEM al crear la base de datos es, por defecto, MANAGER. Como medida de seguridad se recomienda cambiarla cuanto antes. Por lo tanto nos conectaremos bien al SqlPlus mediante sqlplus system/manager, o bien al server manager mediante el comando svrmgrl system/manager.

Crear un Tablespace.

En primer lugar vamos a crear un tablespace llamado Prueba. Esto lo podemos hacer por ejemplo desde el SQLPLUS conectados como system.

Create tablespace prueba datafile '/users/oradata/orcl/prueba01.dbf' size 100M;

Con esta sentencia estamos creando en nuestra base de datos un tablespace nuevo llamado "prueba" y que está formado fisicamente por un fichero (datafile) llamado prueba01.dbf de 100 Mbytes y que está en el directorio "/users/oradata/orcl". Esta sentencia crea fisicamente dicho fichero.

Aumentar de tamaño un Tablespace.

Para aumentar el tamaño de un tablespace que se nos ha quedado ya pequeño, tenemos varias posibilidades. La primera de ellas es crear un nuevo datafile y asignárselo al tablespace que queremos aumentar. Esto lo podemos hacer con la instrucción siguiente.

```
Alter tablespace prueba add datafile
'/users/oradata/orcl/prueba02.dbf' size 50M;
```

Con esta sentencia hemos creado un nuevo fichero físico en nuestro directorio /users/oradata/orcl de 50 Mbytes de tamaño y se lo hemos asignado al tablespace "prueba".

Otra posibilidad es ampliar el tamaño de uno de los ficheros físicos o datafiles que forman el tablespace. Esto lo podemos hacer fácilmente con la siguiente instrucción:

```
Alter datafile '/users/oradata/orcl/prueba01.dbf' resize 150M;
```

Con esta sentencia lo que hacemos es aumentar el datafile que forma parte de nuestro tablespace en 50 Mbytes.

Tanto en la instrucción de creción como en la de aumentar el tamaño de un tablespace se puede observar fácilmente cómo un datafile pertenece solamente a un tablespace ya que en la propia sentencia se crea el fichero físico o datafile.

Borrando un tablespace.

Para eliminar un tablespace de la base de datos se debe utilizar la sentencia:

```
Drop tablespace prueba;
```

2.2.4. Tablespaces Online y Offline

Un tablespace puede estar en dos estados: Online y Offline. Que un tablespace esté online significa que está disponible para operar en él, mientras que si está offline quiere decir que no se puede utilizar. Cuando creamos un tablespace, se crea en estado online y, por lo tanto, podemos crear en dicho tablespace objetos como índices, tablas, etc.

¿Cómo sabemos en qué estado se encuentran nuestros tablespaces?.

Existe una vista que nos da información sobre los tablespaces de nuestra base de datos. Esta vista es la dba_tablespaces. Consultándola podemos conocer qué tablespaces tenemos en nuestra base de datos y en qué estado se encuentran.

select tablespace_name, status from dba_tablespaces;

¿Para qué queremos poner un tablespace offline?.

Hay que tener en cuenta que cuando un tablespace está offline, no se puede acceder a ningún objeto que se encuentre en él, es decir, que si en el tablespace hay tablas, no se podrá hacer consultas ni inserciones ni modificaciones de estas tablas, sin embargo, el resto de los objetos que se encuentran en otros tablespaces de la base de datos si están accesibles. Por lo tanto, pondremos un tablespace offline en general para realizar tareas administrativas.

- Para poder hacer una copia de seguridad del tablespace estando completamente seguros de que nadie está modificando los objetos del tablespace y que no quedan transacciones pendientes sin terminar y que pueden modificar estos objetos.
- Para poder actualizar una aplicación que se basa en los objetos de este tablespace sin que ningún usuario pueda modificar los datos en medio de la actualización.

En un tablespace puede haber objetos de varios tipos, como hemos indicado. Si en un tablespace existen segmentos de rollback activos, no se puede poner offline, primero hay que desactivar los segmentos de rollback activos del tablespace.

¿Cómo sabemos los rollback segments que existen en un tablespace y su estado?
Muy sencillo, con la siguiente sentencia:

```
select rollback_segment, status, tablespace_name from
dba_rollback_segs;
```

Así podremos ver todos los rollback que tenemos, en qué estado se encuentran (online, offline) y en qué tablespace están. Si comprobamos que en el tablespace que vamos a poner offline tenemos algún segmento de rollback online (activo), debemos ponerlo offline antes que el tablespace. Para desactivar un segmento de rollback, ejecutaremos la siguiente sentencia desde el SqlPlus o desde el server manager.

```
alter rollback segment nombre_de_segmento offline;
```

Cuando ya no queden segmentos de rollback en estado online en nuestro tablespace, ya podremos desactivarlo para que no se pueda acceder a él.

```
alter tablespace nombre_de_tablespace offline;
```

Finalmente, cuando terminemos nuestras tareas administrativas sobre dicho tablespace, ya podemos activarlo para que todos sus objetos vuelvan a estar accesibles por los usuarios.

```
alter tablespace nombre_de_tablespace online;
```

Por supuesto, no debemos olvidar que si hemos tenido que desactivar algún segmento de rollback que se encontraba en nuestro tablespace, ahora deberemos volver a activarlo.

alter rollback segment nombre_de_segmento online;

Una curiosidad sobre los tablespaces que no están disponibles (offline), es que, como ya hemos comentado, no se pueden realizar consultas ni modificaciones ni inserciones en los datos de las tablas que están en ellos, pero si que se pueden eliminar objetos de dicho tablespace, que no es lo mismo que borrar datos de objetos del tablespace.

También es muy habitual que en el diseño de las bases de datos, se creen tablespaces para almacenar los índices de la aplicación y otros distintos para almacenar las tablas o datos. En estos casos, si desactivamos el tablespace en el que se encuentran los índices, podemos seguir accediendo a las tablas y realizar consultas sobre ellas porque su tablespace está accesible.

Por otro lado, es posible que si en servidor Oracle se encuentra con graves problemas para escribir en un tablespace, al cabo de varios intentos lo ponga automáticamente offline.

Queremos apuntar aquí, aunque se escapa a los objetivos de este manual, que cuando un tablespace está offline, la información de que esto ha ocurrido se queda almacenadada en el tablespace SYSTEM de esta base de datos. Existe una forma de transportar un tablespace de una base de datos a otra para tener accesibles sus objetos en la segunda, pero si el tablespace está offline en la primera, nunca podrá ponerse online en la base de datos destino ya que, como hemos dicho, la información del estado de este tablespace se encuentra en el tablespace SYSTEM de la base de datos originaria del tablespace.

2.2.5. Tablespaces Read Only

Cuando creamos un tablespace, podemos crear en él todos los objetos que queramos y acceder a ellos y eliminarlos y también consultar los datos de las tablas que se encuentren en este tablespace, así como borrar, insertar y modificar estos datos. Existe la posibilidad de poner un tablespace en un estado en el cual, solamente se pueden consultar los datos de los objetos, no se puede ni borrar ni insertar nada en ellos.

¿Para qué me viene bien un tablespace read only?.
La principal ventaja de un tablespace read only es que, como no se pueden modificar los datos que en él se encuentran, no hace falta hacer backup del mismo. Dependiendo de las aplicaciones que tengamos en nuestra base de datos nos puede interesar tener tablespaces read only o no. Por ejemplo, si tenemos una aplicación en la que se pueden consultar cientos de fotos de animales salvajes o de paisajes, podríamos crear un tablespace en el que introducir estas imágenes y luego ponerlo read only.

Generalmente un tablespace de estas características, que sirve de almacenamiento de fotos o temas similares, suele ocupar mucho espacio, por lo que hacer un backup del mismo todos los días puede resultar muy costoso en tiempo y espacio. Además, si no se modifican nunca estas fotos no tiene mucho sentido hacer copia de seguridad del mismo, y no solo eso, podríamos incluso almacenar dicho tablespace en un CDROM en vez de ocupar espacio en disco.

Para poner un tablespace en estado read only, simplemente debemos ejecutar la siguiente instrucción:

```
alter tablespace nombre_de_tablespace read only;
```

Como hemos indicado, en un tablespace read only solo se pueden realizar consultas de los datos, por lo tanto, si en el instante de ejecutar esta sentencia se están realizando modificaciones o inserciones o borrado de datos, el servidor espera hasta que acaben para poner el tablespace en estado read only. Para ver si ha quedado en estado read only, simplemente ejecutamos la misma select que al principio para ver la información general de los tablespaces:

```
select tablespace_name, status from dba_tablespaces;
```

Si por algún motivo necesitamos modificar los datos que se encuentran almacenados en espace un tablespace read only, simplemente deberemos ponerlo en primer lugar en estado read write y una vez realizada la modificación, volver a ponerlo en su estado read only. La sentencia que debemos ejecutar será:

```
alter tablespace nombre_de_tablespace read write;
```

Tenemos un concepto que debe quedar claro. Un tablespace read only no necesita backup, y por tanto, recovery, pero, esto no hay que tomarlo al pie de la letra. Siempre hay que hacer al menos un backup. En primer lugar creamos un tablespace vacío en el que iremos metiendo poco a poco toda la información que nos interesa que, como en el caso que hemos supuesto anteriormente, pueden ser varias tablas que almacenan fotos de animales y paisajes. Cuando ya no vamos a crear nuevas imágenes es cuando ponemos el tablespace read only, pero ahí si debemos hacer una copia de seguridad, backup, y como ya no vamos a tocar nunca más este tablespace será la única. Si por algún motivo decidimos poner este tablespace otra vez read write para crear o borrar datos, después de volver a ponerlo read only deberemos hacer un backup de los nuevos datos.

También hay que diferenciar dos ideas. Por un lado hemos dicho que en un tablespace read only no se pueden modificar, ni insertar ni borrar datos de sus tablas. Sin embargo y, al igual que en los tablespaces offline, si se pueden borrar objetos enteros del tablespace, como por ejemplo un índice o una tabla.

2.2.6. Tablespaces Temporales

Un tablespace temporal es aquél en el que solamente puede haber objetos temporales. No se pueden crear en él objetos permanentes como pueden ser los índices, las tablas o los segmentos de rollback. Están especialmente preparados para optimizar las operaciones en las que se lleven a cabo ordenaciones. Por lo tanto está muy recomendado tener al menos un tablespace temporal en cada base de datos. Algunas de las operaciones que implican realizar ordenaciones son, las selects que tienen group by, las que tienen order by, la creación de índices y analizar índices para calcularles las estadísticas. En todos estos casos, cuando para realizar la ordenación el servidor no encuentra espacio suficente libre en la memoria, utiliza el tablespace temporal. Los rendimientos son muy superiores comparándolos con los tiempos

que se emplearía en realizar las ordenaciones en tablespaces normales. Esto se debe a que el mecanismo que se utiliza para reservar y desreservar el espacio en los tablespaces temporales es muy distinto que en los normales ya que está orientado a objetos que crecen mucho y rápido y que a continuación disminuyen totalmente su tamaño y desaparecen.

Para crear un tablespace temporal simplemente hay que añadir la palabra TEMPORARY a la instrucción utilizada para crear tablespaces normales. Siguiendo el ejemplo indicado en la creación de tablespaces, podríamos tener lo siguiente:

```
Create tablespace prueba datafile
'/users/oradata/orcl/prueba01.dbf'
size 100M temporary;
```

Para indicar a un usuario de base de datos que sus ordenaciones debe hacerlas en un determinado tablespace temporal, hay que lanzar una sentencia como la que sigue.

Alter user nombre_de_usuario temporary tablespace nombre_de_tablespace;

Y para conocer qué usuarios existen en nuestra base de datos y cual es el tablespace temporal que utilizan, podemos consultar la base de datos de la siguiente manera:

```
Select username, temporary_tablespace from dba_users;
```

Y finalmente, si queremos conocer qué tablespaces tenemos y cuáles son temporales y cuales son permanentes, podemos consultar la vista que nos da la información sobre los tablespaces, es decir, la vista dba_tablespaces;

```
Select tablespace_name, contents from dba_tablespaces;
```

Como nota final apuntaremos que un tablespace permanente puede pasar a temporal y que uno temporal puede pasar a permanente.

2.3. Datafiles

2.3.1. Introducción

Los datafiles son los ficheros físicos en los que se almacenan los objetos que forman parte de un tablespace. Un datafile pertenece solamente a un tablespace y a una instancia de base de datos. Un tablespace puede estar formado por uno o varios datafiles. Cuando se crea un datafile, se debe indicar su nombre, su ubicación o directorio, el tamaño que va a tener y el tablespace al que va a pertenecer. Además, al crearlos, ocupan ya ese espacio aunque se encuentran totalmente vacíos, es decir, Oracle reserva el espacio para poder ir llenándolo poco a poco con posterioridad. Por supuesto, si no hay sitio suficiente para crear un fichero físico del tamaño indicado, se producirá un error y no se creará dicho fichero.

Fig. 2.3.1.1. Datafiles

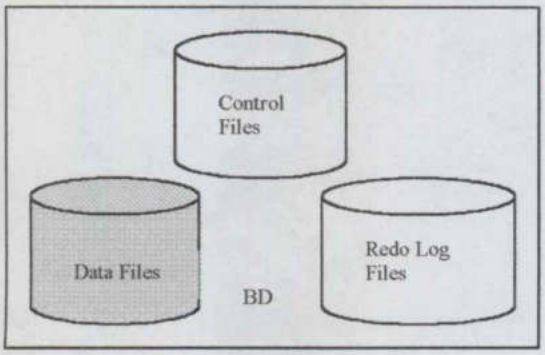
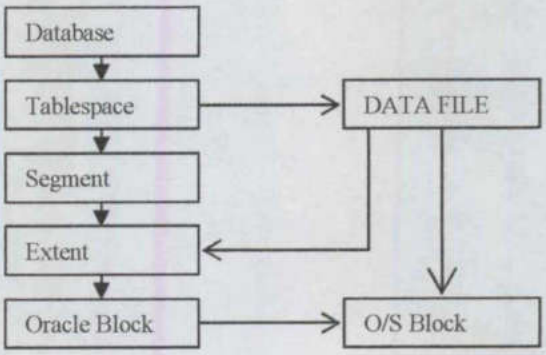


Fig. 2.3.1.2. Relación Datafile



Cuando se van creando objetos en un tablespace, éstos fisicamente se van almacenando en los datafiles *(Fig. 2.3.1.1.)* asignados a dicho tablespace *(Fig. 2.3.1.2.)*, es decir, cuando creamos una tabla y vamos insertando datos en ella, estos datos realmente se reparten por los ficheros fisicos o datafiles que forman parte del tablespace. No se puede controlar en qué fichero fisico se almacenan los datos de un tablespace. Si un tablespace está formado por 2 datafiles y tenemos una tabla en ese tablespace, a medida que vamos insertando filas éstas se almacenarán en cualquiera de los dos datafiles indistintamente, es decir, unas pueden estar en un datafile y otras en otro.

El espacio total disponible en un tablespace es lógicamente la suma de los tamaños que ocupan los ficheros fisicos o datafiles que lo forman. Como hemos indicado estos datafiles, al crearlos, están totalmente vacíos, simplemente es un espacio reservado y formateado por Oracle para su uso. A medida que se van creando objetos en ellos como tablas, índices, etc y se van insertando registros en estas tablas, los datafiles se van llenando o, lo que es lo mismo, el tablespace se va llenando.

2.3.2. Creación y Manipulación

La creación de datafiles está estrechamente relacionada con el tablespace al que va a pertenecer. Tenemos varias formas de crear datafiles. Cada vez que se crea un tablespace nuevo, hay que indicar obligatoriamente cual es el datafile que va a pertenecer a dicho tablespace y, en ese momento, se crea tanto el tablespace como su datafile. También se pueden añadir datafiles nuevos a un tablespace que ya existe. Esto se suele hacer cuando un tablespace se está llenando y está a punto de llegar a su capacidad máxima. Al añadir un datafile a un tablespace, se aumenta el espacio disponible en dicho tablespace en tantos megabytes como tenga el datafile nuevo recién creado.

Creación de un nuevo datafile de 50 megabytes junto con un nuevo tablespace:

```
Create tablespace  nombre_tablespace datafile
'/users/oracle/orcl/nombre_datafile.dbf' size 50M;
```

Una vez creado este tablespace, si con el tiempo queremos añadirle espacio, lo podemos hacer creando un nuevo datafile y asignándoselo al tablespace:

```
Alter tablespace nombre_tablespace add datafile
'/users/oracle/orcl/nombre_datafile2.dbf' size 100M;
```

Con estas dos instrucciones hemos creado un tablespace nuevo en nuestra base de datos en el que caben 150 megabytes de información. Este espacio está formado físicamente por dos ficheros llamados nombre_datafile.dbf y nombre_datafile2.dbf que se encuentran en el directorio /users/oracle/orcl de nuestra máquina y que ocupan 50 y 100 Mbytes respectivamente.

Para conocer los datafiles que forman parte de nuestra base de datos, podemos consultar la vista dba_data_files en la que se nos indica por cada datafile o fichero de datos, a qué tablespace pertenece y cuanto espacio total tiene reservado. Es importante recalcar que el espacio que aparece en esta vista es el espacio total que ocupa el fichero físico y no el espacio utilizado de ese fichero, es decir, que si creamos un datafile de 50Mbytes y acto seguido consultamos esta vista, veremos que ocupa 50Mbytes a pesar de estar totalmente vacío. Este dato indica la cantidad de espacio que ocupa el fichero físico, la cantidad de información que podremos introducir en él.

```
select tablespace_name, file_name, bytes /1024/1024 from
dba_data_files;
```

Tenemos también la posibilidad de aumentar el tamaño de un datafile, es decir, podemos conseguir que un tablespace tenga más sitio vacío aumentando uno o varios de los ficheros físicos que lo forman, en lugar de añadiéndole un nuevo fichero físico. Para aumentar el tamaño de un datafile, podremos utilizar la siguiente instrucción:

```
alter database datafile
'/users/oracle/orcl/nombre_datafile.dbf' resize 100M;
```

Para indicar que queremos que un datafile aumente automáticamente cuando añadimos un nuevo datafile a un tablespace existente podemos utilizar:

```
alter tablespace nombre_tablespace add datafile nombre_datafile
size 100M
autoextend on next 250K maxsize 200M;
```

Con esta instrucción lo que estamos haciendo es añadir un nuevo datafile llamado nombre_datafile a nuestro tablespace nombre_tablespace con 100Mbytes de tamaño. Además, estamos indicando que queremos que aumente por si mismo cada vez que se llene y que aumente en bloques de 250 Kbytes cada vez. Finalmente le ponemos un tope al tamaño total que queremos que tenga nuestro datafile con la instrucción maxsize, por lo que una vez que llegue a 200 Mbytes, si se llena, no volverá a crecer más.

Para indicar en cualquier momento que queremos que un datafile no crezca más automáticamente, podemos utilizar:

```
alter database datafile nombre_datafile autoextend off;
```

Nota: en esta sentencia, se puede indicar que queremos que crezca indefinidamente, sin tome máximo. Esto lo conseguimos con "maxsize unlimited", pero es muy peligroso porque por algún problema descontrolado, nos puede crecer tanto que nos quedemos si disco en la máquina y luego es muy complicado restaurar un tamaño normal.

2.3.3. Renombrando Datafiles

Existe la posibilidad de cambiarle el nombre a un datafile o de cambiarlo de directorio. Esta operación no consiste simplemente en ir al sistema operativo y cambiarle el nombre, ya que si hiciéramos eso, Oracle no se da cuenta de que hemos movido de sitio un datafile y cuando intenta acceder a información de ese datafile muestra mensajes de error indicando que no lo encuentra.
Hay que distinguir entre los datafiles del tablespace SYSTEM y el resto. Los datafiles del tablespace SYSTEM son especiales y no se pueden mover con la misma facilidad que los demás.

Renombrando datafiles que no son del tablespace SYSTEM

En primer lugar, hay que comprobar cual es nombre y el path completo del fichero a mover y el estado en que se encuentra dicho fichero. Para realizar esta comprobación podemos consultar la vista dba_data_files.

```
select file_name, status, bytes from dba_data_files;
```

En file_name se nos indica cual es el nombre del datafile que nos interesa, con todo su path, y además vemos cuanto ocupa. El campo status podremos comprobar si el datafile está disponible (available).

Nota: No se debe mover el datafile físico sin antes poner el tablespace offline.

Hay que señalar que file_name es el nombre que Oracle cree en ese mismo instante que tiene su datafile. Si vamos al sistema operativo y movemos el datafile de sitio, Oracle no es consciente de lo que hemos hecho por lo que si volvemos ha realizar esta select nos seguirá dando los mismos valores. Hay que conseguir decirle a Oracle que realmente hemos movido o renombrado el fichero.

Ahora que sabemos cual es el path y nombre completo de nuestro datafile, tenemos que evitar que se realicen operaciones que modifiquen los datos de los objetos de nuestro tablespace, para que así consigamos tener el contenido del datafile estático. Esto se consigue poniendo el tablespace en estado read only, como se explicó en el tema de los tablespaces.

```
alter tablespace nombre_tablespace read only;
```

Para comprobar que realmente está nuestro tablespace en estado read only, podemos consultar la vista dba_tablespaces. En estos momentos, los usuarios de la base de datos, pueden acceder y modificar la información de cualquier tablespace que no sea el que estamos manipulando, en el cual, solamente podrán realizar operaciones de lectura, nunca inserciones ni modificaciones ni borrados de datos.

Es en este instante, cuando sabemos que no se está modificando el contenido de nuestro tablespace y, por lo tanto, de nuestro datafile, cuando debemos ir al sistema operativo y hacer una copia de nuestro datafile con el nuevo nombre y la nueva ubicación. Una vez copiado, comprobamos también desde el sistema operativo que el nuevo datafile ocupa el mismo espacio que el antiguo, para estar seguros de que no ha habido ningún problema en la copia.

Hasta ahora, no le hemos indicado a Oracle que hemos movido de ubicación o de nombre a uno de sus datafiles, para poder indicárselo, debemos asegurarnos que no hay ningún usuario utilizando el tablespace, ni siquiera en modo consulta. Por lo tanto, debemos deshabilitar el tablespace.

```
alter tablespace nombre_tablespace offline;
```

Y una vez deshabilitado, indicamos a Oracle el cambio de nombre o de ubicación:

```
alter database rename file 'viejo_datafile_con_path' to
'nuevo_datafile_con_path';
```

En estos momentos Oracle ya sabe que cuando tenga que buscar la información de ese datafile debe buscarlo en el nuevo path indicado y con el nuevo nombre. Por lo tanto, si lanzamos la select para ver los datafiles de la base de datos, es decir, la select de la vista dba_data_files, comprobaremos que ha cambiado la información antigua por la nueva. Ahora solamente nos queda activar el tablespace y permitir operaciones de lectura y escritura en él.

```
alter tablespace nombre_tablespace online;
alter tablespace nombre_tablespace read write;
```

Por supuesto, antes de realizar cualquier operación que implique modificación de las estructuras de la base de datos, como el renombrado de un datafile, se debe hacer un backup completo de la misma. Una vez realizada la operación también se recomienda hacer un nuevo backup.

Nota: hay que resaltar una vez más, que no se debe mover el datafile desde el sistema operativo sin haber puesto con anterioridad su tablespace offline. De no ser así, si alguien manipula datos durante el tiempo que tarda en hacerse la copia en el sistema operativo,

Oracle detecta problemas e invalida el datafile, lo que va a provocar que haya que poner en práctica alguna estrategia de backup para recuperar el datafile invalidado.

Renombrando datafiles del tablespace SYSTEM

El tablespace SYSTEM es especial, por lo tanto, para manipular sus datafiles, hay que hacerlo también de manera especial. Nadie puede trabajar con la base de datos. Por ese motivo, se debe apagar la base de datos y levantarla pero sin abrirla. Los conceptos de apagar la base de datos y levantarla no son objeto de este manual por lo que simplemente se indicarán las instrucciones.

Primeramente se debe apagar o, más coloquialmente, tirar abajo la base de datos. Esto lo hacemos desde el Server Manager, no desde SqlPlus. Nos conectamos al Server Manager como el usuario administrador y con privilegios especiales:

```
svrmgrl
connect internal
shutdown;
```

Después de esperar a que se terminen las transacciones activas, las base de datos se apaga y podemos volver a levantarla, también desde el Sever Manager, pero sin abrirla, solamente montándola.

```
startup mount;
```

Con esta instrucción hemos levantado la base de datos pero no la hemos abierto, por lo que nadie, excepto otro administrador, puede estar manipulando sus objetos. Ahora podemos realizar la copia de los datafiles del tablespace SYSTEM al nuevo directorio o con el nuevo nombre. Comprobamos que tanto el fichero nuevo como el antiguo tengan el mismo tamaño y a continuación indicamos a Oracle que hemos movido el datafile de la misma manera que en el apartado anterior:

```
alter database rename file 'viejo_datafile_con_path' to
'nuevo_datafile_con_path';
```

Finalmente podemos levantar la base de datos para que pueda volver a ser utilizada por todos los usuarios:

```
alter database open;
```

Nota: Después de comprobar que la base de datos se levanta correctamente, se pueden borrar los ficheros físicos o datafiles viejos de su ubicación antigua ya que Oracle está utilizando solamente los nuevos.

2.4. Data Blocks - Bloques

2.4.1. Introducción

Oracle almacena la información en unidades lógicas que son los segmentos, las extensiones y los bloques. Estas tres unidades están relacionadas entre sí. Un segmento está formado por una o varias extensiones y cada extensión está formado por varios bloques.

Un bloque es la unidad mínima de almacenamiento de información de Oracle. A los bloques también se les conoce como "bloques de datos", "bloques lógicos" o "bloques oracle". Cada uno de estos bloques está formado por un número determinado de bloques del sistema operativo. A la hora de crear una nueva base de datos se debe indicar cuántos bloques de sistema operativo formarán un bloque de datos o bloque oracle. Es muy importante decidir bien este valor de antemano ya que una vez creada la base de datos ya no se puede modificar más que en migraciones a versiones más actuales del producto.

Un bloque de datos es la mínima unidad de Lectura / Escritura en una base de datos Oracle, es decir, Oracle no lee y escribe en bloques del sistema operativo sino que lo hace en unidades lógicas que son los bloques de datos y que varían de una base de datos a otra en la misma máquina ya que es un valor que se debe indicar en la creación de cada base de datos Oracle.

Oracle recomienda que el tamaño de un bloque de datos o, data block, sea siempre un múltiplo del bloque de datos del sistema operativo.

2.4.2. Estructura de un bloque

Los bloques de base de datos pueden contener información de tablas, índices o segmentos de rollback, pero no importa qué información contengan, siempre tienen la misma estructura, que es la mostrada en la siguiente figura *(Fig. 2.4.2.1.)*.

Fig. 2.4.2.1. Estructura de un bloque

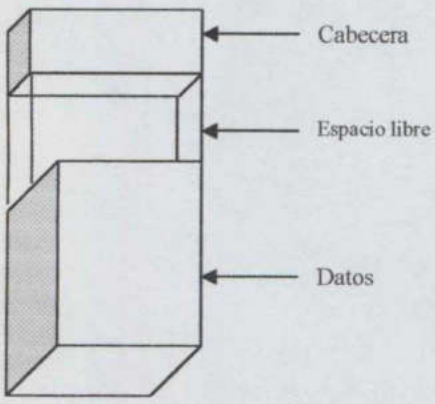
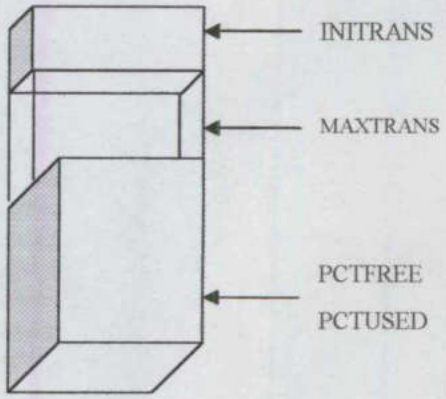


Fig. 2.4.2.2. Parámetros de un bloque



Todo bloque de datos o, data block, está dividido en una cabecera, en un directorio de tablas que utilizan dicho bloque, en un directorio de las filas que se encuentran almacenadas en ese bloque, de espacio aún libre y de las filas de datos de las tablas, índices o segmentos de rollback. Al espacio ocupado por la cabecera más el directorio de tablas y más el directorio de filas se le llama overhead ya que es un espacio del bloque que realmente no se rellena con datos sino que está ocupado por la información que necesita Oracle para saber exactamente qué datos tiene en dicho bloque.

Cabecera:
Contiene información general sobre el bloque como el tipo de segmento al que pertenece (índice, tabla, rollback) o la dirección del bloque.

Directorio de Tablas:
Contiene información acerca de las tablas que tienen datos en el bloque.

Directorio de Filas:
Contiene información sobre las filas que se encuentran en cada momento en el bloque. Esta información incluye la dirección de la fila dentro de la subzona "Datos de Filas" del bloque en la que debe buscar Oracle los datos.

El espacio ocupado por esta subzona va aumentando a medida que se insertan nuevas filas en el bloque, sin embargo nunca se libera el espacio. Si se borran filas de datos que estaban en el bloque, en el directorio de filas desaparecerá la entrada que apuntaba a ellas, pero el espacio permanecerá reservado aunque vacío. A medida que se insertan nuevas filas de datos en el bloque, también se insertan registros en el Directorio de Filas, pero antes de aumentar el tamaño de esta subzona para la nueva entrada, se comprueba si alguna de las entradas que hay está vacía y en ese caso se "ocupa" y no hace falta que crezca más la subzona.

Espacio Libre:
Esta subzona está reservada para la inserción de nuevas filas en el bloque o, para la modificación de campos que requieren más espacio que el que tenían con anterioridad. Esto último ocurre, por ejemplo, con los campos que son de tipo varchar2. Si en un campo de una tabla tenemos un varchar2 de 30 caracteres para almacenar el nombre del empleado, si insertamos un registro con el nombre de 'Jesus', solo ocupa 5 bytes y si posteriormente lo modificamos para poner 'Jesus Maria', se necesitarán 6 bytes más para almacenarlo.

Si en el bloque se están almacenando datos de segmentos de tipo tabla o índice, en la subzona de Espacio Libre también se utiliza para llevar un registro de las transacciones que en cada momento acceden a datos del bloque. Se necesita una entrada de transacción siempre que se realice una insert, update, delete o select for update sobre filas del bloque. El tamaño necesario para cada una de las entradas de transacciones depende del sistema operativo.

Datos de Filas:
En esta subzona se almacenan los datos de las tablas o de los índices del bloque. Se puede dar el caso de que una fila no entre completa en el bloque y tenga que ocupar más de un bloque.

Este caso especial se comentará más a fondo en el apartado de encadenamiento y migración de filas.

2.4.3. Pctfree

Este parámetro se utiliza para modificar el comportamiento de Oracle a la hora de insertar y modificar filas dentro de un bloque de datos o data block, se asigna a la hora de crear la tabla o índice *(Fig. 2.4.2.2.)*, También se puede modificar posteiormente el valor del pctfree alterando la tabla o el índice.

Este parámetro indica el porcentaje mínimo que se debe dejar libre para modificaciones de los datos de las filas que ya existen dentro del bloque. Hay que tener en cuenta que el espacio de un bloque no está compuesto solamente por los datos, sino que también hay un overhead, por lo que si asignamos a un segmento de tipo tabla un pctfree de 20, no estamos dejando para insercciones el 80% sino el 80% menos lo que ocupe el overhead del bloque.

El concepto de pctfree se entiende mejor con un ejemplo. Si a una tabla le asignamos un pctfree de 20, le estamos diciendo que se pueden insertar filas en el hasta que quede libre en dicho bloque solamente el 20 por ciento. A partir de ese instante, todas las filas nuevas que se creen se crearán en otros bloques ya que en este ya no queda sitio para más. Entonces, ¿qué ocurre con este 20%?. Pues muy sencillo, se utiliza para las modificaciones de las filas que ya están en el bloque. Cuando se modifica una fila y se aumenta el contenido de un campo, por ejemplo, el campo "nombre" tenía el valor 'Jesus' y ahora pasa a tener el valor 'Jesus Maria', Oracle echa mano del espacio libre dentro del bloque para poder realizar esta operación.

Por lo tanto, este espacio podría incluso llenarse lo cual, en caso de seguir haciendo modificaciones de este estilo, podría generarnos filas migradas, como se explica más adelante. Sin embargo, el espacio libre en un bloque también puede aumentar, bien borrando filas o bien haciendo updates que disminuyan el valor de los campos de las filas que existen en dicho bloque. Cuando se hace espacio libre suficiente en el bloque se permite otra vez la inserción de registros en el mismo. El concepto de "espacio libre suficiente" para volver a permitir inserciones en el bloque es lo que se define con el parámetro pctused.

2.4.4. Pctused

El concepto de pctused está directamente relacionado con pctfree *(Fig. 2.4.2.2.)*. Supongamos que se crea un segmento (tabla o índice) y se le asigna un pcfree de 20, por lo que todos sus bloques tendrán dicho pctfree. Como ya hemos explicado, esto quiere decir que podremos insertar filas o registros en uno de sus bloques hasta que se llene al 80 por ciento. A partir de ese momento ya no se pueden insertar nuevos registros hasta que se libere espacio en el bloque, o sea, hasta que vuelva a aumentar el espacio libre.

Llegados a este punto nos hacemos 2 preguntas:

¿Qué hay que hacer para que aumente el espacio libre en un bloque?

Muy sencillo, o bien borrar las filas que están en él o bien modificando campos de esas filas disminuyendo el tamaño de los valores que en ellas están guardados.
¿Cuanto espacio libre tiene que haber para poder volver a insertar nuevas filas?
Este valor nos lo indica el parámetro pctused.
Así, si a un bloque le hemos asignado un pctused de 40, lo que conseguimos es que en un bloque no se puedan insertar nuevos registros (después de haberse llenado hasta dejar solamente el pctincrease de espacio libre) hasta que el espacio ocupado por las filas en dicho bloque no baje por debajo de 40, es decir, que el pctused nos indica el límite mínimo por debajo del cual debe bajar el espacio ocupado dentro del bloque antes de volver a estar disponible para aceptar nuevas filas, nuevas inserciones. Hay que resaltar que en los bloques de los segmentos de tipo índice, no tiene utilidad el parámetro pctused debido a la finalidad de los mismos y a su estructura interna en forma de árbol binario.
Para consultar el valor tanto del parámetro pctfree como del parámetro pctused de cada segmento de tipo tabla o de tipo índice, podemos leer las vistas dba_tables y dba_indexes del usuario SYS.

```
Select owner, table_name, pct_free, pct_used from dba_tables;
Select owner, index_name, pct_free from dba_indexes;
```

Para modificar el valor de los parámetros de una tabla o de un índice se pueden utilizar las siguientes sentencias:

```
Alter table nombre_de_tabla  pctfree nuevo_pct_free;
Alter table nombre_de_tabla pctused nuevo_pct_used;
Alter index nombre_de_indice pctfree nuevo_pct_free;
```

2.4.5. Encadenamiento y Migración de Filas

Existen dos circunstancias diferentes por las cuales puede ocurrir que los datos de una fila recién insertada no tengan espacio suficiente dentro del bloque. Hay que intentar por todos los medios evitar que esto se produzca para que no caiga el rendimiento del sistema ya que cuando hay encadenamiento o migracion de filas, los datos de una fila se dispersan por varios bloques, con lo que para obtener esos datos o para modificarlos Oracle debe recorrer varios bloques que, posiblemente, no estén contiguos.

Encadenamiento de filas:

El encadenamiento o chained rows, se da cuando los datos de una fila ocupan tanto espacio que no caben fisicamente en un solo bloque y Oracle debe guardarlos en dos o más bloques de los reservados para ese segmento. Esto suele ocurrir generalmente cuando se utilizan columnas de tipo long o long raw que pueden almacenar grandes cantidades de espacio, por lo que no caben en un solo bloque.

Migración de filas

Este otro caso se da cuando modificamos los datos de una fila de un bloque, aumentándolos de tamaño, es decir, como en le ejemplo anterior, si teníamos un campo varchar2(30) con el valor

'Jesus' solo ocupaba 5 bytes y si lo modificamos para que contenga 'Jesus Maria' necesita 11 bytes. En este caso, si en la subzona en la que tenemos el espacio libre del bloque no disponemos de espacio suficiente, Oracle mueve o mejor dicho, migra toda la fila a un nuevo bloque en el que si que haya espacio para toda la fila. Sin embargo, para no tener que cambiarle a dicha fila el rowid, es decir, el identificador único de la fila, lo que se hace es dejar en el bloque inicial una información mínima de la fila, que será simplemente un puntero hacia la dirección del nuevo bloque en el que se ha reubicado toda esta fila.

2.5. Extensiones

2.5.1. Introducción

Una extensión es una unidad lógica de almacenamiento que está formada por un número determinado de bloques de datos contiguos. La agrupación de una o varias extensiones forman un segmento que puede ser una tabla, un índice, un segmento de rollback o un segmento temporal. Por lo tanto, datos de una tabla, sabemos que están en un solo segmento de tipo tabla, que a su vez estará formado por una o varias extensiones y que, cada una de esas extensiones está formada por un número determinado de bloques de datos.

Cuando se crea un segmento nuevo, es decir, una tabla, un índice o un segmento de rollback, se crea obligatoriamente una extensión en dicho segmento (en el caso de los rollback se crean dos). El tamaño de esta extensión inicial viene dado por el valor parámetro "initial" que se indica en el momento de crear el segmento.

2.5.2. Asignación de Extensiones

Al crear o, mejor dicho, asignar una nueva extensión al segmento, se está reservando espacio en el disco para almacenar los nuevos datos de dicho segmento. Por lo tanto, al crear la nueva extensión está totalmente vacía y todo su espacio está disponible para almacenar los datos del segmento y, además, en el disco debe haber espacio libre para que Oracle reserve todo el tamaño que necesita la extensión, y lo formatea de forma especial para poder utilizarlo. A partir de ese momento, en esa extensión solamente se podrán almacenar datos del segmento al que pertenece.

Cuando se llenan todos los bloques de datos de una extensión, el segmento solicita una nueva extensión al sistema para poder seguir almacenando información.

¿Cómo podemos determinar el número de extensiones y su tamaño de un segmento?

Para establecer el tamaño de las futuras extensiones que irá solicitando un segmento se utilizan varios parámetros a los que hay que dar valor en el momento de la creación de la tabla, índice o segmento de rollback. Estos parámetros se indican en la claúsula STORAGE de la sentencia que crea el segmento y son los siguientes:

Initial
Indica el tamaño en bytes de la primera extensión que tendrá el segmento. Se puede indicar después del valor una "K" o "M" para que el valor sea interpretado como Kilobytes o Megabytes en lugar de bytes. Si no se pone explícitamente este parámetro en la creación del segmento, se hereda por defecto el valor que tenga este parámetro en el tablespace en el que se está creando el segmento y que, si no se ha indicado tampoco, por defecto son 5 bytes. Cuando se crea una extensión oracle redondea el tamaño indicado al siguiente múltiplo superior a 5 bloques de datos. Por lo tanto, si nuestros bloques son de 8192 bytes y creamos un segmento con un inital de 256Kbytes, realmente estamos creando un segmento de 32 bloques y oracle lo redondea a 35 bloques que es el primer múltiplo superior a 5 de 32. Por lo tanto, nuestra initial extent será de 35 * 8192 = 280 Kbytes.

Para comprobar estos datos se puede consultar la tabla dba_segments en la que tenemos un registro por cada segmento distinto:

Select segment_name, initial_extent, next_extent, pct_increase, min_extent,
max_extent from dba_segments;

También se puede consultar la vista dba_extents que es un detalle de dba_segments ya que aquí se detalla por cada segmento todas sus extensiones, con su tamaño en concreto.

```
Select segment_name, extent_id, blocks, bytes from dba_extents;
```

A pesar de lo que aparece en la documentación de Oracle sobre los redondeos a múltiplos de 5 bloques, nos hemos encontrado con muchos casos en los que creamos segmentos con una sola extensión de 256 Kbytes con bloques de 8192 bytes (con el caso anterior), y en la vista dba_extents, al consultar el valor de initial_extent sigue siendo 256 Kbytes, y en dba_extents, que es donde debería redondearse realmente dicho valor a un múltiplo de 5, algunas extensiones aparecen con 256 Kbytes y otras con los 280 Kbytes que teóricamente deberían tener.

Next
Indica el tamaño que tendrá la próxima extensión que se cree cuando no quede más sitio en las extensiones que ya tiene asignadas el segmento. De igual manera que en el caso del parámetro initial, Oracle redondea a un múltiplo de 5 bloques este valor, a la hora de crear la extensión nueva. Cada vez que se crea una nueva extensión, se recalcula el valor de este parámetro en función del valor de pctincrease y se actualiza la vista dba_segments.

Pctincrease
En el momento que se asigna una nueva extensión al segmento, se recalcula el valor que va a tener la próxima que se le asigne y que será, el tamaño de la extensión recién creada (el next que tenía el segmento) aumentado en el porcentaje indicado por pctincrease. Por lo tanto, si qeremos que todas las extensiones de nuestro segmento tengan el mismo tamaño, habrá que asignarles el pctincrease a 0. Oracle recomienda establecer por norma el pctincrease a 0 para evitar que se descontrolen los tamaños de los segmentos, especialmente los temporales,

aunque, curiosamente, su valor por defecto es de 50%. No se puede modificar el pctincrease de los segmentos de rollback que es siempre 0.

Mostremos un ejemplo: tenemos un segmento que en el campo next_extent tiene como valor 262144 y en pct_increase tiene 50 y los bloques son de 8192 bytes. Cuando crezca el segmento y solicite una nueva extensión, ésta se creará del tamaño indicado en next_extent redondeándola al primer múltiplo de 5 bloques superior o igual a dicho valor. En nuestro caso 262144 son 32 bloques así que creará una extensión de 35 que son 286720 bytes. Además, recalcula el valor del siguiente next_extent aumentando en un 50% el valor del antiguo next_extent, es decir 262144 * 1,5 = 393216 bytes.

Nota: el recálculo del siguiente extent (393216) se basa en el valor del anterior next_extent (262144) y no en el valor de la extensión creada al redondear a un múltiplo de 5 bloques (286720), ya que si no, nos habría salido un next_extent de 286720 * 1,5 = 430080. Por lo tanto, al consultar las tabla dba_segments veremos que next_extent es 393216 es decir 48 bloques, aunque, eso si, si se vuelve a llenar esta extensión, se creará realmente una extensión de 50 bloques que son los 430080 bytes.

Minextents
Se indica el número de extensiones que se deben reservar a la vez para un determinado segmento en el momento de su creación. Por defecto es una excepto en los segmentos de rollback que son dos. Puede que las extensiones no estén contiguas. Por supuesto, si en la cláusula storage en la créación del objeto se indican valores para next y pctincrease, las extensiones iniciales que se crean se recalculan para cumplir lo indicado en estos parámetros.

Por ejemplo, creamos un segmento con minextents = 4, con un initial de 262144, next de también 262144, bytes y con un pctincrease del 50%, el resultado será la creación de 4 extensiones de tamaños, 286720, 286720, 409600, 614400 y de un next extent de 884736. Estos tamaños vienen de redondear 32, 48 y 72 a múltiplos de 5 y el next extent es 108 bloques que es el 50% de 72 bloques.

Maxextents
Es el número máximo de extensiones que se pueden crear en ese objeto, contando también la primera. Se puede especificar UNLIMITED con lo que pueden crecer indefinidamente. No se recomienda que a los segmentos de rollback se les asigne unlimited maxextents ya que con operaciones complejas podrían aumentar excesivamente de tamaño y llenarían el disco. Así que hay que tener cuidado a la hora de crear rollback segments, sobretodo porque heredan por defecto el valor del parámetro que tenga asignado el tablespace en el que se crean.

Vamos a poner un ejemplo de creación de una tabla en la que se indican valores para los parámetros de la cláusula storage que acabamos de explicar. Crearemos, por ejemplo, una tabla llamada empleado que contiene un solo campo, nombre, con un initial extent de 256 Kilobytes, con 512 Kilobytes de next extent, un pctincrease de 50, con 3 extensiones iniciales y con un máximo de 10 extensiones:

```
create table empleado (nombre varchar2(50))
```

```
storage (initial 256K  next 512K pctincrease 50 minextents 3
maxextents 10)
```

Si consultamos la vista dba_extents nos mostrará que ha creado las 3 extensiones que le hemos indicado con los siguientes valores:

```
Select extent_id, bytes, blocks from dba_extents
where segment_name = 'EMPLEADO' order by extent_id;
```

0	286720	35
1	532480	65
2	819200	100

Y, al consultar la vista dba_segments o incluso la vista user_tables para este segmento en concreto, observamos el valor de next_extent:

```
Select next_extent from user_extents where segment_name =
'EMPLEADO';
```

1179648 que es el resultado de 512K * 1,5 * 1,5.

Cuando Oracle necesita asignar una nueva extensión a un segmento realiza el siguiente proceso para buscar bloques de datos contiguos en igual número o superior al solicitado por la extensión:

En primer lugar, busca un conjunto de bloques contiguos igual al solicitado por la extensión más uno, para evitar la fragmentación (excepto cuando la extensión es de 5 o menos bloques). Por lo tanto, si la extensión nueva es de 29 bloques oracle buscará un conjunto de justo 30 bloques libres consecutivos.
Si no encuentra ningún conjunto de exactamente ese número de bloques, empieza a buscar un conjunto de más bloques contiguos. Si el primer conjunto que encuentra tiene más de 5 bloques que los que busca, se queda solamente con los que busca, mientras que si la diferencia es de menos de 5 bloques, se coge todo el conjunto.
Por lo tanto, en nuestro caso, si el primer conjunto que encuentra fuera de 35 o más bloques, cogería solo los 30 primeros, mientras que si encuentra un conjunto de entre 31 y 34 se lo quedaría entero.

Nota: En este paso se puede comprobar que, aunque a un segmento le asignemos pctincrease 0, puede que luego nos encontremos en dba_extents algun extensión algo más grande que otras del mismo segmento.

Cuando no encuentra ningún conjunto de bloques de tamaño superior al que busca, realiza un coalesce del tablespace, que es un proceso mediante el cual se unen los distintos bloques que han ido quedándose fragmentados en el tablespace al irse creando y eliminando extensiones mediante este proceso. Una vez hecho el coalesce, Oracle vuelve a repetir los pasos anteriores.

Si, finalmente no encuentra ningún conjunto de bloques para crear la nueva extensión, intenta aumentar el tamaño de alguno de los datafiles del tablespace. Esto solamente lo conseguirá si tienen activado el autoexpand. En caso de no conseguirlo, devolverá un error.

2.5.3 Desasignación de Extensiones

Las extensiones que han sido reservadas por un segmento no son devueltas a Oracle para que se las pueda signar a otro segmento del mismo tablespace hasta que se elimina el objeto mediante la instrucción DROP. Por lo tanto, cuando tenemos una tabla que nos ocupa varias extensiones, a pesar de que borremos todas sus filas, esa tabla seguirá teniendo reservadas las extensiones aunque eso si, estarán vacías.

Existen algunas excepciones. Se pueden devolver todas las extensiones de una tabla excepto las min_extents haciendo un truncate de la misma. Hay que ser muy cuidadoso con esta instrucción ya que se eliminan todos los datos de la tabla y, no hay rollback posible. En los segmentos de rollback, si tienen puesto el parámetro optimal, oracle periódicamente puede desasignarle las extensiones que no usa hasta que vuelve a tener el tamaño óptimo. Finalmente, existe una sentencia que también desasigna las extensiones que no se usan en una tabla:

```
Alter table nombre_de_table deallocate unused;
```

En cuanto se ha desasignado una extensión del segmento al que pertenecía, Oracle ya la puede volver a reclamar para que la puedan utilizar otros segmentos del mismo tablespace. Por lo tanto, cuando un nuevo objeto solicite una nueva extensión, si Oracle no encuentra espacio libre suficiente para crear una nueva, y entre las que ha ido desasignando tampoco encuentra ninguna suficientemente grande como para asignarla, puede unir varias de estas extensiones hasta conseguir una lo suficientemente grande como para poder asignársela al segmento. A esta operación se le llama COALESCE de extensiones.

2.6. Segmentos de datos y temporales

2.6.1. Introducción

Un segmento almacena la información de una estructura lógica de Oracle dentro de un Tablespace. Está formado por una o más extensiones y, a medida que va creciendo el segmento se van asignando nuevas extensiones al mismo. Hay cuatro tipos de segmentos: de datos, de índices, temporales y de rollback.

2.6.2. Segmentos de datos e índices

En un segmento de datos se almacenan todos los datos de una tabla que no esté particionada o que no forme parte de un cluster, de una partición de una tabla particionada o, de un cluster de tablas. Se crea el segmento de datos a la hora de ejecutar la sentencia create que crea la tabla, cluster o partición. En dicha sentencia se indican también los valores de la cláusula storage,

que se han explicado en el capítulo que hace referencia a las extensiones, y va a determinar la forma en que dicho segmento va a ir asignando y desasignando las extensiones.

En el caso de los índices, existe un segmento para cada índice no particionado o para cada partición de un índice particionado. Al igual que con las tablas, los segmentos de índices se crean al ejecutar la sentencia de creación de índices en la cual, también se pueden indicar valores para la cláusula storage y así parametrizar la forma en que se le asignarán las extensiones a medida que vaya creciendo.

2.6.3 Segmentos temporales

Cuando Oracle procesa las consultas se puede ver en la necesidad de utilizar espacio en disco para poder llevar a cabo algunas partes del parsing (análisis) y de la ejecución de la misma. Solamente utilizará este tipo de segmentos cuando no pueda realizar la consulta íntegramente en memoria o cuando no pueda buscarse un método alternativo para realizarla utilizando los índices.

Hay varios tipos de sentencias en las que Oracle se ve en la obligación de utilizar los segmentos temporales:

```
SELECT ... ORDER BY.
CREATE INDEX.
SELECT ... GROUP BY.
SELECT ... UNION ...
SELECT DISTINCT ...
SELECT INSERSEC ...
SELECT MINUS ...
```

Se puede dar el caso en el que algunas consultas en las que intervengan joins en los que no haya índices que faciliten la unión y en las que se den a la vez sentencias del tipo "group by" y "order by" o incluso "distinct", en las que no solo se requiere de un nuevo segmento temporal sino que pueden adquirirse dos segmentos para poder realizar dichas consultas.

Como es natural, cuantas más operaciones se hagan en memoria mejor será el rendimiento del sistema, por lo que si en nuestra aplicación existe un número considerable de consultas de las mencionadas anteriormente, resulta muy apropiado hacer un tunning para decidir si ampliar la zona de memoria reservada para las ordenaciones, aumentando el valor del parámetro SORT_AREA_SIZE.

En Oracle también existen las tablas temporales y los índices temporales para dichas tablas. Estos objetos tambíen utilizan segmentos temporales, pero se les asigna y desasigna de forma diferente a como se hace con las consultas, creación de índices y ordenaciones.

2.6.4. Asignación de segmentos temporales en consultas.

En las consultas, los segmentos temporales se van asignando según se van necesitando y, al terminar la ejecución de la sentencia, se desasignan. Para determinar en qué tablespace se van a crear los segmentos temporales necesarios para estas consultas, a cada usuario de la base de datos, se le asigna un tablespace para dicha función. Esto se realiza con el siguiente comando:

```
Alter user nombre_de_usuario default tablespace
nombre_tablespace_temporal;
```

Por defecto, en la creación del usuario también se le puede asignar un tablespace temporal y, si no se le indica, el sistema le asigna por defecto siempre el tablespace SYSTEM. Es muy importante que ningún esquema o usuario tenga como tablespace temporal el SYSTEM por varios motivos. En primer lugar, porque SYTEM no es un tablespace temporal y por lo tanto, no está optimizado para la gestión de los segmentos temporales, en segundo lugar, al utilizar SYSTEM como tablespace temporal, se aumenta la fragmentación de dicho tablespace por culpa de los segmentos temporales que se van creando y borrando en medio de segmentos de datos e índices con lo que disminuye drásticamente el rendimiento del tablespace principal de la base de datos, y por último, se corre el peligro de que se llene este tablespace por culpa de alguna select mal escrita y que se descontrole aumentando desproporcinadamente el tamaño del segmento temporal creado para ejecutarla.

2.6.5. Asignación de segmentos temporales para tablas temporales

El segmento requerido para una tabla temporal y para sus índices temporales se asigna en el momento de producirse la primera insert en dicha tabla. Este segmento al igual que en el caso anterior, se creará en el tablespace temporal del usuario que está creando la tabla temporal. Para conocer el tablespace por defecto de un usuario y su tablespace temporal, podemos consultar la vista dba_users.

```
Select username, default_tablespace, temporary_tablespace from
dba_users;
```

La explicación del funcionamiento de las tablas temporales excede los objetivos de este manual por lo que no se va a profundizar más en ellas, simplemente se indicará aquí que la eliminación del segmento temporal que se ha creado se hará dependiendo del tipo de tabla temporal que se ha creado. Cuando la tabla temporal es específica para una transacción, Oracle eliminará el segmento creado al finalizar dicha transacción, y si por el contrario, la tabla es específica para la sesión, su segmento se eliminará al terminarse esta sesión.

2.7. Segmentos de Rollback

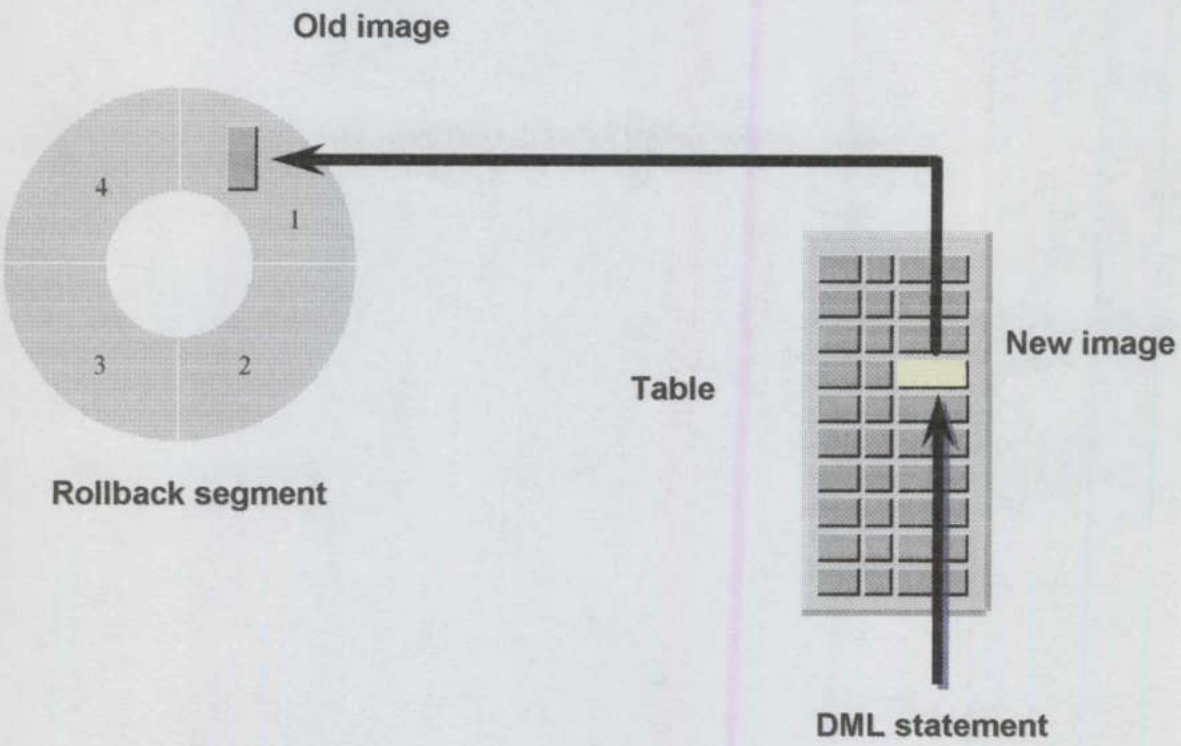
2.7.1. Introducción

En cada base de datos Oracle tenemos uno o más segmentos de rollback en los que se almacena la información que ha sido cambiada por las transacciones. Estas transacciones pueden ser definitivas, es decir, se ha realizado ya el commit de ellas, o puede que aún no se haya hecho dicho commit. Este tipo especial de segmento se utiliza principalmente para poder realizar una lectura consistente de la base de datos Oracle mientras se están modificando los datos y para poder llevar a cabo las recuperaciones de la base cuando ésta cae por algún motivo.

La información de un segmento de rollback se almacena en las llamadas entradas de rollback. En estas entradas se detalla en qué datafile estaba el dato que ha sido modificado, en qué bloque dentro de dicho datafile y también el dato viejo que se ha modificado en la transacción. Además, todas las entradas de rollback de una misma transacción están encadenadas unas con otras para que así, si se deben deshacer los cambios llevados a cabo en esa transacción, resulta más fácil de encontrarlos todos.

Las vistas más importantes de las que podemos extraer información sobre los segmentos de rollback son: v$rollname, dba_rollback_segs y v$rollstat.

Fig. 2.7.1.1. Segmento de Rollback



En una base de datos Oracle, se guardan todos cambios en los bloques modificados en una estructura llamada "logs de rehacer" o, "redo logs". Cuando se crea una nueva entrada de rollback en un segmento de rollback, realmente se está modificando un bloque que se encuentra en dicho segmento de rollback con la información de dicha entrada, por lo que este cambio también se almacena en los log de rehacer. Este doble almacenamiento de la información que se guarda en los segmentos de rollback y en los log de rehacer es fundamental para poder realizar un buen proceso de recuperación de la base de datos en caso de que se produzca en fallo en ella. Cuando se produce una caída de la base de datos, en el momento de la recuperación, se recupera el estado de los segmentos de rollback tanto con las transacciones que ya se habían terminado como con aquellas transacciones cuyo commit aún no se había realizado. El siguiente paso que se produce es el rollback de todas aquellas transacciones que se encuentran en los segmentos de rollback y para las cuales no se había completado el commit.

2.7.2. Utilización de los segmentos de rollback

Como se ha indicado anteriormente, los segmentos de rollback se utilizan para poder deshacer los cambios de las transacciones para las que no se ha hecho un commit y para asegurar la consistencia de lectura. Para facilitar estas tareas, Oracle guarda por cada bloque una tabla de las transacciones que en cada momento se están ejecutando en el mismo. Además, por cada transacción, por cada nuevo cambio que se realiza en los bloques de datos se crea una entrada de rollback que se encadena a las anteriores entradas de rollback asignadas a esa misma transacción de forma ordenada.

Gracias a este sistema, cada vez que se desea restaurar el estado de una transacción al realizar el rollback de la misma, simplemente se debe detectar en qué bloque del segmento de rollback se está almacenando los cambios producidos por dicha transacción mirando en las tablas de transacciones de los bloques del segmento de rollback y, una vez detectado el bloque, se deben seguir una a una las entradas de rollback de la transacción que se encuentran ordenadas y encadenadas, para ir restaurando los valores antiguos en los bloques de datos de forma ordenada.

De la misma manera, se utiliza para facilitar la lectura consistente ya que se detecta el valor antiguo de los bloques navegando por la cadena de las entradas de rollback de la transacción.

¿Cómo se asignan las transacciones a los segmentos de rollback?.

Cada vez que comienza una nueva transacción, se asigna a un determinado segmento de dos formas diferentes:

Se asigna la transacción al siguiente segmento de rollback que se encuentre libre en ese momento de manera automática. Solamente se asigna una transacción cuando se realiza una instrucción de DDL o de DML que no sea una select.
También se puede asignar una transacción a un segmento de rollback en concreto de forma manual. De esta forma, se puede asignar a un segmento de rollback grande una transacción que conocemos de antemano que modifica un gran volumen de datos. Una vez finalizada la

transacción, Oracle vuelve a asignar la siguiente de manera automática al primer rollback que encuentra libre. La instrucción es la siguiente: Set transaction use rollback segment nombre_segmento_rollback;
Cuando se finaliza una transaccion, Oracle libera la información de rollback aunque no la destruye, esto es para soportar la lectura consistente de consultas que han comenzado antes de que se realizara el commit. Para asegurarse que la información se encuentra en los segmentos de rollback el mayor tiempo posible para estas consutas sin borrarla, Oracle va asignando las extensiones a los segmentos de rollback de manera secuencial y, cuando se ha llenado el segmento, se reutilizan las extensiones empezando nuevamente por la primera, como si fuera un segmento circular.

Un segmento de rollback puede tener asignadas solamente un número fijo de transacciones como máximo. Oracle se encarga de asignar las transacciones de una instancia de manera que todos los segmentos tengan el mismo número de transacciones aproximadamente, sin tener en cuenta el tamaño de las mismas ya que, de antemano no lo puede conocer. El número de transacciones que puede contener cada segmento de rollback depende del tamaño del bloque.

2.7.3. Asignación de extensiones

Como hemos indicado anteriormente, un segmento de rollback debe tener al menos dos extensiones y cada una de sus extensiones está formada por un número determinado de bloques. A continuación vamos a explicar cómo se organizan las transacciones en los segmentos de rollback.

En un segmento de rollback pueden estar realizándose a la vez varias transacciones. Estas transacciones pueden estar escribiendo en distintas extensiones o incluso en la misma. Sin embargo, en un bloque de una extensión solamente puede contener información de una transacción, es decir, que no pueden escribir dos transacciones en el mismo bloque de la misma extensión a la vez. Además, como hemos indicado que la escritura de transacciones es secuencial, en cada momento una transacción escribe en una sola extensión. Cuando una transacción se queda sin espacio para escribir en la extensión en la que estaba, puede hacer dos cosas, bien reutilizar una extensión que ya estaba asignada al segmento o bien requerir una nueva extensión para el segmento de rollback.

La primera transacción que necesita más espacio nuevo chequea la siguiente extensión del segmento de rollback, y si no contiene información de transacciones activas, la adquiere. A partir de ese momento, todas las demás transacciones que necesiten espacio utilizarán esta extensión. Si, nuevamente se llena esta extensión y alguna transacción sigue necesitando espacio libre, Oracle vuelve a comprobar si en la siguiente extensión que le toca ocupar, siguiendo el orden secuencial y circular de asignación de extensiones, no se están realizando transacciones activas (insistimos en la naturaleza circular de los segmentos de rollback que, una vez ocupada la última extensión, vuelve a intentar ocupar la primera como si formaran todas un anillo).

Como estamos viendo, Oracle mantiene un anillo formado por las extensiones que ha ido adquiriendo este segmento de rollback y siempre intenta reusar una de las extensiones que lo

forman antes que adquirir una nueva del sistema. Si en algún momento Oracle necesita utilizar una extensión y, en todas los que forman parte del anillo se están escribiendo transacciones activas, se ve obligado a adquirir una nueva extensión del sistema y a añadirla al anillo del segmento de rollback para seguir escribiendo. El número máximo de extensiones que pueden formar parte de un segmento de rollback viene determinado por una parámetro definido en el initSID.ora y que es MAXEXTENTS.

Ya hemos visto cómo se asignan extensiones a un segmento de rollback, pero ¿cómo se van desasignando?

Al borrar un segmento de rollback, todas las extensiones que tenía asignadas el segmento se devuelven al tablespace y pueden ser utilizadas por el resto de objetos que pertenecen al tablespace. Existe otra manera de devolver el espacio utilizado por un segmento sin tener que eliminarlo. A la hora de crear un segmento de rollback se puede indicar un valor en el parámetro OPTIMAL de la cláusula storage que representa el tamaño óptimo de dicho segmento en bytes. Cada vez que Oracle necesista una nueva extensión para el segmento de rollback, compara el tamaño que tiene dicho segmento con el valor del parámetro optimal y, si lo ha sobrepasado, irá devolviendo al tablespace las extensiones más antiguas que se va encontrando en las que ya no quedan transacciones activas ya que, son las que menor probabilidad tienen de tener datos necesarios para mantener la consistencia de lectura. Si puede, liberará tantas extensiones como para quedarse con un tamaño aproximado al indicado en optimal pero siempre por encima.

El valor del parámetro óptimal nuca podrá ser menor que el espacio necesario para la creación del segmento, en el que participan el parámetro initial_extent, next_extent, y min_extents (recordemos que pct_increase no tiene sentido en los segmentos de rollback, todas las extensiones deben ser iguales). Para consultar los valores de estos parámetros podemos utilizar la vista dba_rollback_segs de la siguiente forma:

```
Select segment_name, initial_extent, next_extent, min_extents,
max_extents
from dba_rollack_segs;
```

Y para conocer si nuestros rollback segments tiene asignado un tamaño óptimo:

```
Select name, optsize from v$rollname, v$rollstat
where v$rollname.usn = v$rollstat.usn order by 1;
```

2.7.4. Estados de un segmento de rollback

Un segmento de rollback puede encontrarse en un determinado estado dependiendo de cada momento. Los estados en los que puede encontrarse son:
OFFLINE: No ha sido adquirido por ninguna instancia de la base de datos.
ONLINE: Ha sido adquirido por alguna de las instancias y puede contener datos de transacciones activas.
NEEDS RECOVERY: Contiene datos de transacciones que no se pueden hacer rollback porque sus datafiles están inaccesibles o corruptos.

PARTLY AVAILABLE: Contiene información de una transacción "en duda" que son transacciones en entornos de base de datos distribuidas de las que aún no se ha recibido respuesta.
INVALID: Ha sido borrado.
Esta información se puede obtener de la vista dba_rollback_segs. Las causas por las que un segmento puede pasar de un estado a otro se muestran en la *Tabla 2.7.4.1.*

Tabla 2.7.4.1. Cambios de estado de un Rollback

Estado Inicial	Estado Final	Motivo
Online	Offline	Se pone el segmento de rollback offline con la instrucción alter rollback segment nombre_segmento offline.
Offline	Online	Se pone el segmento de rollback online con la instrucción alter rollback segment nombre_segmento online.
Offline	Invalid	Al borrar el segmento de rollback
Online	Partly Available	Cuando falla la red y hace que una transacción distribuida quede "en duda".
Partly Available	Online	Se resuelve la transacción "en duda".
Partly Available	Offline	No se resuelve la transacción "en duda".
Partly Available	Needs Recovery	Cuando un fallo del medio hace inaccesible la transacción "en duda".
Online	Needs Recovery	Cuando un fallo del medio hace inaccesibles los datafiles o cuando el segmento está corrupto.
Needs Recovery	Offline	Si se soluciona satisfactoriamente la recuperación del medio.
Needs Recovery	Invalid	Si se borra el segmento de rollback

Los estados de Partly Available y Needs recovery son prácticamente iguales. En ambos el segmento contiene información de transacciones que no han sido resueltas. En un estado Partly Available, las transcciones distribuidas no han sido resueltas por culpa de un fallo en la red mientras que en el estado de Needs Recovery, las transacciones no han sido resueltas por un fallo del medio o por estar corrupto el propio segmento.

Una diferencia entre ambos estados es que un adminstrador o el propio Oracle puede poner un segmento Partly Available en estado Online, mientras que un segmento que necesita Recovery, primero debe pasar a estado Offline y luego Online. Además, un segmento en estado Needs Recovery puede ser borrado por el administrado para eliminarlo si estaba corrupto, sin embargo, no se puede borrar un segmento Partly available, primero se debe resolver la transacción en duda.

3 Administración y Optimización de Bases de Datos Oracle (Ejercicios y Ejemplos)

3.1. Introducción a la administración de Bases de datos

3.1.1. Tareas de un administrador de BD

Software

- Instalación, desinstalación y actualización de software sgbdr y clientes
- Instalación y desinstalación de parches
- Pruebas de productos
- Documentación sobre bugs y versiones de productos
- Creación de informes de recomendación de software
- Conocimiento de las aplicaciones de los usuarios

Creacion de BD

- Configuración parámetros de funcionamiento de la BD
- Gestión de
- Usuarios
- Creación, Baja
- Gestión de privilegios

Recursos

- Asignacion de tablespaces, espacio en HD
- Actualización de parámetros del sistema
- Backup y recuperación

Cargas de datos

- Crear y modificar programas de carga
- Ejecutar programas de carga

Mantenimiento de la BD. Altas, Bajas y Modifiaciones de Objetos de la BD (vistas, índices, tablas, etc.)

Monitorización

Accesos de los usuarios

Rendimiento de los procesos

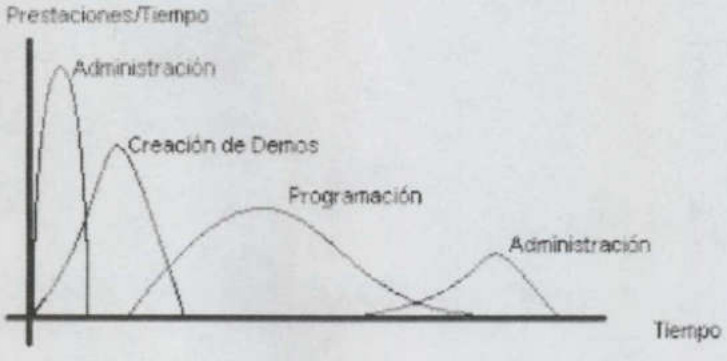
Crecimiento de ficheros

Creación de estadísticas

Optimización, mejoras de rendimiento

3.1.2. Rendimientos obtenidos en el tiempo

Comparativa de las prestaciones relativas subjetivas de la administración de BD frente a otras tareas relacionadas con Ingeniería del Software



3.1.3 Una visión de la evolución de los sistemas de información

Tabla 3.1.3.1. Evolución de los SI

Objetivo	Tecnología	Operaciones más comunes
Gestión	BDR. OLTP. Sistemas Operacionales	On Line: AbMC Batch: Am
Ayuda a la toma de decisiones	BDM. OLAP. Sistemas de Información	On Line: C Batch: A
Toma de decisiones automática	Sistemas Agentes autónomos. Representación en nombre del usuario (Haz en mi lugar)	On Line: C,A

3.1.4. Conceptos de Bases de Datos Multidimensionales

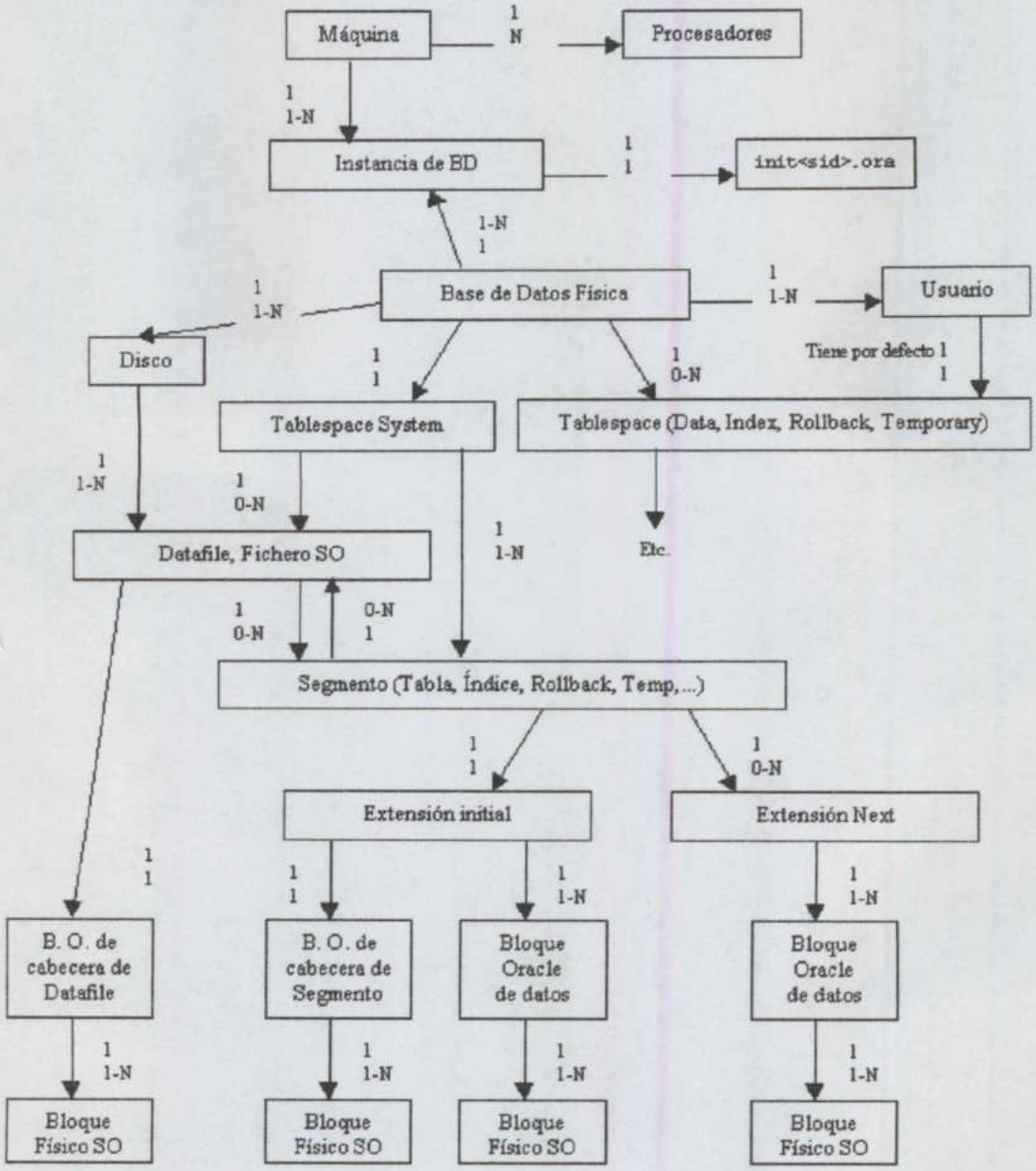
Tabla 3.1.4.1. BD Multidimensionales

ncepto	Descripción
DR	*Base de Datos Relacional*. Sistema de almacenamiento de datos basado en un conjunto de tablas unidas mediante relaciones.
DM	*Base de datos Multidimensional*. Base de datos de estructura basada en dimensiones orientada a consultas complejas y alto rendimiento. Puede utilizar un SGBDR en estrella (Base de datos Multidimensional a nivel lógico) o SGBDM (Base de datos Multidimensional a niveles lógico y físico o Base de datos Multidimensional Pura)
LTP	*On Line Transactional Processing*. Procesamiento Transaccional En Línea. Se trata de los procesos clásicos de tratamiento automático de información, que incluyen Altas, Bajas, Modificaciones y Consultas.
LAP	*On Line Analytical Processing*. Procesamiento Analítico En Línea. Se trata de procesos de análisis de información. Estos sistemas están orientados al acceso en modo consulta.
W	*DataWarehouse*. Sistema almacén de datos que reúne la información generada por los distintos departamentos de una organización. Pretende conseguir que cualquier departamento pueda acceder a información de cualquiera de los otros mediante un único medio, así como obligar a que los mismos términos tengan el mismo significado para todos. Es un almacen de datos historicos, utilizado por una herramienta OLAP para procesar información, elaborar informes y vistas. También se define como un conjunto de datos orientados por tema, integrados, variables en el tiempo y no volátiles que se emplea como apoyo a la toma de decisiones.
atamart	Sistema que mantiene una copia de parte de un DataWarehouse para un uso departamental. Almacén de datos historicos relativos a un departamento de una organización, utilizado por una herramienta OLAP para procesar información, elaborar informes y vistas.
IS	*Executive Information Systems*. Sistemas de información para directivos.
SS	*Decision Support System*. Sistema de ayuda a la toma de decisiones.
ata Mining	Proceso no trivial de análisis de grandes cantidades de datos con el objetivo de extraer información útil. Por ejemplo, se trata de aplicar algoritmos de clasificación de datos para realizar predicciones futuras, o estudios de correlación entre variables aparentemente independientes. Para ello, es común la utilización de Redes Neuronales o Algoritmos Evolutivos.
DD	*Knowdledge Discovery in Databases*.
otación	Cambio de dimensiones en un informe.
rill Down	Descomponer (visualmente) en detalle un dato según una jerarquía de una dimensión.
rill Up	Agregar (visualmente) un dato según una jerarquía de una dimensión.
oll Up	Proceso que calcula para un indicador, y para una o más de las dimensiones por las que ese indicador se mueve, los valores agregados o padres sucesivos a partir de la suma de sus hijos, según las jerarquías especificadas, pudiendo poseer cada dimensión más de una jerarquía. Por ejemplo, es el proceso que suma los ingresos por cada provincia acumulándolos en los ingresos de la comunidad autónoma correspondiente. Se trata de una función que relaciona los valores de dos niveles jerárquicos distintos y adyacentes en una dimensión, transformando un grupo de datos de un nivel en un único dato asignable a otro valor del nivel superior.
pread	Proceso que produce dentro de una dimensión una progresión o algún tipo de reparto proporcional de la cantidad asignada a un elemento entre otros de acuerdo a algún criterio.
imensión	Criterio de clasificación de información. Eje de análisis. Lista de valores que proporciona un índice a los datos. Por ejemplo: <Tiempo>, <Geografía>, <Producto>
dicador, Medida, ipercubo, Variable, órmula	Objeto de estudio. Cada indicador tiene asociada una serie de dimensiones sobre las que se pueden clasificar sus valores, se dice que *se mueve por un cierto número de dimensiones*. Por ejemplo, algunos indicadores son: Ingresos(<Tiempo>, <Geografía>, <Producto>) Número de Empleados(<Tiempo>, <Geografía>)Si el indicador contiene datos almacenados se habla

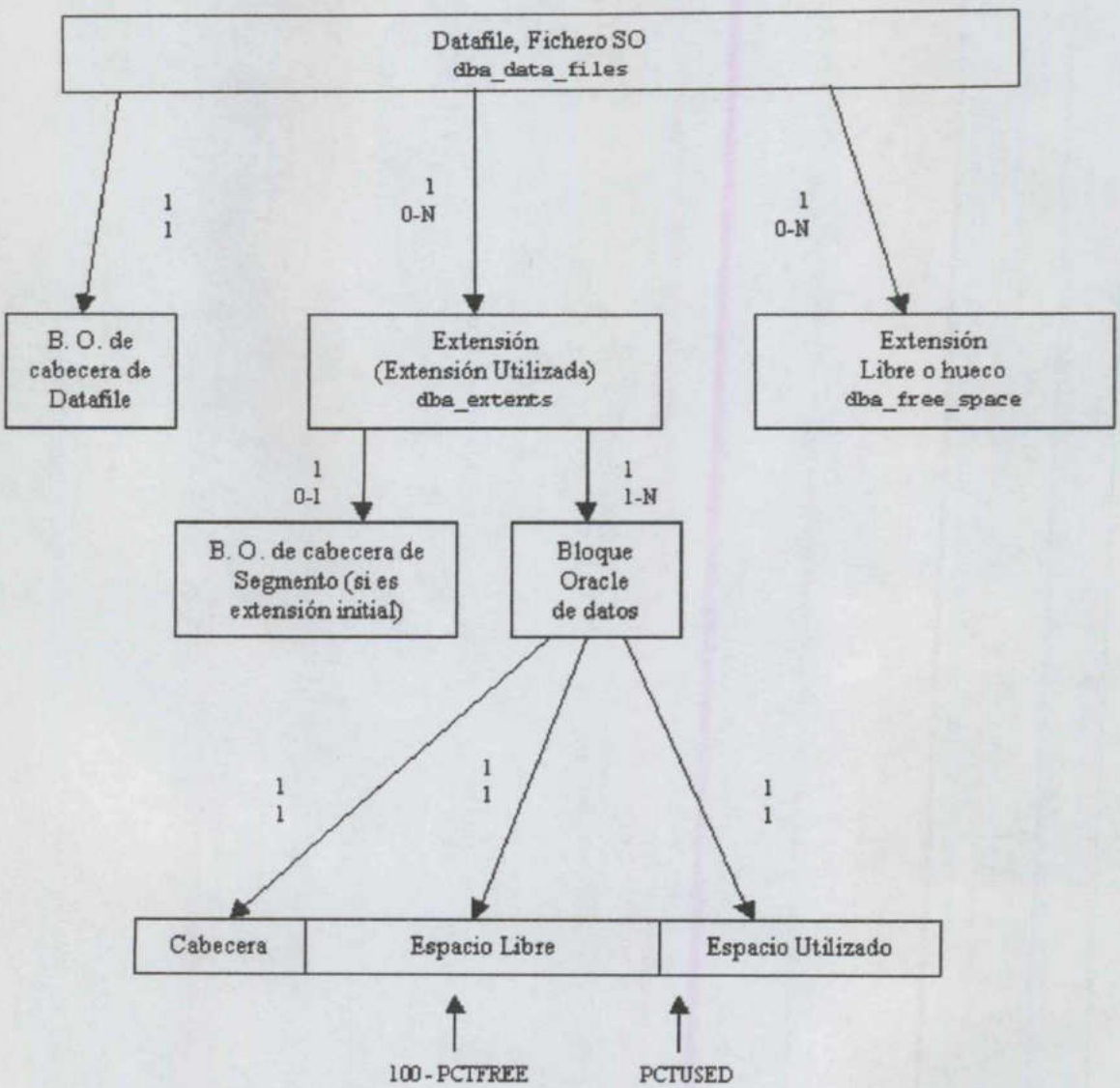
	de *Variable Multidimensional*. Si por el contrario, lo que se almacena es la expresión para calcular esos datos a partir de otros (que puede ser una fórmula o un programa), se habla de *Fórmula Multidimensional*
ementos de una mensión	Posibles valores de un eje de análisis. Por ejemplo, "Enero de 1998", "Trimestre 4 de 1998", o "1996" para la dimensión <Tiempo> y "Bilbao", "Andalucía" o "Zona Norte" para la dimensión <Geografía>
rarquía	Forma de agrupar todos o sólo algunos de los elementos de una dimensión con relaciones padre-hijo. Casi siempre, pero no obligatoriamente, implican que el padre se calcula como la suma de sus hijos. Una dimensión puede tener cero, una o varias jerarquías.
laciones o Atributos	Definen vínculos entre valores de dos dimensiones, de forma que cada valor de una dimensión puede estar relacionado con uno o más valores de otra dimensión
lda	Estructura mínima de almacenamiento formada por la intersección de un valor de cada una de las dimensiones que componen el cubo. Puede contener o no contener datos
QL	*Structured Query Language*. Lenguaje de Consultas Estructurado. *"Select Query Language"*. Lenguaje orientado a la creación de consultas de bases de datos relacionales.
DBMS	*Relational DataBase Management System*. Sistema de gestión de bases de datos relacionales. Programa que sirve para crear, diseñar y manipular bases de datos relacionales
LTP to OLAP	Proceso de migración de datos desde un sistema OLTP a uno OLAP. Esta migración es habitualmentwe el elemento crítico en un desarrollo OLAP
-OLAP	Arquitectura de Base de Datos Multidimensional en la que los datos se encuentran almacenados en una Base de Datos Relacional, normalmente con en forma de estrella (copo de nieve, araña).
-OLAP	Arquitectura de Base de Datos Multidimensional en la que los datos se encuentran almacenados en una Base de Datos Multidimensional, que mejora los tiempos de acceso a costa de mayores necesidades de almacenamiento y retardos en las modificaciones.
-OLAP	Arquitectura que combina las tecnologías ROLAP y MOLAP. En HOLAP, el soporte de almacenamiento de datos y el motor de generación de vistas contienen elementos de ambas tecnologías. Pretende combinar las ventajas de cada una sin sus inconvenientes.

3.2. Gestión de espacio

3.2.1. Estructura física y lógica de una BD Oracle



3.2.2 Espacio Libre en una BD Oracle



3.2.3 Gestión del espacio

Tabla 3.2.3.1. Vistas más relevantes

Significado	dba	user	all
Usuarios	dba_users	user_users	all_users
Tablespaces	dba_tablespaces	user_tablespaces	-
Ficheros que componen los datafiles	dba_data_files	-	-
Segmentos	dba_segments	user_segments	all_segments
Extensiones que forman los segmentos	dba_extents	user_extents	-
Bloques libres	dba_free_space	user_free_space	-
Bloques libres que podrían unirse	dba_free_space_coalesced	-	-

Crear un usuario

```
create user prueba0001 identified by prueba0001;
```

Asignar a un usuario los permisos connect y resource

```
grant connect, resource to scott;
```

Cambio de password

```
alter user prueba0001 identified by prueba0001;
```

Crear un tablespace

```
create tablespace TSprueba0007
 datafile 'DFprueba0007_1.dat' size 60K;
```

Asignar un tablespace a un usuario

```
alter user prueba0001
 default tablespace TSprueba0001
 temporary tablespace TSprueba0001;
```

Borrar un tablespace

```
drop tablespace TSprueba0001
  including contents
  cascade constraints;
```

Crear un tablespace especificando las propiedades de almacenamiento

```
create tablespace TSprueba0001
  datafile 'c:\orant\database\DFprueba0001.dat' size 50K
  default storage (
    initial 10K
    next 10K
    minextents 8
    maxextents 200
    pctincrease 0);
```

Crear un tablespace de varios datafiles

```
create tablespace TSprueba0001
  datafile
    'c:\orant\database\DFprueba0001.dat' size 50 M autoextend
off,
    'c:\orant\database\DFprueba0001.dat' size 50 M autoextend
off,
    'c:\orant\database\DFprueba0001.dat' size 100 M autoextend
on maxsize 200 M
  default storage (
    initial 100 K
    next 100K
    minextents 1
    maxextents 200
    pctincrease 0);
```

Modificar las propiedades de almacenamiento de un tablespace

```
alter tablespace TSprueba0001
  default storage (
    initial 10K
    next 10K
    minextents 1
    maxextents 500
    pctincrease 0);
```

Añadir otro datafile al tablespace

```
alter tablespace TSprueba0001
  add datafile 'c:\orant\database\DFprueba0002.dat'
    size 10 k
    autoextend on
    next 10 k
    maxsize 100 k;
```

Modificar el tamaño de un datafile

```
alter database datafile 'prueba0001.dat' resize 1 M;
```

Eliminar del diccionario un tablespace con datafiles borrados manualmente estando la BD parada

```
shutdown abort
startup
shutdown abort
startup mount
alter database datafile 'c:\mi_carpeta\mi_datafile.dat' offline
drop;
shutdown
startup
```

Unir huecos contiguos en los datafiles de un tablespace

```
alter tablespace TSprueba1 coalesce;
```

Crear extensiones de un segmento

(Se crean automáticamente cuando se necesitan)

```
alter table allocate extent...
```

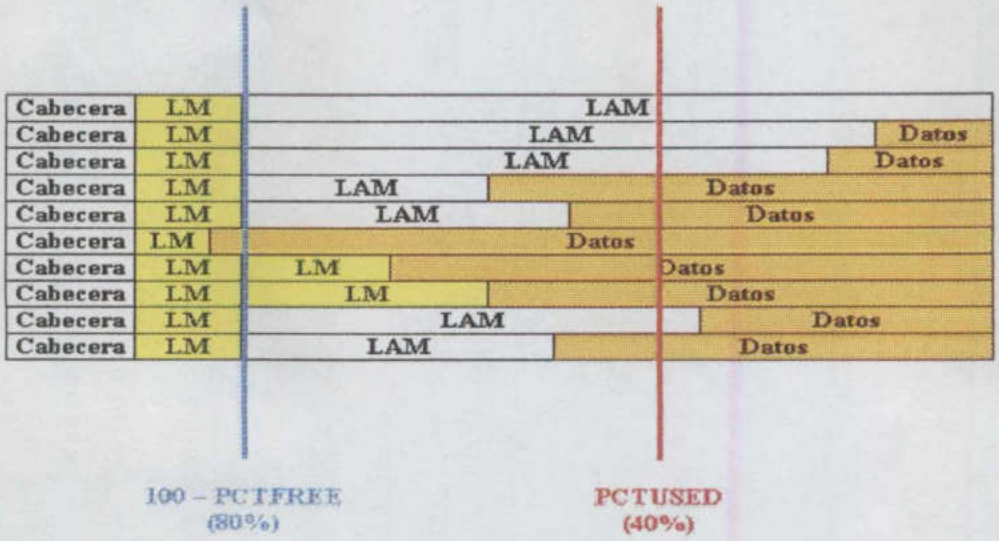
Eliminar extensiones de un segmento

```
alter table deallocate unused...
create rollback segment ... optimal ...
```

3.2.4. Bloque oracle

Una posible evolución de un bloque oracle

LM: Libre para Modificaciones
LAM: Libre para Altas y Modificaciones



3.2.5. Rollback y Redo

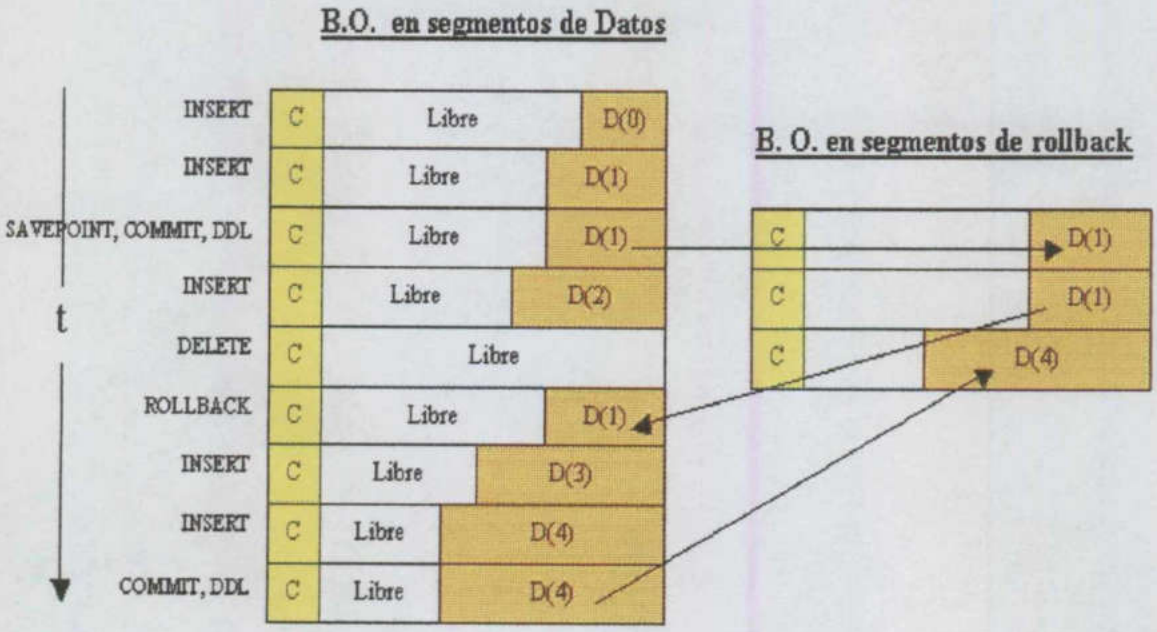
Fases en la ejecución de una sentencia

1. Cacheo de bloques de datos y rollback (si no lo estaban)
2. Bloqueo de datos
3. Copia del viejo valor en segmentos de rollback
4. Copia del nuevo valor en buffers de redo log
5. Modificación de los bloques de datos

Tabla 3.2.5.1. Ejecuciones Rollback y Redo

	Rollback	**Redo**
Alta (INSERT)	Clave	Registro completo
Baja (DELETE)	Registro completo	Clave
Modificación (UPDATE)	Clave y valor modificado anterior	Clave y valor modificado posterior
Consulta (SELECT)	-	-

Efecto de la evolución de un bloque oracle de datos sobre un bloque oracle de rollback



Segmentos de rollback

- Deshacer cambios (SET TRANSACTION, SAVEPOINT, COMMIT, ROLLBACK)
- Consistencia en lectura para otras transacciones
- Recuperación en un estado consistente en caso de fallo

Database log mode

- Montada no abierta
- SVRMGR>shutdown
- SVRMGR>startup mount
- SVRMGR>startup nomount + alter database mount;
- SVRMGR>alter database archivelog;
- SVRMGR>alter database noarchivelog;
- c:\orant\database\log1orcl.ora, ...

Automatic archival

- SVRMGR>archive log list

- SVRMGR>archive log start
- SVRMGR>archive log stop
- initsid.ora>log_archive_start = true
- log_archive_dest
- log_archive_format

```
c:\orant\database\archive\
```

3.3. Autentificación y gestión de usuarios

3.3.1. Usuarios de ORCL

Tabla 3.3.1.1. Diferentes usuarios del ORCL

Usuario	Password por defecto
sys	change_on_install
system	manager
scott	tiger
internal	oracle
sys	oracle

En una Base de Datos Oracle la confidencialidad puede ser relativa a

- los objetos de la Base de Datos fisica
- las operaciones sobre la instancia de Base de Datos, es decir, sobre
- Procesos (servicios, listeners, demonios, programas residentes, ejecutables sin interfaz)
- Recursos (estructuras de memoria RAM, buffers, datos)

3.3.2. Cambio de passwords

Cambiar la password del listener

```
c:\orant\bin>lsnrctl80.exe
```

- editar el fichero listener.ora
- set password vieja_password
- stop
- set password nueva_password
- start

Nota: la password sólo se comprueba en el stop.

Cambio de passwords de usuarios de la base de datos

En cuanto a la confidencialidad relativa a los objetos de la Base de Datos física, los nombres de los usuarios y sus passwords encriptadas se almacenan en la propia base de datos. SQL>select * from dba_users;

Tabla 3.3.2.1. Vista de los usuarios de la BD

Usuario	Comando
sys	alter user sys identified by change_on_install;
system	alter user system identified by manager;
scott	alter user scott identified by tiger;

3.3.3. Métodos de autentificación para operaciones sobre la instancia

En cuanto a la confidencialidad relativa a operaciones sobre la instancia, existen dos formas de autentificación

Métodos de autentificación para operaciones sobre la instancia

1. por sistema operativo
2. por fichero de passwords

Tabla 3.3.3.1. Métodos recomendados

Tipo de Administración	¿hay una conexión segura?	Método recomendado
local	si	por fichero de passwords o por sistema operativo
local	no	por fichero de passwords o por sistema operativo
remota	si	por fichero de passwords o por sistema operativo
remota	no	por fichero de passwords

Tabla 3.3.3.2. Métodos de autentificación utilizados

Sistema Operativo	Método utilizado
Unix y VMS	por sistema operativo
NT	por fichero de passwords

Configuración

En c:\orant\database\initorcl.ora

Tabla 3.3.3.3. Autentificación

Autentificación por Sistema Operativo	Autentificación por fichero de passwords
remote_login_passwordfile = none	remote_login_passwordfile = exclusive remote_login_passwordfile = shared

remote_login_passwordfile = shared indica que el fichero de passwords puede ser compartido por varias bases de datos, pero entonces los únicos usuarios permitidos serán SYS y INTERNAL

En la autentificación por Sistema Operativo, Oracle comprueba que el usuario del Sistema Operativo posee los roles del Sistema Operativo:

- OSDBA
- OSOPER

y los hace equivalentes a los modos de conexión de Oracle:

- SYSDBA
- SYSOPER

```
SQL>connect scott/tiger@orcl as sysdba
SQL>connect scott/tiger@orcl as sysoper
```

Por ejemplo, sys no puede hacer shutdown conectandose como

```
SQL>connect sys/change_on_install
```

pero sí como

```
SQL>connect sys/oracle as sysdba
```

Tabla 3.3.3.4. Suposiciones del Sistema Operativo

El rol del Sistema Operativo	Supone
OSOPER	Permiso para realizar las acciones STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVE LOG, RECOVER, y incluye el privilegio RESTRICTED SESSION
OSDBA	Todos los privilegios con ADMIN OPTION, el OSOPER y el comando CREATE DATABASE

Cambio de la password del usuario internal

1. Borrar o mover el fichero C:\orant\database\pwdorcl.ora
2. Ejecutar

```
c:\orant\bin>orapwd80.exe
file=c:\orant\database\pwdorcl.ora
password=oracle
```

Suplantación de un usuario

- Obtener y copiar la password encriptada

```
SQL>select password from dba_users where
username='SCOTT';
```

- Modificar la password

```
SQL>alter user scott identified by nueva;
```

- Realizar las acciones deseadas
- Restaurar la password

```
SQL>alter user scott identified by values
 'F894844C34402B67';
```

3.3.4. Gestión de privilegios y recursos

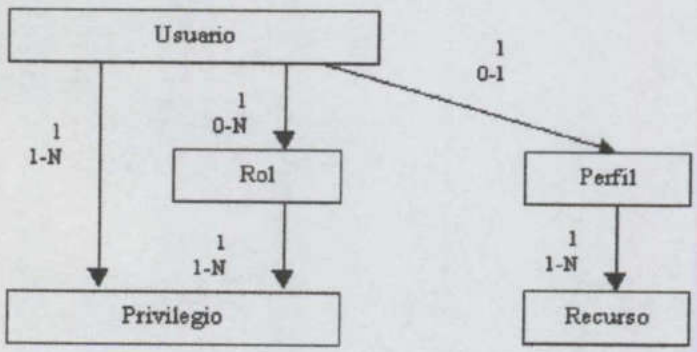


Tabla 3.3.4.1. Conceptos de gestión de privilegios y recursos

Concepto	Significado
Privilegio	Permiso para realizar una acción, asignable a un usuario o un rol
Rol	Conjunto de privilegios, asignables a un usuario o un rol
Usuario	Colección de objetos y privilegios identificado con un nombre y password
Perfil	Conjunto de restricciones relativas al uso de recursos, y asignable a usuarios. Un usuario sólo puede tener un perfil
Recurso	Uso susceptible de ser restringido, asignable a un perfil

Roles Predefinidos por Oracle (select * from dba_roles;)

- CONNECT
- RESOURCE
- DBA
- EXP_FULL_DATABASE
- IMP_FULL_DATABASE
- DELETE_CATALOG_ROLE
- EXECUTE_CATALOG_ROLE
- SELECT_CATALOG_ROLE

Recursos en Oracle (select * from user_resource_limits;)

COMPOSITE_LIMIT

- SESSIONS_PER_USER
- CPU_PER_SESSION
- CPU_PER_CALL
- LOGICAL_READS_PER_SESSION
- LOGICAL_READS_PER_CALL
- IDLE_TIME
- CONNECT_TIME
- PRIVATE_SGA

Limites en uso del espacio en disco (select * from dba_ts_quotas;)

- ALTER USER SCOTT QUOTA UNLIMITED ON USER_DATA;
- ALTER USER SCOTT QUOTA 5M ON TEMPORARY_DATA;
- ALTER USER SCOTT QUOTA 0 ON SYSTEM;

Tabla 3.3.4.2. Comandos para Gestiones

Para la gestión de	**se utilizan los comandos**
Privilegios	GRANT (conceder un privilegio a un usuario o a un rol) REVOKE (denegar un privilegio a un usuario o a un rol)
Roles	CREATE ROLE (crear) ALTER ROLE (modificar) DROP ROLE (borrar) SET ROLE (activar, desactivar) GRANT (conceder un permiso o un rol a un rol) REVOKE (denegar un permiso o un rol a un rol)
Usuarios	CREATE USER (crear) ALTER USER (modificar) DROP USER (borrar) GRANT (conceder un permiso o un rol a un usuario) REVOKE (denegar un permiso o un rol a un usuario)
Perfiles y Recursos	CREATE PROFILE (crear) ALTER PROFILE (modificar) DROP PROFILE (borrar) ALTER USER ... PROFILE (asignar a un usuario) CREATE USER ... PROFILE (asignar a un usuario)

Tabla 3.3.4.3. Activación de perfiles

Estado de la Base de Datos	Acción
Base de Datos Parada	RESOURCE_LIMIT = TRUE en c:\orant\database\initorcl.ora
Base de Datos Arrancada	ALTER SYSTEM SET RESOURCE_LIMIT = TRUE;

Tabla 3.3.4.4. Vistas más relevantes

Significado			
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	-	-
Roles asignados a roles	role_role_privs		
Privilegios de cada rol	role_sys_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_passw ord_limits	-
Límites de recursos en cuanto a espacio máximo en tablespaces	dba_ts_quotas	user_ts_quotas	-

Ejemplos de gestión de privilegios y recursos

Crear un perfil que que sólo permita 2 conexiones concurrentes y asignar ese perfil al usuario SCOTT

```
SYSTEM>create profile solodos limit sessions_per_user 2;
SYSTEM>alter user scott profile solodos;
```

Crear un rol llamado conectarse que incluya los roles connect y resource, y asignar ese rol a scott

```
SYSTEM>create role conectarse;SYSTEM>grant connect, resource to
conectarse;SYSTEM>grant conectarse to scott;
```

Convertir a scott en DBA

```
SYSTEM>grant dba to scott;
```

Crear un usuario especificando que la password debe ser cambiada en la primera conexión

```
SYSTEM>create user prueba1 identified by prueba1 password
expire;
```

Cambiar el espacio reservado en disco para SCOTT

```
SYSTEM>alter user scott quota unlimited on user_data;
SYSTEM>alter user scott quota 5M on temporary_data;
SYSTEM>alter user scott quota 0 on system;
SYSTEM>select * from dba_ts_quotas;
```

Crear un usuario llamado PRUEBA1 y darle únicamente el permiso para conectarse (hacer log on)

```
SYSTEM>create user prueba1 identified by prueba1;
SYSTEM>grant create session to prueba1;
```

Dar al usuario PRUEBA1 permiso para crear tablas

```
SYSTEM>grant create table to prueba1;
SYSTEM>alter user prueba1 default tablespace system quota 10M
on system;
SYSTEM>connect prueba1/prueba1
PRUEBA1>create table tabla1(campo1 number);
PRUEBA1>connect system/manager
```

Desde SCOTT, dar al usuario PRUEBA1 el permiso de consultar la tabla EMP

```
SYSTEM>connect scott/tiger
SCOTT>grant select on emp to prueba1;
PRUEBA1>select * from scott.emp;
PRUEBA1>connect system/manager
```

Desde SCOTT, dar al usuario PRUEBA1 el permiso de insertar en la tabla EMP

```
SYSTEM>connect scott/tiger
SCOTT>grant insert on emp to prueba1;
PRUEBA1>insert into scott.emp values
('john',999,'development');
PRUEBA1>connect system/manager
```

Desde SYSTEM, dar al usuario PRUEBA1 el permiso de consultar la tabla EMP de SCOTT

```
SYSTEM>connect scott/tiger
SCOTT>grant select on emp to system with grant option;
PRUEBA1>connect system/manager
SYSTEM>grant select on scott.emp to prueba1;
SYSTEM>connect prueba1/prueba1
PRUEBA1>select * from scott.emp;
PRUEBA1>connect system/manager
```

Desde SYSTEM, dar al usuario PRUEBA1 el permiso de crear otros usuarios

```
SYSTEM>grant create user to prueba1;
SYSTEM>connect prueba1/prueba1
PRUEBA1>create user prueba2 identified by prueba2;drop user
prueba2;
PRUEBA1>connect system/manager
```

Desde SYSTEM, dar al usuario PRUEBA1 el permiso de crear otros usuarios (por ejemplo, PRUEBA2) de forma que estos, a su vez, puedan crear otros usuarios (por ejemplo, PRUEBA3)

```
SYSTEM>grant create user to prueba1 with admin option;
SYSTEM>grant create session to prueba1 with admin option;
SYSTEM>connect prueba1/prueba1
PRUEBA1>create user prueba2 identified by prueba2;
PRUEBA1>grant create user to prueba2 with admin option;
PRUEBA1>grant create session to prueba2 with admin option;
PRUEBA1>connect prueba2/prueba2
PRUEBA2>create user prueba3 identified by prueba3;
PRUEBA2>grant create session to prueba3;
```

```
PRUEBA2>connect prueba3/prueba3
PRUEBA3>connect system/manager
```

Privilegios sobre objetos

- ALTER
- EXECUTE
- INDEX
- INSERT
- READ
- REFERENCES
- SELECT
- UPDATE
- ALL ó ALL PRIVILEGES

Privilegios del sistema

- ALTER ANY CLUSTER
- ALTER ANY INDEX
- ALTER ANY PROCEDURE
- ALTER ANY ROLE
- ALTER ANY SEQUENCE
- ALTER ANY SNAPSHOT
- ALTER ANY TABLE
- ALTER ANY TYPE
- ALTER ANY TRIGGER
- ALTER DATABASE
- ALTER PROFILE
- ALTER RESOURCE COST
- ALTER ROLLBACK SEGMENT
- ALTER SESSION
- ALTER SYSTEM
- ALTER TABLESPACE
- ALTER USER
- ANALYZE ANY
- AUDIT ANY
- AUDIT SYSTEM
- BACKUP ANY TABLE
- BECOME USER
- COMMENT ANY TABLE
- CREATE ANY CLUSTER
- CREATE ANY DIRECTORY
- CREATE ANY INDEX
- CREATE ANY LIBRARY

- CREATE ANY PROCEDURE
- CREATE ANY SEQUENCE
- CREATE ANY SNAPSHOT
- CREATE ANY SYNONYM
- CREATE ANY TABLE
- CREATE ANY TRIGGER
- CREATE ANY TYPE
- CREATE ANY VIEW
- CREATE CLUSTER
- CREATE DATABASE LINK
- CREATE ANY LIBRARY
- CREATE PROCEDURE
- CREATE PROFILE
- CREATE PUBLIC DATABASE LINK
- CREATE PUBLIC SYNONYM
- CREATE ROLE
- CREATE ROLLBACK SEGMENT
- CREATE SEQUENCE
- CREATE SESSION
- CREATE SNAPSHOT
- CREATE SYNONYM
- CREATE TABLE
- CREATE TABLESPACE
- CREATE TRIGGER
- CREATE TYPE
- CREATE USER
- CREATE VIEW
- DELETE ANY TABLE
- DROP ANY CLUSTER
- DROP ANY DIRECTORY
- DROP ANY INDEX
- DROP ANY LIBRARY
- DROP ANY PROCEDURE
- DROP ANY ROLE
- DROP ANY SEQUENCE
- DROP ANY SNAPSHOT
- DROP ANY SYNONYM
- DROP ANY TABLE
- DROP ANY TRIGGER
- DROP ANY TYPE
- DROP ANY VIEW
- DROP LIBRARY
- DROP PROFILE
- DROP PUBLIC DATABASE LINK

- DROP PUBLIC SYNONYM
- DROP ROLLBACK SEGMENT
- DROP TABLESPACE
- DROP USER
- EXECUTE ANY PROCEDURE
- EXECUTE ANY TYPE
- FORCE ANY TRANSACTION
- FORCE TRANSACTION
- GRANT ANY PRIVILEGE
- GRANT ANY ROLE
- INSERT ANY TABLE
- LOCK ANY TABLE
- MANAGE TABLESPACE
- RESTRICTED SESSION
- SELECT ANY SEQUENCE
- SELECT ANY TABLE
- SYSDBA
- SYSOPER
- UNLIMITED TABLESPACE
- UPDATE ANY TABLE

3.4. SQL dinámico. PL/SQL

3.4.1. SQL dinámico

Ejemplos de generación de SQL mediante SQL

```
create table persona (nombre varchar2 (20));
insert into persona values ('pepe');
insert into persona values ('rosa');
insert into persona values ('juan');
select nombre from persona;
select 'hola ' || nombre from persona;
select nombre, nombre from persona;
select nombre || ' es ' || nombre "Algunas tautologías" from
persona;
select nombre from persona;
create user begoña identified by begoña;
grant connect, resource to begoña;
select 'create user ' || nombre || ' identified by ' || nombre
|| ';' from persona;
select 'grant connect, resource to ' || nombre || ';' from
persona;
set heading off
```

```
set feedback off
spool c:\ejemplo0001.sql
select 'create user ' || nombre || ' identified by ' || nombre
|| ';' from persona;
select 'grant connect, resource to ' || nombre || ';' from
persona;
spool off
set feedback on
set heading on
```

3.4.2. Funciones PL/SQL

Ejercicio 1
Crear una función pl/sql que duplica la cantidad recibida como parámetro

```
--Función que duplica la cantidad recibida como parámetro
CREATE OR REPLACE FUNCTION duplicador(
  valor number
) RETURN number IS
BEGIN
  return (valor * 2);
END;
/
show errors
var x number;
EXEC :x := duplicador(5);
print x
```

Ejercicio 2
Crear una función pl/sql llamada factorial que devuelva el factorial de un número, por ejemplo 5! = 1 * 2 * 3 * 4 * 5 = 120

```
--Cálculo del factorial de un número
CREATE OR REPLACE FUNCTION factorial (
  pNum number
) RETURN number IS
BEGIN
  if pNum = 0 then
    return 1;
  else
    return pNum * factorial(pNum - 1);
  end if;
END;
/
show errors
var x number;
```

```
EXEC :x := factorial(5);
print x;
```

3.4.3. Procedimientos PL/SQL

Ejercicio 1
Crear un procedimiento pl/sql que muestra los números desde el 1 hasta el valor pasado como parámetro

--Mostrar los números del 1 al parametro
```
CREATE OR REPLACE PROCEDURE mostrarNumeros1Ub (
  Ub number
) IS
  vCont number;
BEGIN
  -- vContamos de 1 a Ub
  vCont := 0;
  loop
    vCont := vCont + 1;
    exit when vCont > Ub;
    dbms_output.put_line('Iteración número ' || vCont);
  end loop;
END;
/
show errors
set serveroutput on;
EXEC mostrarNumeros1Ub(5);
```

Ejercicio 2
Modificar el procedimiento del Ejercicio 1 para que muestre números desde un valor inferior hasta uno superior con cierto salto

--Mostrar los números del Lb a Ub con un salto
```
CREATE OR REPLACE PROCEDURE mostrarNumerosLbUbStep (
  pLb    IN     number,
  pUb    IN     number,
  pStep  IN     number   DEFAULT   1
) IS
  vCont  number;
BEGIN
  -- contamos de pLb a pUb
  vCont := pLb-pStep;
  loop
    vCont := vCont + pStep;
    exit when vCont > pUb;
```

```
    dbms_output.put_line('Iteración número ' || to_char((vCont-
pLb+pStep)/pStep));
  end loop;
END;
/
show errors
set serveroutput on;
EXEC mostrarNumerosLbUbStep(1990,1995,0.5);
EXEC mostrarNumerosLbUbStep(1990,1995,0.1);
EXEC mostrarNumerosLbUbStep(1990,1995,2);
```

Ejercicio 3

Modificar el procedimiento del Ejercicio 1 para que inserte los números en una tabla

```
--creación de objetos
create table numeros(numero number);
delete from numeros;
insert into numeros values(1);
insert into numeros values(2);
insert into numeros values(3);
--Mostrar los números del 1 al parametro
CREATE OR REPLACE PROCEDURE mostrarNumeros1Ub (
  Ub number
) IS
  vCont number;
BEGIN
  -- vContamos de 1 a Ub
  vCont := 0;
  loop
    vCont := vCont + 1;
    exit when vCont > Ub;
    insert into numeros values(vCont);
    --dbms_output.put_line('Iteración número ' || vCont);
  end loop;
END;
/
show errors
set serveroutput on;
EXEC mostrarNumeros1Ub(5);
```

Ejercicio 4

Corregir los errores de sintaxis en esta función

Esta función PL/SQL devuelve el número PI (3,141592653589793238462...)
calculado mediante el algoritmo que ideó John Wallis en 1665
PI = 2 *(2/1 * 2/3 * 4/3 * 4/5 * 6/5 * 6/7 *.....)

```
CREATE FUNCTION piWallis(pIteraciones number) number AS
```

```
  vCont number
  vRet number
  vCont = 0
  vRet = 1
  loop
    vCont = vCont + 1;
    when vCont > pIteraciones exit loop;
    if (vCont % 2) = 0
      vRet := vRet * vCont / (vCont + 1);
    else
      vRet := vRet * (vCont + 1) / vCont;
    endif;
  end loop
  return (2 * vRet);
  END;
  /
```

Solución al ejercicio 4:

-- Esta función PL/SQL devuelve el número PI (3,141592653589793238462...)
-- calculado mediante el algoritmo que ideó John Wallis en 1665
PI = 2 *(2/1 * 2/3 * 4/3 * 4/5 * 6/5 * 6/7 *.....) * (--- * --- * --- * --- * --- * --- * ...)

```
CREATE OR REPLACE FUNCTION piWallis(pIteraciones number) RETURN
number IS
  vCont number;
  vRet number;
BEGIN
  vCont := 0;
  vRet := 1;
  loop
    vCont := vCont + 1;
    exit when vCont > pIteraciones;
    if mod(vCont, 2) = 0 then
      vRet := vRet * vCont / (vCont + 1);
    else
      vRet := vRet * (vCont + 1) / vCont;
    end if;
  end loop;
  return (2 * vRet);
END;
/
show errors
var x number;
EXEC :x := piWallis(1);
print x
EXEC :x := piWallis(2);
print x
EXEC :x := piWallis(2000);
```

```
print x
select piWallis(100) from dual;
column piWallis(100) format 9.99999999999999999999999999999
select piWallis(100) from dual;
```

3.4.4. Scripts SQL

Creación de un usuario y un tablespace con un datafile

```
--Script de creción de prueba0001
--sta c:\orant\database\prueba0001\crear_prueba0001.sql
connect system/manager
create user prueba0001 identified by prueba0001;
grant connect, resource to prueba0001;
CREATE TABLESPACE TS_prueba0001
  DATAFILE 'c:\orant\database\prueba0001\prueba0001.dat' SIZE
1M
  DEFAULT STORAGE (
  INITIAL 10K
  NEXT 50K
  MINEXTENTS 1
  MAXEXTENTS 999
  PCTINCREASE 0);
alter user prueba0001
  default tablespace TS_prueba0001
  temporary tablespace TS_prueba0001;
connect prueba0001/prueba0001
create table numeros(numero number, texto varchar2(100));
drop table numeros;
connect system/manager
DROP TABLESPACE TS_prueba0001 INCLUDING CONTENTS CASCADE
CONSTRAINTS;
drop user prueba0001 cascade;
```

Genera y ejecuta un script que une todos los huecos libres

```
--START C:\ORANT\DATABASE\SCRIPTS\GENERA_UNEHUECOS.SQL
--SELECT TABLESPACE_NAME FROM DBA_TABLESPACES;
--ALTER TABLESPACE TSUS1 COALESCE;
--SELECT TABLESPACE_NAME FROM DBA_TABLESPACES;
--SELECT ' DDDD ' || TABLESPACE_NAME || ' DFSDFDS' FROM
DBA_TABLESPACES;
SET SQLPROMPT --SQL>
SET HEADING OFF
SET FEEDBACK OFF
SPOOL C:\ORANT\DATABASE\SCRIPTS\UNEHUECOS.SQL
```

```
SELECT 'ALTER TABLESPACE ' || TABLESPACE_NAME || ' COALESCE;'
FROM DBA_TABLESPACES;
SPOOL OFF
SET HEADING ON
SET FEEDBACK ON
START C:\ORANT\DATABASE\SCRIPTS\UNEHUECOS.SQL
```

Inserta números en una tabla mediante un procedimiento pl/sql

```
--START C:\ORANT\DATABASE\SCRIPTS\GENERANUMEROS.SQL
--DROP TABLE NUMEROS CASCADE CONSTRAINTS;
--CREATE TABLE NUMEROS(CAMPO1 NUMBER);
--Mostrar los números del 1 al parametro
CREATE OR REPLACE PROCEDURE insertarNumeros1Ub (
  Ub number
) IS
  vCont number;
BEGIN
  -- vContamos de 1 a Ub
  vCont := 0;
  loop
    vCont := vCont + 1;
    exit when vCont > Ub;
    --dbms_output.put_line('Iteración número ' || vCont);
    INSERT INTO NUMEROS VALUES(vCont);
  end loop;
END;
/
show errors
set serveroutput on size 400000
DELETE FROM NUMEROS;
EXEC insertarNumeros1Ub(500);
SELECT * FROM NUMEROS;
```

Crea una tabla de usuarios a partir de una tabla de números

```
--START C:\ORANT\DATABASE\SCRIPTS\GENERAUSUARIOS.SQL
--DROP TABLE USUARIOS CASCADE CONSTRAINTS;
--CREATE TABLE USUARIOS(NOMBRE VARCHAR2(200));
--INSERT INTO USUARIOS VALUES ('US1');
--INSERT INTO USUARIOS VALUES ('US2');
--INSERT INTO USUARIOS VALUES ('US3');

--SELECT CAMPO1 FROM NUMEROS WHERE CAMPO1 <= 300;

--SELECT 'aa' || CAMPO1 || 'bb' FROM NUMEROS WHERE CAMPO1 <= 300;
--aa = INSERT INTO USUARIOS VALUES 'US
```

```
--bb = ';
--aa = 'INSERT INTO USUARIOS VALUES "US'
--bb = "';'

SET SQLPROMPT --SQL>
SET HEADING OFF
SET FEEDBACK OFF
SPOOL C:\ORANT\DATABASE\SCRIPTS\INSERTAUSUARIOS.SQL
SELECT 'INSERT INTO USUARIOS VALUES (''US' || CAMPO1 || ''');'
FROM NUMEROS WHERE CAMPO1 <= 300;
SPOOL OFF
SET HEADING ON
SET FEEDBACK ON
DELETE FROM USUARIOS;
START C:\ORANT\DATABASE\SCRIPTS\INSERTAUSUARIOS.SQL
SELECT * FROM USUARIOS;
```

Crea un usuario por cada registro en una tabla de usuarios

```
--START C:\ORANT\DATABASE\SCRIPTS\GENERAOBJETOS.SQL
--DROP USER PEPE CASCADE;
--DROP TABLESPACE TS_PEPE INCLUDING CONTENTS CASCADE CONSTRAINTS;
--CREATE TABLESPACE TS_PEPE DATAFILE 'C:\ORANT\DATABASE\DF_PEPE'
SIZE 6 K;
--CREATE USER PEPE IDENTIFIED BY PEPE DEFAULT TABLESPACE TS_PEPE
TEMPORARY TABLESPACE TS_PEPE;
--GRANT CONNECT, RESOURCE TO PEPE;

--SELECT NOMBRE FROM USUARIOS;

--SELECT NOMBRE || NOMBRE FROM USUARIOS;
--SELECT NOMBRE || NOMBRE || NOMBRE || NOMBRE FROM USUARIOS;
--SELECT NOMBRE FROM USUARIOS;

--SELECT 'A' || NOMBRE || 'B' || NOMBRE || 'C' FROM USUARIOS;
--SELECT 'A' || NOMBRE || 'B' || NOMBRE || 'C' || NOMBRE || 'D' || 'E' || NOMBRE || 'F'
FROM USUARIOS;
--SELECT 'A' || NOMBRE || 'B' FROM USUARIOS;

SET SQLPROMPT --SQL>
SET HEADING OFF
SET FEEDBACK OFF
SPOOL C:\ORANT\DATABASE\SCRIPTS\CREAOBJETOS.SQL

SELECT 'CREATE TABLESPACE TS_' || NOMBRE || ' DATAFILE
''C:\ORANT\DATABASE\DF_' || NOMBRE || ''' SIZE 6 K;' FROM
USUARIOS;
```

```
SELECT 'CREATE USER ' || NOMBRE || ' IDENTIFIED BY ' || NOMBRE
|| ' DEFAULT TABLESPACE TS_' || NOMBRE || ' TEMPORARY
TABLESPACE TS_' || NOMBRE || ';' FROM USUARIOS;
SELECT 'GRANT CONNECT, RESOURCE TO ' || NOMBRE || ';' FROM
USUARIOS;

GRANT CONNECT, RESOURCE TO PEPE;
SPOOL OFF
SET HEADING ON
SET FEEDBACK ON
START C:\ORANT\DATABASE\SCRIPTS\CREAOBJETOS.SQL
SELECT TABLESPACE_NAME FROM DBA_TABLESPACES;
```

Crea usuarios mediante for

```
set serveroutput on;
begin
  for i in 0..25 loop
    dbms_output.put_line ('create user us' || i || ' identified
by us' || i || ';');
  end loop;
end;
/
```

Crea usuarios mediante for

```
set serveroutput on;
begin
  for i in 0..25 loop
    dbms_output.put_line ('create user us' || i || ' identified
by us' || i || ';');
  end loop;
end;
/
```

Crea usuarios mediante for

```
set serveroutput on;
begin
  for i in 0..25 loop
    dbms_output.put_line ('create user us' || i || ' identified
by us' || i || ';');
  end loop;
end;/
```

Ejecución de SQL dinámico

--Ejecución de SQL dinámico

```
connect sys/change_on_install
GRANT EXECUTE ON DBMS_SQL TO scott;
connect scott/tiger
set serveroutput on size 400000
declare
  csent varchar2(32000);
  vcursor number;
  nfilas  integer;
begin
  vcursor := dbms_sql.open_cursor;
  --csent := 'select user from dual';
  --csent := 'select table_name from user_tables';
  --DROP TABLE NUMEROS;
  --CREATE TABLE NUMEROS(CAMPO1 NUMBER);
  --SELECT * FROM NUMEROS;
  csent := 'insert into numeros values(32432)';
  dbms_sql.parse(vcursor,csent,dbms_sql.native);
  nfilas := dbms_sql.execute(vcursor);
  dbms_output.put_line('Han sido tratadas ' || nfilas || '
filas');
  dbms_sql.close_cursor(vcursor);
end;
/
show errors
```

Ejecución de SQL dinámico

--Ejecución de SQL dinámico

```
set serveroutput on;
CREATE OR REPLACE PROCEDURE executeSql(csent varchar2) IS
  vcursor number;
  nfilas  integer;
BEGIN
  vcursor := dbms_sql.open_cursor;
  dbms_sql.parse(vcursor,csent,dbms_sql.native);
  nfilas := dbms_sql.execute(vcursor);
  dbms_output.put_line('Han sido tratadas ' || nfilas || '
filas');
  dbms_sql.close_cursor(vcursor);
END;
/
show errors
set serveroutput on;
```

```
EXEC executeSql('select user from dual');
EXEC executeSql('select table_name from user_tables');
--select * from dba_role_privs;
--select * from dba_role_privs where grantee = 'SCOTT';
--connect sys/change_on_install
--EXEC executeSql('grant dba to scott');
--select * from dba_role_privs where grantee = 'SCOTT';
```

Ejecución de SQL dinámico

--Ejecución de SQL dinámico

```
connect sys/change_on_install
GRANT EXECUTE ON DBMS_SQL TO system;
connect system/manager
set serveroutput on;
CREATE OR REPLACE PROCEDURE executeSql(csent varchar2) IS
  vcursor number;
  nfilas  integer;
BEGIN
  vcursor := dbms_sql.open_cursor;
  dbms_sql.parse(vcursor,csent,dbms_sql.native);
  nfilas := dbms_sql.execute(vcursor);
  dbms_output.put_line('Han sido tratadas ' || nfilas || '
filas');
  dbms_sql.close_cursor(vcursor);
END;
/
show errors
set serveroutput on;
EXEC executeSql('select user from dual');
begin
  for i in 0..25 loop
    dbms_output.put_line ('create user us' || i || ' identified
by us' || i);
    executeSql('select user from dual');
    --executeSql('create user us' || i || ' identified by us'
|| i);
  end loop;
end;
/
```

3.4.5. Cursores SQL

Ejemplo de cursor SQL

```
--Este procedimiento contiene un cursor SQL que recorre y muestra
--los numeros de la tabla NUMEROS que sean menores o iguales al parametro
CREATE OR REPLACE PROCEDURE recorreNumeros1 (
  pUb    IN     number DEFAULT 100
) is
  vNum  number;
  CURSOR cNumeros IS
    SELECT numero FROM NUMEROS WHERE NUMEROS.numero <= pUb;
BEGIN
  OPEN cNumeros;
  loop
    FETCH cNumeros INTO vNum;
    exit when cNumeros%NOTFOUND;
    dbms_output.put_line('Número ' || vNum);
  end loop;
  CLOSE cNumeros;
END;
/
show errors
set serveroutput on;
delete from numeros;
EXEC insertaNumerosLbUbStep(1,10);
select * from numeros;
EXEC recorreNumeros1(5);
```

Otra forma de codificarlo es:

```
--Este procedimiento contiene un cursor SQL que recorre y muestra
--los numeros de la tabla NUMEROS que sean menores o iguales al parametro
CREATE OR REPLACE PROCEDURE recorreNumeros2 (
  pUb    IN     number DEFAULT 100
) is
  CURSOR cNumeros IS
    SELECT numero, texto FROM NUMEROS WHERE NUMEROS.numero <=
pUb;
  regNum  cNumeros%ROWTYPE;
BEGIN
  OPEN cNumeros;
  loop
    FETCH cNumeros INTO regNum;
    exit when cNumeros%NOTFOUND;
    dbms_output.put_line('Número ' || regNum.numero || ' ' ||
regNum.texto);
  end loop;
```

```
  CLOSE cNumeros;
END;
/
show errors
set serveroutput on;
delete from numeros;
EXEC insertaNumerosLbUbStep(1,10);
select * from numeros;
EXEC recorreNumeros2(5);
```

3.5. Arquitectura. Monitorización y optimización

3.5.1. Memoria de la Instancia

La instancia la componen Procesos (programas ejecutables con interfaz o servicios) y recursos (estructuras de memoria RAM, buffers, datos). El mensaje devuelto por show SGA muestra el tamaño total de la memoria compartida SGA.

```
SVRMGR> show sga
Total System Global Area 15077376 bytes
Fixed Size 49152 bytes
Variable Size 12906496 bytes
Database Buffers 2048000 bytes
Redo Buffers 73728 bytes
```

Tabla 3.5.1.1. Procesos de Memoria

Estructuras de Memoria	SGA *System Global Area*	49 Kb	Área Fija. Información adicional sobre procesos	Contiene información sobre: • Usuarios conectados • Bloqueos • Número máximo de bloqueos para DDL y DML • Colas de Entrada/Salida. • Número de db_links permitidos

		12 Mb	Área de memoria compartida. *Shared Pool.* Área Variable. (Shared SQL Area). Se dimensiona con el parámetro `shared_pool_size` en init<sid>.ora, por ejemplo `shared_pool_size = 11534336` Contiene: • Sentencias SQL preanalizadas (parsed) • Procedimientos, librerías • Diccionario de datos	Área compartida para sentencias SQL
				Caché del diccionario (*Dictionary Caché* o *Row Caché*)
		2 Mb	Caché de datos (buffer caché) Buffers de BD. Database Buffers	Contiene los datos y bloques de rollback en uso Se dimensiona con el parámetro `db_block_buffers` en init<sid>.ora, por ejemplo `db_block_buffers = 1000`
		73 Kb	Caché de redo logs (Redo log buffers)	Contiene los datos que son modificados para pasarlos a ficheros históricos y mantener un histórico capaz de reconstruir la BD en caso de fallo
	PGA		Cursores (Áreas privadas para sentencias SQL)	
			Pila de variables	
	Áreas de ordenación		Áreas de ordenación de sentencias. *Sort Area*	
	Ejecutables de Oracle			

3.5.2. Procesos de la Instancia

Tabla 3.5.2.1. Procesos de Instancia

Procesos Background	Estándar	DBWR. *DataBase Writer*	Escribe en los ficheros de la BD los buffers de datos modificados en memoria
		LGWR. *LoG Writer.*	Escribe en los ficheros históricos de la BD (redo log files) cuando los redo log buffers se han llenado o cuando se produce un `commit`. Esto permite el recovery.
		SMON. *System MONitor.*	Trata las recuperaciones en caso de fallo. Limpia los segmentos temporales cuando no están en uso. Realiza el *Coalesce* de tablespaces, uniendo en una sola extensión extensiones contiguas.
		PMON. *Process MONitor.*	Recuperaciones en caso de fallo de un usuario. Libera recursos inactivos. En configuraciones Multi-Threaded Server restablece los procesos dispatcher y servidores caídos.
	Puntos de Ruptura	CKPT	Proceso de Apoyo al LGWR. Actualiza ficheros de control y las cabeceras de ficheros de datos agilizando puntos de ruptura. Este proceso se activa si indicamos en init<sid>.ora la linea `checkpoin_process=true`
	Multithread Server	D000 – Dnnn	Procesos encargados de distribuir las peticiones de usuarios de sentencias SQL entre los procesos compartidos.
		S000 – Snnn	Procesos Compartidos que resuelven sentencias SQL
	Parallel Server	LCKn	Procesos de sincronización de instancias
		LMON	Soluciona problemas de procesos muertos o colgados en Parallel Server
	Parallel Query	P000 – Pnnn	Ejecución de sentencias de forma paralela distribuyendo la carga entre distintas CPU
	Modo Archiver (archive log)	ARCH	Proceso encargado de copiar los históricos (redo log files) en ficheros aparte (archiver files) antes de que sean sobreescritos
	Opción Distribuida	RECO	Proceso encargado del tratamiento de fallos y recuperaciones producidos en transacciones distribuidas debidos a fallos de la red o de la instancia
		SNP0 – SNPn	Procesos para el mantenimiento y refresco de snapshots en BD distribuidas

3.5.3. Cursor vs Select

Creación de objetos

```
CREATE TABLE CLTE (
  CodClte number,
  Nom varchar2(50),
  Edad number
);

CREATE TABLE CREDITO (
  CodClte number,
  Importe number
);

CREATE TABLE DENEGAR (
  CodClte number,
  Nom varchar2(50)
);
DELETE FROM CLTE;
DELETE FROM CREDITO;
DELETE FROM DENEGAR;

INSERT INTO CLTE VALUES (1,'Pepe',30);
INSERT INTO CLTE VALUES (2,'Luis',27);
INSERT INTO CLTE VALUES (3,'Dani',14);
INSERT INTO CLTE VALUES (4,'Rosa',19);
INSERT INTO CLTE VALUES (5,'Bego',18);
INSERT INTO CLTE VALUES (6,'Manu',28);
INSERT INTO CLTE VALUES (7,'Paco',54);

INSERT INTO CREDITO VALUES (1,5000000);
INSERT INTO CREDITO VALUES (2,100000);
INSERT INTO CREDITO VALUES (3,4300000);
INSERT INTO CREDITO VALUES (5,2000000);
INSERT INTO CREDITO VALUES (7,500000);
```

Solución 1

```
INSERT INTO DENEGAR(
  SELECT
    CLTE.CodClte,
    CLTE.Nom
  FROM CLTE, CREDITO WHERE
    CLTE.CodClte = CREDITO.CodClte AND
    CREDITO.Importe > 1000000 AND
    CLTE.Edad <= 25
```

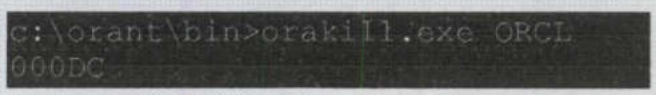
```
);SELECT * FROM DENEGAR;
```

Solución 2

```
CREATE OR REPLACE PROCEDURE cargarDenegar IS
  CURSOR cClte IS SELECT CodClte, Nom, Edad FROM CLTE;
  CURSOR cCredito IS SELECT CodClte, Importe FROM CREDITO;
  regClte cClte%ROWTYPE;
  regCredito cCredito%ROWTYPE;
BEGIN
  COMMIT;
  OPEN cClte;
  OPEN cCredito;
  fetch cClte into regClte;
  fetch cCredito into regCredito;
  loop
    exit when cClte%NOTFOUND OR cCredito%NOTFOUND;
    --tratar registro actual
    if
      regClte.CodClte = regCredito.CodClte AND
      regClte.Edad <25 AND
      regCredito.Importe > 1000000
    then
      INSERT INTO DENEGAR VALUES(regClte.CodClte, regClte.Nom);
    end if;
    --pasar al siguiente
    if regClte.CodClte = regCredito.CodClte then
      fetch cClte into regClte;
      fetch cCredito into regCredito;
    elsif regClte.CodClte < regCredito.CodClte then
      fetch cClte into regClte;
    else
      dbms_output.put_line('Error A0001. El cliente ' ||
regClte.CodClte || ' no existe en la posición esperada en CLTE
o el cliente ' || regCredito.CodClte || ' no existe en la
posición esperada en CREDITO');
      ROLLBACK;
    end if;
  end loop;
  CLOSE cClte;
  CLOSE cCredito;
END;
/
SET SERVEROUTPUT ON
EXEC cargarDenegar;
SELECT * FROM DENEGAR;
```

3.5.4. Monitorización

Tabla 3.5.4.1. Ejemplos de consultas

Descripción	Consulta
Usuarios de la Base de datos	select * from dba_users;
Usuario con el que estamos conectados	select username from user_users; select user from dual;
SID de la Base de datos a la que estamos conectados	select name from v$database;
Máquina a la que estamos conectados	select distinct machine from v$session where type='BACKGROUND';
Sesiones activas. Usuarios que estan conectados en este momento	select distinct sid from v$sesstat; select username, sid, serial# from v$session; select username, sid, serial#, program from v$session; select spid, osuser, s.username, s.program from v$process p, v$session s where p.addr=s.paddr; select spid, osuser, s.username, s.program from v$process p, v$session s where p.addr=s.paddr and s.program like '%SQL%';
Finalizar la sesion de un usuario	select username, sid, serial# from v$session; --siendo 11,9 los campos sid, serial# alter system kill session '11,9'; --También desde MS-DOS, consultando select spid, osuser, s.username, s.program from v$process p, v$session s where p.addr=s.paddr; --siendo 000DC el campo SPID 
Estadisticas de uso de CPU para todas las sesiones activas. Por ejemplo, la estadistica "5 user rollbacks" aumentará cada rollback realizado desde scout	select * from v$sysstat;
Estadisticas de uso de CPU para una de las sesiones activas	select v$sesstat.sid, v$sysstat.name, v$sesstat.value from v$sysstat , v$sesstat where v$sysstat.STATISTIC# = v$sesstat.STATISTIC# and v$sesstat.sid=1;
Caché de sentencias sql	select sql_text from v$sqlarea;
Tamaño de todas las estructuras de memoria (en orden descendente)	select * from v$sgastat order by bytes desc;
Cálculo del porcentaje de fallos en los accesos a Row Cache (Caché del diccionario) calculado como Fallos / (Aciertos + Fallos). Si es > 15% se debería incrementar el tamaño de la Shared Area (Shared Pool) mediante shared_pool_size en initorcl.ora	select sum(gets) "(Aciertos+Fallos)", sum(getmisses) "Fallos" from v$rowcache;

3.5.5. Optimización

Temas de optimización

- En diseño: redundancia para rapidez (mas disco, menos CPU).Ej calculo de seguros, memorizar tablas de datos precalculados. relaciones 1 a 1 para evitar tablas con muchos campos inútiles
- Gestión de espacio en disco. Propiedades de almacenamiento en TS y tablas
    - INITIAL, NEXT, MINEXTENTS, MAXEXTENTS, PCTINCREASE
    - PCTFREE, PCTUSED
    - DB_BLOCK_SIZE
- Particionamiento de tablas grandes (manual y automático)
- Aspectos de administración de tablas grandes, datawarehouses, BBDD multidimensionales
- Gestión de BD distribuidas
- Funcionamiento del optimizador de oracle. Distintas sentencias SQL o procedimientos PL/SQL con el mismo objetivo
- Índices
- Segmentos de rollback
- SQL Trace: rendimiento de diferentes sentencias SQL de forma individual, mostrando información referente a
    - Tiempo de CPU consumido.
    - Lecturas físicas y lógicas.
    - Número de filas procesadas.
    - Fallos en la caché.

Para interpretar los ficheros de traza resultantes es necesario usar la utilidad TKPROF, incluida en C:\ORANT\BIN\TKPROF80.exe. TKPROF: tiempo de ejecución de sentencias, numero de accesos a disco

Tabla 3.5.5.1. *Algunas recomendaciones*

Recomendación	**Acción**
SQL idénticas (mayúsculas/minúsculas) y sin comodines	Normativa, Uso de constantes, Funciones de generación de SQL
Tablespace SYSTEM exclusivo para el diccionario	
Número de extensiones mínimo (evitar creación de extensiones)	
Máxima cantidad razonable de RAM para Oracle	shared_pool_size en initorcl.ora
Máximo número razonable de CPU para Oracle	cpu_count en initorcl.ora
Particionamiento de tablas en varios datafiles y estos distribuidos en varios discos	
Si se usa AUTOEXTEND ON siempre con MAXSIZE	
Los Rollback Segments (para consistencia en lectura y recuperaciones en caso de error) en un tablespace exclusivo debido a su uso elevado. Se recomienda un segmento de rollback por cada 4 transacciones concurrentes sobre la misma instancia	
Es preferible muchas sentencias SQL pequeñas antes que pocas grandes	
Borrar índices antes de los procesos batch, recreándolos después	

4 Conclusiones

4.1. Conclusiones personales

Como pudimos darnos cuenta, el manejo y la Administración de Bases de Datos no es tan difícil, si desde el principio se realiza una adecuada planeación y diseño del sistema. Si desde el principio sabemos que la información ahí depositada es integra y confiable, los resultados serán igual.

Es muy importante que el administrador conozca los controles que debe llevar a cabo para evitar que los usuarios alteren esa información, tomar medidas de seguridad muy estrictas pues en las manos del administrador de la Base de Datos está depositado el activo más importante de cualquier empresa: "La Información".

Después de revisar este trabajo, lo único que podemos concluir es que no basta con tener la teoría sino que se debe poner en práctica de la manera más eficiente para poder ofrecer resultados fidedignos y que puedan soportar una toma de decisiones, pues a fin de cuentas de éstas depende el crecimiento de la organización y es de vital importancia que esas decisiones sean lo más acertadas posible, y si aquellas se van a tomar con base en la información, es lógico que ésta debe ser muy precisa y la tarea principal de cualquier Administrador de Bases de Datos es mantenerla bajo estas condiciones.

4.2. Bibliografía

Referencias Bibliográficas

Newan Aaron y Theriault Marlene. *Oracle Security Handbook*, 3a. Edición, U.S.A., Mc Graw Hill, 2001

Douglas Scherer. Oracle 8i Tips & Techniques, 2ª. Edición, U.S.A. Mc Graw Hill, 2001.

Loney, Kevin. *Oracle8 Manual del Administrador,* 3ª. Edición, España, Mc Graw Hill, 2000.

Referencias Electrónicas

Bases de Datos" Soluciones Avanzadas
http://hp.fciencias.unam.mx/revista/soluciones/30s/No36/dw-1.html

Cómo diseñar grandes variables en bases de datos multidimensionales (Herrán Gascón, Manuel de la; Castellar-Busó, Vicent)

http://www.revista.unam.mx/vol.1/art5/index.html

http://www.oracle.com/datawarehouse/index.html

Oracle Express Objects
http://www.oracle.com/olap/html/oeo.html

OLAP Glossary
http://www.oracle.com/olap/html/glossary.html

OLAP
http://sie.efpol.ua.es/webolap/pagolap.htm

Arquitectura Inteligente para Descubrimiento de Conocimiento en Bases de Datos
http://www.mor.itesm.mx/~portillo/Thesis/kdd.html

http://www.datawarehouse.com/

http://www.sun.es/success/warehouse

http://www.datamining.com

http://research.microsoft.com/research/datamine