

# ADMINISTRACIÓN DE ORACLE



## **INDICE DEL CURSO**

### **INTRODUCCIÓN**

- [Roles y Responsabilidades del DBA de Oracle](#)
- [Tareas básicas del DBA](#)
- [Tareas adicionales del DBA](#)
- [La Base de Datos](#)
- [La Capa Física](#)
- [La Capa Lógica](#)
- [Los Tablespaces y los Datafiles](#)
- [Segmentos, Extensiones y Bloques](#)
- [El Esquema de la base de datos](#)
- [Arquitectura de Oracle](#)

### **LA INSTANCIA ORACLE**

- [El Área Global del Sistema \(SGA\)](#)
- [Procesos de la Instancia](#)
- [El Área Global de Programas \(PGA\)](#)
- [Las Transacciones](#)

### **CREACIÓN DE UNA BASE DE DATOS**

- [Generalidades](#)
- [Creación de una Instancia](#)
- [Arranque de la Instancia](#)
- [Creación de una base de datos](#)

### **AREAS LOGICAS Y ARCHIVOS FISICOS**

- [Tablespaces y Datafiles](#)
- [Creación de un Tablespace](#)
- [Eliminación de un Tablespace](#)

- [Manipulación de Datafiles](#)
- [Los segmentos de Rollback](#)
- [Creación de un segmento de Rollback](#)
- [Estados de un segmento de Rollback](#)
- [Los archivos "Redo Log"](#)

### **MANEJO DE DATOS**

- [Export](#)
- [Import](#)

### **ADMINISTRACIÓN DE CUENTAS DE USUARIO**

- [Creación de Usuarios](#)
- [Modificación de Usuarios](#)
- [Eliminación de Usuarios](#)
- [Creación de Perfiles](#)
- [Creación de Roles](#)

### **OBJETOS DE LA BASE DE DATOS\***

- [Tablas](#)
- [La cláusula \*storage\*](#)
- [Tablas particionadas](#)
- [Las Cláusulas PCTFREE y PCTUSED](#)
- [Vistas](#)
- [Sinónimos](#)
- [Índices](#)
- [Tipos de índices](#)
- [Consideraciones en el diseño de índices](#)
- [Indices particionados](#)
- [Secuencias](#)

### **GLOSARIO DE TÉRMINOS**

### **EJERCICIO DEMOSTRATIVO**

- [Creación de Usuarios](#)
- [Creación de Tablespace](#)
- [Creación de Tabla](#)
- [Revisión de las extensione](#)

## **INTRODUCCIÓN**

Este manual está dirigido a usuarios inexpertos en la administración de Oracle y que se topan por primera vez con la necesidad de efectuar una administración básica de la base de datos, movidos por el afán de personalizarla según los requerimientos del proyecto al que se vean enfrentados. La versión de Oracle que se utilizará para efectos prácticos es la número 8 y el conjunto de herramientas gráficas de administración al que se hará mención en algunas partes del manual corresponde al producto "*DBA Studio*" que se proporciona con las versiones actuales de Oracle.

Se asume que el lector posee conocimientos básicos de SQL y que ha trabajado con bases de datos en el pasado y, por lo tanto, está familiarizado con algunos conceptos y objetos propios de estos ambientes. De todas formas, se recomienda revisar el anexo de términos, al final de este manual, para repasar algunos conceptos que pudieran haberse olvidado.

También es pertinente recomendar que cada vez que se trabaja con Oracle es muy conveniente disponer de la ayuda en línea del software, principalmente de aquellos programas que ayudan a construir la sintaxis de los comandos o los nombres de los objetos principales (tablas o vistas del sistema), que se pueden olvidar con facilidad. Ejemplos de programas útiles son PL/SQL Developer, TOAD o SQL Navigator.

### **Características del software:**

La versión 8 de la base de datos Oracle incluye una herramienta de administración gráfica que es mucho más intuitiva y cómoda de utilizar. Se emplea en forma alternativa a los comandos de línea de texto que se usan para efectuar administración. Por lo tanto, no es necesario disponer de esta herramienta en forma obligatoria porque siempre es posible administrar una base de datos desde la línea de comandos, sin embargo, es mucho más recomendable por la facilidad de uso y rapidez para efectuar la mayoría de los comandos.

### **Instalación de las Herramientas Administrativas:**

Cuando se instalan las aplicaciones cliente de la base de datos, aparece la pantalla siguiente, con todas las opciones que se indican. Para efectos de este curso, se seleccionará la opción "Administrator", que ocupa 356 MegaBytes de espacio en disco y que nos proporcionará las herramientas administrativas gráficas que se mencionaron en el párrafo anterior y otros servicios.

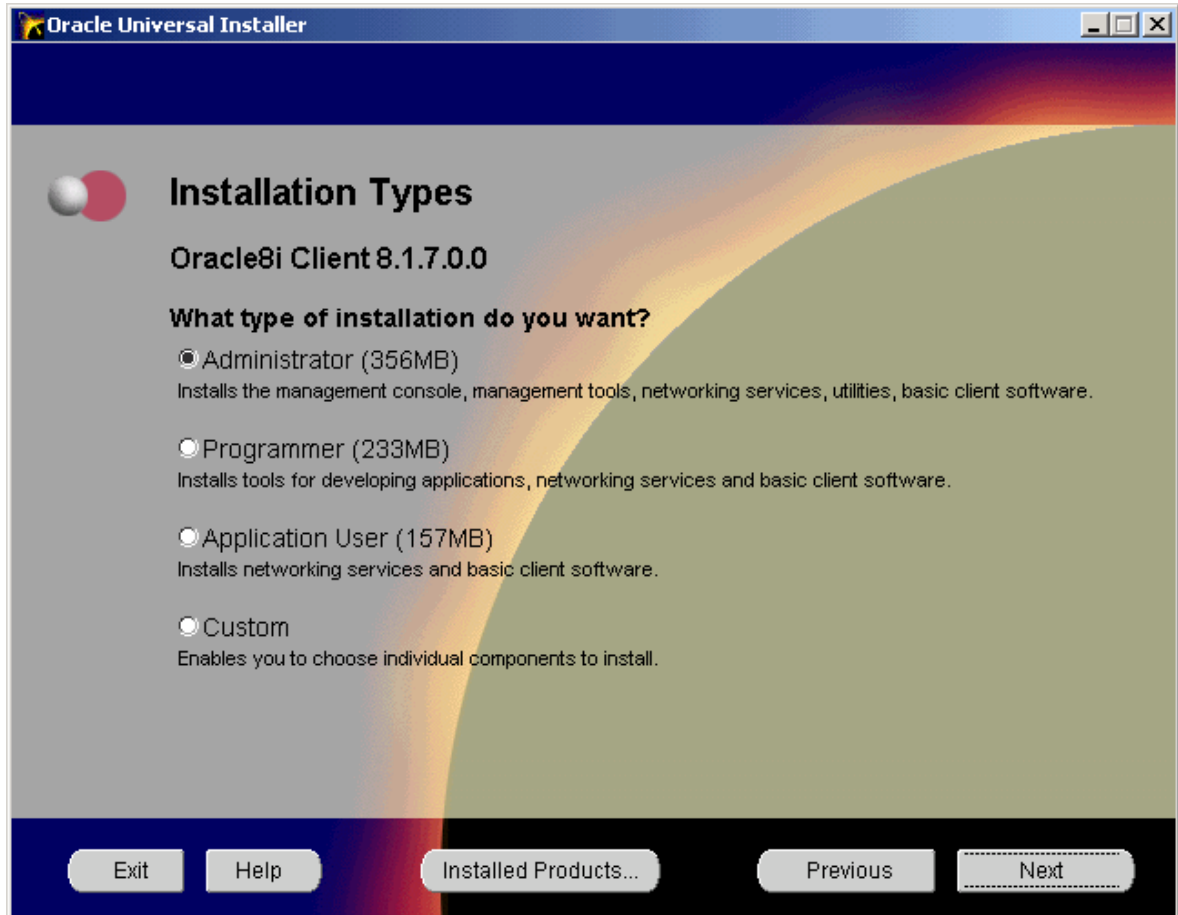


Figura No. 1 Pantalla de instalación del cliente de Oracle

### ***Roles y Responsabilidades del DBA de Oracle***

El administrador de la base de datos de una empresa es siempre considerado como la persona con más experiencia en el área de bases de datos. Por lo anterior, es conveniente tener muy claras las expectativas que se generan en torno a su trabajo y cuáles son los principales roles que debe asumir dentro del marco corporativo o de un proyecto.

#### **Tareas básicas del DBA**

- *Instalación de nuevos componentes del software*

Una de las tareas principales del DBA consiste en la instalación periódica de nuevas actualizaciones de software de Oracle, tanto en lo referente a programas de aplicaciones como a herramientas administrativas. También es recomendable que el propio DBA y otros usuarios de Oracle prueben la instalación y nuevas configuraciones antes de migrarlas a los ambientes de producción.

- *Interacción con el administrador del sistema*

En la mayoría de los casos los programas sólo pueden ser instalados o accedidos por el administrador del sistema. En este caso, el DBA debe trabajar siempre muy bien coordinado con él para garantizar que tanto la instalación y configuración de software como de hardware permita un adecuado funcionamiento del motor de base de datos y de las aplicaciones.

- *Garantizar la seguridad del sistema*

El DBA debe siempre monitorear y administrar la seguridad del sistema. Esto involucra la incorporación y eliminación de usuarios, administración de espacios de disco (cuotas), auditorías y una revisión periódica para detectar probables problemas de seguridad.

- *Monitorización*

El DBA debe monitorear continuamente el rendimiento del sistema y estar preparado para efectuar ajustes de sintonización de éste. En ciertas oportunidades esto involucra cambiar sólo algunos parámetros y otras veces reconstruir índices o reestructurar tablas.

- *Respaldos*

Debido a que la tarea más importante del DBA es proteger la integridad de los datos, se deberá desarrollar una estrategia efectiva de respaldos y recuperación de datos para mantener la estabilidad de toda la información guardada. Las frecuencias de estos respaldos deberán decidirse dependiendo de la cantidad de procesos que alteran los datos a través del tiempo.

- *Prevención de riesgos*

Otra tarea del DBA es la de calendarizar mantenciones a las bases de datos (archivos lógicos) o cooperar en el mantenimiento de las máquinas al administrador del sistema. El DBA debe fortalecer sus esfuerzos en orden a eliminar problemas o situaciones potencialmente peligrosas.

### **Tareas adicionales del DBA**

Otras tareas de importancia que corresponden con frecuencia realizar a un DBA son:

- Analizar datos y efectuar recomendaciones concernientes a mejorar el rendimiento y la eficiencia en el manejo de aquellos datos que se encuentran almacenados.
- Apoyar en el diseño y optimización de modelos de datos.
- Asistir a los desarrolladores con sus conocimientos de SQL y de construcción de procedimientos almacenados y *triggers*, entre otros.
- Apoyar en la definición de estándares de diseño y nomenclatura de objetos.
- Documentar y mantener un registro periódico de las mantenciones, actualizaciones de hardware y software, cambios en las aplicaciones y, en general, todos aquellos eventos relacionados con cambios en el entorno de utilización de una base de datos.

## La Base de Datos

La base de datos de Oracle tiene una capa lógica y otra física. La capa física consiste de archivos que residen en el disco y los componentes de la capa lógica son estructuras que mapean los datos hacia estos componentes físicos.

### La Capa Física

Ya se dijo que consiste de archivos físicos que se encuentran en los discos. Estos pueden ser de tres tipos diferentes:

- Uno o más *datafiles*
- Los *datafiles* almacenan toda la información ingresada en una base de datos. Se pueden tener sólo uno o cientos de ellos. Muchos objetos (tablas, índices) pueden compartir varios *datafiles*. El número máximo de *datafiles* que pueden ser configurados está limitado por el parámetro de sistema MAXDATAFILES.
- Dos o más archivos *redo log* (de *deshacer*)
- Los archivos del tipo *redo log* almacenan información que se utiliza para la recuperación de una base de datos en caso de falla. Estos archivos almacenan la historia de cambios efectuados sobre la base de datos y son particularmente útiles cuando se necesita corroborar si los cambios que la base de datos ya ha confirmado se han efectuado realmente en los *datafiles*.
- Uno o más *control files*
- Estos archivos contienen información que se utiliza cuando se levanta una instancia, tal como la información de dónde se encuentran ubicados los *datafiles* y los archivos *redo log*. Estos archivos de control deben encontrarse siempre protegidos.

### La Capa Lógica

La capa lógica de una base de datos consta de los siguientes elementos:

- Uno o más *tablespaces*
- El esquema de la base de datos (*schema*), el cual consiste de objetos como tablas, clusters, índices, vistas, procedimientos almacenados, triggers, secuencias y otros.

### Los Tablespaces y los Datafiles

Como se mencionó, una base de datos se encuentra dividida en una o más piezas lógicas llamadas *tablespaces*, que son utilizados para separar la información en grupos y así simplificar la administración de los datos. Los *tablespaces* pueden ocupar uno o más *datafiles*. Si se decide que utilice varios *datafiles*, el administrador del sistema puede gestionar que éstos queden localizados en discos diferentes, lo que aumentará el rendimiento del sistema, principalmente por la mejora en la distribución de la carga de entrada / salida.

En la figura siguiente se aprecia la diferencia entre estos tres conceptos. Una base de datos de ejemplo contiene tres *tablespaces* lógicos (parte superior de la figura) que utiliza para almacenar información del sistema, de los datos del usuario y de los índices de las tablas. Asimismo, existen los espacios físicos (*datafiles*) que guardan esta

información en los diferentes discos disponibles y que se señalan en la parte inferior del dibujo.

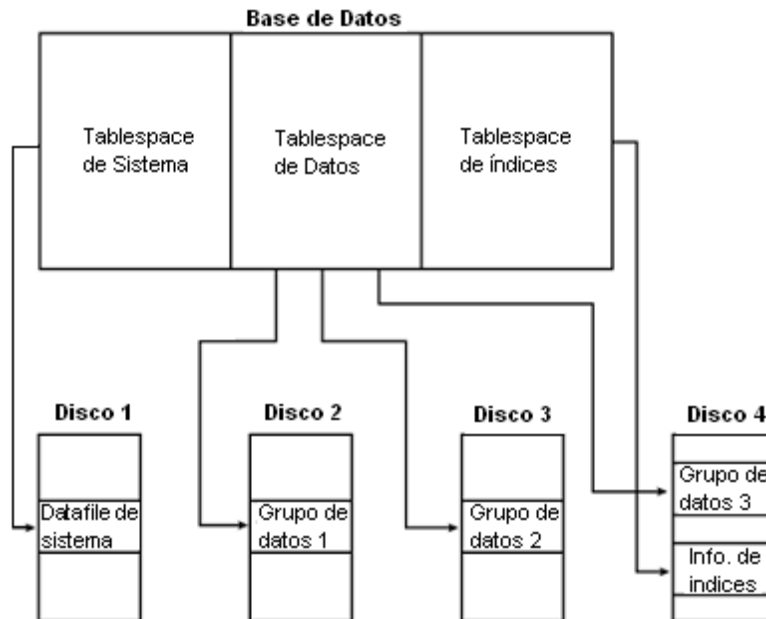


Figura No. 2 Relación entre la base de datos, los *tablespaces* y los *datafiles*

### Segmentos, Extensiones y Bloques

Dentro de los *tablespaces* y *datafiles*, el espacio utilizado para almacenar datos es controlado por el uso de ciertas estructuras; éstas son las siguientes:

- **Bloques:** Un bloque es la unidad de almacenamiento más pequeña en una base de datos Oracle. Contiene una pequeña porción de información (*header*) referente al bloque en sí y el resto a los datos que guarda. Generalmente, un bloque de datos ocupará aprox. 2 KB de espacio físico en el disco (asignación típica).
- **Extensiones:** Es un grupo de bloques de datos. Se establecen en un tamaño fijo y crecen a medida que van almacenando más datos. También se pueden redimensionar para aprovechar mejor el espacio de almacenamiento.
- **Segmentos:** Es un grupo de extensiones utilizados para almacenar un tipo particular de datos. Existen 4 tipos de segmentos: datos, índices, rollback y temporales.

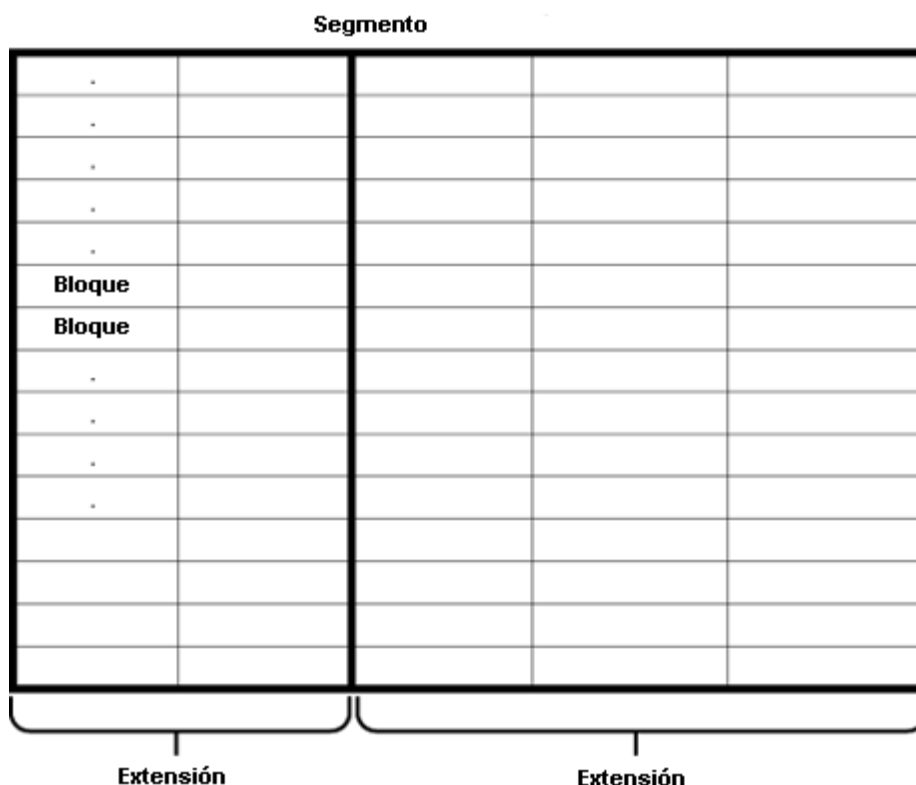


Figura No. 3 Relación entre bloques, extensiones y segmentos

### El Esquema de la base de datos

Un esquema es una colección de objetos lógicos, utilizados para organizar de manera más comprensible la información y conocidos como objetos del esquema. Una breve descripción de los objetos que lo componen es la siguiente:

- **Tabla:** Es la unidad lógica básica de almacenamiento. Contiene filas y columnas (como una matriz) y se identifica por un nombre. Las columnas también tienen un nombre y deben especificar un tipo de datos. Una tabla se guarda dentro de un *tablespace* (o varios, en el caso de las tablas particionadas).
- **Cluster:** Un cluster es un grupo de tablas almacenadas en conjunto físicamente como una sola tabla que comparten una columna en común. Si a menudo se necesita recuperar datos de dos o más tablas basado en un valor de la columna que tienen en común, entonces es más eficiente organizarlas como un cluster, ya que la información podrá ser recuperada en una menor cantidad de operaciones de lectura realizadas sobre el disco.
- **Índice:** Un índice es una estructura creada para ayudar a recuperar datos de una manera más rápida y eficiente. Un índice se crea sobre una o varias columnas de una misma tabla. De esta manera, cuando se solicita recuperar datos de ella mediante alguna condición de búsqueda (cláusula *where* de la sentencia), ésta se puede acelerar si se dispone de algún índice sobre las columnas-objetivo.
- **Vista:** Una vista implementa una selección de varias columnas de una o diferentes tablas. Una vista no almacena datos; sólo los presenta en forma dinámica. Se utilizan para simplificar la visión del usuario sobre un conjunto de tablas, haciendo transparente para él la forma de obtención de los datos.
- **Proced. Almacenado:** Son programas que permiten independizar el manejo de datos desde una aplicación y efectuarla directamente desde el motor de base de



datos, disminuyendo así el tráfico de información a través de la red y mejorando el rendimiento de los procesos implementados mediante estos programas.

- *Trigger*: Un *trigger* es un procedimiento que se ejecuta en forma inmediata cuando ocurre un evento especial. Estos eventos sólo pueden ser la inserción, actualización o eliminación de datos de una tabla.
- *Secuencias*: El generador de secuencias de Oracle se utiliza para generar números únicos y utilizarlos, por ejemplo, como claves de tablas. La principal ventaja es que libera al programador de obtener números secuenciales que no se repitan con los que pueda generar otro usuario en un instante determinado.

### Arquitectura de Oracle

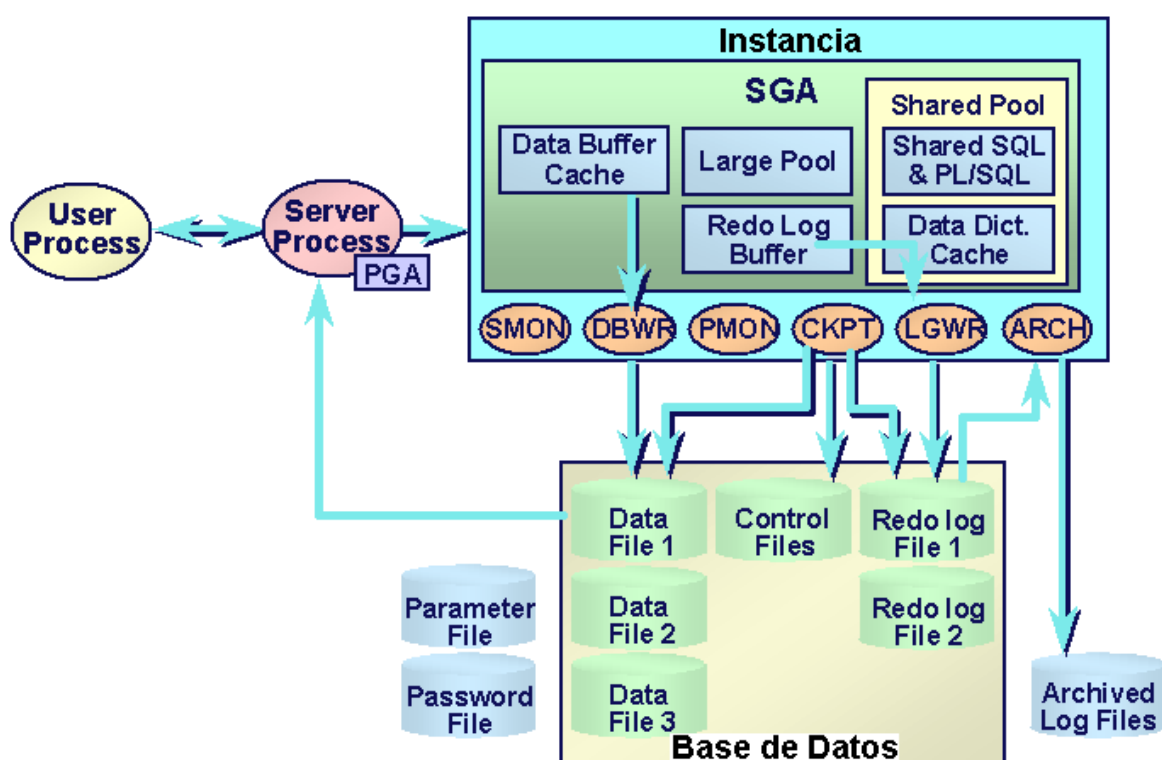


Figura No. 4 Vista general de la Arquitectura de Oracle

La Arquitectura general de Oracle consiste de varios procesos corriendo en la máquina donde reside la instancia, más los espacios de memoria dedicados a ejecutar procesos específicos o al almacenaje de información de cada proceso y la base de datos física propiamente tal, con sus archivos de control, de datos y de transacciones.

## LA INSTANCIA ORACLE

Una instancia de Oracle está conformada por varios procesos y espacios de memoria compartida que son necesarios para acceder a la información contenida en la base de datos.

La instancia está conformada por procesos del usuario, procesos que se ejecutan en el background de Oracle y los espacios de memoria que comparten estos procesos.

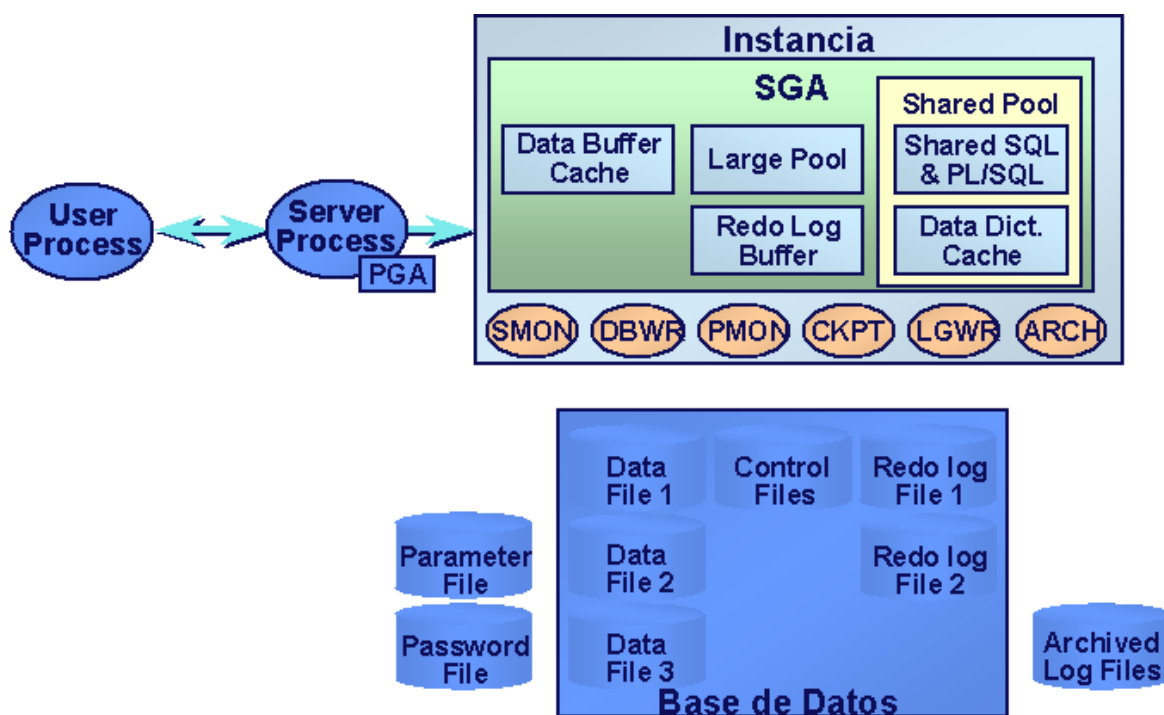


Figura No. 5 Arquitectura de la Instancia de Oracle

### El Área Global del Sistema (SGA)

El SGA es un área de memoria compartida que se utiliza para almacenar información de control y de datos de la instancia. Se crea cuando la instancia es levantada y se borra cuando ésta se deja de usar (cuando se hace *shutdown*). La información que se almacena en esta área consiste de los siguientes elementos, cada uno de ellos con un tamaño fijo:

El buffer de caché (*database buffer cache*)

- Almacena los bloques de datos utilizados recientemente (se hayan o no confirmado sus cambios en el disco). Al utilizarse este buffer se reducen las operaciones de entrada y salida y por esto se mejora el rendimiento.
- El buffer de *redo log*: Guarda los cambios efectuados en la base de datos. Estos *buffers* escriben en el archivo físico de redo log tan rápido como se pueda sin perder eficiencia. Este último archivo se utiliza para recuperar la base de datos ante eventuales fallas del sistema.

- El área *shared pool*: Esta sola área almacena estructuras de memoria compartida, tales como las áreas de código SQL compartido e información interna del diccionario. Una cantidad insuficiente de espacio asignado a esta área podría redundar en problemas de rendimiento. En resumen, contiene las áreas del *caché de biblioteca* y del *caché del diccionario de datos*.
- El *caché de biblioteca* se utiliza para almacenar código SQL compartido. Aquí se manejan los árboles de *parsing* y el plan de ejecución de las *queries*. Si varias aplicaciones utilizan la misma sentencia SQL, esta área compartida garantiza el acceso por parte de cualquiera de ellas en cualquier instante.
- El *caché del diccionario de datos* está conformado por un grupo de tablas y vistas que se identifican la base de datos. La información que se almacena aquí guarda relación con la estructura lógica y física de la base de datos. El diccionario de datos contiene información tal como los privilegios de los usuarios, restricciones de integridad definidas para algunas tablas, nombres y tipos de datos de todas las columnas y otra información acerca del espacio asignado y utilizado por los objetos de un esquema.

### **Procesos de la Instancia**

Según lo que se advierte en la figura 5, los procesos que se implementan en una instancia de Oracle y su función principal son los siguientes:

DBWR (*database writer*): Es el responsable de la escritura en disco de toda la información almacenada en los *buffers* de bloques que no se han actualizado.

LGWR (*log writer*): Es el responsable de escribir información desde el *buffer de log* hacia el *archivo redo log*.

CKPT (*checkpoint*): Es el responsable de advertir al proceso DBWR de efectuar un proceso de actualización en el disco de los datos mantenidos en memoria, incluyendo los *datafiles* y *control files* (para registrar el *checkpoint*). Este proceso es opcional, si no está presente, es el proceso LGWR quien asume la responsabilidad de la tarea.

PMON (*process monitor*): Su misión es monitorizar los procesos del servidor y tomar acciones correctivas cuando alguno de ellos se interrumpe en forma abrupta, limpiando la *caché* y liberando los posibles recursos que pudieran estar asignados en ese momento. También es responsable por el restablecimiento de aquel proceso que se ha interrumpido bruscamente.

SMON (*system monitor*): Levanta una instancia cuando se le da la instrucción de partida (al comienzo del trabajo, encontrándose previamente en *shutdown*). Enseguida limpia los segmentos temporales y recupera las transacciones que pudieran haberse interrumpido debido a una falla del sistema. Además disminuye la fragmentación del sistema agrupando aquellas extensiones libres que existen dentro de la base de datos.

ARCH (*archiver*): La función de este proceso es la de *respaldar* la información almacenada en los archivos *redo log* cuando éstos se llenan. Este proceso está siempre activo cuando se ha establecido el modo ARCHIVELOG. Si el sistema no está operando en este modo se hace más difícil recuperar el sistema sin problemas luego de una falla general.

### **El Área Global de Programas (PGA)**

Esta área de memoria contiene datos e información de control para los procesos que se ejecutan en el servidor de Oracle (relacionados con la base de datos, por supuesto). El tamaño y contenido de la PGA depende de las opciones del servidor que se hayan instalado.

### **Las Transacciones**

El término transacción describe a una unidad lógica de trabajo que está compuesta de una o más sentencias SQL, que deben terminar con una instrucción *commit* o *rollback*. En ese instante, una nueva transacción dará comienzo y estará activa hasta que se ejecute alguno de esos dos comandos otra vez.

Cabe destacar que una transacción no se considera confirmada hasta que ésta se termina de escribir en el archivo de *redo log*.

## CREACIÓN DE UNA BASE DE DATOS

### *Generalidades*

En este capítulo no se discutirá en detalle como se debe crear una instancia o activar sus servicios porque se supone conocido el mecanismo de conectarse a una base de datos o instancia ya creada. Sin embargo, se repasarán los principales comandos que un DBA debiera reconocer para configurarla porque es un hecho que siempre utilizará alguna herramienta gráfica que le permita con mucha facilidad crear instancias y cree automáticamente los archivos de configuración. Un repaso no viene nada de mal.

En primer lugar debemos suponer que el software de Oracle ya se encuentra instalado o que estamos en ello. En la misma operación de instalación se nos preguntará si deseamos crear una instancia y, posteriormente, una base de datos dentro de ella.

Si no es el caso y debemos configurar cada una de ellas ya sea porque no existen, porque las que existen no nos satisfacen o están relacionadas con otros temas o porque no disponen de suficiente espacio, entonces la secuencia correcta es la siguiente:

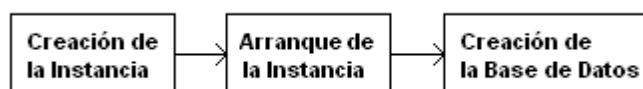


Figura No. 6 Secuencia de creación de Instancias y Bases de Datos

### *Creación de una Instancia*

Cada vez que se crea una instancia con alguna herramienta de administración (generalmente gráfica) como *DBA Studio* (por ejemplo), y queda correctamente configurada, se actualizan todos los archivos que sean necesarios y se puede reconocer posteriormente con un nombre corto que la identifica en forma única y que se conoce como el SID (*system identifier*).

Para crear una instancia desde la línea de comandos del sistema operativo donde se encuentra instalado Oracle, se puede utilizar el utilitario **ORADIM80** que se proporciona con la versión 8 del software. La sintaxis es la siguiente:

```
ORADIM80 -NEW -SID mkt -INTPWD mypass
```

Donde se han omitido los parámetros opcionales y los nombres "*mkt*" y "*mypass*" corresponden a los valores elegidos para nombrar la *instancia* y el *password* de la cuenta **internal**, que es el usuario DBA por defecto que se crea.

### *Arranque de la Instancia*

Una instancia de Oracle puede ser arrancada de forma manual o automática. La primera opción puede efectuarse tanto desde la línea de comandos como desde una interfaz gráfica (*Oracle Enterprise Manager* o *DBA Studio*).

Para la configuración del arranque automático debe establecerse esta opción en algún lugar del sistema operativo. Así, en Windows NT se configura como un servicio y en *Unix*, por ejemplo, se establecen las opciones en un archivo del sistema.

En el caso de tener que arrancar la instancia en forma manual, se puede utilizar el siguiente comando:

#### STARTUP *parámetros*

Y los parámetros pueden ser:

- **PFILE** = *archivo\_de\_parámetros* Si se desea especificar una serie de parámetros de inicialización agrupados.
- **MOUNT** *base\_de\_datos* Si se desea levantar (montar) al mismo tiempo una base de datos (pero no abrirla).
- **NOMOUNT** No monta ninguna base de datos.
- **OPEN** *base\_de\_datos* Levanta la instancia y luego monta y abre una base de datos.
- **RESTRICT** Levanta la base de datos en modo restringido, es decir, sólo los usuarios que tengan el privilegio "RESTRICT SESSION" podrán acceder a ella.

### ***Creación de una base de datos***

Diseñar una base de datos y definir sus propiedades y características de implementación (lógicas y físicas) pensando en los sistemas que harán uso de ella es una tarea muy compleja. Todo el esfuerzo que se debe invertir en esta etapa tendrá como resultado que su administración se haga más fácil o más compleja en el futuro.

Una base de datos se comienza creando los archivos de *redo log*, los archivos de *control* y el *tablespace* de sistema (de nombre *system*). Este último almacena una estructura muy importante que es el diccionario de datos (*data dictionary*) que es el área que contiene toda la información de los *datafiles*, los esquemas y el resto de información relevante de la base de datos. Al igual que en el caso de las instancias, es mucho más cómodo utilizar alguna de las herramientas gráficas mencionadas con anterioridad. En la secuencia de creación de una base de datos se deberá ingresar una gran cantidad de información de configuración, tal como:

- Nombre, SID, password de la cuenta *internal*
- Ruta del archivo de inicialización (initxxx.ora; donde xxx corresponde al SID)
- Ruta de los archivos de control y tamaño de sus *datafiles*
- Datos de tamaño de *datafiles* para los *tablespaces* de usuarios, de sistema y temporal, entre otros
- Tamaño de los archivos *redo log*
- etc.

## AREAS LOGICAS Y ARCHIVOS FISICOS

### *Tablespaces y Datafiles*

Ya hemos dicho que un *tablespace* es una unidad lógica que denota el espacio de almacenamiento de datos dentro de una base de datos y que están constituidos por uno o más *datafiles*, que son los archivos físicos que ocupan efectivamente el espacio en el disco duro. Cuando se crea una base de datos, hay que crear al menos un *tablespace*, por lo que durante el proceso de creación de ésta siempre se indica el *tablespace* principal, de nombre SYSTEM. Su correspondiente *datafile* será entonces el fichero físico al que habrá que asignar una ruta, un nombre y un tamaño.

Los usuarios con características de DBA que se generan automáticamente al crear una instancia son SYS y SYSTEM. Es a partir del trabajo de ellos que la base de datos comienza a crecer y es posible configurar nuevos usuarios, otras áreas de datos (*tablespaces*) e implementar en forma física un modelo de datos en algún esquema.

No es recomendable crear nuevos usuarios o procesos que compartan el *tablespace* del sistema, por lo que una de las primeras tareas del DBA consiste en crear nuevos esquemas (cuentas de usuario) y asignarles *tablespaces* diferentes (que también se deberán crear).

### Creación de un Tablespace

Para crear un *tablespace* desde la interfaz de comandos, se debe escribir la siguiente sentencia:

```
CREATE TABLESPACE nombre DATAFILE 'ruta_y_nombre_del_datafile' SIZE tamaño;
```

Ejemplo:

```
create tablespace datos_prueba datafile 'c:\oracle81\oradata\mkt\tb_mkt01.dbf' size 100M;
```

La cursiva representa valores a escoger para nombrar el *tablespace*, la ruta de su *datafile* y el *tamaño* del mismo. Más tarde se pueden seguir añadiendo *datafiles* al mismo *tablespace* para otorgar más espacio de almacenamiento. Con la sentencia anterior se está creando un *tablespace* llamado "datos\_prueba", al cual se le ha asociado un *datafile* ubicado en el directorio "c:\oracle81\oradata\mkt" de nombre tb\_mkt01.dbf (la extensión **dbf** es siempre obligatoria) y que ocupa 100 megabytes de espacio en el disco.

Una práctica muy habitual y recomendada para quienes deben configurar los *tablespaces* de una base de datos es que implementen espacios diferentes para almacenar los índices de las tablas y otros distintos para almacenar las tablas y sus datos. Y si además sus correspondientes *datafiles* (para los índices y para los datos) se encuentran en discos separados se acelerará el acceso a los datos por partida doble.

## Eliminación de un Tablespace

Para eliminar un *tablespace* que no se vaya a ocupar más, el DBA debe en primer lugar asegurarse que éste no está albergando objetos que se estén utilizando en alguno de los sistemas que se encuentren en explotación (o desarrollo).

Una de las primeras medidas de seguridad que se deben considerar es no eliminar el *tablespace* inmediatamente, sino que dejarlo "deshabilitado" un tiempo prudente mientras se espera a recibir algunas incidencias de los usuarios por este hecho (que no podrán acceder a él, como si se hubiese eliminado). Si se comprueba que efectivamente el *tablespace* ya no es necesario, entonces se puede proceder a eliminarlo sin problemas.

La sintaxis para deshabilitar un *tablespace* es la siguiente:

```
ALTER TABLESPACE nombre OFFLINE;
```

Y para habilitarlo de nuevo:

```
ALTER TABLESPACE nombre ONLINE;
```

Y para eliminarlo definitivamente:

```
DROP TABLESPACE nombre;
```

Otra utilidad de poner un *tablespace* fuera de línea (deshabilitado) es la de poder efectuar tareas administrativas sobre él, ya que esa condición nos garantiza que ningún usuario podrá estar accediendo a los objetos que contiene (tablas, vistas, etc.), por lo que se podrían efectuar, por ejemplo, labores de respaldo o mantención de los objetos, entre otras.

## Manipulación de Datafiles

Mediante el manejo de los archivos físicos de una base de datos (*datafiles*) podemos redimensionar los *tablespaces*, permitiendo la asignación de más espacio.

Para aumentar el tamaño de un *tablespace* se puede optar por alguno de estos dos caminos, representados por las instrucciones que permiten implementar la medida:

- Agregar un *datafile* (por ejemplo, al *tablespace* *datos\_prueba*):

```
alter tablespace datos_prueba add datafile 'c:\oracle81\oradata\mkt\tb_mkt02.dbf' size 50M;
```

- aumentar el tamaño de un *datafile* ya existente:

```
alter datafile 'c:\oracle81\oradata\mkt\tb_mkt01.dbf' resize 150M;
```

La primera instrucción indica que se va a crear un nuevo *datafile* para el *tablespace* que se ha quedado pequeño, aumentando su capacidad en 50 megabytes.



En el segundo ejemplo, no se menciona el *tablespace* porque lo que se hace es redimensionar un *datafile*, cuyo nombre es único en la ruta mencionada y que Oracle ya conoce que está asociado a algún *tablespace* (*datos\_prueba* en el ejemplo). Su tamaño se debe escribir de nuevo, por lo que realmente no se han añadido 150 megabytes como dice la instrucción, sino sólo 50, porque ya tenía 100 megabytes al inicio.

### ***Los segmentos de Rollback***

Los segmentos de *rollback* son áreas lógicas de la base de datos que contienen información de las transacciones que se encuentran en curso y que aún no han sido confirmadas o deshechas. Recuerde que todas las transacciones deben confirmarse en la base de datos en algún momento, con la instrucción **COMMIT** de SQL. Asimismo, se puede deshacer un grupo de transacciones completamente (mientras no se haya hecho el *commit*) mediante la instrucción **ROLLBACK**.

Mientras las transacciones se ejecutan, los cambios se van almacenando en estos segmentos de *rollback* para disponer de ellos en la eventualidad que haya que deshacerlos. Estos segmentos se utilizan en forma concurrente por una o más transacciones. Es labor del DBA el ajustar sus parámetros adecuadamente para proveer un uso eficiente del espacio que utilizan.

Siendo un área que almacena datos, ocupa también extensiones, que son grupos lógicos de bloques de datos. Cada una de estas extensiones va almacenando la información de las transacciones pendientes de confirmarse y va liberando espacio a medida que éstas se van confirmando. Cada vez que una extensión se completa se busca más espacio y se toma otra extensión. Este algoritmo de búsqueda de extensiones va a verificar siempre que la primera se haya desocupado (verificando que las transacciones que almacena ya se han confirmado) y volverá a utilizarla. Por lo anterior se debe pensar en un segmento de *rollback* como un buffer circular, ya que intenta utilizar siempre las mismas extensiones de datos.

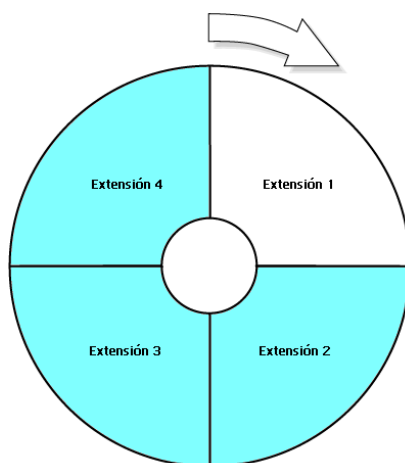


Figura No. 7 Extensiones en un segmento de *rollback*

### **Creación de un segmento de Rollback**

Para crear un segmento de *rollback* desde la línea de comandos de SQL, se debe respetar la sintaxis siguiente:

```

CREATE [PUBLIC o PRIVATE] ROLLBACK SEGMENT nombre_segmento_rollback

TABLESPACE nombre_tablespace

STORAGE (

INITIAL número_en_K_o_M

NEXT número_en_K_o_M

OPTIMAL número_en_K_o_M

MINEXTENTS número

MAXEXTENTS número

);

```

Donde los parámetros de la cláusula STORAGE se refieren a lo siguiente:

- *Initial*: Tamaño de la extensión inicial en Kilobytes (K) o Megabytes (M).
- *Next*: Tamaño de las extensiones sucesivas del segmento de rollback.
- *Optimal*: Tamaño óptimo de crecimiento. Oracle intenta dejar todas las extensiones con este tamaño.
- *MinExtents*: Número mínimo de extensiones que se deberán asignar al segmento.
- *MaxExtents*: Número máximo de extensiones. El segmento sólo crecerá hasta alcanzar este número.

### Estados de un segmento de Rollback

Un segmento de rollback puede encontrarse en cualesquiera de los siguientes estados:

- OFFLINE: No ha sido asociado a 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 pueden hacer *rollback* porque alguno de sus *datafiles* se encuentra inaccesible o corrupto.
- 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: El segmento ha sido borrado.

Para cambiar el estado de un segmento de rollback se debe ejecutar una instrucción cuya sintaxis es como sigue:

```

ALTER ROLLBACK SEGMENT nombre_segmento estado;

```

Para conocer qué segmentos de *rollback* existen en todos los *tablespaces* y el estado en que se encuentran, podemos ejecutar la siguiente sentencia:

```
SELECT segment_name, tablespace_name, status FROM dba_rollback_segs;
```

Que ciertamente, por los objetos a los que accede, sólo podrá ejecutar un DBA.

Esto es particularmente importante si se desea poner algún *tablespace* en estado *offline*, ya que en primer lugar deberían encontrarse también *offline* todos los segmentos de *rollback* que contiene.

### Los archivos "Redo Log"

Los archivos de "deshacer" se utilizan para almacenar la información de todas las transacciones que se llevan a cabo en la base de datos. De esta manera, se cuenta con un registro fiable de las operaciones que se han llevado a cabo para poder reconstruirlas en un eventual proceso de recuperación de la base de datos, si se hubiera producido una falla.

Una base de datos usualmente mantiene dos o más archivos de *redo log*, los que van guardando todas las transacciones que se van efectuando. De hecho, la instrucción COMMIT no se completa mientras no se efectúa la escritura en esos archivos.

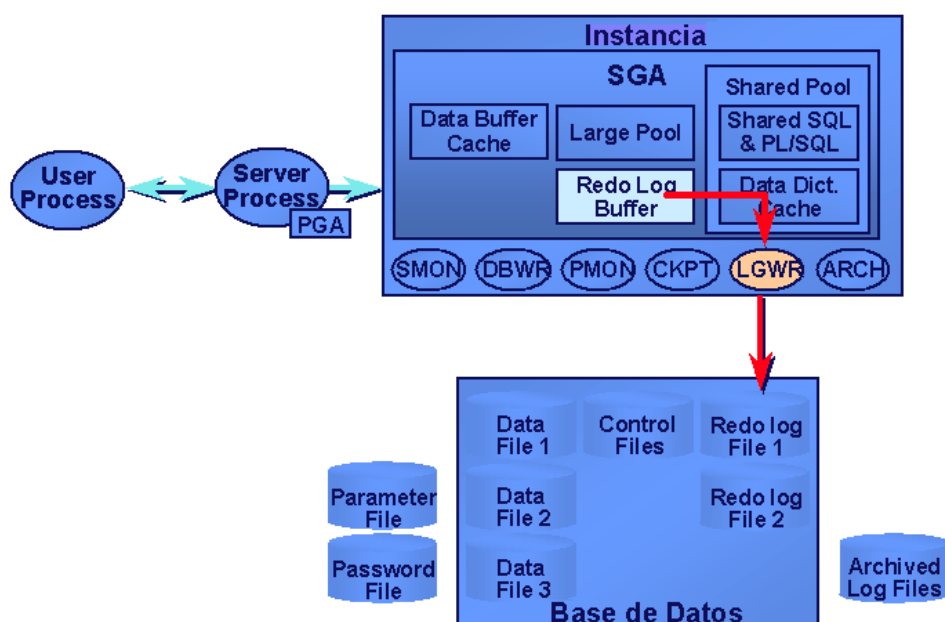


Figura No. 8 Mecanismo de escritura en los archivos *redo log*

Para establecer el tamaño apropiado de un archivo de este tipo deberá considerarse el tamaño del dispositivo que contendrá el respaldo del *redo log*, es decir, si se va a almacenar en una cinta de 525 MB, entonces el tamaño de un archivo de este tipo no debiera superar los 520 MB.

## MANEJO DE DATOS

Como se ha mencionado en los capítulos anteriores, una de las tareas fundamentales de un DBA consiste en la eficiente y completa manipulación de los conjuntos de datos que componen la base de datos de los sistemas que se encuentran en explotación (y desarrollo).

Hay varias formas diferentes (o utilidades) que implementan esta tarea, pero las más comunes son:

- **EXPORT:** Genera un archivo binario con toda la información de estructura y contenido de una base de datos. Estos archivos sólo pueden ser leídos por la utilidad de importación de Oracle (*import*).
- **IMPORT:** Realiza un volcado de la información contenida en un archivo binario (previamente generado con un *export*) en una base de datos.

### ***Export***

Este utilitario está diseñado para registrar en un archivo especial todas las definiciones de objetos y los datos que se deseen dentro de una base de datos. Este archivo es conocido como "el archivo de export" y su formato es únicamente reconocido por el utilitario *Import* de Oracle. Las diferentes intenciones que podrían movernos para efectuar una exportación de datos pueden ser:

1. Respaldo la base de datos: El utilitario *Export* puede ser usado para efectuar un respaldo total de la base de datos (aunque no sea el mecanismo más eficiente para ese propósito).
2. Mover datos entre bases de datos: Los datos y objetos exportados desde una base de datos pueden perfectamente ser recuperados en otra diferente.
3. Reconstruir una base de datos: Si su base de datos tiene los *tablespaces* demasiado fragmentados, ésta es una buena opción para volver a compactarlos.
4. Reorganizar los *datafiles*: Siguiendo la misma lógica anterior, también se puede redistribuir la información en los archivos físicos que se desee.

Bajo Windows NT el programa que permite efectuar las exportaciones es "EXP80" y en otros sistemas operativos (como Unix) es simplemente "EXP".

Sintaxis completa del comando:

**EXP80** *usuario/password* [opciones...]

Para simplificar la utilización del comando cuando se va a repetir varias veces con los mismos parámetros u opciones, éstas se pueden escribir en un archivo de texto y referenciar desde la línea de comandos de la siguiente manera:

**EXP80** *usuario/password* PARFILE=*archivo* [opciones...]

Y las opciones son siempre del tipo *PARÁMETRO=valor*.

Ya sea que se utilice un archivo paramétrico o no, la mayoría de los parámetros que se pueden utilizar en la sintaxis de este comando son:

- **BUFFER=bytes** Especifica el tamaño del buffer de copia (en bytes) usado por el utilitario. Si el valor es cero, entonces se recuperan las filas de a una.
- **COMPRESS=[Y o N]** Este parámetro indica cómo deberá tratarse la extensión inicial. Si el parámetro está establecido como "Y", entonces toda la información se consolidará dentro de una única extensión. Si se establece en "N", se utilizarán los parámetros vigentes para la cláusula *storage*. El valor por defecto es "Y".
- **CONSISTENT=[Y o N]** Si se indica "Y", entonces esperará a que la información que se está actualizando sea confirmada, para tener siempre la versión más fiable mientras dura el procedimiento de exportación. Es una opción muy costosa en tiempo y recursos. El valor por defecto es "N".
- **CONSTRAINTS=[Y o N]** Permite especificar si se desea exportar o no las restricciones de las tablas. Por defecto siempre las exporta.
- **FILE=nombre\_archivo** Especifica el nombre del archivo de salida, es decir, del archivo de exportación.
- **FULL=[Y o N]** Permite indicar si se desea efectuar una exportación completa de la base de datos. El valor por defecto es "N".
- **GRANTS=[Y o N]** Permite indicar si se deben exportar los permisos (*grants*) de cada usuario sobre los objetos que son exportados.
- **INDEXES=[Y o N]** Este parámetro especifica si se deben exportar los índices o no. El valor por defecto es "Y".
- **ROWS=[Y o N]** Se utiliza para exportar todos los datos de las tablas ("Y") o solamente la estructura de los objetos ("N").
- **OWNER=usuarios** Es la lista de usuarios (esquemas) desde donde se realizará la exportación. Puede ser más de uno y se separan por coma.
- **TABLES=(tabla1, tabla2...)** Lista de tablas que se van a exportar. Es válido cuando sólo se exporta un solo esquema de usuario.

Finalmente, se puede utilizar el parámetro **HELP** para obtener una lista de las posibles opciones disponibles con la utilidad. En ese caso deberíamos escribir:

**EXP80 HELP=Y;**

Y entonces se desplegará una pantalla con todas las opciones posibles para el comando **EXP80** (o **EXP**) sin ejecutar ninguna acción de exportación.

## ***Import***

La utilidad de importación se utiliza en conjunto con la de exportación, esto es por que no se puede importar ningún archivo que no sea el resultado de una exportación de datos hecha con anterioridad.

Las opciones de esta utilidad son similares a las de exportación; a continuación sólo presentaremos algunas de ellas, que no son comunes a ambos programas. En este caso, como en la exportación siempre será posible obtener una lista de las opciones disponibles escribiendo:

IMP80 HELP=Y;

Otros parámetros útiles son los siguientes:

- FROMUSER=*usuario* Indica el esquema desde el cual se efectuara la importación. Esto se especifica para no importar el archivo completo, ya que dentro de él se pueden encontrar varios esquemas diferentes.
- TOUSER=*usuario* Es el esquema de destino hacia donde se desean importar los objetos desde el archivo de origen.
- IGNORE=[Y o N] Este parámetro le indica al sistema cómo deberá comportarse ante una probable falla en la importación de algún objeto. Al establecer el valor en "Y", no se hará ninguna advertencia ni se detendrá la ejecución del programa ante alguna eventualidad; en caso contrario, la importación se detendrá para que el administrador tome alguna medida correctiva.
- TABLES=(*tabla1, tabla2,...*) Es la lista de tablas que se desean importar desde el archivo.

## ADMINISTRACIÓN DE CUENTAS DE USUARIO

En este capítulo se conocerá cómo se definen y modifican los **usuarios, perfiles y roles** de una base de datos.

Es una tarea bastante común de cualquier DBA, ya que constantemente se están incorporando nuevos usuarios al sistema o modificando las opciones de éstos. Aquí se repasarán todas las opciones que permiten manejar estas características y se aprenderá a simplificar la carga mediante la administración de perfiles y roles, que son conceptos que controlan diferentes tipos de recursos.

- Rol: Un *rol* es utilizado para asignar privilegios a los usuarios y que les permiten acceder a diferentes objetos y operaciones.
- Perfil: Un *perfil* denota la cantidad de recursos del sistema que se permite consumir a un usuario o grupo de ellos.
- Un usuario puede ser incluido en ambas entidades al mismo tiempo.

### *Creación de Usuarios*

Cuando se da de alta a un usuario basta, como mínimo, con indicar el nombre y el password de la cuenta (esquema) que se está creando. Enseguida se asigna un espacio físico al nuevo esquema dentro de la base de datos con los parámetros por defecto.

Para crear un usuario especificando las opciones adecuadas sin considerar los valores por defecto, se debería respetar la siguiente sintaxis:

CREATE USER *nombre\_usuario*

IDENTIFIED BY *password*

[DEFAULT TABLESPACE *nombre\_tablespace*]

[TEMPORARY TABLESPACE *nombre\_tablespace*]

[QUOTA [*número*, K o M o UNLIMITED] ON *nombre\_tablespace1*]

[, QUOTA [*número*, K o M o UNLIMITED] ON *nombre\_tablespace2*]

[PROFILE *nombre\_perfil*]

[PASSWORD EXPIRE]

[ACCOUNT LOCK o ACCOUNT UNLOCK]

Donde los parámetros corresponden a lo siguiente:

- *Username*: Nombre del usuario que se está creando.
- *Password*: Clave de inicio que se le otorga al usuario. Luego él podrá reemplazarla por la que desee.

- *Default Tablespace*: Es el *tablespace* por defecto al que se conectará el usuario cada vez que ingrese a la base de datos y donde guardará todos sus objetos. Si no se especifica, entonces se conectará al *tablespace system*.
- *Temporary Tablespace*: Es el *tablespace* temporal que utilizará el usuario en todas sus conexiones.
- *Quota*: Cuota de disco (en Kilobytes o Megabytes) que le es otorgada al usuario en cada uno de los *tablespaces* a los que puede acceder. Si se indica "UNLIMITED" entonces el usuario podrá utilizar todo el espacio que quiera dentro del *tablespace* designado.
- *Profile*: Es el nombre del perfil que ha sido asignado a este usuario.
- *Password expire*: Establece que el *password* del usuario expirará en forma automática y, por lo tanto, deberá cambiarlo al iniciar su próxima sesión.
- *Account lock (o unlock)*: Permite establecer si la cuenta debe permanecer bloqueada o no inmediatamente después de crearla.

### ***Modificación de Usuarios***

La forma de modificar usuarios a través de comandos es utilizando la opción *Alter User*, cuya sintaxis completa es muy similar a la de creación de usuarios. Todos los parámetros que fueron establecidos en el instante de la creación pueden ahora modificarse con esta instrucción:

*ALTER USER nombre\_usuario*

*IDENTIFIED BY password*

*[DEFAULT TABLESPACE nombre\_tablespace]*

*[TEMPORARY TABLESPACE nombre\_tablespace]*

*[QUOTA [número, K o M o UNLIMITED] ON nombre\_tablespace1]*

*[, QUOTA [número, K o M o UNLIMITED] ON nombre\_tablespace2]*

*[PROFILE nombre\_perfil]*

*[PASSWORD EXPIRE]*

*[ACCOUNT LOCK o ACCOUNT UNLOCK]*

Es tan poderosa que da la sensación de estar creando al usuario de nuevo, pero no es así porque todos los objetos que tuviera creados bajo su esquema siguen permaneciendo allí.

### ***Eliminación de Usuarios***

Para eliminar un usuario se ejecuta la siguiente instrucción:

*DROP USER nombre\_usuario [CASCADE]*



Y la opción *Cascade* se hace obligatoria cuando el usuario posee objetos en su esquema (tablas, vistas, etc.) y debemos borrarlos junto con él. Sin usar esta opción no podríamos eliminar un usuario con objetos.

### ***Creación de Perfiles***

Los perfiles se crean para limitar las posibilidades de los usuarios del sistema de base de datos. Por ejemplo, se pueden establecer 3 tipos de usuarios:

- Administradores: Que podrían tener acceso a recursos ilimitados dentro del sistema.
- Desarrolladores: Que podrían disponer de un número ilimitado de sesiones pero restringida la utilización de la CPU.
- Otros.

En síntesis, los perfiles se utilizan para suavizar las tareas de administración de la seguridad, manteniendo siempre bajo control los accesos a los recursos de todos los usuarios, por muchos que éstos puedan llegar ser.

Los perfiles se crean y modifican con los comandos `CREATE PROFILE` y `ALTER PROFILE`. La sintaxis de estos comandos no necesitan ser repetidas aquí porque vienen extensamente explicados en la ayuda del software; sólo vale la pena destacar que algunas de las cláusulas hacen referencia a cuánto ciclo de CPU se le asignará a cada usuario, cuántas sesiones concurrentes podrán tener, etc.

### ***Creación de Roles***

Los Roles constituyen la forma más segura y rápida de asignar recursos a los grupos de usuarios. Es una tarea muy tediosa para cualquier DBA tener que asignar o revocar permisos a todos los usuarios, de a uno por uno, y es por eso que agrupando un conjunto de usuarios bajo las mismas características es posible manejar sus permisos como un grupo.

Para crear roles se utiliza la siguiente sintaxis:

`CREATE ROLE nombre_rol/NOT IDENTIFIED o IDENTIFIED BY password`

Y para asignar el rol a un usuario o para comenzar a asignar / quitar ciertos privilegios al rol se debe utilizar los comandos siguientes:

- Grant: Otorga privilegios a un rol (o a un usuario cualquiera) o también asigna un rol a un usuario.
- Revoke: Elimina privilegios otorgados previamente a un rol (o a un usuario).

Los roles o privilegios se pueden asignar varios al mismo usuario o grupo en una sola línea de comandos, siguiendo la sintaxis siguiente:

`GRANT nombre_rol o nombre_privilegio [, nombre_rol o nombre_privilegio]`

`TO nombre_usuario o nombre_rol o PUBLIC [, nombre_usuario o nombre_rol]`

[WITH ADMIN OPTION]

Ejemplo:

1. Creación del rol ROLE\_DML:
2. `CREATE ROLE role_dml NOT IDENTIFIED;`
3. Asignar el privilegio de **Select** al rol recién creado:
4. `GRANT select TO role_dml;`
5. Asignar el rol a los usuarios JPEREZ y LGONZALEZ:

`GRANT role_dml TO jperez, lgonzalez;`

Esto hace que los usuarios anteriores posean el privilegio de SELECT. Cada uno de los usuarios que se incorporen a este rol, tendrá el mismo privilegio recién mencionado.

Si los privilegios se otorgan con la cláusula "*with admin option*" esto quiere decir que los usuarios que reciben los privilegios pueden a su vez otorgarlos a otros.

## **OBJETOS DE LA BASE DE DATOS**

### ***Tablas***

Una tabla se crea en un segmento. Este segmento posee una o más extensiones. Si la tabla crece hasta alcanzar el tamaño máximo de una extensión, entonces se crea uno nuevo para esa tabla. Las extensiones crecen de la manera en que se definieron cuando se creó la tabla, dentro de la cláusula *Storage*. Cuando la cláusula anterior no se define para una tabla, entonces se utilizan los parámetros por defecto definidos dentro del *tablespace* donde se está usando. Si tampoco existen, entonces se utilizan los parámetros del sistema.

#### **La cláusula *storage***

La sintaxis de la cláusula mencionada cuando se crea una tabla es la siguiente:

```
CREATE TABLE nombre_tabla

(nombre_columna tipo_columna,

...)

TABLESPACE nombre_tablespace

STORAGE

(INITIAL tamaño

NEXT tamaño

PCTINCREASE porcentaje

MINEXTENTS número

MAXEXTENTS número o UNLIMITED

);
```

- INITIAL: Es el tamaño en *bytes* de la extensión inicial; la primera que se crea, en el instante mismo en que se crea la tabla (aún sin datos). También se pueden utilizar las letras **K** o **M** seguidas del número para denotar *kilobytes* o *megabytes*.
- NEXT: Análogo al anterior, pero aplica a los tamaños de las extensiones posteriores.
- PCTINCREASE: Este parámetro especifica el tamaño de las extensiones posteriores a la segunda. Así como *initial* indica la extensión de la primera extensión, *next* lo indica para la segunda y *pctincrease* es el porcentaje en que se incrementarán los tamaños de las extensiones en adelante. El valor 0 (cero)

indica que todas las extensiones tendrán el mismo tamaño que lo indicado en *next* y el valor 100 que se incrementaran en un 100% con respecto a ese valor (es decir, el doble de *next*).

- MINEXTENTS: Con este parámetro se puede indicar cuántas extensiones se crearán en el momento en que se cree el objeto, todas respetando el valor de lo indicado en *initial*.
- MAXEXTENTS: Permite indicar el número máximo de extensiones que podrá tener el objeto creado.

### Tablas particionadas

Estos objetos siguen correspondiendo a las tablas que conocemos hasta ahora, pero la diferencia radica en cómo se va a almacenar la información físicamente.

En efecto, al instante de crear una tabla podemos elegir qué rangos de datos van a quedar almacenados en un *tablespace* u otro. Y aunque lo anterior no denote espacio físico de almacenamiento (un *tablespace* es un segmento lógico), recordemos que sí podemos elegir dónde estarán ubicados (en qué discos) los *datafiles* de esos *tablespace* y entonces sí que podremos decir que estamos escogiendo el lugar físico donde se grabarán ciertos rangos de datos de una tabla, lo que nos da las siguientes ventajas:

- Segmentos de datos más pequeños: esto influye directamente en el rendimiento de las búsquedas porque cada partición es tratada como si fuera una tabla diferente; Oracle siempre sabrá en que partición buscar cuando se referencia a la tabla particionada, entonces debe buscar en un trozo más pequeño.
- Índices más pequeños: con la partición por rangos es posible crear índices individuales para cada partición.
- Respaldo más rápido: ya que los datos se encuentran en segmentos separados, el mecanismo de respaldo puede correr en paralelo.
- La sintaxis de la creación de una tabla particionada es la siguiente:

```
CREATE TABLE [esquema.] nombre_tabla
```

```
(nombre_columna tipo_columna)
```

```
PARTITION BY RANGE (lista_columnas)
```

```
(PARTITION [nombre_particion] VALUES LESS THAN valor_columna
```

```
TABLESPACE nombre_tablespace
```

```
[, (PARTITION [nombre_particion] VALUES LESS THAN valor_columna
```

```
TABLESPACE nombre_tablespace)]
```

Por ejemplo, si deseamos tener una tabla que almacene los países del mundo, con la siguiente estructura:

Código	Nombre	Población	Continente

Y deseamos particionarla por el código, entonces la sintaxis de creación tendría que ser como sigue:

```
CREATE TABLE paises
```

```
(codigo number(3), nombre varchar2(40), población number(12), ....)
```

```
PARTITION BY RANGE (codigo)
```

```
(PARTITION VALUES LESS THAN 2 -- (regiones con código=1)
```

```
TABLESPACE ts_reg1
```

```
, PARTITION VALUES LESS THAN 3 -- (regiones con código=2)
```

```
TABLESPACE ts_reg2;
```

### Las Cláusulas PCTFREE y PCTUSED

Al momento de crear una tabla, es posible indicar, mediante dos parámetros al momento de su creación, ciertas condiciones de almacenamiento especiales que dicen relación con la volatilidad de los datos y cómo gestionar mejor el espacio (bloques) asignado a cada extensión del objeto. Estos parámetros se denominan PCTFREE y PCTUSED.

**PCTFREE:** Determina el porcentaje de espacio que se reservará en cada bloque de datos de una tabla para futuras actualizaciones de los registros que se graben en ese mismo bloque. El valor que se asigne al parámetro implica conocer la frecuencia de *updates* que se harán a la tabla.

Los valores sugeridos para distintas frecuencias de actualización de filas proyectada para la tabla, son los siguientes:

- **Alta:** Cuando hay muchas actualizaciones que no necesariamente puedan hacer crecer el registro de la tabla, se puede establecer un porcentaje igual a 10.
- Cuando se incrementa el tamaño de la fila en las actualizaciones, y éstas son además de **alta** periodicidad, un valor de 20% es suficiente.
- Cuando casi no existen actualizaciones o la frecuencia es muy baja, basta con reservar un 5% de espacio para permitir actualizaciones dentro del mismo bloque.

**PCTUSED:** Este parámetro está relacionado con la frecuencia de inserciones que se pueden hacer a una tabla. Determina el mínimo porcentaje de espacio usado que será mantenido para cada bloque de datos, antes de crear el próximo segmento.

Si la actividad de inserción es alta o baja, los valores sugeridos para setear este parámetro son los siguientes:

- **Alta:** Establézcase el porcentaje cercano o igual a 40.
- Si es **alta** y además con mucha frecuencia de actualizaciones, establézcase un valor de 60.
- Si la frecuencia de inserciones de filas es **baja**, un porcentaje del 60% también es válido para este parámetro.

## **Vistas**

Una vista es una especie de *ventana* dentro de una tabla. Es una estructura lógica que tiene la apariencia de una tabla, sin llegar a serla. El objetivo de crear vistas es el de tener que prescindir de la tabla cuando se desea permitir la manipulación de datos a otros usuarios; así como también de prevenir que se altere de manera involuntaria el contenido de la información más sensible que pudiera encontrarse en ciertas tablas.

Las vistas se forman haciendo una selección de campos de una o varias tablas. También se puede reemplazar una vista que ya existe con una sintaxis diferente, manteniendo su nombre.

La sintaxis de la creación de vistas es:

```
CREATE [OR REPLACE] VIEW nombre_vista AS
```

```
SELECT columna1, columna2, ...
```

```
FROM tabla1, ...
```

```
WHERE columna > valor;
```

## **Sinónimos**

Los sinónimos son objetos del sistema que apuntan a otros objetos. Implementan **alias** de tablas, vistas, secuencias o unidades de programas. Por lo general se utilizan para esconder ciertos detalles del objeto que representan al usuario final.

Los sinónimos pueden ser públicos o privados. Los primeros son aquellos que caen dentro del esquema PUBLIC y son vistos por todos los usuarios de la misma base de datos. Los sinónimos privados se crean dentro del esquema de un usuario en particular y sólo estará visible para quienes él estime conveniente.

Sintaxis de creación de sinónimos:

```
CREATE [PUBLIC] SYNONYM nombre_sinonimo FOR [esquema.] nombre_objeto ;
```

## ***Indices***

Un índice es una estructura diseñada para obtener un acceso más rápido a los datos contenidos dentro de una tabla.

Un índice es independiente de los datos almacenados en la tabla y cuando se encuentra bien definido, es decir, cuando se forma atendiendo a la gran mayoría de las consultas que se harán sobre una tabla, reduce significativamente la búsqueda, aumentando el rendimiento.

Inmediatamente luego de creado el índice, Oracle comienza a mantenerlo de acuerdo a las inserciones, actualizaciones y eliminaciones de registros de la tabla en la cual se ha implementado.

### Tipos de índices

Existen tres tipos de índices cuya naturaleza depende de la forma en que haya sido creado. Estos tipos son:

- Un **índice único** es aquel que tiene la restricción adicional de que el grupo de columnas indexadas define una única fila. Sin embargo, si no van a existir más grupos de columnas con esta características dentro de una misma tabla, se recomienda crear el conjunto como una clave primaria ya que de todas formas Oracle asociará un índice único a esta restricción (la clave primaria).
- Un **índice no único**, que es aquel que no impone la restricción de que las filas no deban repetirse.
- Un **índice compuesto** es aquel que agrupa varias columnas de la tabla. Este tipo es muy útil cuando las sentencias de selección (SELECT) efectúan búsquedas por varios criterios (columnas) en una misma tabla. Es importante el orden en que se ponen las columnas al crear el índice; la columna más referenciada debería ser puesta en primer lugar y así sucesivamente.

Cuando se crea un índice (de cualquier tipo) también se crea un segmento de datos para guardar esa información, que también se verá afectada por la misma cláusula *storage* que se estudió para el caso de las tablas.

### Consideraciones en el diseño de índices

Un índice sólo es efectivo cuando es utilizado. Es por eso que debe asegurarse que la frecuencia de uso sea muy alta y que su implementación redunde en mejoras de rendimiento de las consultas efectuadas a la tabla donde reside el índice. Sin embargo, no debe explotarse el uso de los índices dentro de una misma tabla porque con cada operación de inserción, actualización o eliminación que se lleva a cabo sobre una tabla, sus índices se deben recrear, con el consiguiente *overhead* que se produce. A menudo es conveniente eliminar o desactivar temporalmente un índice cuando sabemos que se va a efectuar una operación de carga/actualización/eliminación masiva en la tabla para evitar este *overhead* y más tarde volver a crearlo, cuando la operación haya finalizado.

Considere las siguientes reglas de indexación para cuando se enfrente a la tarea de decidir *qué tablas indexar*:

- Indexe solamente las tablas cuando las consultas (*queries*) no accedan a una gran cantidad de filas de la tabla. Use índices cuando una *query* acceda a un porcentaje menor al 5% de las filas de una tabla.
- No indexe tablas que son actualizadas con mucha frecuencia.
- Indexe aquellas tablas que no tengan muchos valores repetidos en las columnas escogidas. Recuerde que finalmente el índice hace una búsqueda secuencial dentro de un conjunto de filas objetivo.
- Las *queries* muy complejas (en la cláusula WHERE) por lo general no toman mucha ventaja de los índices. Cuando posea más experiencia podrá corroborar esta afirmación y estará preparado para arreglar estas situaciones.

También es importante decidir *qué columnas indexar*. Siga las siguientes reglas cuando tenga que tomar esta decisión:

- Escoja las columnas que se utilizan con mayor frecuencia en las cláusulas WHERE de las consultas.
- No indexe aquellas columnas que tengan demasiados valores repetidos en ellas.
- Las columnas que toman valores únicos son excelentes candidatas para indexar. Oracle automáticamente indexa las claves primarias de las tablas.
- Indexe las columnas que sirven para unir una tabla con otras (*join* en las consultas).
- Si hay columnas que no tienen valores únicos por sí solas pero que en conjunto con otra columna forman una dupla única o con pocas repeticiones (menos que las columnas individualmente), entonces conviene indexarlas (siempre y cuando existan consultas que las utilicen en conjunto). Estos índices reciben el nombre de *índices compuestos*.

Sintaxis de creación de índices:

```
CREATE INDEX nombre_indice ON [esquema.]nombre_tabla (columna1 [, columna2, ...])
```

```
TABLESPACE nombre_tablespace ;
```

### Índices particionados

Tal como en el caso de las tablas, los índices también pueden ser almacenados en *tablespaces* separados. La sintaxis de creación de los índices de este tipo es similar a la de creación de las tablas particionadas:

```
CREATE INDEX nombre_indice ON [esquema.]nombre_tabla (columna1 [, columna2, ...])
```

```
PARTITION BY RANGE (columna1 [, columna2, ...])
```

```
PARTITION particion1 VALUES LESS THAN (valor) TABLESPACE tablespace1
```



[PARTITION *particion2* VALUES LESS THAN (MAXVALUE) TABLESPACE *tablespace2*] ;

Observe que el valor representado por la palabra reservada MAXVALUE será siempre el mayor valor presente en la tabla para la columna especificada.

## ***Secuencias***

A menudo es preciso generar números en forma ordenada para implementar, por ejemplo, una clave primaria en una tabla o garantizar que esos números no se repiten y van siempre en un orden predefinido por el desarrollador (no necesariamente secuenciales).

La forma tradicional de efectuar lo anterior sería almacenar el último número utilizado en un registro especial, bloquearlo, obtener el próximo valor, actualizar el registro, desbloquearlo y utilizar el número. Sin embargo, para eso Oracle implementa los objetos denominadas *secuencias*, que permiten hacer lo anterior de manera transparente para el usuario.

Cuando se define una secuencia se deben indicar, como mínimo, el valor de partida (valor mínimo) y el incremento.

La sintaxis de creación de una secuencia es la siguiente:

CREATE SEQUENCE *nombre\_secuencia*

INCREMENT BY *número*

START WITH *número*

MINVALUE *número* [o NOMINVALUE]

MAXVALUE *número* [o NOMAXVALUE]

NOCYCLE [o CYCLE] ;

Los parámetros significan lo siguiente:

- *Increment by*: Indica la cantidad de incremento de la secuencia.
- *Start with*: Es el valor de partida de la secuencia.
- *Minvalue*: Indica cuál será el valor mínimo de la secuencia.
- *Maxvalue*: Corresponde al valor máximo que puede tomar la secuencia.
- *Nocycle*: Es el valor por defecto para establecer si la secuencia deberá comenzar nuevamente a generar valores una vez que ha alcanzado el máximo.

## **GLOSARIO DE TÉRMINOS**

La siguiente es una lista de los términos más utilizados cuando se trabaja con bases de datos Oracle. Las definiciones ayudarán a comprender con mayor claridad algunos conceptos que se mencionan a lo largo de los diferentes capítulos de este manual.

- **Administrador de Base de Datos**

El administrador o DBA es el principal responsable de la operación, configuración y rendimiento de una base de datos. Su principal tarea consiste en resguardar la integridad de los datos almacenados en la base, proveyendo para esto mecanismos de respaldo, efectuando monitorizaciones periódicas al sistema, implementando medidas de seguridad, etc.

- **Bloque**

Un bloque es la unidad más pequeña de almacenamiento en una base de datos Oracle. El tamaño mínimo es de 2 KB y el máximo no debiera superar los 16 KB.

- **Buffer**

Este término se refiere a una cantidad de memoria utilizada para almacenar información. Un *buffer* comúnmente almacena datos que están a punto de ser usados o se acaban de utilizar recientemente. En la mayoría de los casos son copias exactas de datos que se encuentran almacenados en el disco y se mantienen en memoria con el fin de lograr un acceso más rápido y ayudar de esa manera a mejorar el rendimiento de un sistema.

En Oracle, los *buffers* del SGA almacenan los bloques de datos usados más recientemente. El conjunto de *buffers* que guardan estos bloques reciben el nombre de *database buffer cache*; y aquellos que se utilizan para guardar temporalmente las entradas del tipo *redo log* hasta que se escriben en el disco, se conocen como *redo log buffers*.

- **Caché**

Es un área de almacenamiento implementada en la memoria RAM del computador que permite accesos más rápidos a la información ya que es mucho más veloz que la memoria. En Oracle, los *buffers* de bloques y el área *shared pool* son consideradas áreas caché. Estas guardan los datos que se utilizan con mayor frecuencia y los mantienen disponibles por si son requeridos en los procesos de consulta hasta que nuevos datos más frecuentemente usados los reemplazan.

- **Checkpoint**

Un *checkpoint* es una operación que fuerza a que todos los cambios registrados en bloques de datos en memoria, sean escritos en el disco.

- **Clean buffer**

Un buffer de este tipo es aquel que no ha sido modificado y que por lo tanto el proceso DBWR no utilizará para confirmar los cambios en el disco (porque no ha sufrido cambios).

- **Concurrencia**

Este término se refiere a la capacidad de permitir muchas funciones al mismo tiempo. Oracle provee a muchos usuarios el acceso simultáneo a sus servicios, implementando de esta forma la concurrencia.

## **DBA**

Vea *Administrador de la Base de Datos*

## **DBMS**

El *database management system* o DBMS corresponde al software y grupo de herramientas que permiten manejar la base de datos. Un RDBMS es un DBMS relacional, es decir, cuya naturaleza es la formación de relaciones al interior del mismo.

## **DDL (comandos DDL)**

Los comandos DDL (*data definition language*) son utilizados en la creación y modificación de objetos del esquema. Proveen la habilidad de crear, alterar e incluso eliminar objetos de un esquema, otorgar y revocar privilegios y roles a los usuarios, establecer opciones de auditoria e incluso agregar comentarios al diccionario de datos del sistema. Estos comandos están estrechamente relacionados con las labores de administración de la base de datos.

## **Diccionario de Datos**

El diccionario de datos es un grupo de tablas de Oracle que se utilizan para almacenar información sobre el resto de las tablas, índices, clusters y otros objetos de la base de datos.

## **DML (comandos DML)**

Los comandos DML (*data manipulation language*) son menos poderosos que los comandos DDL en cuanto a administración se refiere, de hecho, implementan modificaciones sobre la información que se guarda en los objetos de una base de datos. Estas sentencias son del tipo DELETE, INSERT, SELECT y UPDATE, principalmente.

## **Esquema**

Un esquema es una colección de objetos asociados dentro de una base de datos.

## Función

Una función es un grupo de sentencias SQL, escritas generalmente en PL/SQL que implementan una serie de rutinas que devuelven un valor. Son casi idénticas a los procedimientos y sólo se diferencian en esa última condición. Implementando funciones en el servidor de base de datos se reduce el tráfico de comunicaciones en la red, ya que sólo se envían a la función los parámetros de entrada y ésta sólo devuelve el valor al final de todo el proceso, el que es ejecutado en la misma máquina donde reside la base de datos mejorando así el rendimiento general del sistema.

## Memoria Virtual

Indica la memoria que puede ser utilizada por programas que corren en un sistema operativo y que está implementada físicamente en sectores del disco y no en la RAM. El proceso de copiar datos de la RAM al disco (o memoria virtual) se llama paginación (*paging*, en inglés). El archivo resultante es llamado el "*swap file*" y cada vez que un programa accede a esta memoria virtual disminuye el rendimiento del mismo debido a que realmente está accediendo al disco y no a la RAM.

## Procedimiento

Un Procedimiento almacenado es un grupo de sentencias SQL o PL/SQL que implementan un programa que se ejecuta en el servidor de base de datos, pero que a diferencia de las funciones, no devuelve un valor. Al igual que las funciones su implementación permite reducir el tráfico en la red, potenciando el rendimiento del sistema.

## Query

Es una consulta efectuada contra la base de datos en lenguaje SQL. Se genera utilizando la sentencia SELECT. Su principal característica es que no efectúa cambios en la base de datos; por este motivo es llamada también una *transacción de sólo lectura*.

## System Global Area (SGA)

El SGA es un área compartida de memoria que utiliza Oracle para guardar información de control en una instancia. Se asigna un espacio a esta área en cuando la instancia se levanta (*startup*) y se elimina cuando ésta se baja (*shutdown*). Cada instancia de Oracle maneja su propia SGA y guarda información de los *buffers* y la *shared pool*.

## Tablas de rendimiento dinámicas

Estas tablas son creadas cuando se levanta una instancia y se usan para guardar información acerca del rendimiento de ésta.. Esta información incluye notas acerca de la conexión, datos que manejan los procesos de entrada/salida, valores de los parámetros de inicialización , entre otros.

## Transacción

Una transacción es una unidad lógica de trabajo que consiste de una o más sentencias SQL, que pueden finalizar con un *commit* o un *rollback*. Las métricas de rendimiento

utilizan comúnmente las unidades "transacciones por segundo" o "transacciones por minuto".

## Trigger

Un *trigger* es un mecanismo que permite escribir procedimientos que son ejecutados en forma automática (sin una orden explícita del usuario o programador) cuando ocurre un evento de INSERT, UPDATE o DELETE sobre una tabla o vista. Generalmente se utilizan los *triggers* para forzar las restricciones de integridad entre las tablas o automatizar alguna otra función específica.

## Unidad de Almacenamiento

La información es almacenada dentro del computador en forma binaria. Las unidades que se refieren a estos conceptos (datos binarios) son las siguientes:

<i>Término</i>	<i>Definición</i>	<i>Comentario</i>
<i>bit</i>	La unidad más pequeña de almacenamiento	Un bit representa un <b>1</b> ó un <b>0</b>
<i>nibble</i>	4 bits	Este término no se utiliza con demasiada frecuencia
<i>byte</i>	8 bits	Es la unidad de almacenamiento más utilizada
<i>Word</i>	Término que depende de la arquitectura	En muchos sistemas, una "palabra" corresponde a 16 bits. También puede representar 32 ó 64 bits
<i>kilobyte (KB)</i>		En términos computacionales, un <i>kilobyte</i> equivale a 1024 <i>bytes</i>
<i>megabyte (MB)</i>		Este término denota a 1024 KB o 1048576 <i>bytes</i>
<i>gigabyte (GB)</i>		Un gigabyte corresponde a 1024 <i>megabytes</i> o 1,073,741,824 <i>bytes</i>
<i>terabyte (TB)</i>		Un <i>terabyte</i> son 1024 <i>gigabytes</i> o 1,099,511,627,776 <i>bytes</i>

## EJERCICIO DEMOSTRATIVO

En las próximas páginas se demostrarán las principales funcionalidades que se pueden llevar a cabo en una sesión típica de administración cuando comienza un proyecto.

Se mostrará cómo se efectúa la creación de los usuarios, de los espacios de tablas, cómo se asignan y se eliminan los privilegios mediante roles y otras funcionalidades, utilizando la herramienta gráfica *DBA Studio*.

### *Creación de Usuarios*

Para crear un usuario se debe seleccionar, en primer lugar, la opción correspondiente en la aplicación, de la siguiente manera:

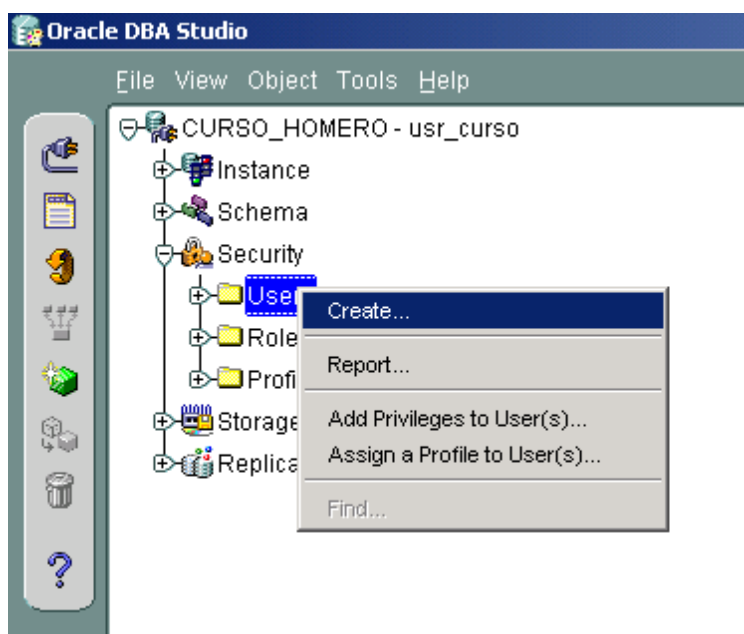


Figura No. 9 Menú flotante de creación de usuarios

Haciendo clic con el botón derecho una vez que estamos sobre la opción "Users" del administrador de seguridad (figura anterior), aparece la interfaz de creación de usuarios, que tiene las siguientes características y que se rellena con los datos que se muestran:

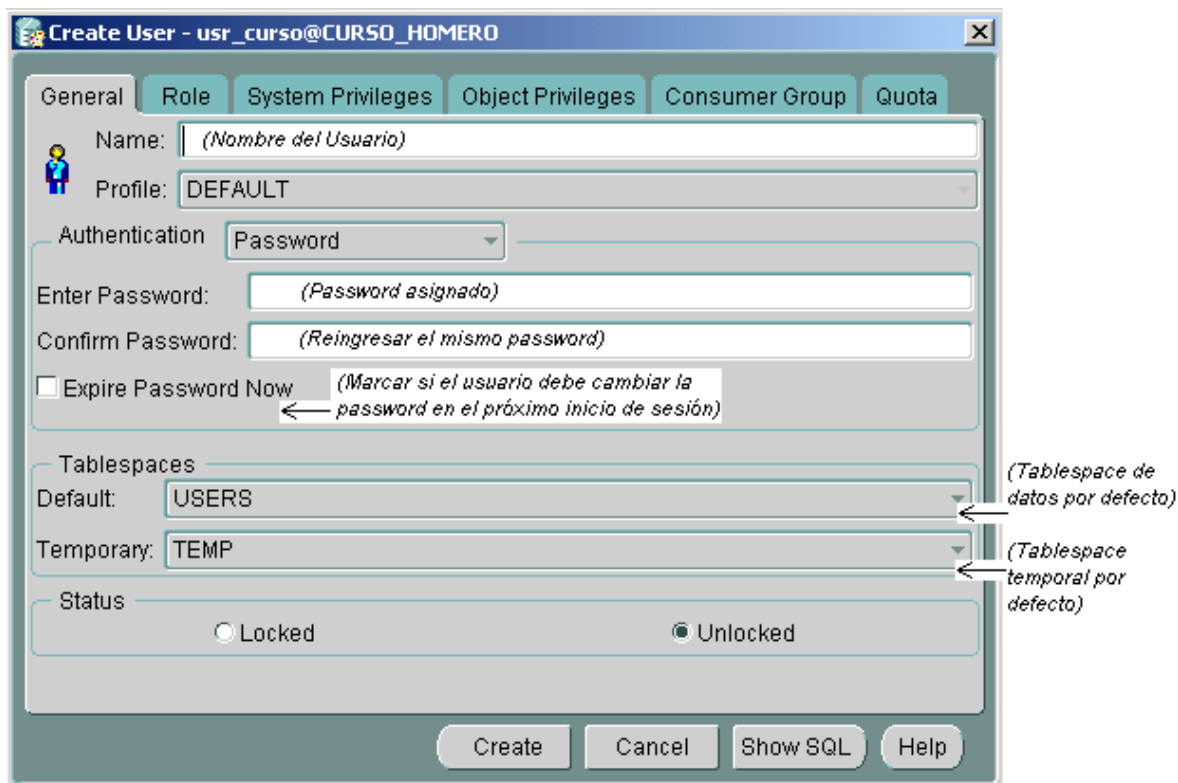


Figura No. 10 Ventana principal de creación de usuarios

Además, como se observa en la figura anterior, existen otras fichas que permiten asociar al usuario algún rol, privilegios sobre objetos comunes o del sistema y cuotas de espacio, entre otros.

Otras opciones que es necesario configurar la primera vez se refieren a brindar la capacidad al usuario de poder conectarse a una base de datos e iniciar una sesión por primera vez, dándosele también la oportunidad de crear objetos en su espacio o esquema, asignando un volumen máximo a ese espacio, para cada usuario que se crea.

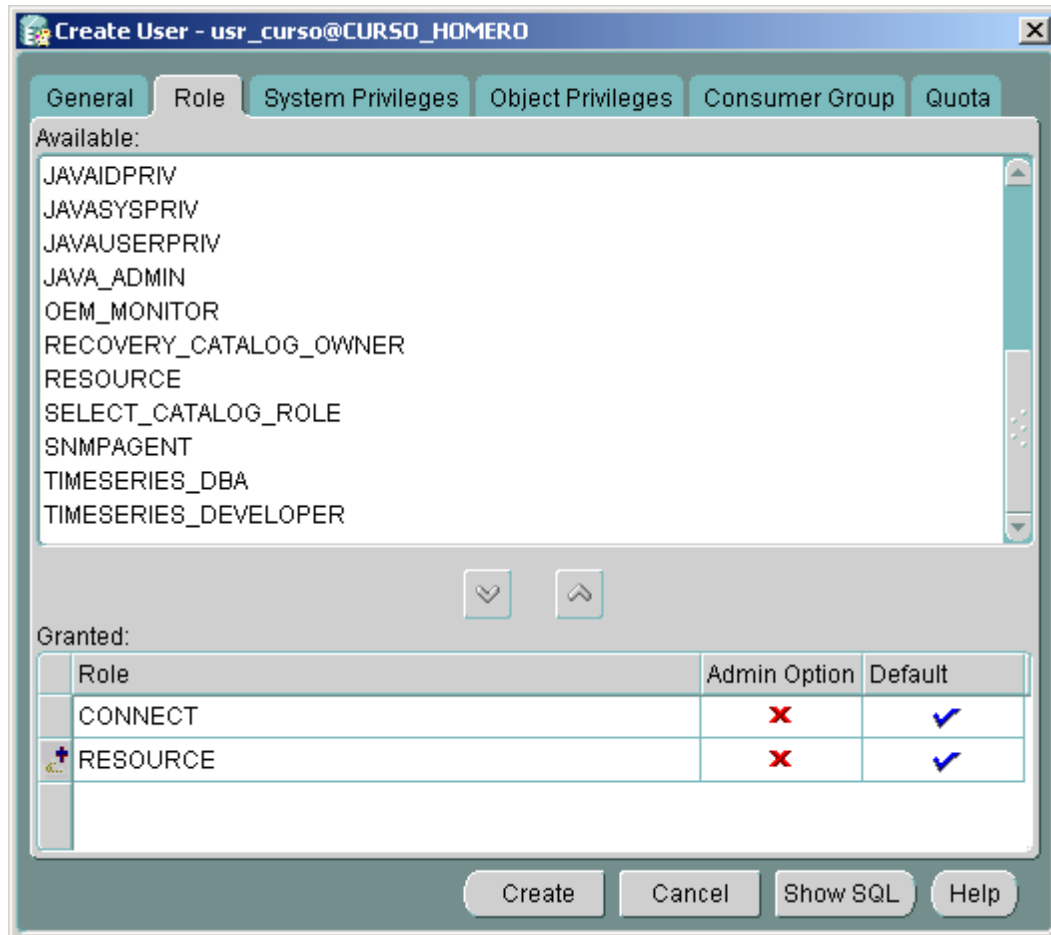



Figura No. 11 Roles concedidos al usuario

**Rol Connect:** Permite al usuario iniciar una sesión en la base de datos.

**Rol Resource:** Permite crear objetos, entre otros.

El símbolo  a la izquierda del rol *resource* significa que esa es una línea que se está agregando en la lista. En efecto, sólo se asigna por defecto el rol *connect* y nosotros debemos agregar el segundo cada vez para permitir al usuario crear objetos en su esquema.

Enseguida, para definir la cuota de espacio, tenemos que abrir la última pestaña de la ventana de creación de usuarios y empezar a asignar, *tablespace* por *tablespace*, el espacio definido para este usuario en particular. De esta forma podemos establecer las cuotas de cada usuario en cada uno de los espacios definidos en el sistema.



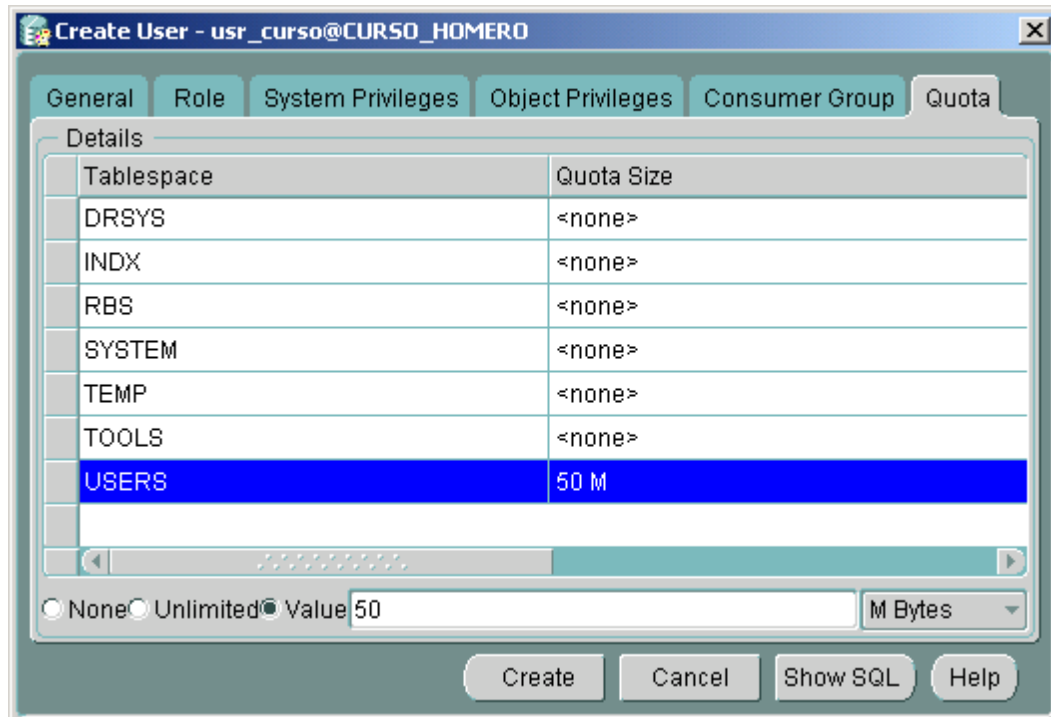


Figura No. 12 Cuota del usuario por *tablespace*

### ***Creación de Tablespaces***

Para efectos prácticos vamos a suponer que los usuarios creados en este ejercicio deben estar asignados a un espacio de tablas diferente a los ya existentes. Por lo tanto, no nos sirve que tengan el *tablespace* **users** asignado por defecto.

Para crear un nuevo *tablespace* y asignarlo a los usuarios creados, se debe proceder de la siguiente manera, si estamos usando *DBA Studio*:

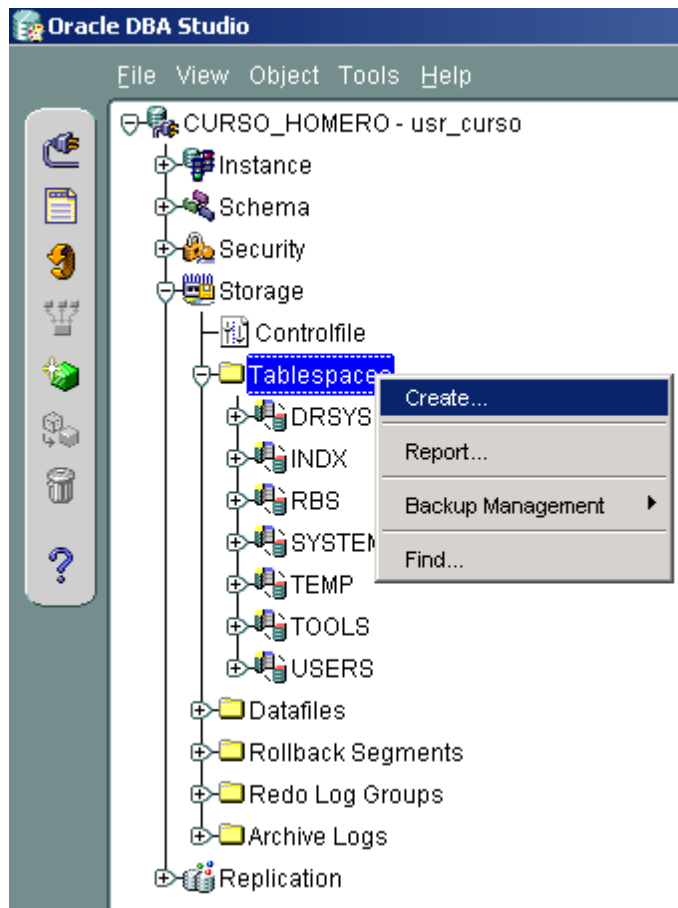


Figura No. 13 Como crear un *tablespace*

La interfaz principal de creación de los *tablespaces* aparece cuando seleccionamos la opción mostrada en la figura anterior y es la siguiente:

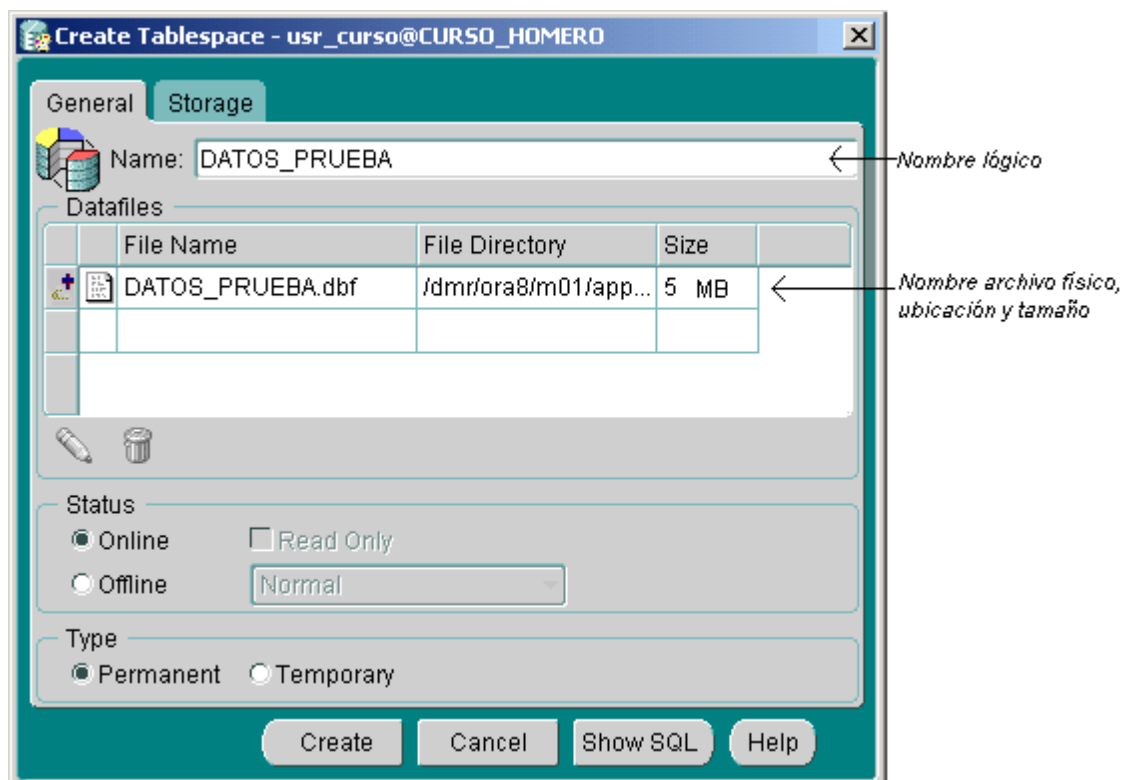


Figura No. 14 Detalle de la creación de un *tablespace*

En la figura anterior se observa la relación entre *tablespace* y *datafile*, éste último corresponde al archivo físico de extensión DBF que se muestra en la línea de detalle.

Para el ejemplo, el *tablespace* creado se llama **datos\_prueba**.

Ahora, para asignar el espacio de tablas recién creado a nuestros usuarios, basta con editar sus características (botón derecho sobre el nombre del usuario) y asignar el nuevo *tablespace* a cada uno de ellos.

Posteriormente, con el fin de poder otorgarle al usuario la posibilidad de crear tablas en su esquema, no debemos olvidarnos de asignar una cuota de espacio a cada usuario dentro del *tablespace*.

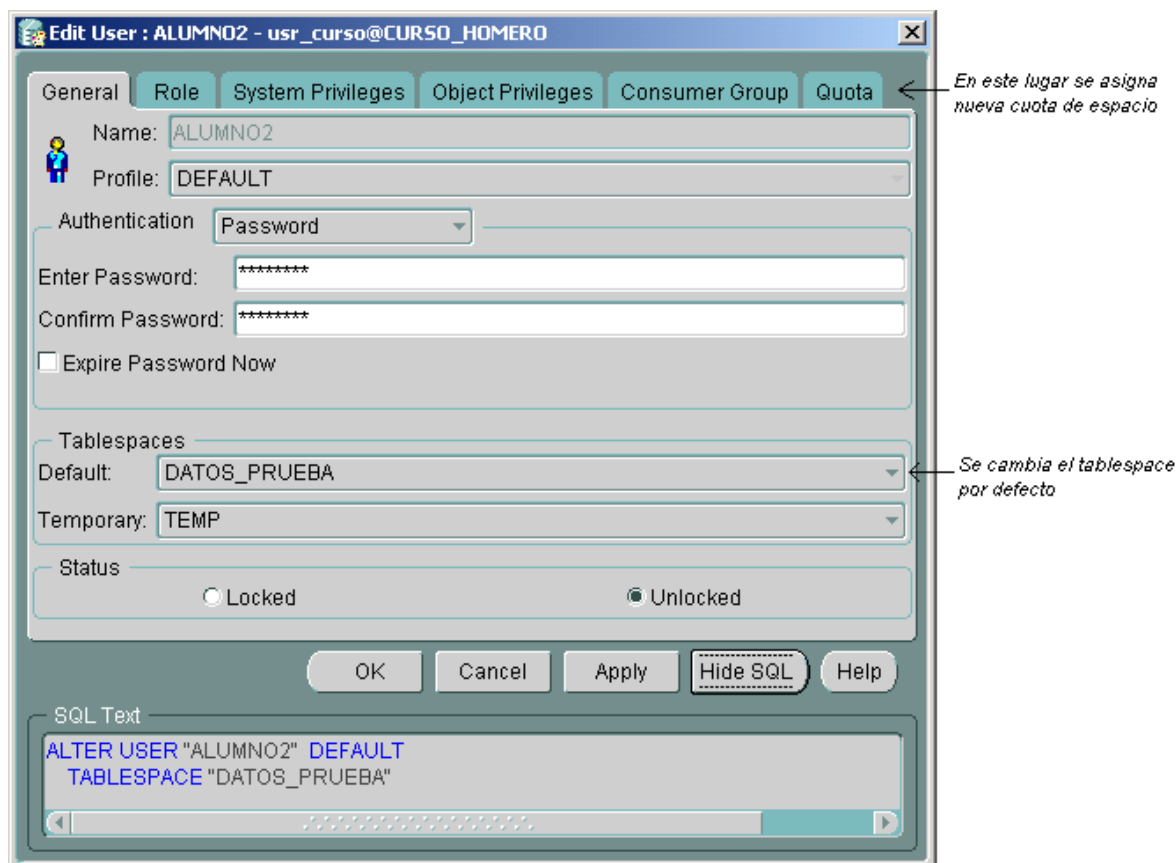


Figura No. 15 Modificación de datos de un usuario

### ***Creación de Tablas***

Enseguida, con el fin de conocer en la práctica cómo trabaja Oracle las extensiones de las tablas y aprender a monitorearlas para evitar que crezcan demasiado y puedan llegar a causar detenciones de la base de datos.

Un error muy común es que una tabla haya alcanzado el máximo posible de sus extensiones (valor indicado al crearla) y que por lo tanto no pueda seguir creciendo en tamaño, por lo que cada vez que se intente insertar datos o actualizarla incrementando su tamaño, aparezca un error de Oracle que nos lo impida.

#### **Ejemplo:**

Al crear una tabla, los parámetros que identifican los tamaños y cantidad de extensiones posibles para una tabla son los siguientes:

General Constraints Cluster Columns Partitions Storage Options

Name: TABLA\_PRUEBA

Schema: ALUMNO1

Tablespace: DATOS\_PRUEBA

Table: ☒ Standard ☐ Organized Using Index (IOT) ☐ Object Table

☒ Define Columns ☐ Define Query ☐ Object Table

Name	Schema	Datatype	Size	Scale	Ref	Nulls?	Default Value
COL_1	<None>	VARCHAR2	10				
COL_2	<None>	NUMBER	8	2			

Create Cancel Show SQL Help

Figura No. 16 Creación de una tabla (paso 1)

Las opciones que le permiten a Oracle efectuar el manejo del espacio de almacenamiento se ingresan (considerando la figura anterior), en la pestaña "Storage":

**Create Table - usr\_curso@CURSO\_HOMERO**

General Constraints Cluster Columns Partitions **Storage** Options

☒ Explicit ☐ Auto Calculation

**Extents**

Initial Size:  K Bytes ← *Tamaño inicial*

Next Size:  K Bytes ← *Tamaño del incremento*

Increase Size by:  % ← *Porcentaje de incremento*

Minimum Number:  ← *Extensiones mínimas*

Maximum Number: ☐ Unlimited ☒ Value  ← *Extensiones máximas*

**Space Usage** ← *PCTFREE*

% Free:  % Used:  ← *PCTUSED*

**Number of Transactions**

Initial:  Maximum:

**Free Lists**

Free Lists:  Groups:

**Buffer Pool**

Buffer Pool:

Create Cancel Show SQL Help

Figura No. 17 Creación de una tabla (paso 2)

Finalmente, introduciendo ciertos valores que deberán establecerse en rigor luego de un exhaustivo análisis del objeto que se está creando (porcentaje de volatilidad, crecimiento esperado, restricciones de tamaño en los discos, etc.), un Dba podría establecer que para esta tabla podrían aplicar ciertos valores, que para este ejemplo, se muestran a continuación, junto con la sentencia SQL que podría haberse escrito en lugar de utilizar la forma gráfica:

**Create Table - usr\_curso@CURSO\_HOMERO**

General Constraints Cluster Columns Partitions **Storage** Options

☒ Explicit ☐ Auto Calculation

**Extents**

Initial Size: 8 K Bytes

Next Size: 2 M Bytes

Increase Size by: 0 %

Minimum Number: 1

Maximum Number: ☐ Unlimited ☒ Value 200

**Space Usage**

% Free: 10 % Used: 40

**Number of Transactions**

Initial: Maximum:

**Free Lists**

Free Lists: Groups:

**Buffer Pool**

Buffer Pool: DEFAULT

Create Cancel Hide SQL Help

**SQL Text**

```
CREATE TABLE "ALUMNO1"."TABLA_PRUEBA"("COL_1" VARCHAR2(10) NOT
NULL,"COL_2" NUMBER(8, 2) NOT NULL)
TABLESPACE "DATOS_PRUEBA" PCTFREE 10 PCTUSED 40
STORAGE ( INITIAL 8K NEXT 2M MINEXTENTS 1 MAXEXTENTS 200
PCTINCREASE 0)
```

Figura No. 18 Sintaxis de creación de la tabla del ejemplo

### ***Revisión de las extensiones***

Para conocer la cantidad de espacio utilizada por una tabla en el disco, disponemos de otras dos tablas de apoyo donde se guarda esta información. La primera de ellas, la tabla **user\_segments**, guarda un resumen de todo el segmento o tabla en cuanto a las condiciones en que fue creada (tamaño y número de las extensiones, etc.). La segunda, de nombre **user\_extents**, guarda la información de detalle de cada extensión que se va agregando al segmento, insertándose un registro en esta tabla por cada extensión que se crea para un objeto.

Las descripciones de las tablas y de sus campos más importantes (para fines de este curso) son las siguientes:

#### **USER\_SEGMENTS:**

Campo	Significado
SEGMENT_NAME	Nombre del segmento de datos. Por ejemplo, una tabla
PARTITION_NAME	Nombre de la partición, si se trata de una tabla particionada
SEGMENT_TYPE	Tipo del segmento, que puede ser una tabla, secuencia, etc.
TABLESPACE_NAME	Nombre del <i>tablespace</i> donde reside el objeto indicado por el segmento
BYTES	Tamaño en <i>bytes</i> del objeto. Divídase por 1.024 para tener el valor en Kilobytes o por 1.048.576 para expresarlo en Megabytes
BLOCKS	Cantidad de bloques de datos utilizados por el objeto
EXTENTS	Cantidad de extensiones utilizadas por el objeto. Este número corresponde con el total de registros que vamos a encontrar para el mismo objeto en la tabla <i>user_extents</i>
INITIAL_EXTENT	Tamaño de la extensión inicial (en <i>bytes</i> )
NEXT_EXTENT	Tamaño de las extensiones consecutivas (en <i>bytes</i> )
MIN_EXTENTS	Número mínimo de extensiones definidas para el objeto
MAX_EXTENTS	Número máximo de extensiones definidas para el objeto
PCT_INCREASE	Porcentaje de incremento de los tamaños de las extensiones siguientes

**USER\_EXTENTS:**

Campo	Significado
SEGMENT_NAME	Nombre del segmento de datos al que está asociada la extensión
PARTITION_NAME	Nombre de la partición, si se trata de una tabla particionada
SEGMENT_TYPE	Tipo del segmento, que puede ser una tabla, secuencia, etc.
TABLESPACE_NAME	Nombre del <i>tablespace</i> donde reside el objeto indicado por el segmento
EXTENT_ID	Correlativo de la extensión dentro del segmento al que pertenece
BYTES	Tamaño en <i>bytes</i> de la extensión
BLOCKS	Cantidad de bloques de datos utilizados por la extensión