



# Administración de Bases de Datos

## Tema 3. Nivel Físico de una Base de Datos

# Índice

1. EL SGBD ORACLE
2. Tablespaces y Datafiles
3. Vistas Dinámicas del Diccionario
4. Gestión del Espacio Lógico
  - a. Bloques de Datos, Extensiones y Segmentos
5. Estructura de la Memoria
6. Estructura de los Procesos
7. Objetos del Esquema. Tablas, Clusters, Índices
8. Administración del SGBD ORACLE
9. Herramientas
10. Iniciar/finalizar ORACLE

# EL SGBD ORACLE

## ► Estructuras básicas:

- E. Lógica: Representa de los datos y sus relaciones (esquema conceptual).
- E. Física: Almacenamiento de datos.

## ► Estructura Lógica de una BD Oracle:

- Objetos del esquema (*schema objects*): Definición de tablas, vistas, índices, sinónimos, procedimientos almacenados...
- Espacios de Tablas (*tablespaces*): Es un área lógica de almacenamiento.
  - Informan cómo debe ser utilizado el espacio físico de la BD.
  - Describen el almacenamiento físico, gestionando el espacio físico que usa la BD.
  - Cada BD tiene al menos un *tablespace*, aunque puede tener más para mejorar su gestión (uno para usuarios, aplicaciones, *rollback...*).
  - Cada *tablespace* pertenece sólo a una BD y se divide en 1 ó más ficheros de datos.

**Bibliografía:** La mejor sobre este tema es el **Manual de Oracle** de la última versión.  
Concepts:  
<https://docs.oracle.com/en/database/oracle/oracle-database/18/cncpt/index.html>  
Administrator's Guide:  
<https://docs.oracle.com/en/database/oracle/oracle-database/18/admin/index.html>  
SQL Reference:  
<https://docs.oracle.com/en/database/oracle/oracle-database/18/sqlrf/index.html>

# Almacenamiento

- ▶ Cuándo guardamos un dato en la base de datos ¿dónde se almacena realmente?
- ▶ Si se produce un error en la base de datos o un disco falla ¿se pierden las transacciones finalizadas?
- ▶ ¿Cómo sabe el SGBD dónde están los ficheros?
- ▶ Si se produce algún fallo ¿cómo podemos saberlo?



# EL SGBD ORACLE

## ▶ Estructura Física de una BD Oracle: Tipos de Ficheros:

- ▶ Datos: Existen uno o más ficheros que contienen los datos actuales. Todos los datos se almacenan en estos ficheros en un formato propiedad de Oracle de modo que no puedan ser leídos por otros programas.
- ▶ Ficheros del Registro de Rehacer (*redo log*): Registran los cambios efectuados, para poder efectuar operaciones de recuperación (*recovery*).
- ▶ Ficheros de Control: Información general, como nombre de la BD, nombres de sus ficheros, sus localizaciones, fecha de creación, histórico de *backups*...
- ▶ Ficheros para Rastrear (*trace files*) y para Registrar Alarmas (*alert log*): Se registran las operaciones por las que han pasado determinados procesos y los eventos importantes acaecidos a la BD.

# EL SGBD ORACLE

## ▶ Mecanismos para almacenar los ficheros:

### ▶ Mediante el S.O. (el más usado):

- ▶ Los archivos se almacenan en un sistema de archivos
- ▶ El S.O. asigna y libera espacio en disco en los archivos.
- ▶ Cada archivo tiene un nombre y se ofrece como un espacio de direcciones contiguo.
- ▶ La base de datos puede crear, leer, escribir, cambiar el tamaño y eliminar archivos.

### ▶ Oracle Automatic Storage Management (Oracle ASM):

- ▶ Es un gestor de volumen y proporciona un sistema de archivos diseñado exclusivamente para uso de la base de datos

### ▶ Raw device:

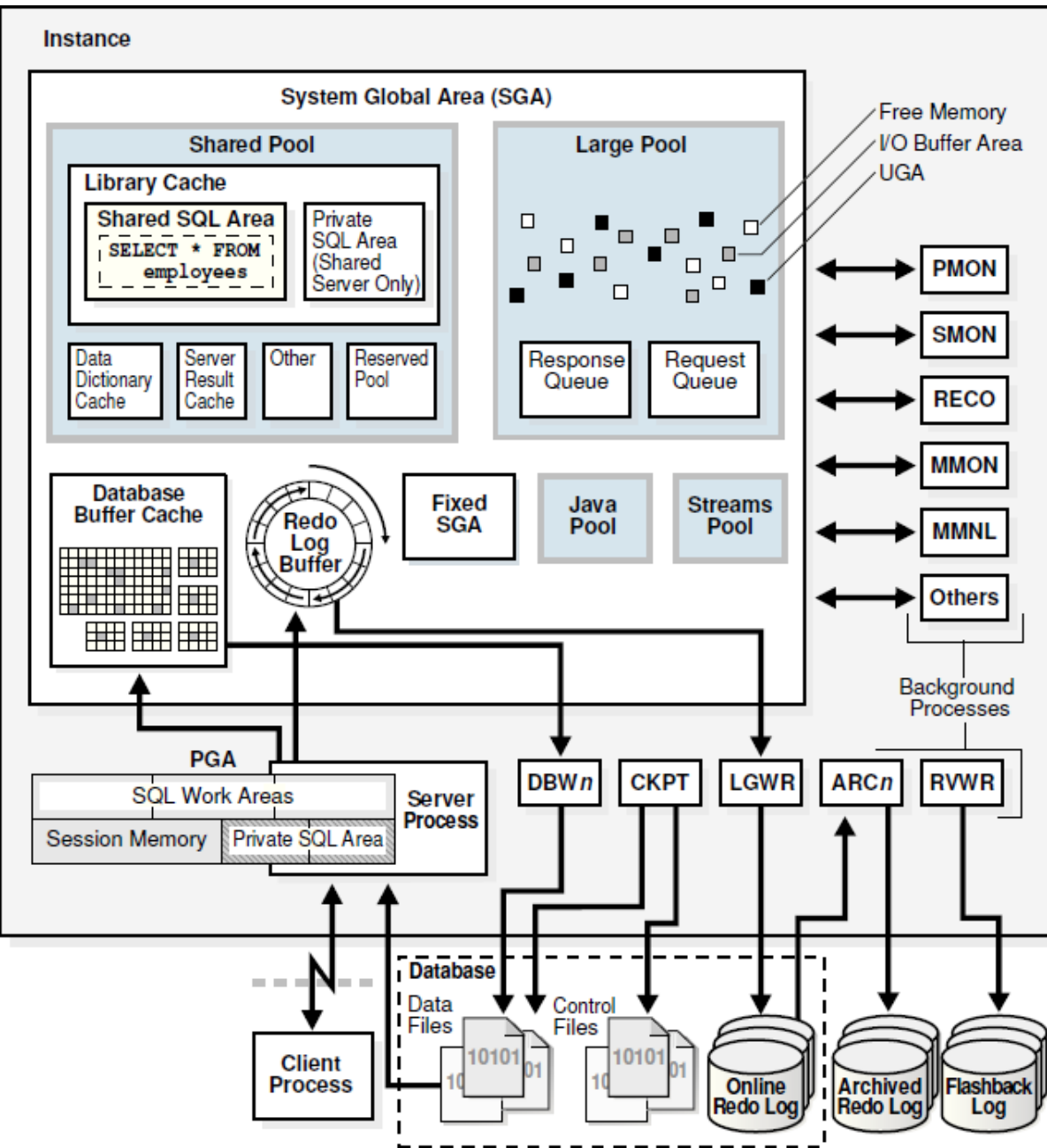
- ▶ Son particiones de disco o volúmenes lógicos no formateados con un sistema de archivos.

### ▶ Cluster File System:

- ▶ Varias computadoras comparten el almacenamiento de archivos, manteniendo consistente la distribución del espacio y el contenido del archivo.
- ▶ Si un equipo del cluster falla, el sistema de archivos sigue disponible.

### ▶ Combinaciones de las anteriores

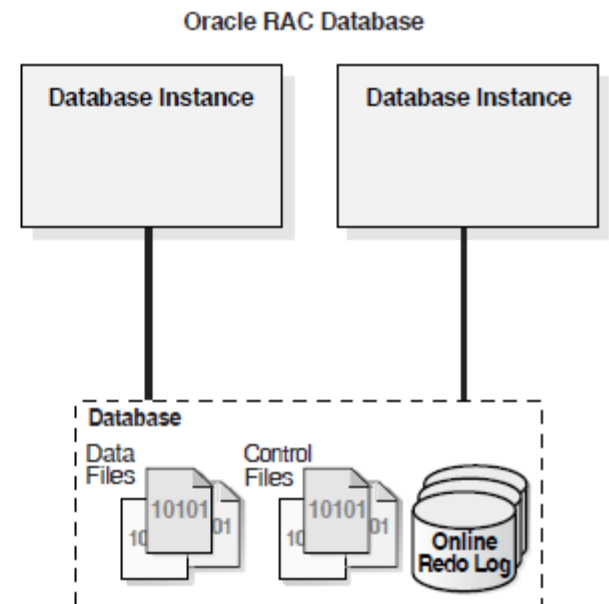
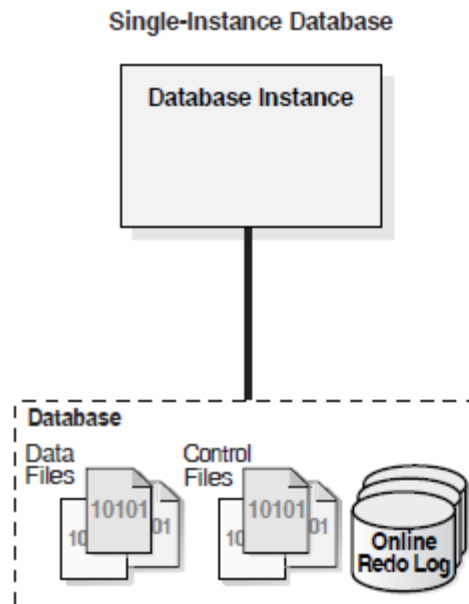
# Instancias de ORACLE



► Para acceder a los ficheros se necesita una instancia

# Instancias de ORACLE

- ▶ **Instancia o Servidor de BD:** Conjunto de estructuras de memoria y procesos que acceden a los archivos de una BD. Distintas instancias *pueden* acceder a la misma BD. Pero 2 BD no pueden ser montadas sobre una instancia (Relación 1 a N).
- ▶ A Partir de Oracle 12, una instancia puede ser de solo lectura → INSTANCE\_MODE → READ\_ONLY | READ\_WRITE



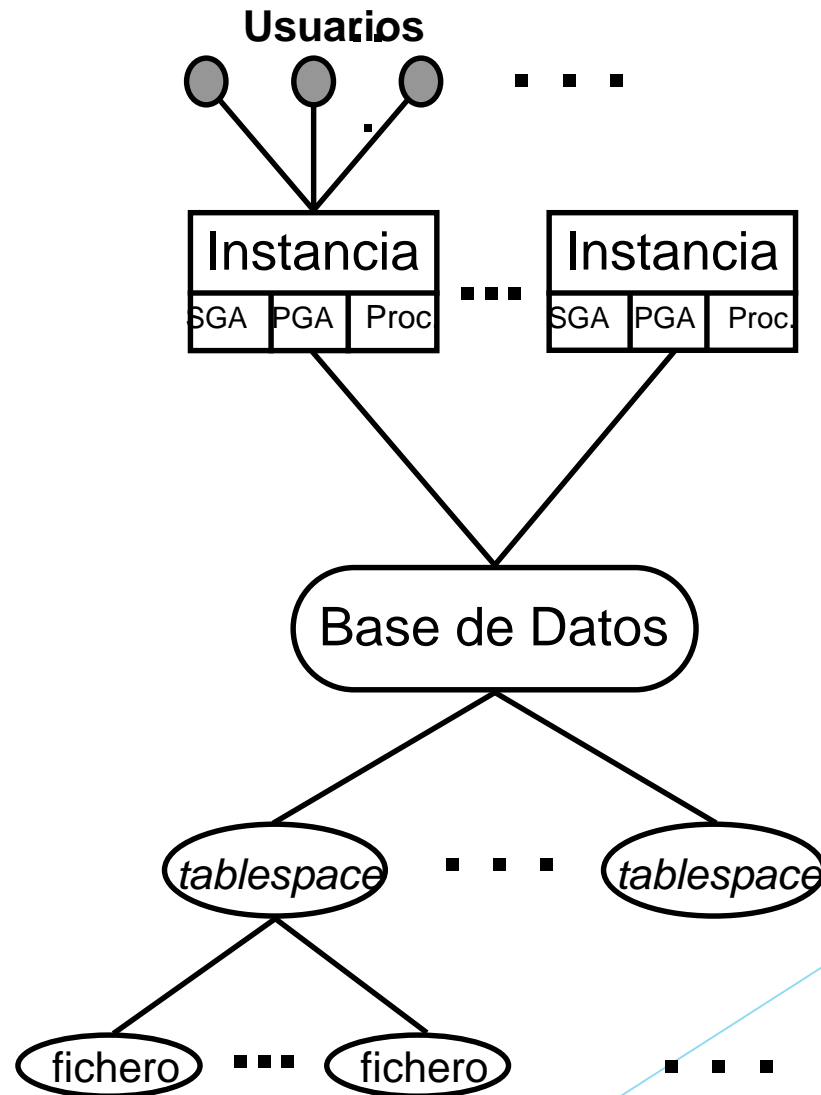


# Instancias de ORACLE (Oracle Instance)

## ► Partes de una instancia:

- **System Global Area (SGA)**: área de memoria con la información de la BD que pueden compartir los usuarios. Se crea cuando se empieza a usar una BD concreta. Puede verse alguna información, usando el comando **SHOW SGA**. Está compuesta por:
  - **Caché de BD**: Con los bloques de BD más recientemente accedidos, para reducir los accesos a disco.
  - **Buffer del Registro de Rehacer (Redo Log Buffer)**, para el fichero de *redo log*.
  - **Memoria compartida**: Para consultas SQL y otros procesos.
- **Program Global Area (PGA)**: *Buffer* de memoria con información sobre los procesos.
- **Procesos de Usuario**: Aplicaciones que ejecuta el usuario.
- **Procesos de Oracle**: Procesos del servidor (para atender a los usuarios...) y procesos de segundo plano (*background*), para tareas de registro, monitorización...

# Instancias de ORACLE (Oracle Instance)





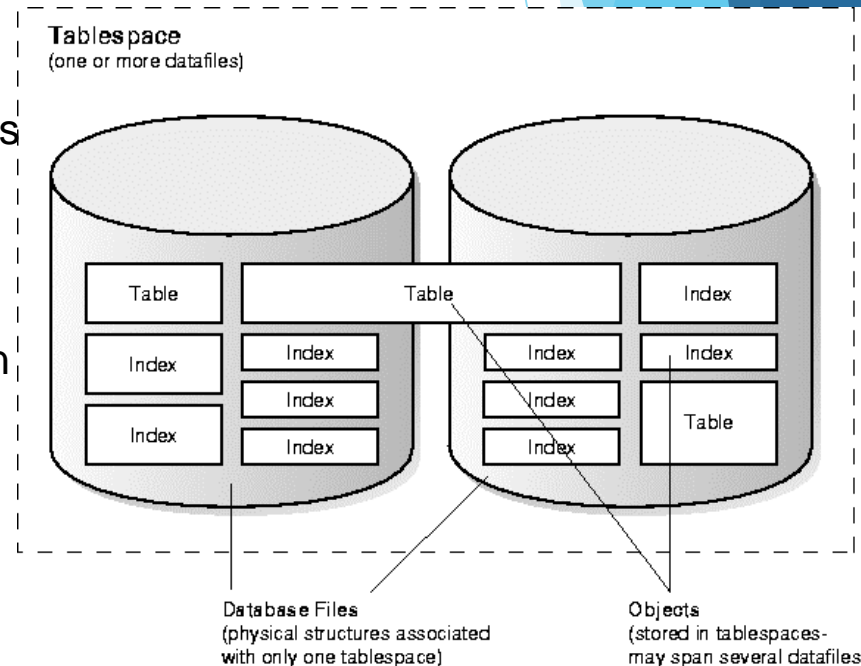
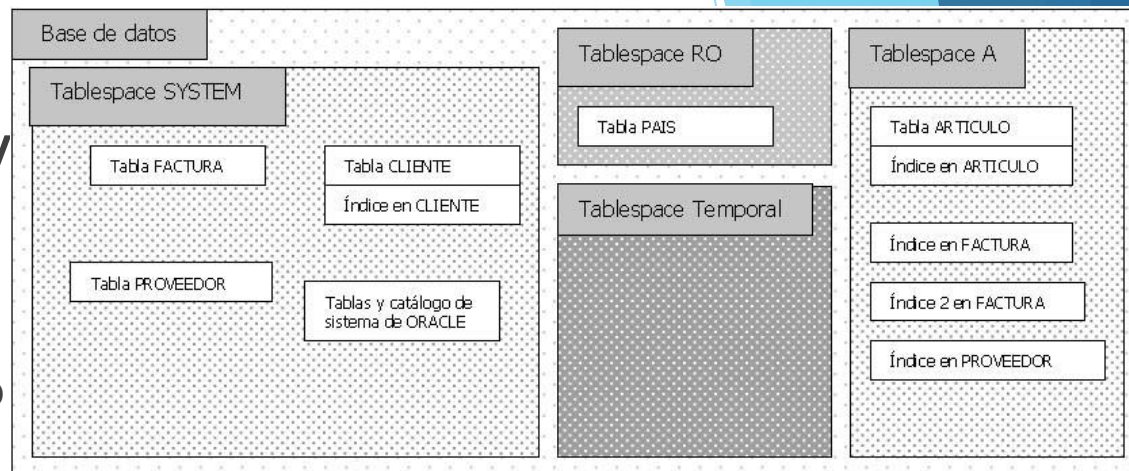
## 2. Tablespaces y Datafiles

# Tablespaces y Datafiles

- ▶ Oracle almacena los datos lógicamente en *tablespaces* y físicamente en *datafiles*.

- ▶ Tablespaces:

- ▶ Una BD está dividida en uno o más *tablespaces*
- ▶ El Administrador los usa para:
  - Controlar la **creación de espacios en disco** para los datos de la BD.
  - Asignar **cuotas específicas** para los usuarios.
  - Controlar la **accesibilidad** de los datos (poniendo un *tablespace* en modo *online/offline* o *read-only/read-write*).
  - Realizar operaciones parciales de **backups/restore**.
  - Repartir los datos en **varios discos** para mejorar el **rendimiento**.



# Tablespaces y Datafiles

- **Tablespaces**

El **Administrador** puede: crear y borrar *tablespaces*, añadir ficheros a los *tablespaces*, añadir o alterar segmentos del *tablespace*, hacer que un *tablespace* sea temporal o permanente...

Podemos **Crear *Tablespaces*** temporales o no con la instrucción SQL:

```
CREATE TABLESPACE tablespace TEMPORARY / PERMANENT...
```

También podemos **Cambiar** el estado de un ***Tablespace*** con:

```
ALTER TABLESPACE tablespace TEMPORARY;
```

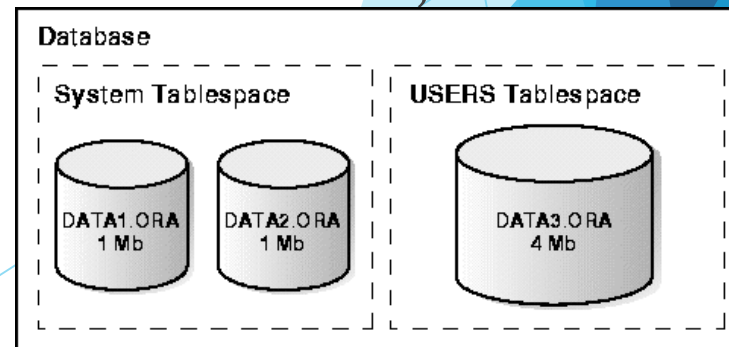
Borrar un *tablespace*:

```
DROP TABLESPACE prueba;
```

# Tablespaces y Datafiles

- ▶ **El *Tablespace* SYSTEM**: Cada BD Oracle contiene un *tablespace* llamado **SYSTEM**, creado automáticamente al crear la BD.
  - ▶ Contiene las **tablas del Diccionario de Datos** para toda la BD.
  - ▶ Una BD pequeña puede necesitar sólo el *tablespace* **SYSTEM**.
    - ▶ Sin embargo, es recomendable crear, al menos, un *tablespace* adicional a fin de separar la información del diccionario de los datos.
      - ▶ Más flexibilidad en las tareas de administración
      - ▶ Reduce los problemas del acceso concurrente al mismo *tablespace*.
  - ▶ Evidentemente, el *tablespace* **SYSTEM** está siempre *online* mientras la base de datos esté abierta.
- ▶ **Tablespaces Read-Only**: Eliminan la necesidad de realizar *backups* y recuperaciones de porciones de la base de datos.
  - ▶ Oracle nunca actualiza los ficheros de un *tablespace read-only* y, por tanto, estos *tablespaces* pueden residir en **dispositivos de sólo lectura**, como CD-ROM.

**ALTER TABLESPACE nombre READ ONLY**



# Tablespaces y Datafiles

## ► Tablespaces Online y Offline:

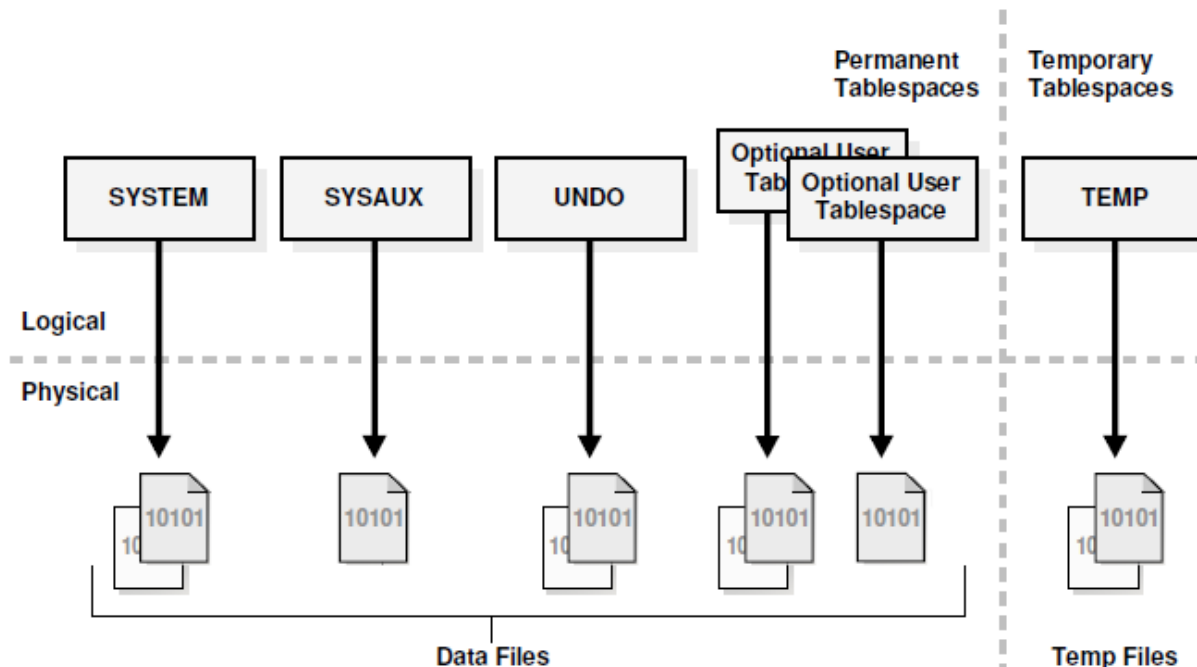
Un Administrador de la BD puede poner a los *tablespaces* accesibles o no, mientras la BD está abierta.

- Normalmente, un *tablespace* está **online** para que los usuarios tengan accesibles los datos.
- Los motivos por los que un Administrador puede poner **offline** a un *tablespace* pueden ser:
  - Para dejar inaccesible una porción de la BD mientras el resto sigue accesible.
  - Para realizar un **backup offline** del *tablespace*.
  - Para hacer que algunas aplicaciones y algunas tablas queden temporalmente inaccesibles al objeto de realizar operaciones de mantenimiento o modificación.
- **No** se pueden poner **offline** *tablespaces* que contengan segmentos de *rollback* que estén en uso, ni el *tablespace* **SYSTEM**.
- Oracle puede poner un *tablespace* **offline** si se ha producido algún **error** importante (como un error de disco...).

# Tablespaces y Datafiles


## ► Tablespaces Temporales:

- Se puede utilizar espacio para la realización de operaciones de ordenación (*sort*) de forma más eficiente creando *tablespaces* temporales para este exclusivo uso.
- Un *tablespace* temporal puede ser utilizado únicamente para contener segmentos *sort*.
- Objetos permanentes no pueden residir en un *tablespace* temporal.
- A partir de Oracle 12, se pueden definir tablespaces temporales locales a cada instancia





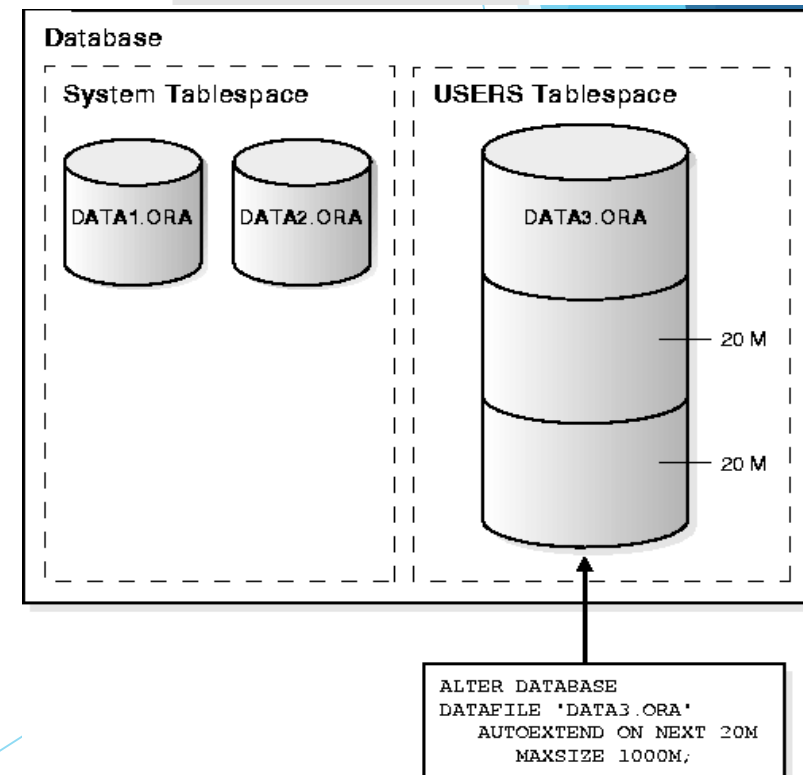
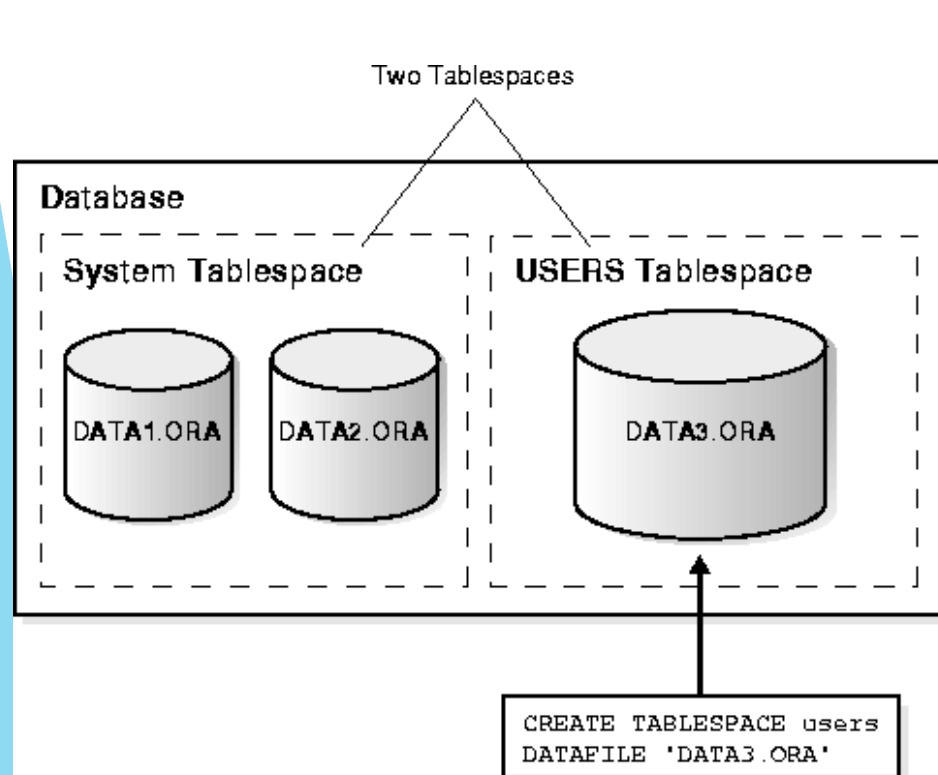
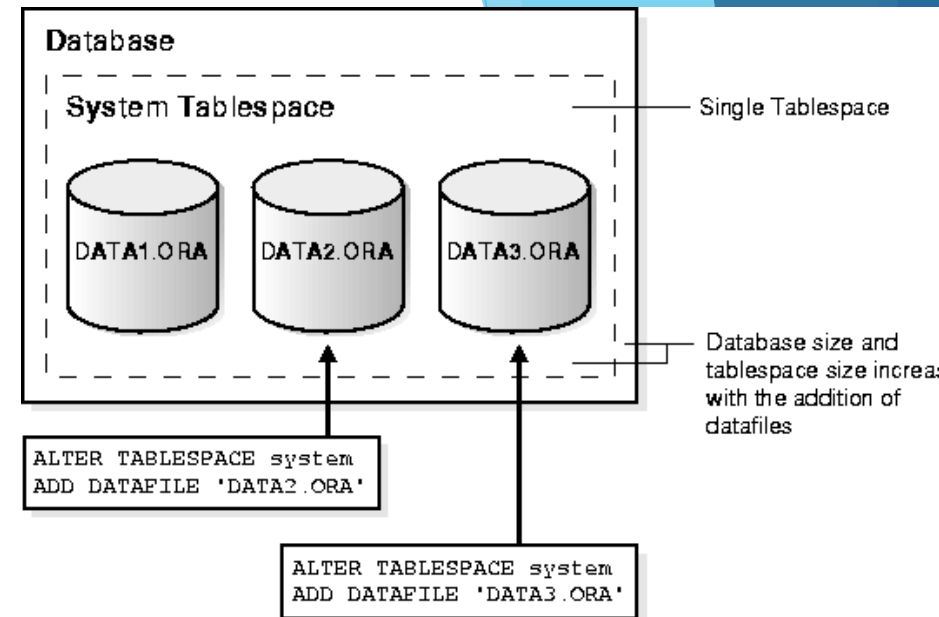
# Tablespaces y Datafiles

- ▶ Un **tablespace** de una BD consiste en uno o más **datafiles**:
  - ▶ Un *datafile* puede estar asociado con un único *tablespace*.
  - ▶ El primer ***tablespace*** en cualquier BD Oracle siempre es el *tablespace SYSTEM* y para éste se construye el primer *datafile* cuando creamos la base de datos.
- ▶ **Contenido del Datafile:**
  - ▶ Cuando un *datafile* se crea, su espacio es formateado para que pueda contener los datos de usuario.
  - ▶ Los datos asociados con los objetos del esquema serán almacenados físicamente en *datafiles*, pero debe tenerse en cuenta que no existe una correspondencia directa entre estos objetos y los *datafiles*.
  -  ▶ Un objeto no se corresponde con un *datafile*, sino con un *tablespace*.
- ▶ **Tamaño de los datafiles:**
  - ▶ Podemos alterar el tamaño de un *datafile* después de haber sido creado.
  - ▶ Esto permite facilitar las tareas de administración de la base de datos.
  - ▶ La instrucción correspondiente está dentro de **ALTER DATABASE**.

# Tablespaces y Datafiles

► Podemos hacer Creacer la BD de tres maneras:

1. Añadiendo un *datafile* al *tablespace*
2. Añadiendo un nuevo *tablespace*
3. Incrementando el tamaño de un *datafile*





# 3. Vistas Dinámicas del Diccionario

# Vistas Dinámicas del Diccionario

## ► Tablas de Ejecución Dinámica:

- Tablas virtuales con información sobre la actividad de la BD durante su funcionamiento.
- No son verdaderas tablas y no son accesibles para la mayoría de los usuarios. Los Administradores de la BD pueden consultar y crear **vistas** sobre esas tablas (llamadas *fixed views*) y autorizar el acceso a esas vistas por parte de otros usuarios.
- El propietario es **SYS** y los nombres de las mismas comienzan por **v\_\$**. Para estas tablas se crean vistas y para ellas se crean sinónimos que comienzan todos por **v\$**.

❑ ¿Por qué se llaman *fixed views*?

# Vistas Dinámicas del Diccionario

## ► Ejemplos:

- **V\$DATABASE** contiene información sobre la BD (nombre...).
- **V\$DATAFILE** contiene información sobre los *datafiles*.
- **V\$FIXED\_TABLE** contiene información sobre todas las tablas de ejecución dinámica y vistas de la BD.
- **V\$PROCESS** contiene información sobre los procesos activos.
- **V\$CONTROLFILE** contiene una lista con los ficheros de control.
- **V\$LOGFILE** ficheros de REDO. **V\$LOG** estado de los ficheros
- **V\$SESSION** contiene información sobre las sesiones actuales (usuario, comando actual, programa y terminal de acceso...).

Para matar una sesión:

```
ALTER SYSTEM KILL SESSION 'sid,serial#';
```

- **Generar las instrucciones para matar las sesiones de un usuario determinado**





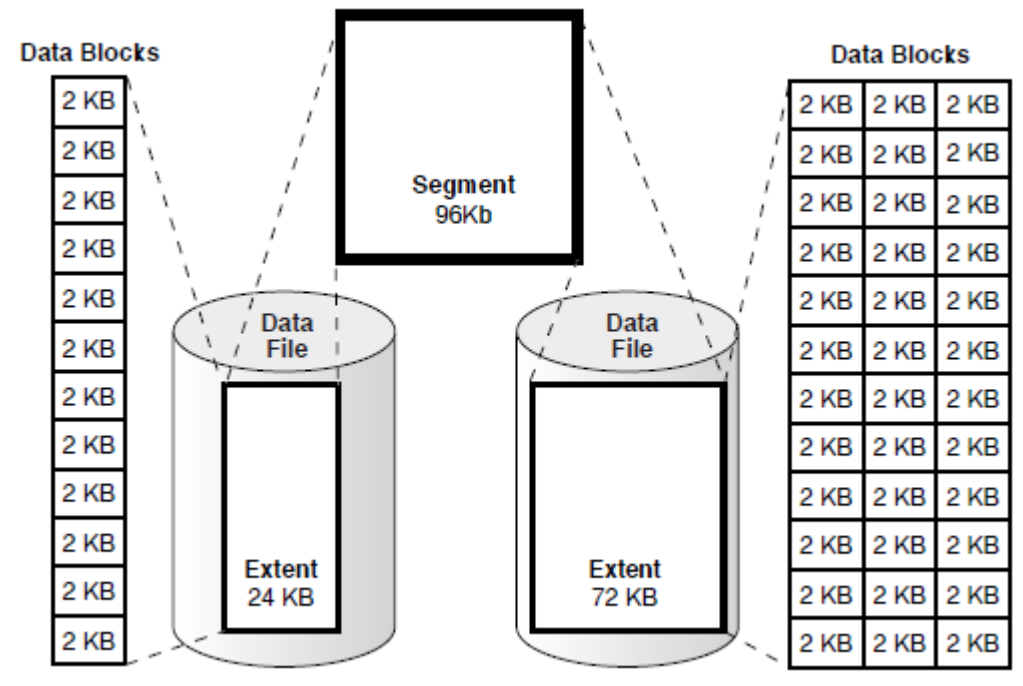
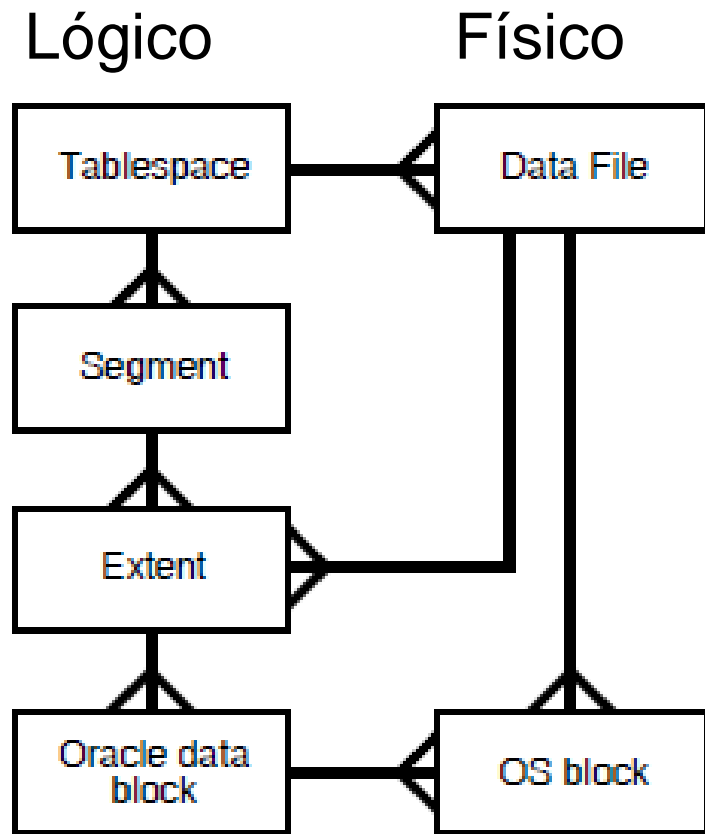
# 4. Gestión del Espacio Lógico

Bloques de Datos, Extensiones y Segmentos

# Gestión del Espacio Lógico

- ▶ El SGBD gestiona las extensiones dentro de un Tablespace (localizar una libre o liberar una que ya no se usa)
  - ▶ Tablespaces Gestionados Localmente (por defecto)
    - ▶ Automático, **Automatic Segment Space Management** (ASSM) → Sólo hay que configurar un parámetro, PCTFREE
    - ▶ Manual (MSSM) → PCTFREE, PCTUSED, FREELISTS, y FREELIST GROUPS
  - ▶ Gestionados por el diccionario. En el propio diccionario se guardan las extensiones libres y utilizadas → Se produce SQL recursivo, lo cual es menos eficiente

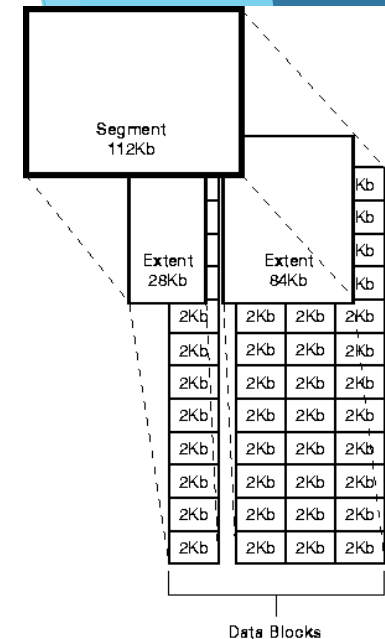
# Almacenamiento Lógico vs Físico





# Bloques de Datos, Extensiones y Segmentos

- ▶ **Bloque de Datos o Página (*data blocks*):** Unidad mínima de asignación de espacio en la Base de Datos.
  - ▶ Es la menor unidad de E/S que puede utilizar la BD (independientemente de que el tamaño de bloque del S.O. sea menor).
- ▶ **Extensión (*extent*):** Conjunto de **data blocks** *contiguos*, con un tipo de información específico.
- ▶ **Segmentos (*segment*):** Conjunto de **extensiones** que almacenan un determinado tipo de datos.
  - ▶ Al crear una estructura de datos, Oracle le asigna un **segmento** con una única **extensión**.
  - ▶ Cuando se llena esa extensión se le asignan otras **extensiones** a ese **segmento**. Por eso, las distintas extensiones no suelen ocupar espacios consecutivos, como sería deseable.
  - ▶ Un **segmento** completo se almacena en un *tablespace*, que puede distribuir sus **extensiones** en distintos ficheros.
  - ▶ Cada **extensión** se almacenará siempre en un único fichero.



# Bloques de Datos

## ► Tamaño de Datablock:

- Especificado por el DBA cuando crea la BD.
- Debe ser múltiplo del tamaño de bloque del S.O.
- No se puede cambiar a no ser que se cree la BD de nuevo
- Se pueden crear Tablespaces con un tamaño de bloque distinto

## ► Formato de un Datablock:

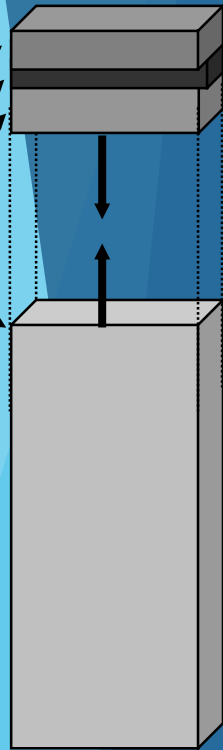
- Overhead: Formado por 3 zonas:

- **Block Header**: Información general, como la dirección en el disco, el tipo de segmento al que pertenece (de datos, de índice, de rollback...).
- **Transaction entries**: También se guarda aquí información sobre las transacciones (**INSERT**, **UPDATE** y **DELETE**) sobre las filas de esta página
- **Table Directory**: Información sobre las tablas con filas en este datablock.
- **Row Directory**: Información sobre las filas almacenadas en este datablock.

- Row Data: Zona con datos de tablas o índices. Una fila puede estar en varias páginas.

- Free Space: Espacio libre para insertar nuevas filas (**INSERT**) o nuevos valores en las filas ya existentes (**UPDATE**), si requieren más espacio.

Common and Variable Header  
Table Directory  
Row Directory  
Free Space  
Row Data

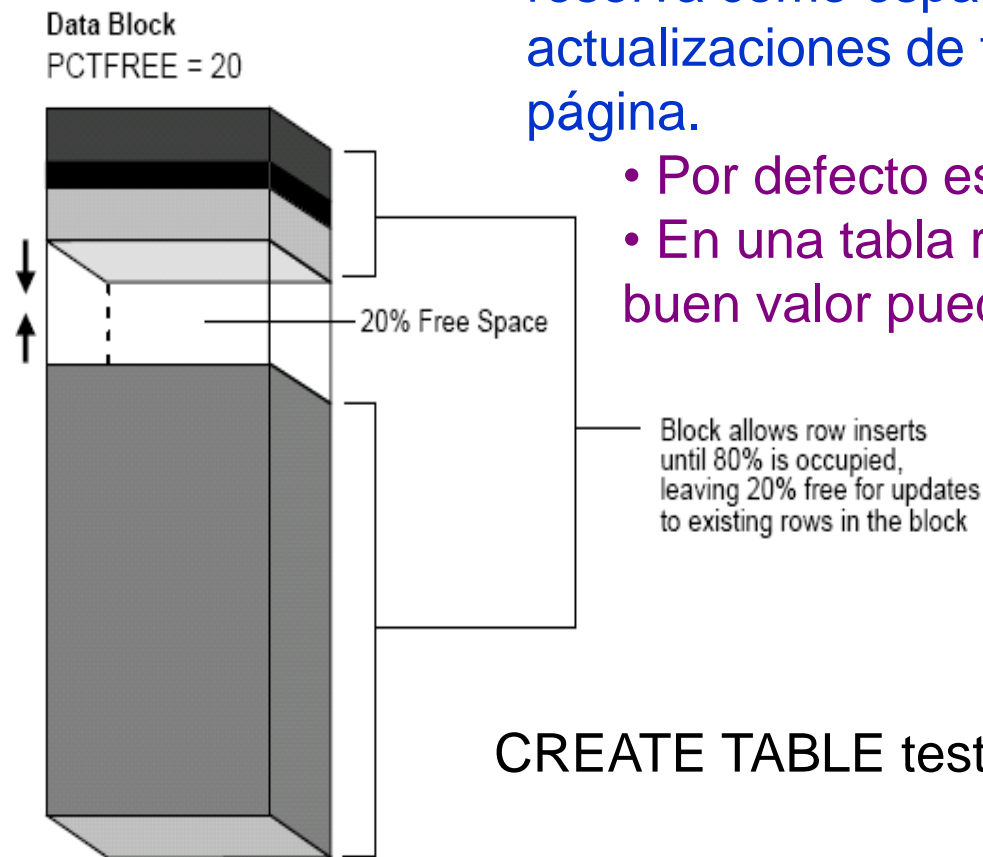


# Bloques de Datos

- Control del Espacio Libre para Inserción y Actualización de Filas: Existen dos parámetros que se especifican cuando se crea o altera una tabla o un índice:

**PCTFREE**: Mínimo porcentaje de página que se reserva como espacio libre para futuras actualizaciones de filas que ya existen en la página.

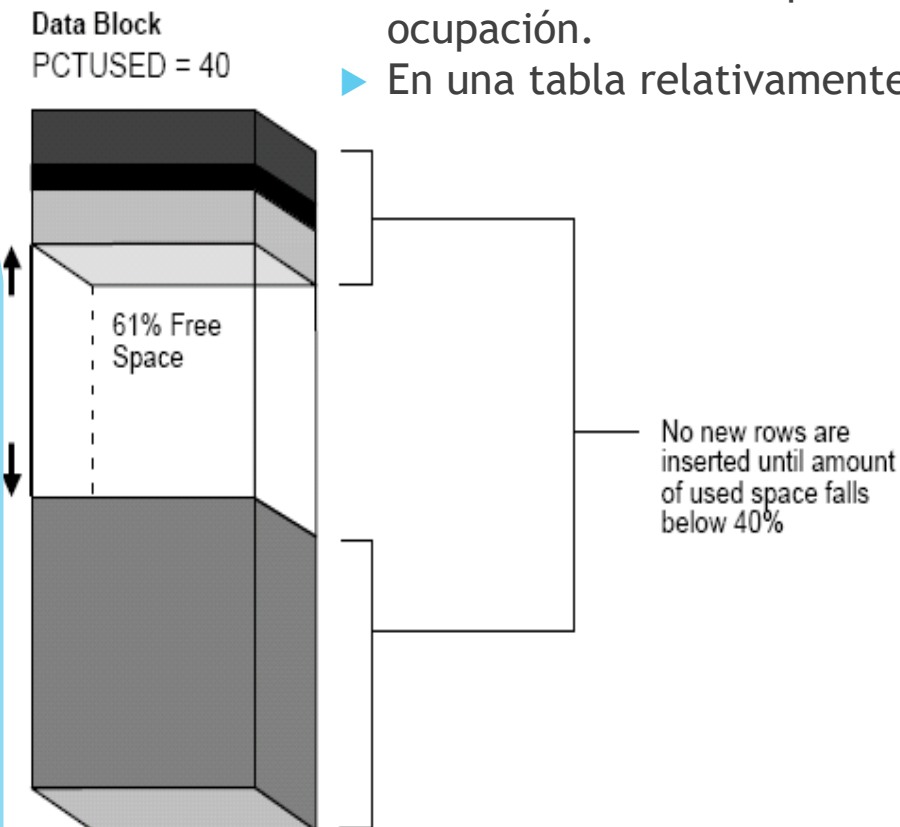
- Por defecto es el 10%.
- En una tabla relativamente estática un buen valor puede ser el 5%.



```
CREATE TABLE test_table (n NUMBER) PCTFREE 20;
```

# Bloques de Datos

- ▶ **PCTUSED** (gestión de Tablespaces manual): Después de que una página sea considerada llena en función del límite especificado en el **PCTFREE**, Oracle no vuelve a introducir ninguna nueva fila en la misma hasta que el porcentaje de página ocupada sea menor que **PCTUSED**. Hasta entonces, el espacio libre sólo será dedicado a la actualización de las filas ya existentes en la página.
  - ▶ Por defecto deja el 40%, que indica que cuando esa página se llena no volverá a estar libre para inserciones hasta que tenga menos del 40% de ocupación.
  - ▶ En una tabla relativamente estática un buen valor puede ser el 75%.

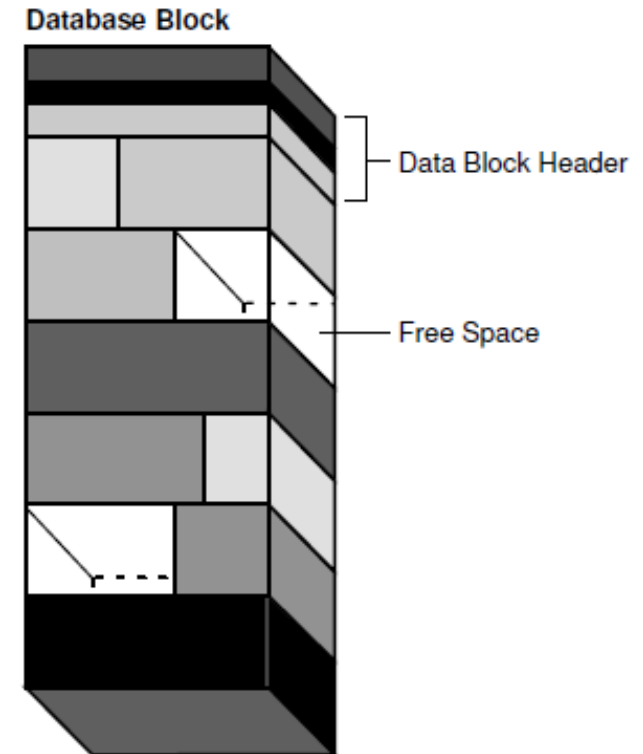


- Espacio libre para inserciones:  
**Tama\_Página - Overhead - PCTFREE**
- Cuando se produce un **INSERT**, Oracle mira la lista de páginas que están disponibles (*free list*) y selecciona la primera que encuentra.
- Para actualizaciones (**UPDATE**) cualquier espacio libre puede ser utilizado.

# Bloques de Datos

## ► Reagrupación del Espacio Libre en un Datablock:

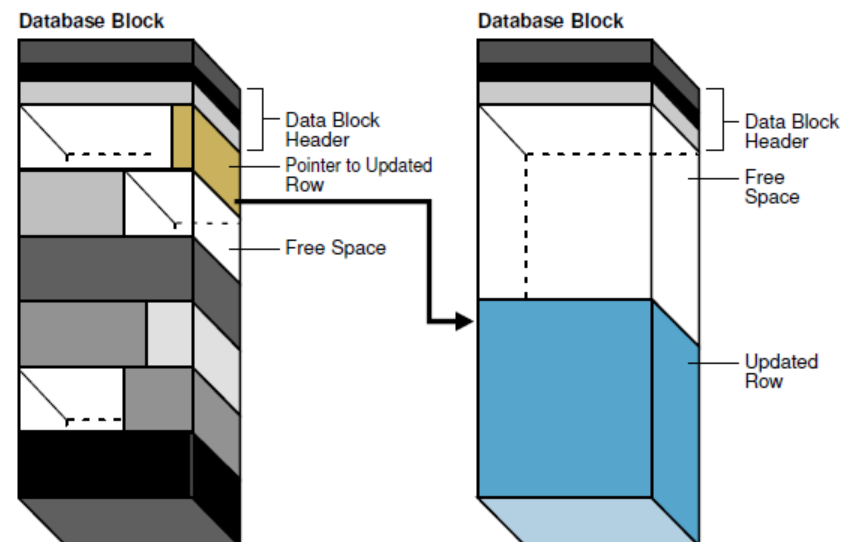
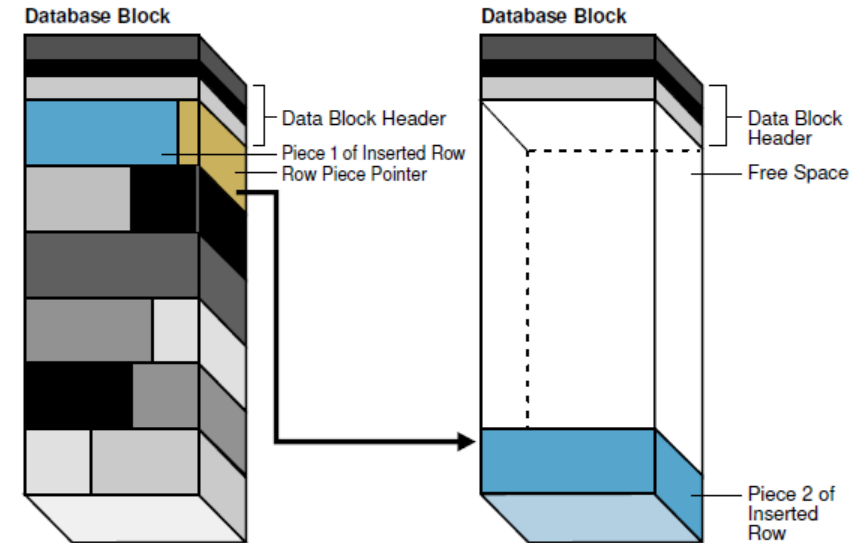
- El **espacio libre** aumenta por las instrucciones **DELETE** o **UPDATE** (si la actualización establece valores que ocupan menos espacio).
- Todo ese **espacio libre** podrá usarlo una instrucción **INSERT**:
  - Si la instrucción **INSERT** está en la misma transacción que la instrucción que ha generado el espacio libre (y situado después, naturalmente).
  - Si la instrucción **INSERT** está en otra transacción y la transacción que deja el espacio libre ya efectuó su **COMMIT**. Posiblemente ambas transacciones sean de distintos usuarios.



# Bloques de Datos

## ► Encadenamiento y Migración de Filas:

- Los datos de una fila pueden ser demasiado grandes para caber en una página:
  - **La fila es muy grande:** Oracle almacena la fila en una cadena de páginas del mismo segmento.
    - Con datos grandes (como el tipo `LONG`) esta fragmentación es inevitable.
  - **Una fila es actualizada y sus nuevos valores no caben en su página actual:** Oracle traslada toda la fila a una nueva página, suponiendo que caben en una nueva página.
    - Oracle conserva la cabecera de la fila en su página inicial, apuntando a la nueva dirección en la nueva página. Así, el identificador de fila (`ROWID`) no cambia.
- Esto hace que la eficiencia al tratar esta fila sea menor, ya que Oracle debe leer más de una página para recuperar la información de esa fila.



# Extensiones

## ▶ Un Segmento es un Conjunto de Extensiones.

- ▶ Si un segmento se llena, Oracle crea una nueva extensión para ese segmento (*incremental extent*) del mismo tamaño o superior.

## ▶ Hay Dos Formas de Gestionar las Extensiones:

- ▶ Extensiones Gestionadas Localmente (LOCAL): Al crear un *tablespace* se pueden especificar las siguientes opciones:
  - ▶ **AUTOALLOCATE:** Son gestionadas por el sistema. Se especifica el tamaño de la extensión inicial y el tamaño del resto es calculado por Oracle, con un mínimo de 64KB.
  - ▶ **UNIFORM:** El tamaño especificado es para todas las extensiones (1MB por defecto).
- ▶ Extensiones Gestionadas por el Diccionario de Datos (DICTIONARY): Utilizan como valores por defecto los valores almacenados en el Diccionario de Datos de la Base de Datos. Esos valores por defecto pueden modificarse en cualquier momento.
  - ▶ Estos valores son **INITIAL** (tamaño del primero), **NEXT** (tamaño del segundo) y **PCTINCREASE** (porcentaje de incremento en el tamaño del siguiente respecto al anterior).

# Extensiones (extents)

- ▶ **Eliminar Extensiones:** En general, las extensiones de un segmento **no** son liberadas (*deallocated*) a no ser que borre el objeto almacenado en el segmento (mediante una instrucción **DROP TABLE** o **DROP CLUSTER**).

- ▶ No obstante se producen algunas excepciones. Por ejemplo, el Administrador puede “desasignar” extensiones no utilizadas mediante la instrucción:

```
ALTER TABLE nombre_de_tabla DEALLOCATE UNUSED;
```

- ▶ También utilizando el paquete DBMS\_SPACE\_ADMIN



# Segmentos

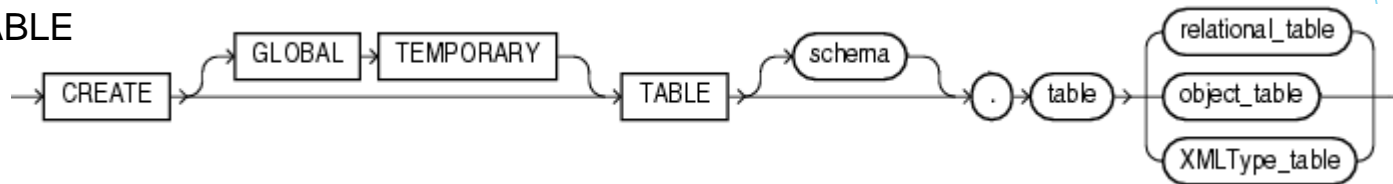
- ▶ Un Segmento es un Conjunto de Extensiones que contienen todos los datos de una estructura lógica específica (una tabla, un índice...) en un *tablespace*.
- ▶ Cuatro Tipos de Segmentos:
  - ▶ Segmentos de Datos (*data segments*)
  - ▶ Segmentos de Índices (*index segments*)
  - ▶ Segmentos Temporales (*temporary segments*).
  - ▶ Segmentos de Rollback (*Undo segments*).

# Segmentos de datos

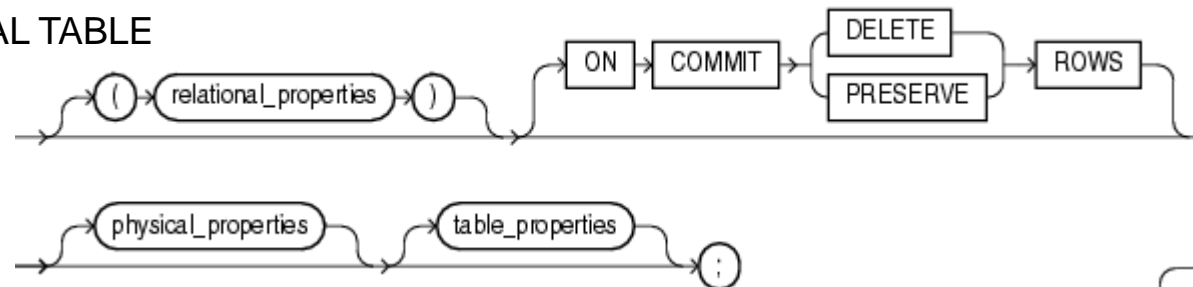
## ► Data segments:

- Se crean con la sentencia **CREATE** (para tablas “*nonclustered*”, *snapshots*, *clusters*...). Por ejemplo, CREATE TABLE ...
- Los parámetros de almacenamiento de datablocks y extensiones se asignan con **CREATE** o **ALTER**, y afectan a la eficiencia en el almacenamiento y en la recuperación de datos.

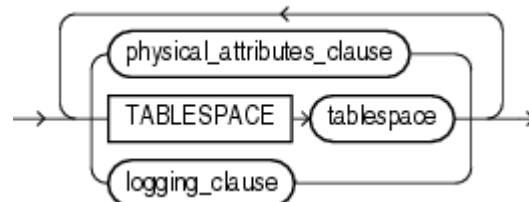
### CREATE TABLE



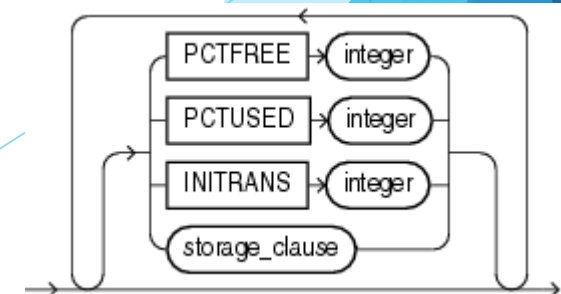
### RELATIONAL TABLE



### PHYSICAL PROPERTIES



### PHYSICAL ATTRIBUTES CLAUSE



# Segmentos de índices

## ▶ *Index segments:*

- ▶ La sentencia **CREATE INDEX** crea un segmento también y pueden fijarse los parámetros de almacenamiento.
- ▶ Una tabla y sus índices pueden tener segmentos en distinto *tablespace*.
- ▶ Esto permite que el índice y los datos se almacenen en distintos ficheros e incluso, en distintos discos, lo cual mejora el rendimiento

# Segmentos Temporales

## ► *Temporary segments:*

- Cuando se procesa una consulta, Oracle requiere espacio temporal para realizar las operaciones intermedias de la instrucción SQL.
- Para ello, automáticamente crea un espacio en disco: **Segmento Temporal**.
- Operaciones que usan segmentos temporales: **CREATE INDEX**, **SELECT...ORDER BY**, **SELECT DISTINCT**, **SELECT...GROUP BY**, **SELECT...UNION**, **SELECT...INTERSECT** y **SELECT...MINUS**.
  - Normalmente solo se requiere este espacio cuando se necesita ordenar un resultado que, además, no cabe en memoria.
- **Oracle** crea los segmentos temporales que necesita durante una sesión de usuario en el ***tablespace* temporal del usuario** que realiza la instrucción.
  - Este *tablespace* es especificado en la instrucción **CREATE USER** o con **ALTER USER** y usando la opción **TEMPORARY TABLESPACE**.
  - Si el Administrador no ha definido estos parámetros, por defecto el *tablespace* temporal será el ***tablespace* TEMP** o el **SYSTEM**.
- Una vez la instrucción se completa, **Oracle borra el segmento temporal** asignado.
- Es razonable Crear un *Tablespace* especial para contener los segmentos temporales debido a que la creación y borrado de segmentos temporales ocurre frecuentemente → distribuir las entradas/salidas por distintos discos mejorando los tiempos de respuesta.
- También se pueden utilizar para las ***tablas temporales***

# Segmentos de Rollback

## ► *Undo segments:*

- Almacenan los viejos valores de los datos modificados por las transacciones → mantener la consistencia, realizar *rollbacks* y permitir la recuperación (*recovery*) de la BD.
  - Cada BD contiene **uno o más segmentos de *rollback*** → **undo tablespace**
  - Un segmento de *undo* contiene numerosos registros o *rollback entries*. Por ejemplo, qué dato ha sido modificado (fichero *filenumber* y *block ID*), así como el contenido de éste (si existía antes de realizar la operación).
  - **Oracle** enlaza cada entrada de una **transacción** de manera que todas ellas son fácilmente localizables en caso de tener que deshacerla.
  - Al grabar las **entradas de *rollback***, cambian los datablocks del segmento de *undo* y Oracle guarda todos los cambios en los datablocks, incluyendo estas entradas de *rollback*, en el **Redo Log** (registro de rehacer).
    - Este segundo almacenamiento de la información de *rollback* es muy importante para las transacciones activas (sin terminar con **COMMIT** o **ROLLBACK**).
    - **Si el sistema cae**, se restaura la información de los segmentos de *undo*, incluyendo las entradas para las transacciones activas.
    - Cuando se realiza un **COMMIT**, Oracle libera la información de *rollback*.

# Ejercicio

- ▶ Piensa 2 casos en los que sea necesario leer los segmentos de Rollback
- ▶ ¿Es lo mismo un segmento de rollback que un segmento de Undo?
- ▶ ¿Es lo mismo un segmento de rollback que el redo log?



# 5. Estructura de la Memoria

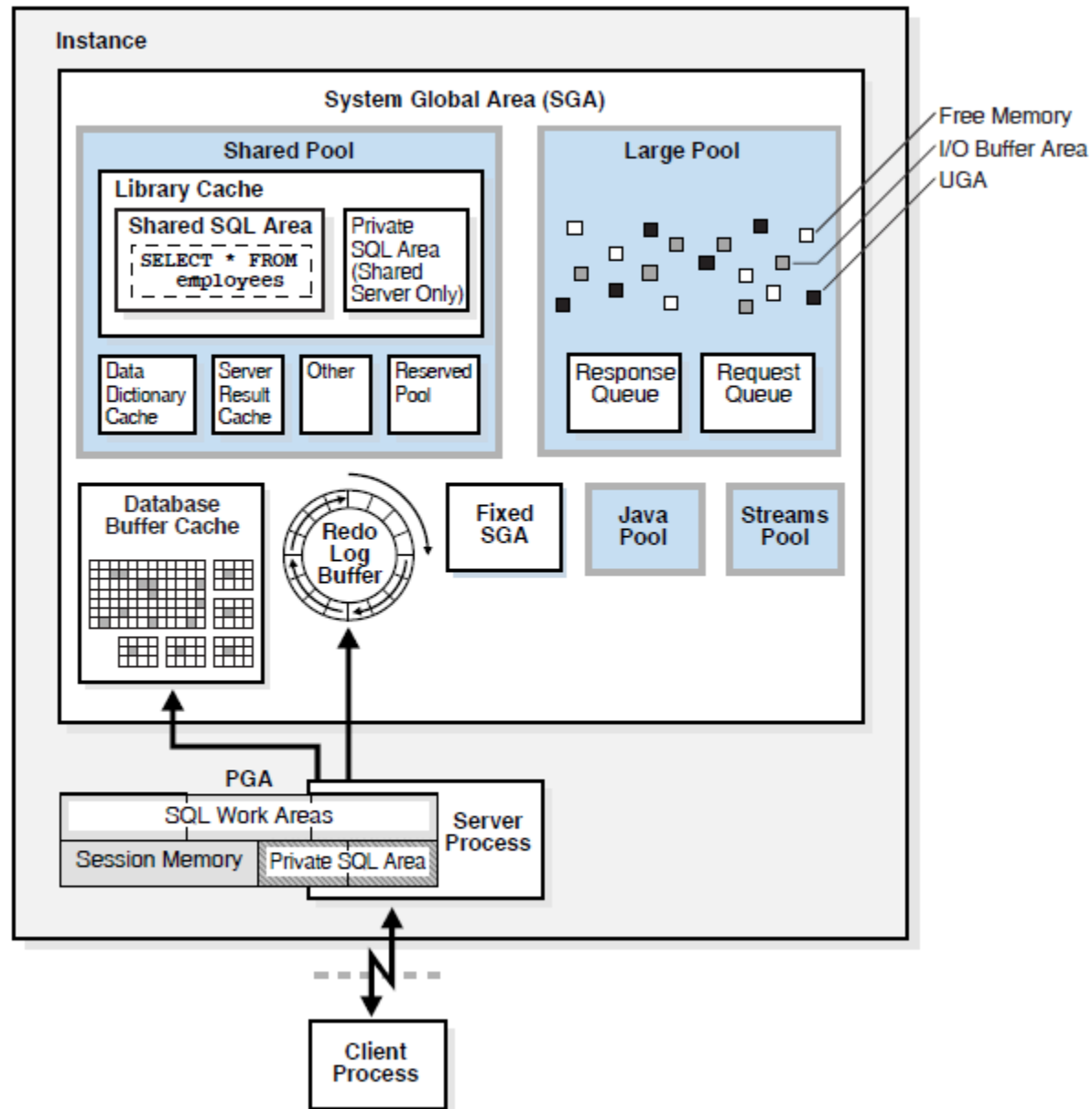
# Estructura de la Memoria en ORACLE

- ▶ Oracle Utiliza la Memoria para Almacenar:
  - ▶ Código del programa que se ejecuta.
  - ▶ Información sobre las sesiones conectadas.
  - ▶ Información necesaria sobre las ejecuciones de los programas (estados de las consultas...).
  - ▶ Información compartida entre procesos (sobre los bloqueos...).
  - ▶ Caché de datos.



# Estructura de la Memoria en ORACLE

- ▶ Estructuras Básicas de Memoria de una instancia de Oracle:
  - ▶ Área Global del Sistema (SGA, System Global Area).
  - ▶ Áreas Globales de Programas (PGA, Program Global Areas).
  - ▶ Área Global de Usuario (UGA).
  - ▶ Áreas de Código de Software (SCA, Software Code Areas).

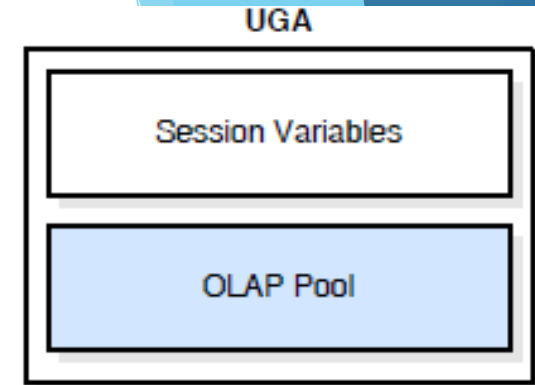


# Gestión de la memoria

- ▶ Automática (por defecto): Se especifica el tamaño objetivo de la instancia (target size). La memoria se distribuye automáticamente entre la SGA y la PGA
- ▶ Compartida. Se especifican los tamaños de la SGA y la PGA se puede gestionar como un todo o individualmente
- ▶ Manual. Se especifican los tamaños de cada parte de la instancia

# User Global Area

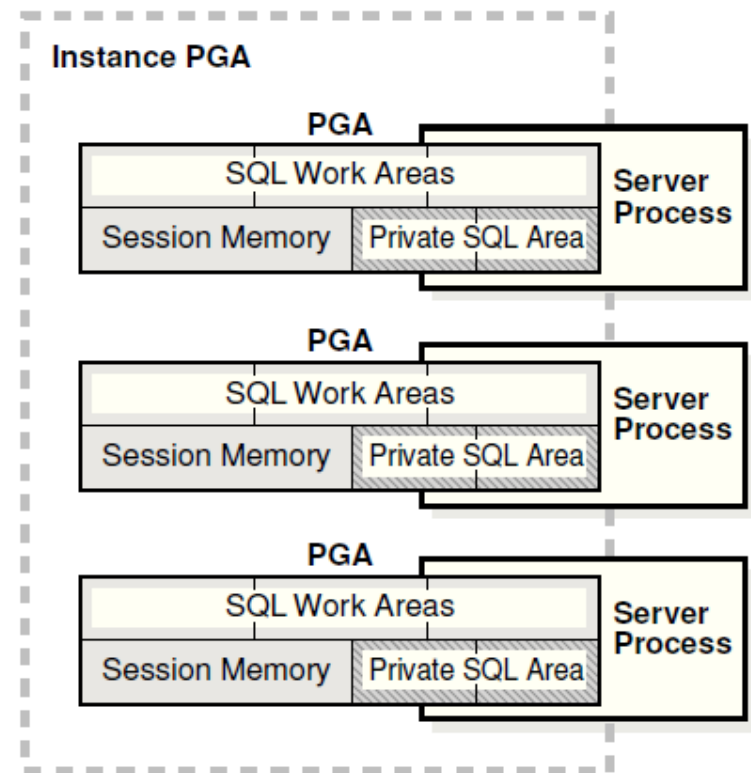
- ▶ Memoria para variables de sesión
  - ▶ Información de logon
  - ▶ Estado de la sesión
  - ▶ Variables de un paquete
- ▶ En OLAP (Data Warehouse) se almacenan las OLAP page pools para gestionar mejor los cubos
- ▶ La UGA debe estar disponible mientras la sesión esté abierta
- ▶ Puede residir en la SGA (servidor dedicado) o en la PGA (servidor compartido)



# PGA (Program Global Areas)

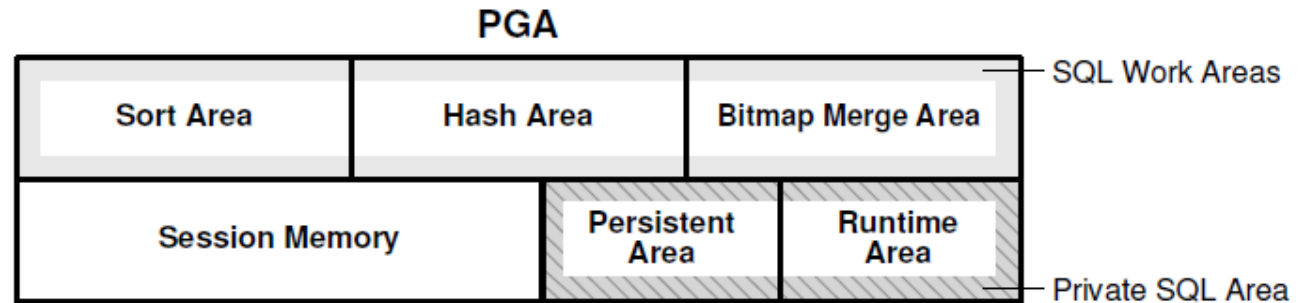
## ▶ Áreas Globales de Programa (PGA).

- ▶ Un **PGA** es una región de memoria que contiene datos e información del control para un solo proceso (de servidor o de *background*).
- ▶ Un **PGA** es un área de memoria no compartida en la que puede escribir un proceso.
- ▶ Analogía: Trozo del mostrador usado por el dependiente para un cliente. El dependiente es el proceso que trabaja para el cliente



# PGA (Program Global Areas)

- ▶ Para cada proceso del servidor se asigna un **PGA**.
  - ▶ Ese **PGA** es exclusivo para ese proceso
  - ▶ Ese **PGA** se lee y se escribe por Oracle, que actúa en nombre de ese proceso.
  - ▶ Un **PGA**, pues, es asignado cuando un usuario se conecta a una base de datos y crea una sesión.
  - ▶ Un **PGA** siempre contiene un espacio que contiene las variables de la sesión, y alguna otra información. Este espacio es denominado *stack space*.
  - ▶ El tamaño de un **PGA** depende del sistema operativo específico.



# PGA (Program Global Areas)

## ► Ejemplo:

```
SELECT * FROM employees e JOIN departments d
ON e.department_id=d.department_id
ORDER BY last_name;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		106	9328	7 (29)	00:00:01
1	SORT ORDER BY		106	9328	7 (29)	00:00:01
*2	HASH JOIN		106	9328	6 (17)	00:00:01
3	TABLE ACCESS FULL	DEPARTMENTS	27	540	2 (0)	00:00:01
4	TABLE ACCESS FULL	EMPLOYEES	107	7276	3 (0)	00:00:01

- Run-time → Número de filas devueltas.
- Hash area → join de las 2 tablas
- Sort area → ORDER BY

# SGA

- ▶ **Área Global del Sistema (SGA, System Global Area).**
  - ▶ Un **SGA** es un grupo de estructuras de memoria compartida que contienen datos e información de control de una Instancia de una BD.
  - ▶ Si a una Instancia están conectados múltiples usuarios concurrentes, los datos de éstos se comparten en el **SGA**.
  - ▶ El **SGA** se crea cuando se arranca una Instancia de una BD y se destruye cuando se cierra la Instancia.
  - ▶ El **SGA** es de lectura y escritura, y contiene las siguientes estructuras de datos:
    - ▶ Caché de los Buffers de la BD (*Database Buffer Cache*).
    - ▶ Registro de Rehacer (*Redo Log Buffer*).
    - ▶ El “Pool” Compartido (*Shared Pool*).
    - ▶ Pool Grande
    - ▶ Pool de Java
    - ▶ Streams Pool
    - ▶ SGA fijo (*fixed SGA*): Parte del **SGA** con información general sobre el estado de la BD y la Instancia, a la que los procesos de “background” necesitan acceder

# SGA

- ▶ **Caché de los *Buffers*** de la BD (*Database Buffer Cache*).
  - ▶ Es la parte del **SGA** que contiene copias de las **páginas leídas** de los *datafiles*. Todos los procesos de usuarios que están conectados a la Instancia comparten el acceso a esta caché.
  - ▶ Cuando se hace COMMIT se escriben los buffers del redo pero no los datablocks: escritura perezosa en background
  - ▶ Los ***buffers*** en la caché están organizados en dos listas:
    - ▶ **Lista “en espera” (*write list*)**: Contiene *buffers* en espera (*dirty buffers*) en el sentido de que contienen datos que han sido modificados pero que aún no han sido escritos a disco.
    - ▶ **Lista LRU** (menos recientemente usada, **Least Recently Used**): La lista LRU contiene *buffers* libres, *buffers* que están siendo accedidos actualmente (*pinned buffers*) y, por último, los *buffers* “en espera” que aún no han sido movidos a la lista “en espera”.
- ▶ La primera vez que un proceso requiere un dato, **se busca en la caché**:
  - ▶ Si se encuentra el dato en uno de estos *buffers*, se lee directamente de la memoria (*cache hit*), acelerando el proceso de lectura.
  - ▶ Si no se encuentra (*cache miss*), entonces debe obtenerse una copia de la página en disco y pasarla a un buffer de la caché antes de proceder a leerlo.



# Estructura de la Memoria: SGA

- ▶ Registro de Rehacer (*Redo Log Buffer*):
  - ▶ Es un **buffer circular** en el **SGA** que contiene información sobre los cambios que se producen en la BD.
  - ▶ Las entradas (*redo entries*) contienen información necesaria para reconstruir o **rehacer los cambios** efectuados a la BD.
  - ▶ Estas entradas son copiadas por el servidor de procesos de Oracle desde el espacio de memoria del usuario hasta estos **buffers del SGA**, donde los registros van escribiéndose de forma secuencial.
    - ▶ Ahí esperan a que el proceso de *background LGWR* (*Log Writer*) escriba el buffer al correspondiente fichero activo (*online*) de “*Redo Log*” en disco.
  - ▶ Si se produce un fallo en el sistema, los datos que están en la ‘write list’ se perderían. Pero todos los cambios se van guardando en los Ficheros del Registro de Rehacer (redo log). Cuando el sistema arranca de nuevo, se rehacen los cambios no grabados (database recovery).
  - ▶ Modificar el archivo sobre el que se vuelcan los cambios:
    - ▶ `ALTER SYSTEM SWITCH LOGFILE`
  - ▶ Ver el estado:
    - ▶ `SELECT * FROM V$LOG`

# Estructura de la Memoria: SGA

## ► El “Pool” Compartido (*Shared Pool*):

Su tamaño total está determinado por el parámetro de inicialización

**SHARED\_POOL\_SIZE**. Contiene:

### ► Caché de Biblioteca (*Library Cache*):

Incluye la siguiente información:

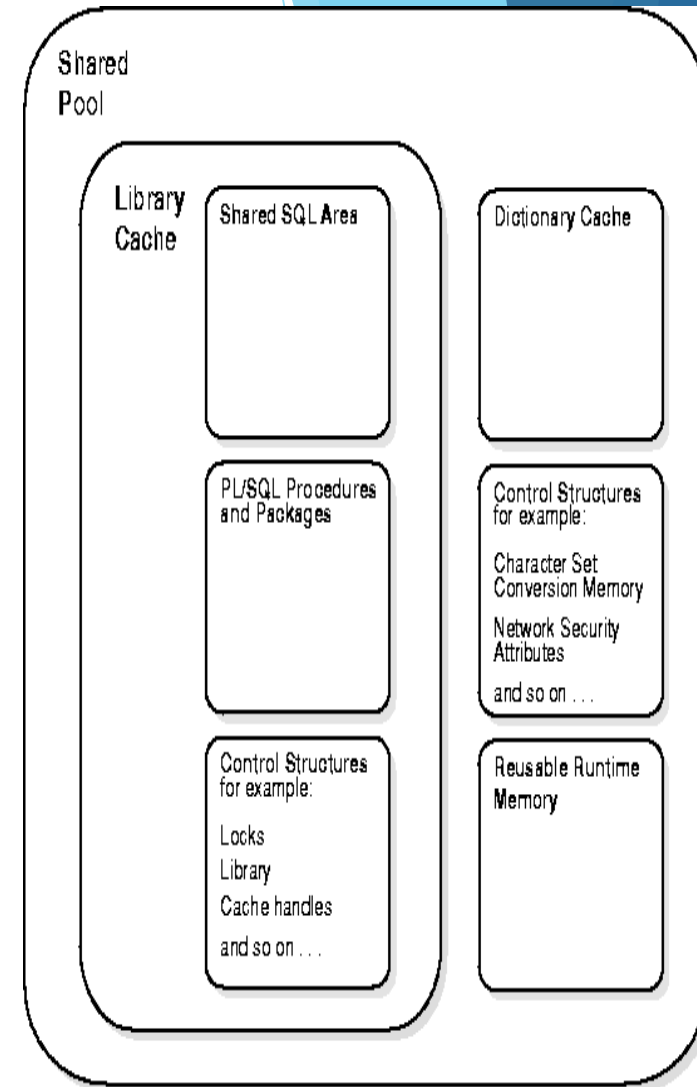
#### ► Áreas compartidas y privadas de SQL.

- Cada instrucción de SQL ejecutada tiene un área de SQL compartida y un área privada.
- Cuando dos usuarios están ejecutando la misma declaración de SQL se reutiliza el área de SQL compartida para esos usuarios, ahorrándose tiempo y memoria (muchos usuarios, misma aplicación):
- Sin embargo, cada usuario debe tener una copia separada de la instrucción en el área de SQL privada.

#### ► Procedimientos y paquetes PL/SQL.

- Evitan tener varias copias de cada programa PL/SQL, aunque una parte es privada para cada usuario.
- Las sentencias SQL contenidas dentro de un programa utilizan también las áreas SQL compartidas y privadas.

#### ► Estructuras de control (caché de gestión de bloqueos, de librerías...).



# Estructura de la Memoria: SGA

## ▶ El “Pool” Compartido (*Shared Pool*):

- ▶ Caché del Diccionario (*Dictionary Cache*): La continua necesidad de acceso al diccionario hace que Oracle disponga de una caché específica para el diccionario.
- ▶ Estructuras de Control (*Control Structures*): Juegos de caracteres, sistemas de conversión, atributos de seguridad...
- ▶ ALTER SYSTEM SET SHARED\_POOL\_SIZE = 64M;

## ▶ El “Pool” Grande (*LARGE\_POOL\_SIZE*).

- ▶ Estructura de memoria opcional.
- ▶ Utilidades de backup y recuperación como RMAN.

## ▶ El “Pool” de JAVA (*JAVA\_POOL\_SIZE*).

- ▶ Guarda información acerca de la interpretación de las sentencias java que se ejecutan.
- ▶ Sólo necesario si se instala y usa Java.

# Estructura de la Memoria: SGA

## ► Gestión Automática de la Memoria Compartida

- **MEMORY\_TARGET** (dinámico) y **MEMORY\_MAX\_TARGET** (estático):
  - Si se fija el **MEMORY\_TARGET**, Oracle modifica dinámicamente el tamaño de los subcomponentes según sea necesario
  - Si **MEMORY\_TARGET** tiene un valor mayor que **MEMORY\_MAX\_TARGET** en el startup, entonces **MEMORY\_MAX\_TARGET** se aumenta.
  - Después del startup, **MEMORY\_TARGET** puede ser modificado dinámicamente, pero no puede exceder del valor de **MEMORY\_MAX\_TARGET** calculado en el startup

## ► Gestión manual o compartida. Parámetro de Inicialización **SGA\_MAX\_SIZE**:

- El tamaño de la instancia puede crecer hasta el valor marcado por **SGA\_MAX\_SIZE**. Pero si los valores iniciales de sus componentes superan **SGA\_MAX\_SIZE**, Oracle ignora este último.
  - **DB\_CACHE\_SIZE** Tamaño de la caché de bloques estándar.
  - **LOG\_BUFFER** Número de bytes asignados para el “redo log buffer”.
  - **SHARED\_POOL\_SIZE** Tamaño en bytes del área dedicada a sentencias compartidas de SQL y PL/SQL
  - **LARGE\_POOL\_SIZE** Tamaño del large pool; por defecto, 0.
  - **JAVA\_POOL\_SIZE** Tamaño del Java pool.

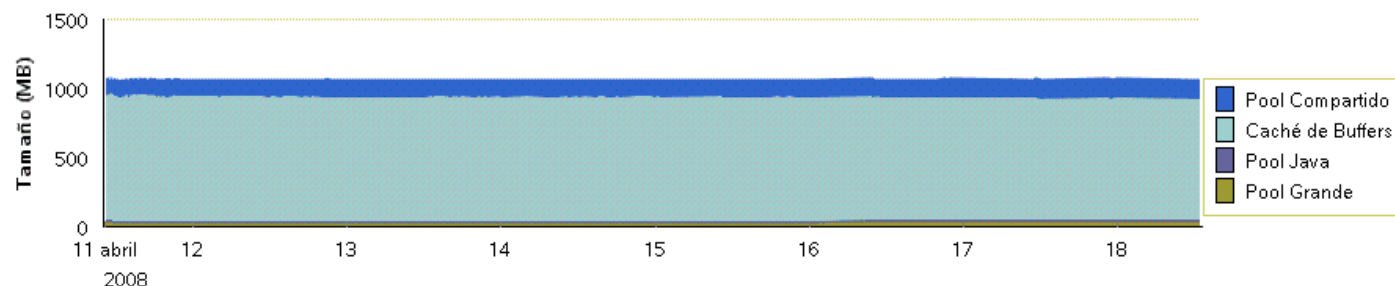
# Estructura de la Memoria: SGA

SGA | PGA

El Área Global del Sistema (SGA) es un grupo de estructuras de memoria compartida que contiene datos e información de control para una base de datos Oracle. SGA está en la memoria cuando se inicia la instancia de base de datos Oracle.

## Historial de Asignaciones

En este gráfico se muestra el historial de los componentes de SGA.



## Asignación Actual

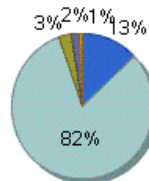
Gestión Automática de Memoria Compartida **Activado**

Desactivar

Tamaño de SGA Total (MB) 1024

Consejo

Componente de SGA	Asignación Actual (MB)
Pool Compartido	132
Caché de Buffers	836
Pool Grande	32
Pool Java	16
Otros	8



Pool Compartido (12,9%)  
Caché de Buffers (81,6%)  
Pool Grande (3,1%)  
Pool Java (1,6%)  
Otros (0,8%)

## Tamaño Máximo de SGA

El tamaño máximo de SGA especifica la memoria máxima que puede asignar la base de datos. Si especifica el tamaño máximo de SGA, más tarde puede cambiar de forma dinámica el tamaño de SGA anterior (siempre que el tamaño de SGA total no exceda el tamaño máximo de SGA).

Tamaño Máximo de SGA\* (MB) 1024

# Área de Código de *Software* (SCA):

- ▶ Son zonas de memoria destinadas a **almacenar el código de Oracle** en ejecución o que puede ejecutarse. Este código de Oracle se almacena en una zona distinta, y más protegida, que las zonas dedicadas a almacenar los códigos de programas de usuarios.
- ▶ Son **áreas de sólo lectura** de tamaño estático que solamente cambian cuando el *software* se instala y su tamaño depende del S.O.
- ▶ **Pueden ser compartidas o no compartidas:** Las primeras son más eficientes porque el mismo código puede ser usado por distintos usuarios.



## 6. Estructura de los Procesos

# Estructura de los PROCESOS

- ▶ **Todo usuario** que se conecta a Oracle debe **ejecutar dos módulos** de código para acceder a la instancia de la BD Oracle, que se ejecutan como procesos normales bajo el control del S.O.:
  - ▶ **Aplicación o Herramienta Oracle**: Se trata de una aplicación normal que se conecte a Oracle, o bien una herramienta de Oracle (como Oracle Enterprise Manager o SQL\*Plus).
    - ▶ Estos programas permiten ejecutar sentencias SQL.
  - ▶ **Código del Servidor de Oracle**: Es una parte de Oracle que se ejecuta para el usuario y que interpreta y procesa las órdenes SQL.
- ▶ **Sistemas Oracle Multiproceso**: Distintos procesos ejecutan distintas partes de Oracle, además de los procesos de los usuarios.
  - ▶ La mayoría de los SGBD son **multiusuario**, ya que es una de sus principales ventajas.
  - ▶ Dividiendo el trabajo en **distintos procesos** se consigue un mejor rendimiento dando servicio a múltiples usuarios y aplicaciones.
- ▶ **Tipos de Procesos**:
  - ▶ **Procesos de Usuario** (*User Processes*).
  - ▶ **Procesos de Oracle** (Oracle Processes):
    1. Proc. de Servidor (*server*)
    2. Proc. en 2º Plano, o de Fondo (*background*)



# Estructura de los PROCESOS

- ▶ **Procesos de Usuario**: Cuando el usuario ejecuta una aplicación, o una herramienta de Oracle, se crea un proceso de usuario.
  - ▶ **Conexión**: Es una vía de comunicación entre un proceso de usuario y una Instancia. Establece el mecanismo de comunicación.
  - ▶ **Sesión**: Es una conexión específica de un usuario a una Instancia a través de un proceso de usuario.
    - ▶ Por ejemplo, un usuario establece una **conexión** usando SQL\*Plus, después introduce su *username* y *password* y, posteriormente, inicia una **sesión** (a través de la ejecución de un proceso de usuario).
    - ▶ Un mismo usuario puede tener varias sesiones.



# Estructura de los PROCESOS

- ▶ Procesos de Oracle: Pueden ser de 2 tipos:
  - ▶ 1. Procesos de Servidor: Se crean para cada una de las aplicaciones de usuario
  - ▶ 2. Procesos de *Background*: Para maximizar el rendimiento y posibilitar el acceso simultáneo de múltiples usuarios
    - ▶ Algunos de estos procesos son creados automáticamente cuando se inicia una **Instancia**. No todos ellos están siempre presentes.

# Estructura de los PROCESOS

## ► Procesos de Background:

- **Database Writer (DBW0 o DBWn):** Escribe el contenido de los *buffers* que han sido modificados a los *datafiles*.
- **Log Writer (LGWR):** Es el responsable del funcionamiento de los *buffers* del *Redo Log* mediante la escritura de su contenido al fichero de *Redo Log*.
- **Checkpoint (CKPT):** Cuando se realiza un *checkpoint*, Oracle actualiza las cabeceras de todos los *datafiles* que almacenan datos a causa de este *checkpoint*.
- **System Monitor (SMON):** Este proceso tiene dos funciones:
  - Maneja la recuperación de la BD a partir del fallo de una Instancia (esto es, cuando las estructuras de memoria y los procesos que componen la Instancia no pueden continuar por algún motivo).
  - Chequea periódicamente los espacios de disco para determinar si une pequeños fragmentos de espacios libres.

# Estructura de los PROCESOS

- ▶ **Process Monitor (PMON):** Actúa cuando falla un proceso de usuario, liberando los recursos que tuviera asignados (memoria...) y deshaciendo los cambios que hubiese realizado desde su último **COMMIT**.
- ▶ **Recoverer (RECO):** Es un proceso de *background* opcional para las BD distribuidas que gestiona las transacciones distribuidas.
- ▶ **Archiver (ARCH):** Es opcional. Copia los ficheros del *Redo Log* a un dispositivo predeterminado cuando éste está lleno. Este proceso solamente se presenta cuando el *Redo Log* se usa en modo **ARCHIVELOG** y el archivo automático está activado.
- ▶ **Lock (LCKn):** Opcional. Solamente para servidores paralelos. Gestiona los bloqueos entre distintas Instancias.
- ▶ **Job Queue (SNPn):** Opcional, para BD distribuidas.
- ▶ **Queue Monitor (QMn):** Opcional. Monitoriza el orden de salida de los mensajes.
- ▶ **Dispatcher (Dnnn):** Opcional. Permite a los procesos de usuario compartir procesos del servidor, de manera que el servidor pueda soportar un mayor número de usuarios.
- ▶ **Shared Server (Snnn):** Cada uno de estos procesos sirve las peticiones de múltiples clientes en configuraciones como la anterior, compartiendo los procesos del servidor.

# Estructura de los PROCESOS

## ► Listener:

- El listener es un proceso que escucha peticiones de conexión de cliente.
- Se configura en el fichero `listener.ora`, con una dirección de protocolo que identifica la base de datos
- Por ejemplo:

```
(DESCRIPTION=
```

```
(ADDRESS= (PROTOCOL=tcp) (HOST=my-server) (PORT=1521)))
```

- Además, el listener conecta al usuario a los dispatchers o servidores dedicados (el listener es parte de Oracle Net Services, no de Oracle).
- Cuando una instancia arranca, el listener establece una ruta de comunicación con Oracle.
- También puede establecer una comunicación entre bases de datos
- En bases de datos distribuidas sirve para definir servicios y las instancias que sirven esos servicios
- Arrancar | parar el listener:
  - `Lsnrctl start | stop` ← Desde el S.O.
  - En Windows también se puede hacer arrancando/parando el servicio



# 7. Objetos del Esquema. Tablas, Índices, Clusters

# TABLAS

- ▶ Unidad básica de organización de datos
- ▶ Organización:
  - ▶ Heap. Las filas no se guardan en ningún orden
  - ▶ IOT (Index-Organized Table). Las filas se guardan en el orden de la clave primaria
  - ▶ Tablas Externas. Sus metadatos se guardan en la base de datos, pero sus datos, no.
- ▶ Los datos pueden ser permanentes o temporales
- ▶ Una tabla puede tener columnas virtuales:

```
create table productos (  
  codigo number primary key,  
  descripcion varchar2(50),  
  precio_netto number (9,2),  
  tipo_iva number(2),  
  total as (precio_netto+ precio_netto * tipo_iva /100))
```

- ▶ Una tabla puede estar partida → PARTITION en función de diversos criterios. Cada partición puede ir a un tablespace distinto

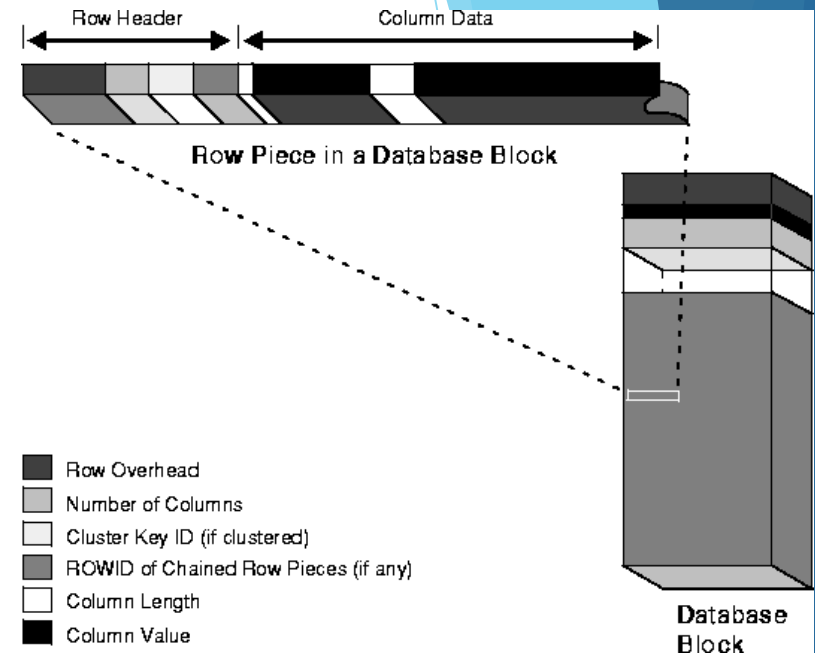
# TABLAS

Cuando se **crea**, Oracle asigna un segmento de datos de un *tablespace* para contener los futuros datos de la tabla. Podemos controlar la generación de este espacio a partir de los parámetros correspondientes (**PCTFREE**, **PCTUSED...**).

**Oracle** almacena habitualmente cada **fila** de una tabla en una única página (**row piece**):

**Una fila** se fragmenta en varias páginas (varios **row pieces**) si la fila no puede ser almacenada completamente en la página: Se usan varias páginas, encadenándolas.

**Cada fila** de una tabla se almacena en dos partes: una que contiene información de cabecera (**row header**) y otra con el contenido de los datos (**column data**).



- ▶ **Row header:** En una fila de un único bloque ocupa 3 bytes como mínimo.
- ▶ **Column data:** Almacena la longitud de cada columna y sus datos.
  - ▶ El valor **NULL** se representa almacenando cero en la longitud y nada en los datos.
- ▶ Borrar una columna es costoso, pero si se usa la opción **SET UNUSED** de **ALTER TABLE**, sólo marca la columna como borrada posponiendo el borrado.



# TABLAS TEMPORALES

- ▶ Sus **datos** duran mientras dure la **transacción o sesión** actuales
- ▶ Los **metadatos** son permanentes (su definición no se borra). Por tanto, sí se pueden crear vistas, triggers, índices sobre las tablas temporales
- ▶ Sentencia de creación:

```
CREATE GLOBAL TEMPORARY TABLE <tabla>  
ON COMMIT [DELETE|PRESERVE] ROWS;
```

- ▶ Debe especificarse si son datos específicos de la transacción o de la sesión:
  - ▶ **ON COMMIT DELETE ROWS:** Borra el contenido de la tabla al efectuar un **COMMIT**, es decir, los datos son específicos de la transacción.
  - ▶ **ON COMMIT PRESERVE ROWS:** NO borra el contenido de la tabla al hacer **COMMIT**, sino que los borra al finalizar la sesión.
- ▶ Son datos privados de cada sesión: No pueden compartirse.

# TABLAS EXTERNAS

- ▶ Accede a datos en fuentes externas como si los datos estuvieran en una tabla de la base de datos. Se puede utilizar SQL, PL/SQL y Java para consultar los datos externos.
- ▶ Se puede crear una tabla externa, copiar el archivo en la ubicación especificada en la definición de la tabla externa, y utilizar SQL para consultar los registros en el archivo de texto
- ▶ Se usan mucho para Data Warehouse en operaciones ETL (Extract, Transform, Load)
- ▶ Se usan sobre todo para leer datos, pero con los **drivers** adecuados, también se puede escribir

```
CREATE TABLE (...) ORGANIZATION EXTERNAL (datos del  
fichero)
```

# TABLAS ORGANIZADAS POR ÍNDICE

- ▶ **IOT** (*index-organized tables*): Son tablas en las que los datos están contenidos en el índice asociado (*B-tree*).
  - ▶ Con la sentencia **CREATE TABLE** y su cláusula **ORGANIZATION INDEX**.
  - ▶ Las **modificaciones** en los datos de la tabla (insertar, actualizar o borrar) tienen como efecto una actualización del índice.
  - ▶ Realmente, **el índice es la tabla**: En vez de estar compuesto de valor de clave y puntero (**ROWID**), se compone de **valor de clave y resto de valores de la fila**.
    - ▶ No duplica el almacenamiento de las claves como en una tabla ordinaria con su índice.
  - ▶ Son idóneas para **accesos por clave primaria** pero no recomendadas para otro tipo de accesos.
  - ▶ Pueden crearse **índices adicionales** sobre este tipo de tablas para acceder eficientemente por otras columnas.
  - ▶ **Diferencias principales** entre estas tablas y las tablas ordinarias:

Tabla Ordinaria	Tablas Organizadas por Índice
<b>ROWID</b> identifica una fila.	La llave primaria identifica una fila.
Llave primaria opcional.	Llave primaria obligatoria.
Acceso por el <b>ROWID</b> .	Acceso por la llave primaria.
Análisis secuencial para recuperar todas las filas.	Un análisis completo del índice recupera todas las filas ordenadas por la PK.
Pueden almacenarse en un cluster.	No pueden almacenarse en un cluster.
Pueden contener columnas de tipos <b>LONG</b> y <b>LOB</b> .	Pueden contener columnas <b>LOB</b> , pero no <b>LONG</b> .

# ÍNDICES

- ▶ Estructura opcional asociada con una tabla o un *cluster*.
  - ▶ Se crean sobre una o varias columnas, para acelerar la ejecución de sentencias SQL.
    - ▶ Tras cada columna puede especificarse `ASC` o `DESC`.
  - ▶ Una tabla puede tener cualquier **cantidad de índices**, si la combinación de columnas en cada uno sea diferente. Incluso, cambiando el orden:
    - ▶ Ejemplos:
      - ▶ `CREATE INDEX Pieza_idx1 ON Pieza (Nombre, Cantidad);`
      - ▶ `CREATE INDEX Pieza_idx2 ON Pieza (Cantidad, Nombre);`
  - ▶ Podemos utilizar distintos **Tipos de Índices** que permiten una funcionalidad distinta de cara al rendimiento de la BD: Árboles B sobre las tablas, árboles B sobre los clusters, índices *hash* sobre clusters, índices de clave inversa e índices de mapas de bits.
  - ▶ Los índices son lógicamente y físicamente **Independientes de los Datos** de las tablas asociadas y son mantenidos dinámicamente y **automáticamente**.
    - ▶ Se puede crear o borrar un índice sin ningún efecto lateral sobre los datos de la tabla u otros índices.

# ÍNDICES

- ▶ Pueden ser únicos (*unique*) o no únicos (*nonunique*), según exijan o no que las columnas del índice admitan o no valores duplicados en distintas filas: `CREATE [UNIQUE] INDEX...`
  - ▶ Si esa restricción existe en la tabla (`UNIQUE`), Oracle crea un índice único automáticamente.
  - ▶ Oracle no recomienda crear índices únicos explícitamente.
- ▶ Es recomendable que sean **únicos** y que, al menos, se cree **uno por cada clave primaria o externa** de cada tabla, así como por cada columna que contenga valores de búsqueda usuales, sin excedernos.

# ÍNDICES

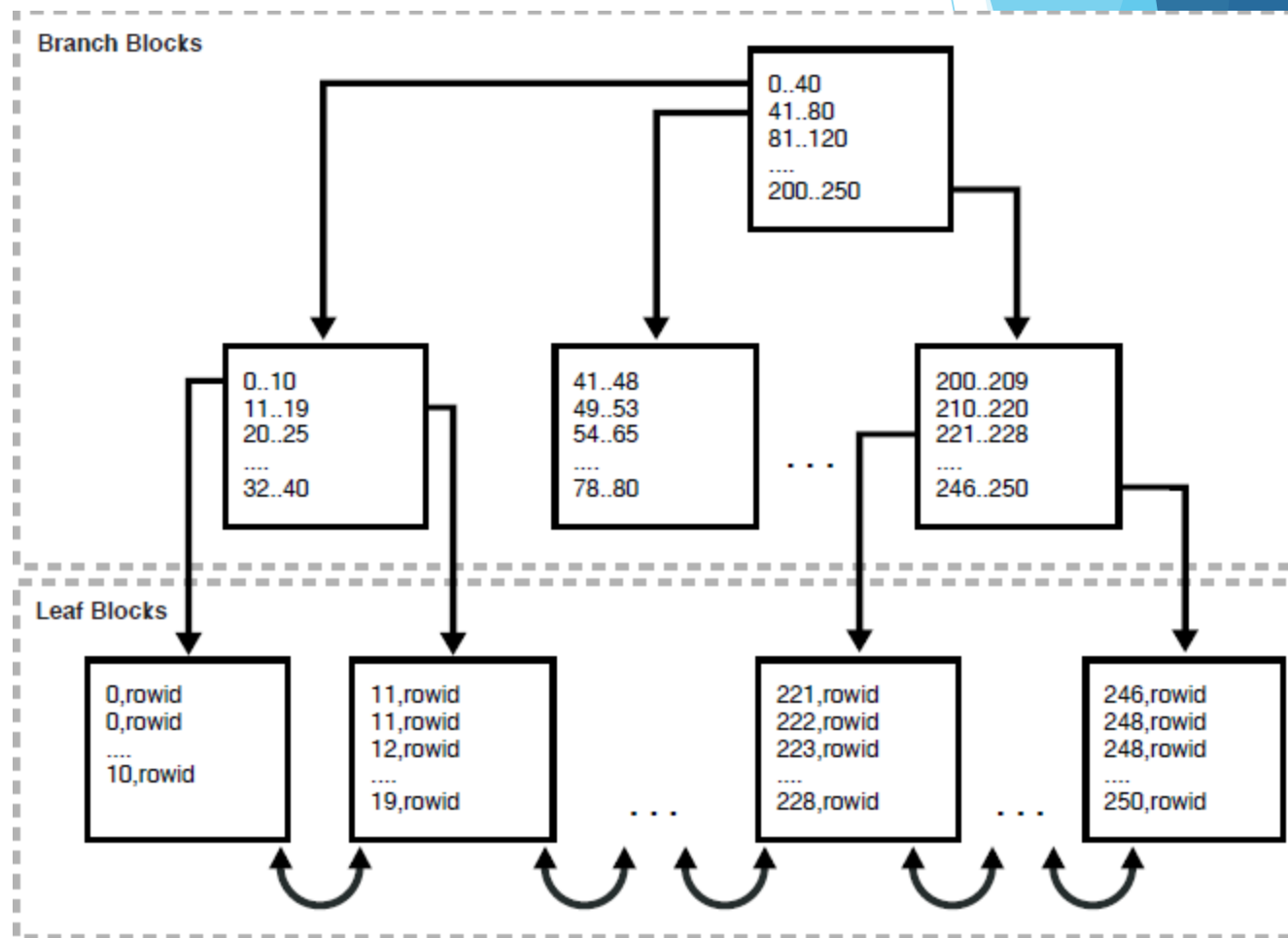
- ▶ Índices Basados en Función (*function-based indexes*): Se puede construir un índice sobre una función determinista definida por el usuario.
  - ▶ Ej. 1: Índice que será usado en consultas como la propuesta:
    - ▶ `CREATE INDEX idx ON table_1 (a + b * (c - 1), a, b);`
    - ▶ `SELECT a FROM table_1 WHERE a + b * (c - 1) < 100;`
  - ▶ Ej. 2: Para facilitar búsquedas insensibles a mayúsculas/minúsculas:
    - ▶ `CREATE INDEX uppercase_idx ON Pieza (UPPER(Nombre));`
    - ▶ `SELECT * FROM Pieza WHERE UPPER(Nombre) = 'TORNILLO';`

## ▶ Cuando se Crea un Índice:

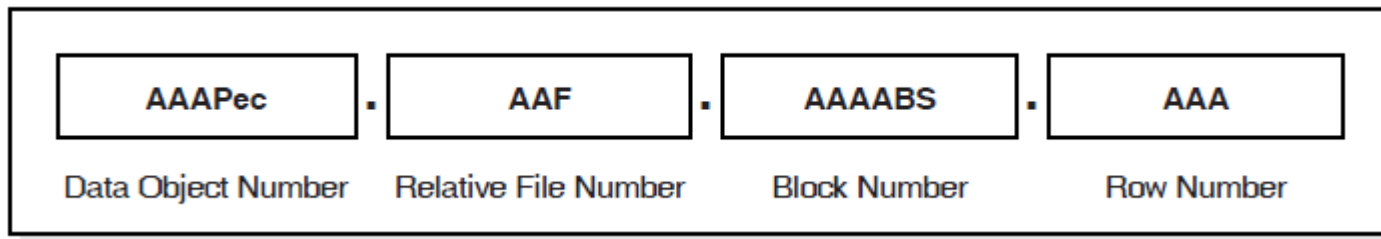
- ▶ Se le asigna un segmento de índice para contener sus valores en el *tablespace* correspondiente.
  - ▶ Es preferible que este *tablespace* no sea el mismo en el que está contenida la tabla asociada y que ambos *tablespaces* estén almacenados en discos diferentes, para que Oracle pueda leerlos en paralelo.
- ▶ Al crear un índice, Oracle ordena las columnas del índice y almacena el valor de los índices junto con el ROWID de las filas.
- ▶ Los índices pueden crearse en orden ascendente (**ASC**), descendente (**DESC**), comprimidos (**COMPRESS**) o no comprimidos (**NOCOMPRESS**).

# Cómo se crean los índices: Árboles B

- ▶ Branch Blocks: Para la búsqueda
- ▶ Leaf Blocks: Almacena los valores



# Formato del ROWID



- ▶ Data Object Number: Identifica el segmento
- ▶ Relative File number: Identifica el Datafile dentro del Tablespace
- ▶ Block Number: Número de datablock dentro del datafile
- ▶ Row Number: Fila dentro del datablock



# ÍNDICES DE MAPAS DE BITS

## ► (CREATE BITMAP INDEX)

- Para cada valor de clave se utiliza un mapa de bits en vez de el ROWID.
- Cada bit del mapa de bits corresponde a un ROWID, y vale 1 si la fila pertenece a ese valor clave, y 0 si no pertenece.
- Una función de mapeo convierte la posición del bit en un ROWID

Ejemplo: - TABLA:

CODIGO	E_CIVIL	...	PROVINCIA
101	Soltero	...	Málaga
102	Casado	...	Madrid
103	Soltero	...	Barcelona
104	Divorciado	...	Barcelona
105	Soltero	...	Madrid
106	Casado	...	Madrid

- ÍNDICE:

VALOR	MAPA DE BITS					
Barcelona	0	0	1	1	0	0
Madrid	0	1	0	0	1	1
Málaga	1	0	0	0	0	0

- Si el número de valores distintos de clave es pequeño (la cardinalidad de la columna es baja), los índices de mapas de bits son muy eficientes:
- Un índice **no** puede ser **BITMAP** y **UNIQUE** a la vez.
- En general, para columnas cuyos valores se repiten más de cien veces.
- Para columnas con valores con un bajo índice de repetición, también se muestra muy eficiente en casos en los que la definición del índice responde a **numerosas condiciones** de una cláusula **WHERE**, ya que las filas que satisfacen algunas, pero no todas las condiciones, son filtradas y desconsideradas antes de que la tabla sea accedida.

# CLUSTERS

- ▶ Grupo de tablas que comparten las mismas páginas porque tienen alguna columna en común y suelen usarse juntas.
  - ▶ Ejemplo: Si construimos un *cluster* con las tablas **EMPLEADOS** y **DEPARTAMENTOS** utilizando la columna común **CODIGO\_DEPARTAMENTO**, los empleados de un mismo departamento, junto con los datos del departamento, compartirían las mismas páginas, y cada valor clave del *cluster* se almacenará sólo una vez.
    - ▶ Si cambia el valor clave del *cluster* para una fila, Oracle la realojará.
- ▶ Pueden ser:
  - ▶ Cluster de índice, para las columnas de la clave del *cluster*.
  - ▶ Cluster HASH: Se accede a los datos mediante una **Función hash**: Función que se aplica a la columna o columnas de la clave del *cluster*, para obtener la página que corresponde a las filas de esa clave.
    - ▶ Esto ahorra tener que leer el índice para localizar o insertar un dato, ya que la función *hash* NO requiere lecturas de disco.



# 8. Administración del SGBD ORACLE

# Administración del SGBD Oracle

## ► Funciones del Administrador o DBA:

- **Instalación y actualización del *software* de Oracle** (servidor y aplicaciones).
  - Cambiar claves iniciales de las 2 cuentas DBA que Oracle crea automáticamente al crear una BD:
    - **sys**: Todas las tablas y vistas del diccionario de datos pertenecen al esquema de **sys**, y nadie debería modificarlas. Tampoco se deben crear nuevas tablas de la BD en las cuentas del DBA.
    - **SYSTEM**: Crea nuevas tablas y vistas con información administrativa.
- **Evaluación del *hardware***: Evaluar discos, memoria... asignar espacios de almacenamiento y planificar requerimientos futuros.
- Planificación de los **parámetros de creación** de la BD.
- **Creación de la BD**:
  - Estructuras de almacenamiento (*tablespaces...*) .
  - Implementación del diseño de la BD: Objetos (tablas, vistas...) y restricciones.
- Modificar la **estructura de la BD** cuando sea necesario.
- **Apertura y cierre** de la BD.
- **Gestión de usuarios y Sistemas de seguridad**: Permisos y Roles.
- **Auditoría**: Controlar y monitorizar el acceso a la BD.
- **Copias de seguridad** (*backup*) y sus recuperaciones (*recovery*).
- **Afinamiento** de la BD (optimizar su rendimiento).

# Administración del SGBD Oracle

## ► Utilidades de Administración:

- SQL\*Loader: Se usa para cargar datos desde ficheros del S.O. a tablas de la base de datos.
  - Puede usarse por usuarios y administradores.
  - Permite especificar el formato de entrada de los datos.
- Export e Import: Permiten mover datos entre distintas BD Oracle.
  - **Export** guarda los datos en ficheros, e **Import** lee esos ficheros y carga los datos en las tablas: Pueden usarse como medios para *backup*.
  - Puede ser entre versiones de Oracle de distintos S.O.

- Ver las Versiones de los distintos productos Oracle (núcleo, PL/SQL...): Se puede hacer consultando la vista del diccionario de datos llamada:

**PRODUCT\_COMPONENT\_VERSION.**

# Administración del SGBD Oracle

- ▶ Existen Dos Privilegios Importantes para la Administración (no son roles):
  - ▶ SYSOPER: Puede hacer casi todas las tareas de administración, excepto conceder la administración a otros, crear una BD y poco más.
  - ▶ SYSDBA: Contiene todos los privilegios del sistema **WITH ADMIN OPTION** (incluyendo **SYSOPER**) y permite usar **CREATE DATABASE**.
  - ▶ Se conceden normalmente : **GRANT SYSDBA TO scott;**
  - ▶ Y se revocan de similar forma : **REVOKE SYSDBA FROM scott;**
  - ▶ Usuario se puede conectar con: **CONNECT scott/tiger AS SYSDBA**
    - ▶ Se conecta al esquema por defecto (**PUBLIC** y **SYS** respectivamente), y no al esquema asociado al usuario, por lo que éste no podrá ver sus tablas sin cualificarlas con su nombre de usuario.

# Crear una BD Oracle

- ▶ Creación de la Base de Datos: Pasos para crear una BD Oracle:
  - ▶ Realizar Copias de Seguridad de otras posibles BD preexistentes.
  - ▶ Crear el Fichero de Parámetros: Las instancias se arrancan a partir de un fichero de parámetros.
    - ▶ Para cada base de datos, debe tenerse un fichero de parámetros de este tipo.
      - ▶ Oracle suministra un fichero de parámetros de inicialización por defecto que puede ser editado y modificado para cada base de datos.
    - ▶ Los parámetros son:
      - ▶ **DB\_NAME**: Nombre local de la base de datos que no puede ser cambiado.
      - ▶ **DB\_DOMAIN**: Localización lógica de la BD en la red (por ejemplo SCI.UMA.ES). En combinación con **DB\_NAME** debe identificar un único nombre en la red.
      - ▶ **CONTROL\_FILES**: Si no especifica nada, Oracle creará uno por defecto. Se recomienda tener al menos dos ficheros de control en distintos discos.
      - ▶ **DB\_BLOCK\_SIZE**: Es el tamaño de página de la BD. Por defecto es el mismo tamaño que el de un bloque del S.O. (4096 ó 8192 bytes).
      - ▶ **PROCESSES**: Máximo número de procesos que pueden conectarse a la BD simultáneamente. Debe incluir los 5 procesos de *background* más uno por cada usuario. Si se estiman 50 usuarios concurrentes como máximo, este valor puede ser de 55.
      - ▶ **ROLLBACK\_SEGMENTS**: Es una lista de los segmentos de *rollback* que la Instancia asigna cuando arranca la BD.

# Fichero de parámetros

- ▶ Server Parameter File (SPFILE). Por defecto.
  - ▶ Sólo uno por base de datos. Debe residir en el mismo host que la base de datos
  - ▶ Leído y escrito por el SGBD (no por aplicaciones cliente)
  - ▶ Es binario y no puede ser modificado externamente
  - ▶ Se puede modificar un parámetro con un comando SQL y almacenarlo en el SPFILE  
`ALTER SYSTEM SET parametro=valor SCOPE=SPFILE;`
- ▶ Fichero de Parámetros de Texto (PFILE)
  - ▶ Fichero de texto con una lista de parámetros y sus valores
  - ▶ Puede residir en el ordenador donde se ejecute la aplicación cliente que arranca la BD
  - ▶ Para modificarlo se hace desde editor (ALTER SYSTEM no lo modifica)
- ▶ Los parámetros estáticos de la base de datos se han de modificar en los ficheros de parámetros
- ▶ Crear PFILE una vez arrancada la BD con el SPFILE:  
`Create pfile from spfile`



# Crear una BD Oracle (continuación)

- ▶ Arrancar la Instancia: Mediante la instrucción **STARTUP** de SQL\*Plus
- ▶ Crear la BD: Se hace mediante la instrucción **CREATE DATABASE**, lo que provoca que se realicen las siguientes operaciones automáticamente:
  - ▶ Crea los ficheros de datos para la BD (*datafiles*): **ALTER TABLESPACE**
  - ▶ Crea los ficheros de control (*control files*): **CREATE CONTROL FILE**
  - ▶ Crea los registros de rehacer (*redo log*).
  - ▶ Crea el *tablespace* SYSTEM y el segmento de *rollback* SYSTEM:  
**CREATE TABLESPACE**
  - ▶ Crea el diccionario de datos.
  - ▶ Crea a los usuarios **SYS** y **SYSTEM**.
  - ▶ Especifica el conjunto de caracteres que se almacenarán.
  - ▶ Monta y abre la BD para su uso.
- ▶ Realizar una Copia de Seguridad de la BD.

# Algunas Sentencias SQL del DBA

- ▶ El Comando **CREATE** para Crear objetos puede sustituirse por **DROP** y **ALTER** para las Borrar y Modificar el objeto en cuestión:
  - ▶ **CREATE USER:** Crea un usuario, una cuenta para acceder a la BD.
  - ▶ **CREATE ROLE:** Crea un conjunto de privilegios con un nombre.
  - ▶ **CREATE SYNONYM:** Crea un sinónimo. Puede establecerse como sinónimo público (sinónimo accesible para todos los usuarios).
  - ▶ **CREATE TABLESPACE:** Crea un *tablespace*, espacio en la BD que puede contener objetos.
  - ▶ **CREATE ROLLBACK SEGMENT:** Crea un segmento de anulación (*rollback*), un objeto donde Oracle almacena los datos para deshacer modificaciones.
  - ▶ **GRANT:** Otorga roles y permisos (o privilegios) del sistema o de objetos a usuarios. Los privilegios se retiran con el comando **REVOKE**.
  - ▶ **ANALYZE:** Almacena o borra en el Diccionario de Datos estadísticas sobre el objeto que se especifique. Por ejemplo, para una tabla el resultado se almacenará en **USER\_TABLES**.
  - ▶ **AUDIT:** Realiza un seguimiento sobre las operaciones ejecutadas o sobre objetos accedidos (usuario, tipo de operación, objeto implicado, fecha y hora). Para detener la auditoria usar **NOAUDIT**. Los datos se guardan en tablas del diccionario con el texto **AUDIT\_** en su nombre, como **DBA\_AUDIT\_OBJECT**, **DBA\_AUDIT\_TRAIL...**



# 9. Herramientas

# Herramientas ORACLE

## ▶ Enterprise Manager

- ▶ Tareas administrativas: crear objetos del esquema (tablespaces, tablas e índices),
- ▶ Manejar seguridad de usuarios, backup, y recuperación, importación/exportación de datos.
- ▶ Visualizar estado de rendimiento.
- ▶ `http://hostname:portnumber/em`



•Anotar al instalar

# Enterprise Manager

## **Almacenamiento**

Archivos de Control, Tablespaces, Grupos de Tablespaces, Temporales, Archivos de Datos, Segmentos de Rollback, Grupos de Redo Logs, Archive Logs

## **Configuración de la Base de Datos**

Parámetros de Memoria, Gestión de Deshacer, Todos los Parámetros de Inicialización, Uso de Funciones de la Base de Datos

## **Planificador de Base de Datos**

Trabajos, Cadenas, Planificaciones, Programas, Clases de Trabajos, Ventanas, Grupos de Ventanas, Atributos Globales

## **Gestión de Estadísticas**

Repositorio de Carga de Trabajo Automática, Gestionar Estadísticas del Optimizador

## **Cambiar Base de Datos**

Migrar a ASM, Gestionar Tablespace Localmente

## **Gestor de Recursos**

Monitores, Grupos de Consumidores, Asignaciones de Grupos de Consumidores Planes

## **Políticas**

Biblioteca de Políticas, Violaciones de Política

# Enterprise Manager

## Esquema

### Objetos de Base de Datos

Tablas, Índices, Vistas, Sinónimos, Secuencias, Enlaces de Base de Datos, Objetos de Directorio, Reorganizar Objetos

### Programas

Paquetes, Cuerpos de Paquetes, Procedimientos, Funciones, Disparadores, Clases, Java, Orígenes Java,

### Base de Datos XML

Configuración, Recursos, Listas de Control de Acceso, Esquemas XML, Tablas de Tipo XML, Vistas de Tipo XML

### Usuarios y Privilegios

Usuarios, Roles Perfiles, Valores de Auditoría

### Vistas Materializadas

Vistas Materializadas, Logs de Vistas Materializadas, Grupos de Refrescamiento

### BI & OLAP

Dimensiones, Cubos, Dimensiones de OLAP, Carpetas de Medidas

### Tipos Definidos por el Usuario

Tipos de Matrices, Tipos de Objetos, Tipos de Tablas

### Administración de Enterprise Manager

Administradores, Planificación de Notificación, Interrupciones

# Herramientas ORACLE

## ▶ Administración

- ▶ Oracle Universal Installer (OUI)
  - ▶ Instala el Software de Oracle y las opciones.
- ▶ Database Configuration Assistant (DBCA)
  - ▶ Crea una base de datos usando plantillas proporcionadas por Oracle, Permite copiar una base de datos semilla, etc.
- ▶ Database Upgrade Assistant
  - ▶ Asistente que guía en el paso de una base de datos existentes a una nueva versión.
- ▶ Oracle Net Manager
  - ▶ Guía en la configuración de la red Oracle.



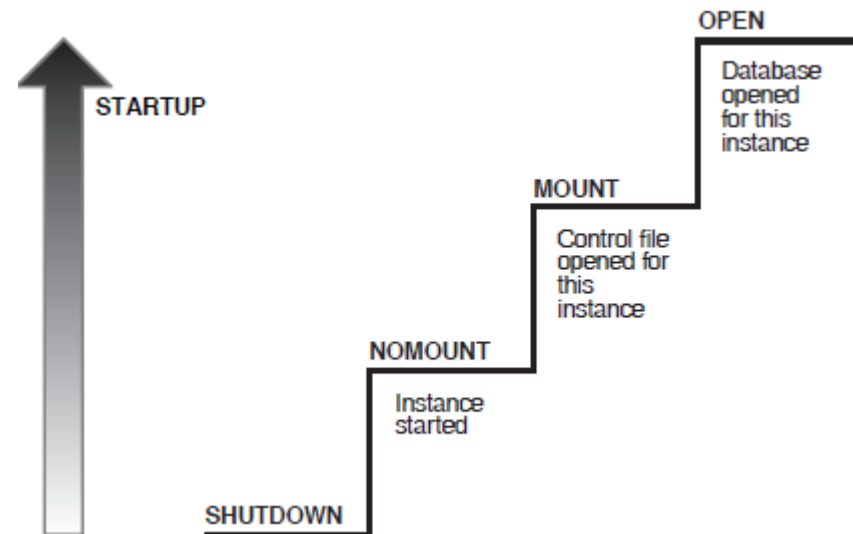
# 10. Iniciar/Finalizar

## ORACLE



# Iniciar/Finalizar ORACLE

- ▶ Inicialización (*Startup*): Es necesaria para que el SGBD pueda utilizarse.
  - ▶ Crear una Instancia: Crear el SGA y los procesos de *background* (se lee el fichero de parámetros).
  - ▶ Montar una BD: Asocia la instancia ya creada a una BD concreta. Para montar la BD es necesario leer los *ficheros de control*.
  - ▶ Abrir la BD: Establece la BD como disponible para sus operaciones. Si la base de datos fue cerrada anormalmente, se realiza el *recovery*.



# Iniciar/Finalizar ORACLE

## ► Finalización (*Shutdown*): Es el proceso inverso:

### ► Cerrar la BD.

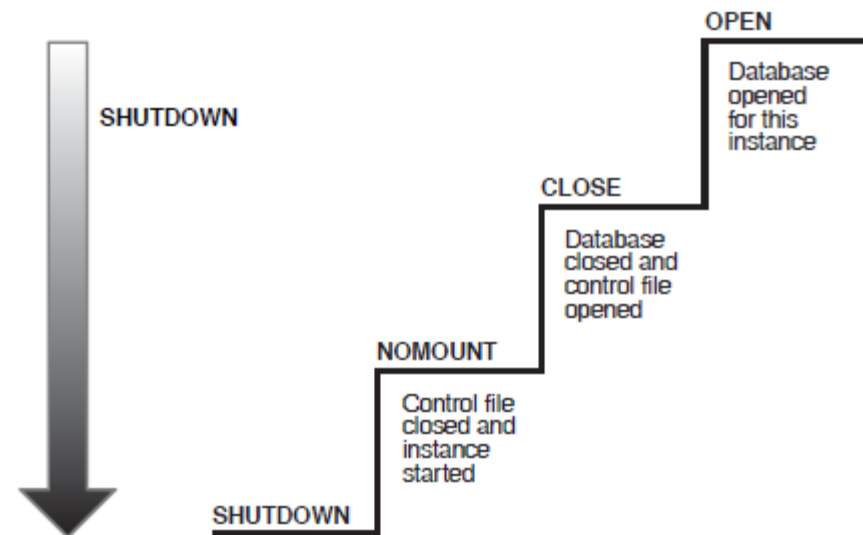
- Se escriben los datos de la base de datos y los datos de recovery de la SGA a los datafiles y ficheros de redo, respectivamente.
- Se cierran los datafiles y ficheros de redo.
- Los ficheros de control permanecen abiertos

### ► Desmontar la BD.

- Se cierran los ficheros de control. La SGA se mantiene en memoria.

### ► Borrar la Instancia Oracle.

- La SGA se borra de la memoria



# Iniciar/Finalizar ORACLE

## ► Permisos necesarios:

- SYSDBA y SYSOPER son privilegios especiales del sistema que permiten el acceso a una instancia de base de datos, incluso cuando la base de datos no está abierta. El control de estos privilegios se encuentra fuera de la propia base de datos.
- Cuando se conecta con el privilegio del sistema SYSDBA, se usa el esquema SYS.
- Cuando se conecta como SYSOPER, se usa el esquema PUBLIC.
- Los privilegios de SYSOPER son un subconjunto de los privilegios de SYSDBA.

# Apertura y Cierre, con SQL\*Plus

## ▶ APERTURA de la BD:

- ▶ Con la sentencia **STARTUP** (que arranca la instancia).

```
STARTUP [PFILE=filename] [EXCLUSIVE] [PARALLEL]  
[MOUNT [dbname] | OPEN [open_options] [dbname] | NOMOUNT]
```

- ▶ **PFILE:** Especifica el fichero de parámetros.
  - ▶ **EXCLUSIVE:** La instancia se asociará a la BD en exclusiva y no permite otras instancias.
  - ▶ **PARALLEL:** Si se van a usar varias instancias para acceder a la BD.
  - ▶ **MOUNT:** Monta la BD con el nombre **dbname**, pero no la abre. Si no se especifica nombre lo toma del parámetro de inicialización **DB\_NAME**.
  - ▶ **OPEN:** Monta y abre la BD especificada con las opciones **open\_options**:  

```
READ {ONLY | WRITE [RECOVER]} | RECOVER
```
  - ▶ **NOMOUNT:** No monta (ni abre) la BD.
- ▶ Si se arranca sin montar la BD: **ALTER DATABASE <nombre> MOUNT;**
  - ▶ Si montamos la BD sin abrirla : **ALTER DATABASE <nombre> OPEN <modo>;**
    - ▶ donde **<modo>** es opcional y puede ser:
      - ▶ **READ ONLY:** No puede modificarse. No puede ser **READ WRITE** en otra instancia.
      - ▶ **READ WRITE RESETLOG:** Borra toda la información del registro de rehacer.
      - ▶ **READ WRITE NORESETLOG:** No borra toda esa información.

# Apertura y Cierre, con SQL\*Plus

## ► CIERRE de la BD:

- **SHUTDOWN [NORMAL]:** Por defecto. Espera a que terminen las conexiones (usuarios) y no admite nuevas.
- **SHUTDOWN TRANSACTIONAL:** No se permiten transacciones nuevas pero se espera a que las que están en curso se cierren.
- **SHUTDOWN IMMEDIATE:** Se terminan las instrucciones en curso, se desconecta a los usuarios y las transacciones activas hacen *rollback*. Se realiza un *checkpoint* y se cierran los ficheros.
- **SHUTDOWN ABORT:** El más rápido. Borra la instancia sin cerrar o desmontar la BD. Requiere hacer un *recovery* al arrancar de nuevo (sólo usar en caso de emergencia).