TEMA 6.

ESPACIO DE “UNDO”.

TEMA 6.
ESPACIO DE "UNDO".

- Concepto de Transacción.
- Espacio de "UNDO" (DESHACER).
  - Caídas del sistema.
  - Consistencia en lectura.
  - Retroceso de transacción.

- Gestión de Espacio de "UNDO" (DESHACER).
- Modo Automático de "UNDO".
  - Parámetros de inicialización.
  - Espacio de almacenamiento.
  - Creación, modificación, borrado e intercambio de espacio "undo".
  - Vistas.
  - Cálculo espacio "undo".

TEMA 6.
ESPACIO DE "UNDO".

- Modo manual de "UNDO".
  - Segmentos y sus tipos: segmentos de "rollback".
  - Escritura en un segmento de "rollback".
  - Segmento rollback system.
  - Segmentos de "rollback" públicos y privados.
  - Creación. Decremento y borrado. Parámetro optimal.
  - Puesta en/fuera de línea.
  - Modificación de parámetros.
  - Asignación explícita a una transacción.
  - Vistas estáticas.

TRANSACCIÓN.

- Unidad lógica de trabajo que contiene una o más sentencias SQL; se trata de una unidad indivisible o atómica. Los efectos de las sentencias de una transacción pueden ser todos validados (aplicados a la base de datos) o retrocedidos.

- Comienza en la primera sentencia SQL ejecutable.

- Termina cuando es validada o retrocedida, de forma explicita, mediante las sentencias COMMIT o ROLLBACK, o implícitamente, en el caso de sentencias DDL.

ESPACIO DE "UNDO" (DESHACER).

- Espacio de "undo": Conjunto de registros que guardan información, relativa a acciones realizadas por una transacción, necesaria para:
  - Recuperación de la base de datos.
  - Proporcionar consistencia en lectura (imagen de los datos).
  - Retroceder transacciones ("rollback").

ESPACIO DE "UNDO" Y CAÍDAS DEL SISTEMA.

- En caso de producirse una **caída del sistema** y quedar transacciones activas (sin validación –commit- ni retroceso – rollback-), Oracle recupera la información del espacio de "undo" y una vez hecho se realiza el "rollback" de dichas transacciones.

- En la recuperación de base de datos y una vez aplicados los cambios guardados en los ficheros de "redo", el espacio de "undo" sirve para deshacer los efectos de transacciones no validadas.

- Este proceso recibe el nombre de "rolling back" o "transaction recovery".

ESPACIO DE "UNDO" Y CONSISTENCIA EN LECTURA.

- Se usa la información en el espacio de "undo" para crear un conjunto de datos coherente respecto a un punto en el tiempo.

- Al validar la transacción ("commit") se libera la información pero no se destruye inmediatamente sino que permanece un tiempo para asegurar la consistencia en lectura de las consultas que comenzaron antes de la validación.

- Los cambios realizados por otras transacciones que suceden durante la ejecución de la consulta no son tenidos en cuenta por esta. Los bloques alterados son reconstruidos a partir del espacio de "undo", y los datos obtenidos enviados a la consulta.

ESPACIO DE "UNDO" Y CONSISTENCIA EN LECTURA.

- En ciertos casos, no pude devolverse un conjunto coherente de resultados, "snapshot", para consulta voluminosa. Ocurre porque no puede almacenarse suficiente información en el espacio de "undo"como para reconstruir los datos requeridos.

- Generalmente se produce si existe una gran actividad que fuerza a que se sobrescriban datos necesarios para lograr la consistencia. Se genera el error:

  ORA-01555 snapshot too old: rollback segment number "string" with name "string" too small

  - La solución es disponer de más espacio de "undo".

ESPACIO DE "UNDO" Y RETROCESO DE TRANSACCION.

- Retroceder una transacción ("rolling back") es deshacer cualquier cambio realizado a los datos por sentencias SQL de una transacción no validada.

- En el **retroceso de una transacción:**

  - Se aplican todos los cambios almacenados en orden inverso hasta llegar al dato original.

  - Se libera cualquier bloqueo de datos efectuado por la transacción.

  - Finaliza la transacción.

GESTION DE ESPACIO DE "UNDO" (DESHACER).

- Existen dos formas de gestionar el espacio de "undo":

  - Usando espacios de almacenamiento de "undo" (modo automático). Oracle recomienda que se trabaje de esta forma, dado que es menos complejo de implementar y más eficiente en su gestión.

  - Usando segmentos de "rollback" (modo manual).

  Ambas formas no pueden simultanearse.

- La forma de gestión, manual o automática, se determina en el arranque de la base de datos mediante el parámetro de inicialización UNDO_MANAGEMENT.

MODO AUTOMATICO "UNDO". PARAMETROS INICIALIZACION.

- El parámetro de inicialización UNDO_MANAGEMENT debe tener el valor AUTO.

UNDO_MANAGEMENT = AUTO

- Al arrancar se busca un espacio de almacenamiento ("tablespace") de "undo" (existente desde la creación de la bd o creado posteriormente). Si no existe, se usa el segmento de "rollback" SYSTEM, y se genera un mensaje de error en el fichero de alertas.

MODO AUTOMATICO "UNDO". PARAMETROS INICIALIZACION.

Otros parametros de inicialización relacionados son:

- *UNDO_RETENTION*. Parámetro dinamico (*alter system set undo_retention= <valor>)* que indica en segundos -por defecto 900-, cuanto tiempo ha de permanecer la informacion de "undo" disponible -importante en largas transacciones-.

  Si se necesita espacio para las transacciones activas y no hay existe suficiente , se reutiliza el existente y puede provocar el fallo de consultas muy largas.

- *UNDO_TABLESPACE*. Parámetro dinámico (*alter system ...*) que indica el espacio de almacenamiento de "undo" a usar en el arranque. Si se indica en modo manual, provoca error y falla el arranque.

MODO AUTOMATICO "UNDO". PARAMETROS INICIALIZACION.

Si se omite, se elige el primer espacio de almacenamiento de "undo" disponible.

- *UNDO_SUPPRESS_ERRORS.* Parámetro dinámico (*alter system ...*) que permite suprimir los mensajes de erro generados cuando sentencias SQL propias de la gestión en modo manual se usan en modo automático.Por ejemplo, si se usa *SET TRANSACTION USE ROLLBACK SEGMENT* se generará el error ORA-30019.

Al arrancar en modo automático, cualquier parámetro relativo al modo manual es ignorado.

MODO AUTOMATICO "UNDO". ESPACIO DE ALMACENAMIENTO.

- Para el uso en modo automático es necesario un espacio de almacenamiento ("tablespace") de "undo", que está reservado exclusivamente para esta función (no pueden crearse allí objetos de datos).

- Distintas operaciones pueden llevarse a cabo con este espacio de almacenamiento:

  - Creación.
  - Modificación.
  - Borrado.
  - Intercambio entre distintos espacios de "undo".

MODO AUTOMATICO “UNDO”. CREACION DE ESPACIO “UNDO”.

- Al crear la base de datos, mediante la clausula UNDO TABLESPACE de la sentencia CREATE DATABASE:

```
CREATE DATABASE CURSO25 ...
    UNDO TABLESPACE undotbs_01 DATAFILE
    '/u03/oradata/CURSO25/undo01.dbf';
```

- Mediante la sentencia CREATE UNDO TABLESPACE (identica a CREATE TABLESPACE):

```
CREATE UNDO TABLESPACE undotbs1
    DATAFILE '/u03/oradata/CURSO25/undo01.dbf'
    SIZE 10M AUTOEXTEND ON;
```

MODO AUTOMATICO “UNDO”. CREACION DE ESPACIO “UNDO”.

- Restricciones:

  En la creacion sólo puede especificarse la clausula DATAFILE (localización del fichero), determinando Oracle el resto de atributos.

MODO AUTOMATICO "UNDO". MODIFICACION ESPACIO "UNDO".

- Mediante la sentencia ALTER TABLESPACE. Se permite:

  - Añadir un fichero de datos.

    ALTER TABLESPACE UNDOTBS ADD DATAFILE '/u03/oradata/CURSO25/undo02.dbf' AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED;

  - Redimensionar un fichero de datos.

    ALTER DATABASE DATAFILE '/u03/oradata/CURSO25/undo01.dbf' RESIZE 20M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED;

MODO AUTOMATICO “UNDO”. MODIFICACION ESPACIO “UNDO”.

- Renombrar un fichero de datos.

```
ALTER DATABASE RENAME FILE
'/u03/oradata/CURSO25/undo01.dbf' TO
'/u03/oradata/CURSO25/tbsp_undo01.dbf';
```

- Poner en linea o fuera de linea un fichero de datos.

```
ALTER DATABASE DATAFILE
'/u03/oradata/CURSO25/undo01.dbf'
  ONLINE/OFFLINE;
```

MODO AUTOMATICO “UNDO”. BORRADO ESPACIO “UNDO”.

- Se emplea la sentencia DROP TABLESPACE:

DROP TABLESPACE <nombre_tbsp>;

- Sólo es posible borrar si el espacio de “undo” no está en uso. Al borrar se elimina todo su contenido.

MODO AUTOMATICO “UNDO”. INTERCAMBIO ESPACIO “UNDO”.

- Se usa la sentencia ALTER SYSTEM SET para asignar un nuevo espacio de “undo”, que sustituye al que anteriormente se utilizaba:

ALTER SYSTEM SET
UNDO_TABLESPACE=<nombre_tbsp>;

- Se producirá error en caso de que el nuevo espacio de almacenamiento no exista, no sea de “undo” o se este usando por otra instancia.
- La bd está en línea mientras se realiza la operación; y pueden ejecutarse transacciones, al terminar todas aquellas comenzadas despues de la sentencia se asignan al nuevo espacio de “undo”.

MODO AUTOMATICO “UNDO”. INTERCAMBIO ESPACIO “UNDO”.

- Si existen transacciones pendientes de validar, “commit”, en el antiguo espacio de “undo”, este pasa al estado PENDING OFFLINE. Las transacciones siguen su curso pero no se usa para nuevas transacciones.
- En el estado PENDING OFFLINE, un espacio de “undo” no puede borrarse. Una vez finalizadas todas las transacciones pasa al estado OFFLINE.

- Para deasignar un espacio de “undo”, en caso de querer realizar una gestión manual:

```
ALTER SYSTEM SET UNDO_TABLESPACE='';
```

MODO AUTOMATICO "UNDO". VISTAS.

- V$UNDOSTAT. Estadísticas para monitorizar y ajustar el espacio de "undo"
- V$ROLLSTAT. Informa sobre el comportamiento de los segmentos "undo" en el espacio de "undo".
- V$TRANSACTION. Transacciones activas en el sistema.
- DBA_UNDO_EXTENTS. Extensiones en el espacio de "undo".

MODO AUTOMATICO “UNDO”. CALCULO ESPACIO “UNDO”.

- El espacio de “undo” necesario para retener la informacion un tiempo determinado (UNDO_RETENTION) es:

  Espacio = UR * (Tasa_transaccion * tamaño_bloque)

- Donde UR es el valor de UNDO_RETENTION, en segundos, y la tasa de transaccion el numero de bloques “undo” por segundo (columna UNDOBLKS de V$UNDOSTAT).

MODO MANUAL "UNDO". PARAMETROS INICIALIZACION.

- El parámetro de inicialización UNDO_MANAGEMENT debe tener el valor MANUAL, o bien no indicarse.

UNDO_MANAGEMENT = MANUAL

MODO MANUAL "UNDO". PARAMETROS INICIALIZACION.

Parametros de inicialización relacionados son:

- *ROLLBACK_SEGMENTS*. Asigna segmentos a la instancia.
- MAX_ROLLBACK_SEGMENTS. Número máximo de segmentos "online" para la instancia.
- *TRANSACTIONS*. Indica el número máximo de transacciones concurrentes. Valores mayores incrementan el tamaño de la SGA y pueden incrementar el numero de segmentos reservados.
- TRANSACTIONS_PER_ROLLBACK_SEGMENT. Indica el número de transacciones concurrentes que cada segmento se espera que maneje.

SEGMENTOS Y SUS TIPOS.

- Segmento: Un conjunto de extensiones que contiene todos los datos para una estructura lógica de almacenamiento especifica en un “tablespace”.

- Tipos de segmentos:

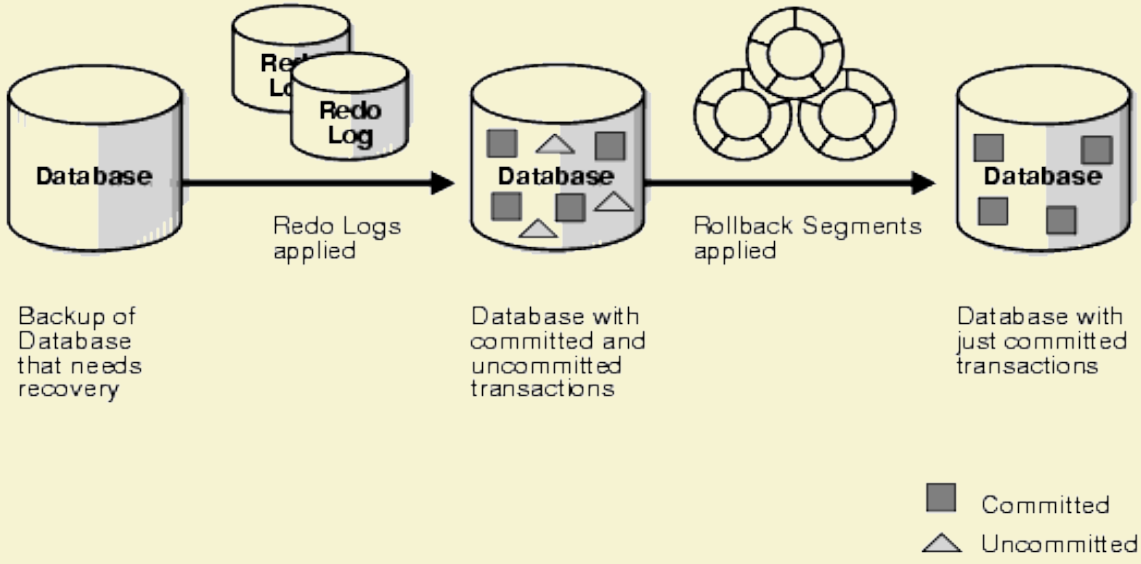
  - Segmentos de datos.
  - Segmentos de índices.
  - Segmentos temporales.

  - **Segmentos de “rollback”.**

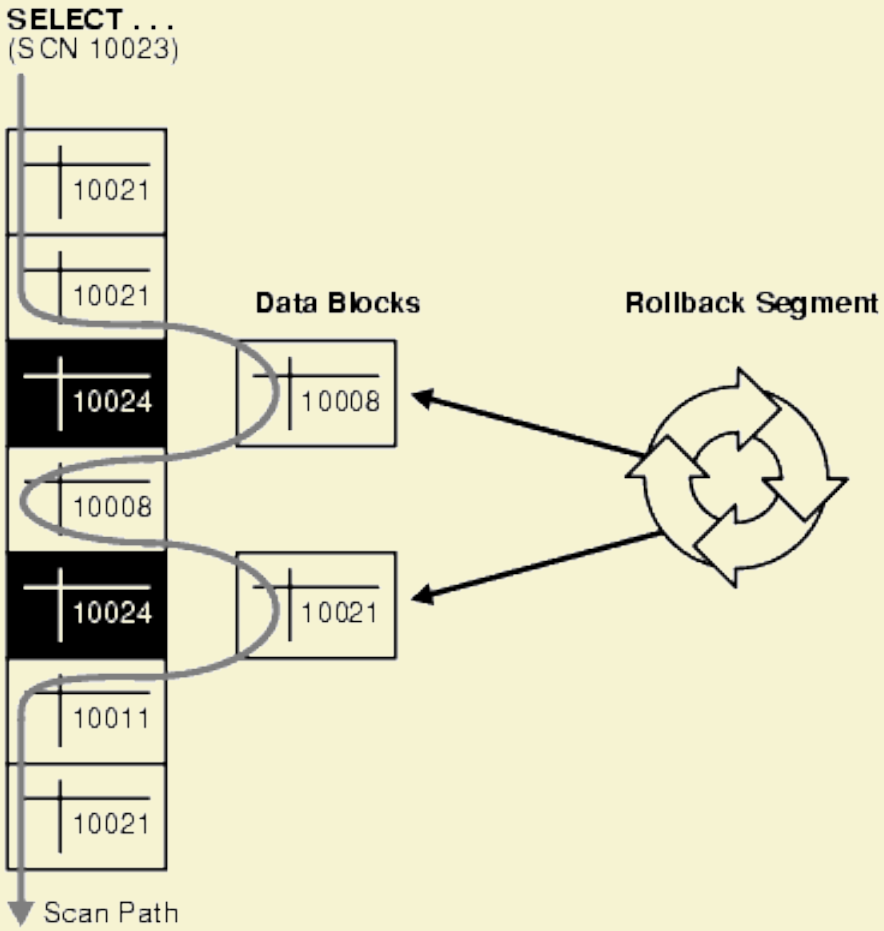
SEGMENTOS DE “ROLLBACK”.

- Toda base de datos posee uno o más segmentos de “rollback”.
- Contiene/n los valores antiguos de datos modificados por cada transacción.
- La información se dispone en múltiples entradas de “rollback” que contienen información de bloque y el dato tal como existía antes de la operación involucrada en la transacción.
- Estas entradas modifican los bloques del segmento de “rollback” y Oracle almacena todos los cambios hechos en la bitácora (“redo log”).

S.”ROLLBACK”.
Y “ROLLING BACK”.



S.R.
Y CONSISTENCIA EN LECTURA.



TRANSACCIONES Y SEGMENTOS DE "ROLLBACK".

- Una transacción es asociada a un segmento:

  - **Automáticamente**, al siguiente segmento libre que exista.

    Oracle distribuye las transacciones entre los segmentos activos de forma que todos ellos trabajen con, aproximadamente, el mismo número. Esto no depende del tamaño de los segmentos.

  - **Por asignación**. Al comienzo de la transacción puede indicarse el segmento de "rollback" apropiado a usar (SET TRANSACTION USE ROLLBACK SEGMENT …).

ESCRITURA EN UN SEGMENTO DE "ROLLBACK".

- Durante la transacción se escribe la información de "rollback" en el segmento asignado de forma secuencial. Cada transacción escribe en una única extensión del segmento en un momento dado.

- Por cada segmento existe una **tabla de transacciones**: Lista de todas las transacciones que lo usan y las entradas en el mismo para cada modificación realizada por dichas transacciones.

- Muchas transacciones activas pueden escribir concurrentemente en un segmento, pero cada bloque de datos de una extensión de un segmento solo puede contener información de una transacción.

ESCRITURA EN UN SEGMENTO DE "ROLLBACK".

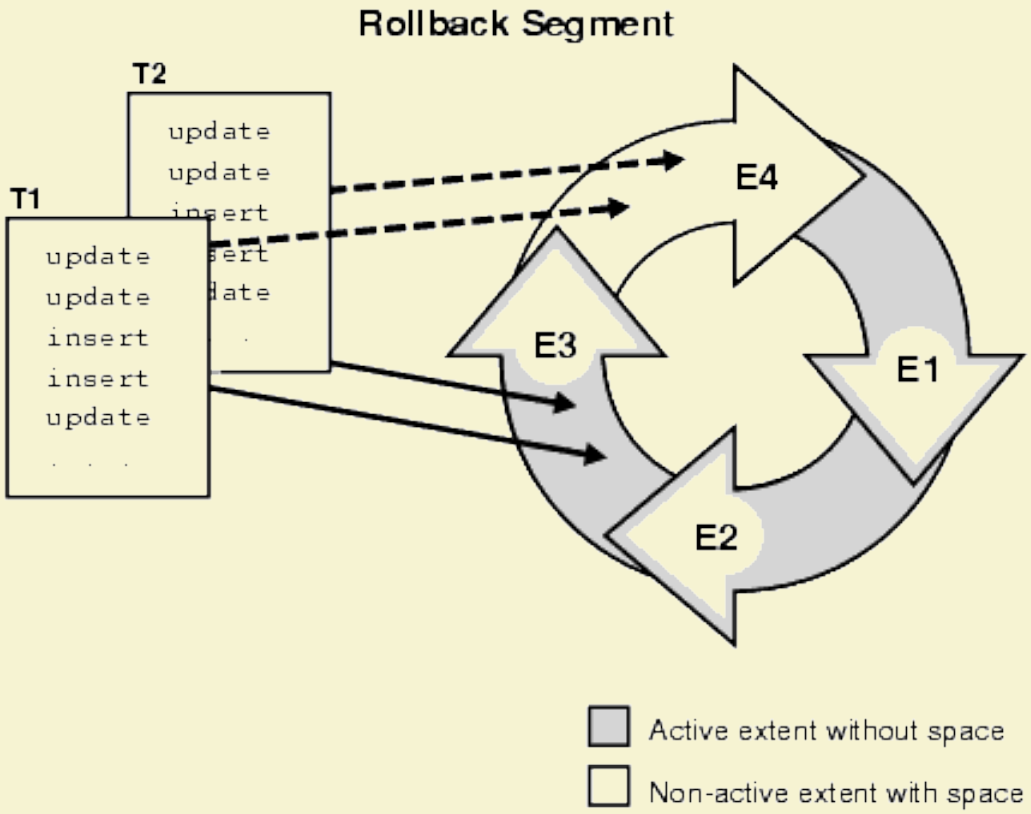
- Cada segmento de "rollback" debe tener al menos dos extensiones asignadas.

- Si una transacción agota el espacio libre en la extensión actual y debe proseguir la escritura, se localiza una extensión en el mismo segmento:

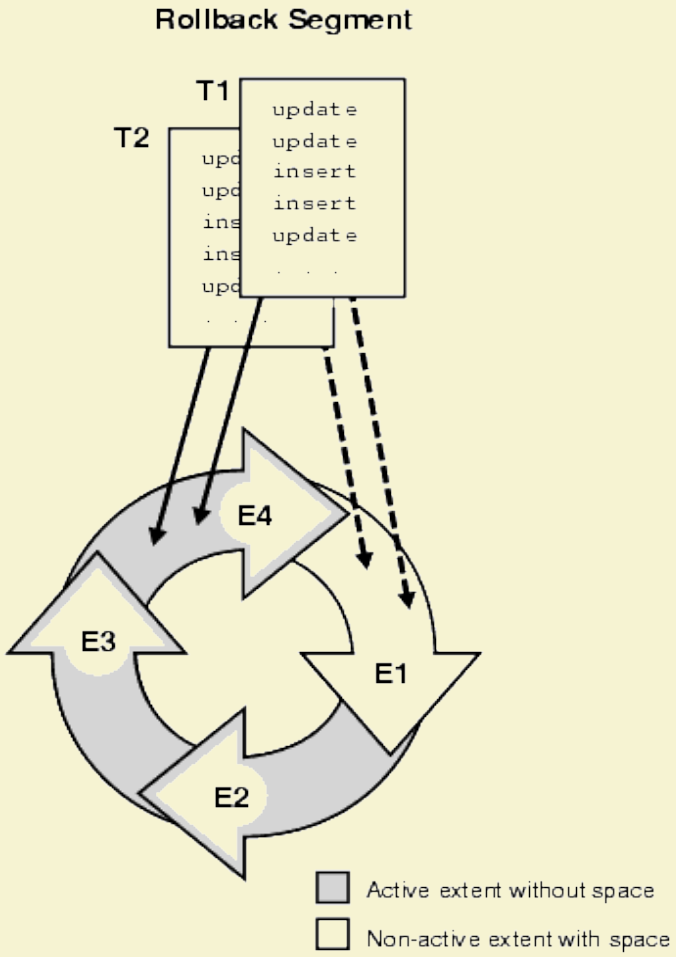
  - Bien se reutiliza una extensión ya asignada al segmento. Se comprueba la siguiente extensión y si no contiene información de una transacción activa, se convierte en la extensión actual.

  - O bien se asigna una nueva extensión al segmento.

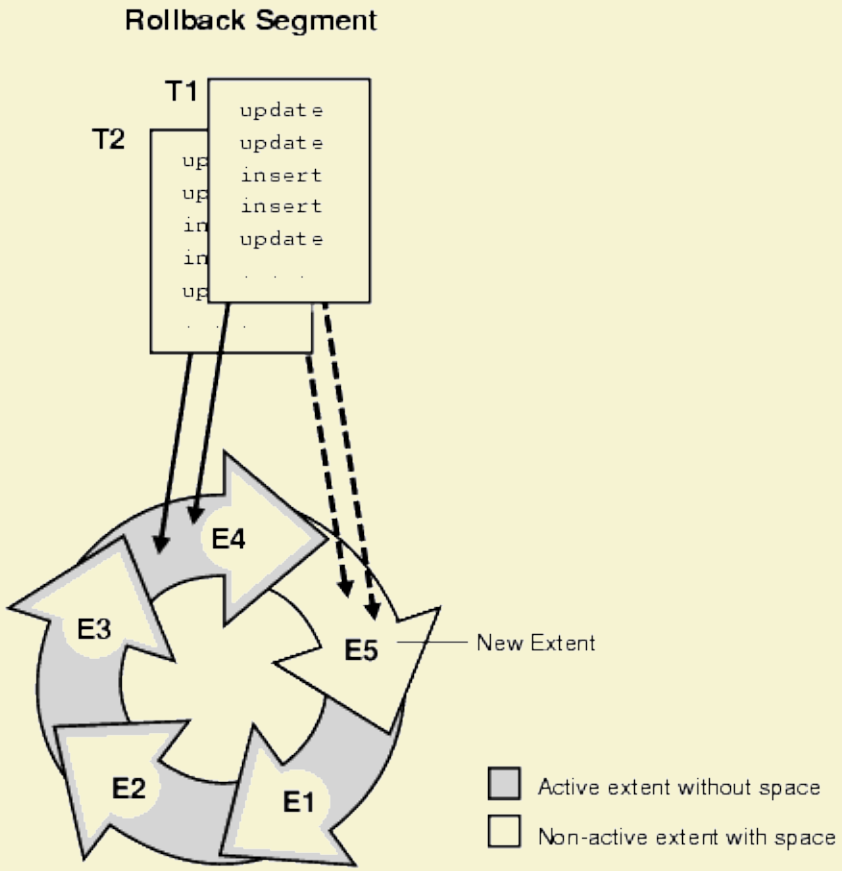
ESCRITURA EN UN SEGMENTO DE "ROLLBACK".



ESCRITURA EN UN SEGMENTO DE "ROLLBACK".



ESCRITURA EN UN SEGMENTO DE "ROLLBACK".



SEGMENTO ROLLBACK SYSTEM.

- Se crea al mismo tiempo que la base de datos.
- Reside en el "tablespace" SYSTEM y usa sus parámetros de almacenamiento por defecto.
- No puede borrarse.
- Si existen diversos segmentos de "rollback", se emplea para transacciones especiales del sistema, y las transacciones de usuario son distribuidas entre otros segmentos de "rollback".
- Se recomienda crear segmentos adicionales al SYSTEM tras instalar la base de datos.
- Si hay excesivas transacciones para el resto de segmentos, el segmento SYSTEM también es usado.

SEGMENTOS DE "ROLLBACK" PÚBLICOS Y PRIVADOS.

- Un segmento privado se adquiere **explícitamente** por la instancia si es nombrado en el fichero de inicialización (parámetro ROLLBACK_SEGMENTS). También puede usarse al ponerlo en línea de forma manual.

- Los segmentos públicos forman un conjunto que cualquier instancia puede usar. El número de segmentos públicos adquiridos automáticamente depende de los valores de TRANSACTIONS *y* TRANSACTIONS_PER_ROLLBACK_SEGMENT.

- La distinción tiene sentido si se usa la opción Oracle9i Real Application Clusters.

RECOMENDACIONES.

- Se recomienda crear un espacio de almacenamiento exclusivo que albergue a los segmentos de “rollback”.

- Ventajas:

  - Puede ser mantenido en línea de forma permanente.

  - No impide que otros espacios de almacenamiento sean puestos fuera de línea.

CREACIÓN.

- Debe poseerse el privilegio “create rollback segment”.
- El espacio de almacenamiento destino debe estar en línea.
- Sintaxis:

```
CREATE <PUBLIC> ROLLBACK SEGMENT <NOMBRE>
TABLESPACE <NOMBRE_TBSP>
STORAGE (INITIAL <XX>
                NEXT <XX>
                MINEXTENTS <XX>
                MAXEXTENTS <XX>
                OPTIMAL <XX> );
```

CREACIÓN. RECOMENDACIONES.

- INITIAL y NEXT deben ser del mismo valor: Se consiguen así extensiones de tamaño uniforme.
- Debe crearse un adecuado número de extensiones iniciales para minimizar la necesidad de extensión.
- No debe fijarse MAXEXTENTS=UNLIMITED. Se evita así que se extienda de forma ilimitada debido a un error de programación.

CREACIÓN.
PARÁMETRO OPTIMAL.

- Especifica el tamaño optimo del segmento. Oracle intenta mantener este tamaño para el segmento deasignando dinámicamente las extensiones cuando sus datos no son necesarios para otras transacciones activas.

- Cuando una transacción necesita escribir en otra extensión, se compara el tamaño actual del segmento con el optimo; si es mayor, y la extensión siguiente a la que se ha llenado es inactiva, se desalojan extensiones hasta llegar al optimo (siempre las mas antiguas).

- Su valor no puede ser menor que el espacio asignado inicialmente, especificado por los parámetros initial, next y el numero mínimo de extensiones.

CREACIÓN. PARÁMETRO OPTIMAL.

- Las estadísticas generadas en la vista V$ROLLSTAT (valores SHRINKS, AVESHRINK, AVEACTIVE y OPTSIZE) dan idea de lo adecuado o no del parámetro OPTIMAL.
  - SHRINKS (bajo) y AVESHRINK (bajo): Si AVEACTIVE es cercano a OPTSIZE, OPTIMAL es correcto. En caso contrario, OPTIMAL es demasiado grande.
  - SHRINKS (bajo) y AVESHRINK (alto): OPTIMAL correcto.
  - SHRINKS (alto) y AVESHRINK (bajo): OPTIMAL demasiado pequeño.
  - SHRINKS (alto) y AVESHRINK (alto): Aumentar OPTIMAL hasta que SHRINKS disminuya.

TAMAÑO SEGMENTO "ROLLBACK".

- Aunque los segmentos de "rollback" pueden manejar transacciones de cualquier dimensión:
  - Si un sistema ejecuta sólo transacciones cortas es preferible que los segmentos sean pequeños (siempre permanecen en memoria pero se corre el peligro de generar el ORA-01555).
  - Si son transacciones de larga duración es mejor usar segmentos de gran tamaño.
- Lo ideal es crear un número de segmentos de tamaño apropiado para cada tipo de transacción y asignar explicitamente las transaccion atípicas a aquellos que corresponda (por ejemplo, transacciones largas).

TAMAÑO SEGMENTO "ROLLBACK".

- Se recomienda que cada segmento típico tenga un 10% del tamaño de la mayor tabla de la bd.

PUESTA EN/FUERA DE LÍNEA.

- Cuando se crea un segmento esta fuera de línea y debe ser puesto en línea de forma explicita.

- Si se intenta poner fuera de línea un segmento activo y en uso solo se consigue cuando las transacciones que lo usan finalizan. Permanecerá fuera de línea hasta que explícitamente sea puesto en línea o la instancia rearrancada.

- Sintaxis:

ALTER ROLLBACK SEGMENT <NOMBRE> ONLINE;

ALTER ROLLBACK SEGMENT <NOMBRE> OFFLINE;

MODIFICACIÓN DE PARÁMETROS.

- Sintaxis:

```
ALTER ROLLBACK SEGMENT <NOMBRE>
STORAGE (INITIAL <XX>
        NEXT <XX>
        MINEXTENTS <XX>
        MAXEXTENTS <XX>
        OPTIMAL <XX> );
```

DECREMENTO.

- Manualmente puede reducirse el tamaño de un segmento de"rollback". El tamaño final depende del espacio libre en el segmento y de cuantas transacciones activas usan el segmento.

- Si no se especifica un tamaño concreto se intenta ajustar al del parámetro de creación OPTIMAL. Si no se ha determinado, al del parámetro MINEXTENTS.

- Sintaxis:

ALTER ROLLBACK SEGMENT <NOMBRE>
SHRINK TO <XX> K/M;

ALTER ROLLBACK SEGMENT <NOMBRE> SHRINK;

BORRADO.

- Debe poseerse el privilegio “drop rollback segment”.
- El segmento debe estar fuera de línea.
- Sintaxis:

DROP *ROLLBACK SEGMENT <NOMBRE>;*

ASIGNACIÓN EXPLICITA A UNA TRANSACCIÓN.

- El segmento debe estar en línea y usarse como primera sentencia de la transacción:

  SET TRANSACTION USE ROLLBACK SEGMENT <NOMBRE>;

- Un ejemplo de uso es asignar transacciones que insertan, actualizan o borran grandes cantidades de información a segmentos lo bastante grandes como para contener la información de "rollback" de la transacción.

VISTAS.

- V$ROLLNAME. Nombres de los segmentos de "rollback" en línea.
- V$ROLLSTAT. Estadísticas sobre segmentos de "rollback".
- V$TRANSACTION. Transacciones activas en el sistema.
- DBA_ROLLBACK_SEGS. Información sobre segmentos de "rollback" de la bd.
- DBA_SEGMENTS, donde tipo de segmento sea ROLLBACK

TEMA 7.

GESTIÓN DE USUARIOS Y RECURSOS.

TEMA 7.
GESTIÓN DE USUARIOS Y RECURSOS.

- Usuarios y su autentificación.
- Creación, modificación y borrado de usuarios.
- Perfiles.
- Creación, modificación, asignación y borrado de perfiles.
- Privilegios. Privilegios de sistema y sobre objetos.
- Privilegios de sistema. Otorgar y revocar. Restricciones.

TEMA 7.
GESTIÓN DE USUARIOS Y RECURSOS.

- Privilegios sobre objetos. Otorgar y revocar privilegios sobre objetos.
- Roles. Roles predefinidos.
- Creación, modificación, asignación y deasignación de roles a usuarios.
- Parámetro max_enabled_roles.
- Borrado de roles.

- Roles por defecto. Activación y desactivación de roles.
- Vistas estáticas y dinámicas.

USUARIOS Y SU AUTENTIFICACIÓN.

- Cada base de datos tiene una lista valida de usuarios. Para acceder a la misma un usuario debe ejecutar un aplicación y conectarse a la instancia usando un nombre valido previamente definido.

- Las formas más comunes de autentificar a un usuario son:

  - Por base de datos.

  - Por sistema operativo (autentificación externa).

USUARIOS Y SU AUTENTIFICACIÓN.

- En la autentificación por base de datos la administración de la cuenta de usuario, contraseña, que se guarda encriptada, y la autentificación es realizada por Oracle.

- En la autentificación externa la cuenta es mantenida por Oracle pero la administración de la contraseña y la autentificación de usuario es realizada externamente.

  El ejemplo más común es la autentificación por sistema operativo; los típicos usuarios ops$ o "identified externally". El prefijo ops$ no es obligatorio y viene determinado por el parámetro de inicialización, fichero init.ora, OS_AUTHENT_PREFIX (define el prefijo a añadir al comienzo de toda cuenta de usuario identificado por S.O. Y su valor por defecto es OPS$).

CREACIÓN DE USUARIOS.

- Necesario el privilegio de sistema *CREATE USER*. Normalmente sólo lo tiene el usuario administrador.
- El creador del usuario puede indicar cuota sobre espacios de almacenamiento aunque él no las posea.
- No es posible la conexión del usuario creado hasta que posea el privilegio de sistema *CREATE SESSION*.

CREACIÓN DE USUARIOS.

- Sintaxis:

CREATE USER <usuario>
IDENTIFIED BY <contraseña>/EXTERNALLY
DEFAULT TABLESPACE <espacio>
TEMPORARY TABLESPACE <espacio>
QUOTA <xx>K/UNLIMITED ON <espacio>
PROFILE <perfil>
PASSWORD EXPIRE
ACCOUNT LOCK/UNLOCK;

CREACIÓN DE USUARIOS.

- Nombre de usuario.

  Debe ser único respecto a otros nombres de usuario y roles. Cada usuario tiene asociado un esquema y dentro del mismo cada objeto debe tener un único nombre.

- Identificación.

  Un usuario autentificado de forma externa se ha de crear con la cláusula "*IDENTIFIED EXTERNALLY*".

CREACIÓN DE USUARIOS.

- DEFAULT TABLESPACE.

  Indica aquel espacio de almacenamiento donde se crearan los objetos del esquema del usuario cuando al hacerlo no se indica ninguno en particular.

  Si no se indica es el espacio SYSTEM (¡Error!).

- TEMPORARY TABLESPACE.

  Indica el espacio de almacenamiento donde se almacenan los segmentos temporales requeridos por el usuario

  No debe indicarse cuota y el espacio temporal por defecto es el SYSTEM (¡Error!).

CREACIÓN DE USUARIOS.

- QUOTA.

  Indica la cantidad de espacio reservada en un determinado espacio de almacenamiento para el usuario.

  Por defecto un usuario no tiene cuota en ningún espacio de almacenamiento.

  Indicando *UNLIMITED*, es ilimitado el espacio a usar.

  Puede revocarse el acceso a un espacio de almacenamiento asignando cuota cero sobre el mismo. Los objetos ya creados permanecen pero no pueden crecer ni crearse ninguno más.

CREACIÓN DE USUARIOS.

- PROFILE.

  Indica el perfil a asignar al usuario.

- PASSWORD EXPIRE.

  Fuerza al usuario a cambiar la clave antes de conectarse a la base de datos.

- ACCOUNT.

  "*ACCOUNT LOCK*", bloquea la cuenta de usuario y deshabilita el acceso. "*ACCOUNT UNLOCK*", desbloquea la cuenta de usuario y permite al acceso.

MODIFICACIÓN DE USUARIOS.

- Los usuarios pueden cambiar sus propias claves, sin embargo para cambiar cualquier otro parámetro es necesario el privilegio "*ALTER USER*".

- Sintaxis:

ALTER USER <usuario>
IDENTIFIED BY <contraseña>/EXTERNALLY
DEFAULT TABLESPACE <espacio>
TEMPORARY TABLESPACE <espacio>
QUOTA <xx>K/UNLIMITED ON <espacio>
DEFALT ROLE <role>/ALL/ALL EXCEPT <role>/NONE
PROFILE <perfil>
PASSWORD EXPIRE
ACCOUNT LOCK/UNLOCK;

MODIFICACIÓN DE USUARIOS.

- DEFAULT ROLE.

  Indica los roles otorgados por defecto al usuario en la conexión. Se refiere a roles otorgados de forma directa al usuario.

  Oracle activa los roles sin necesidad de especificar sus contraseñas.

BORRADO DE USUARIOS.

- Al borrar un usuario el esquema asociado, con todos sus objeto, desaparecen.
- Una posible solución para que permanezca el usuario y los objetos pero impedir la conexión es revocar el privilegio "*CREATE SESSION*".
- No es posible eliminar un usuario que permanezca conectado a la base de datos. Debe esperarse a que concluya o forzar su terminación (*ALTER SYSTEM KILL SESSION*).
- Es necesario tener el privilegio de sistemas "*DROP USER*".

BORRADO DE USUARIOS.

- Es conveniente estudiar las implicaciones que sobre otros esquemas tiene el borrado del usuario y de su esquema.

- Sintaxis:

DROP USER <usuario>;

DROP USER <usuario> CASCADE;

PERFILES.

- Está constituido por un conjunto de límites de recursos de la base de datos. Diferentes perfiles pueden ser asignados a diferentes usuarios.

- Habilitar o deshabilitar la limitación de recursos mediante perfiles puede hacerse:

  - Mediante el parámetro de inicialización *RESOURCE_LIMIT* (init.ora), asignando valores *TRUE* o *FALSE* (por defecto).

  - Mediante la sentencia *ALTER SYSTEM SET RESOURCE_LIMIT = TRUE/FALSE*.

CREACIÓN DE PERFILES.

- Es necesario el privilegio de sistema "*CREATE PROFILE*".
- Existe un perfil por defecto o *DEFAULT*. Inicialmente todos los recursos designados en el están con valor *UNLIMITED*, por lo que es conveniente modificarlo.
- Aquellos recursos para los que en el perfil asignado no se ha definido un valor, toman el valor designado en el perfil por defecto.
- Sintaxis:

```
CREATE PROFILE <perfil>
 LIMIT <parámetros> <valor>/UNLIMITED/DEFAULT;
```

CREACIÓN DE PERFILES.

- *Parámetros de recursos:*

  SESSIONS_PER_USER. Numero de sesiones concurrentes.

  CPU_PER_SESSION. Tiempo de UCP por sesión (centésimas de segundo).

  CPU_PER_CALL. Tiempo de UCP para una llamada (parse, execute, o fetch) en centésimas de segundo.

  CONNECT_TIME. Tiempo total para una sesión(minutos).

  IDLE_TIME. Tiempo de inactividad en una sesión (minutos).

CREACIÓN DE PERFILES.

- Parámetros de recursos:

  LOGICAL_READS_PER_SESSION. Numero de bloques de datos leídos en una sesión (memoria o disco).

  LOGICAL_READS_PER_CALL. Numero de bloques de datos para una llamada de una SQL (parse, execute, o fetch).

  PRIVATE_SGA. Cantidad de espacio, en bytes, para uso privado reservado en la "shared pool" de la SGA (se emplea K o M para indicar kilobytes o megabytes). Solo en "Shared Server".

  COMPOSITE_LIMIT. Coste total en recursos por sesión expresado en unidades de servicio *(CPU_PER_SESSION, CONNECT_TIME, LOGICAL_READS_PER_SESSION, y PRIVATE_SGA).*

CREACIÓN DE PERFILES.

- Parámetros de contraseña:

FAILED_LOGIN_ATTEMPTS. Numero de intentos fallidos de conexión antes del bloqueo.

PASSWORD_LIFE_TIME. Numero de días en que la clave es valida, esta expira si no se cambia en este periodo.

PASSWORD_REUSE_TIME . Numero de días en los cuales la contraseña no puede usarse.

PASSWORD_REUSE_MAX . Numero de cambios de clave necesarios antes de poder reusar la clave actual.

CREACIÓN DE PERFILES.

- Parámetros de contraseña:

  PASSWORD_LOCK_TIME. Numero de días que la cuenta estará bloqueada después de un cierto numero de fallos de conexión.

  PASSWORD_GRACE_TIME. Periodo de gracia donde se permite la conexión pero se notifica la necesidad de cambiarla.

CREACIÓN DE PERFILES.

- Valor *UNLIMITED*. En el caso de parámetros de recursos indica que puede usarse una cantidad ilimitada del mismo, en el caso de parámetros de contraseña que no ha sido fijado limite.

- Valor *DEFAULT*. Si se indica DEFAULT o se omite en el perfil algún parámetro, al ser asignado a un usuario toma para dicho parámetro el valor indicado en el perfil DEFAULT.

MODIFICACIÓN DE PERFILES.

- Es necesario poseer el privilegio de sistema "*ALTER PROFILE*".
- Los valores modificados no afectan a las sesiones en curso.
- Sintaxis:

```
ALTER PROFILE <perfil>
 LIMIT <parámetros> <valor>/UNLIMITED/DEFAULT;
```

ASIGNACIÓN DE PERFILES.

- Los perfiles no pueden asignarse a roles ni a otros perfiles, solo a usuarios.
- Se puede realizar durante la creación del usuario (*CREATE USER*) o posteriormente (*ALTER USER*).
- Un usuario sólo puede tener un perfil asignado a la vez.
- Las asignaciones de perfiles no afectan a las sesiones activas.

BORRADO DE PERFILES.

- Debe poseerse el privilegio de sistema *DROP PROFILE*.
- Para eliminar un perfil asignado a un usuario debe usarse la opción *CASCADE*.
- Si se borra un perfil asociado a un usuario, a este se le asigna de forma automática el perfil *DEFAULT*.
- El borrado de un perfil no afecta a las sesiones en curso.
- EL perfil *DEFAULT* no puede borrarse.

BORRADO DE PERFILES.

- Sintaxis:

DROP PROFILE <perfil>;

DROP PROFILE <perfil> CASCADE;

PRIVILEGIOS.

- Derecho a ejecutar un tipo particular de sentencia SQL o a acceder a un objeto de otro usuario.

- Se distinguen dos tipos:

  - De sistema: Permite realizar determinadas acciones en la base de datos (Por ejemplo, crear espacios de almacenamiento, crear usuarios, …).

  - Sobre Objetos: Permite a un usuario acceder y manipular o ejecutar objetos concretos (tablas, vistas, secuencias, procedimientos, funciones o paquetes).

PRIVILEGIOS DE SISTEMA.

- Pueden clasificarse en:
  - Privilegios sobre operaciones del sistema: *CREATE SESSION, CREATE TABLESPACE*, …
  - Privilegios que permiten el manejo de objetos en el esquema propio de un usuario: *CREATE TABLE, CREATE PROCEDURE*, …
  - Privilegios que permiten el manejo de objetos en cualquier esquema: *CREATE ANY TABLE, CREATE ANY INDEX*, …

PRIVILEGIOS DE SISTEMA.

DATABASE LINKS	*CREATE DATABASE LINK* *CREATE PUBLIC DATABASE LINK* *DROP PUBLIC DATABASE LINK*
PROFILES	*CREATE PROFILE* *ALTER PROFILE* *DROP PROFILE*
ROLES	*CREATE ROLE* *ALTER ANY ROLE* *DROP ANY ROLE*
ROLLBACK SEGMENTS	*CREATE ROLLBACK SEGMENT* *ALTER ROLLBACK SEGMENT* *DROP ROLLBACK SEGMENT*

PRIVILEGIOS DE SISTEMA.

TABLAS / INDICES	*CREATE ANY TABLE / CREATE ANY INDEX* *ALTER ANY TABLE / ALTER ANY INDEX* *DELETE ANY TABLE* *DROP ANY TABLE / DROP ANY INDEX* *INSERT ANY TABLE / SELECT ANY TABLE* *UPDATE ANY TABLE*
TABLESPACE	*CREATE TABLESPACE* *ALTER TABLESPACE* *DROP TABLESPACE* *UNLIMITED TABLESPACE*
USER	*CREATE USER* *ALTER USER* *DROP USER*

PRIVILEGIOS DE SISTEMA.

- La cláusula *ANY* en cualquier privilegio indica que los usuarios a los que se les conceda tienen dicho privilegio en cualquier esquema.

- Notas:

  - No existe el privilegio *CREATE INDEX*.
  - *CREATE TABLE* incluye las sentencias *CREATE INDEX* y *ANALYZE*.
  - Privilegios como *CREATE TABLE* o *CREATE PROCEDURE* incluyen el borrado de dichos objetos.
  - *UNLIMITED TABLESPACE* no puede otorgarse a un rol.

OTORGAR PRIVILEGIOS DE SISTEMA.

- Sintaxis:

  GRANT <privilegio>/ALL_PRIVILEGES TO <usuario>/<rol>/PUBLIC;

  GRANT <privilegio>/ALL_PRIVILEGES TO <usuario>/<rol>/PUBLIC WITH ADMIN OPTION;

- Al especificar *ALL PRIVILEGES* se otorgan todos los privilegios de sistema (¡Peligro!).

- La cláusula *PUBLIC* otorga el privilegio a todos los usuarios (¡Peligro!).

- La cláusula *ADMIN OPTION* permite a aquel a quien se le concede el privilegio poder otorgarlo (¡Peligro!).

RESTRICCIONES EN PRIVILEGIOS DE SISTEMA.

- El parámetro *O7_DICTIONARY_ACCESSIBILITY* permite restringir los privilegios de sistema. Impide el acceso al esquema SYS a través de los privilegios que conceden acceso a cualquier esquema.

- Si su valor es *TRUE*, se permite el acceso a los objetos del esquema *SYS*.

- Por defecto su valor es *FALSE;* en este caso, por ejemplo, *SELECT ANY TABLE* permite acceder a vistas y tablas en otros esquemas pero no seleccionar objetos del esquema SYS (diccionario de datos).

REVOCAR PRIVILEGIOS DE SISTEMA.

- Sintaxis:

  REVOKE *<privilegio>/ALL_PRIVILEGES* FROM *<usuario>/<rol>/PUBLIC;*

- Cualquier usuario con la opción *ADMIN OPTION* sobre un privilegio puede revocarlo. Quien lo hace no tiene porque ser el usuario que originalmente lo otorgo.

- Al retirar ciertos privilegios determinados objetos pueden quedar inconsistentes (procedimientos o vistas consultadas merced al privilegio *SELECT ANY TABLE*).

- En el caso de *ADMIN OPTION* no hay un efecto en cascada cuando se retira un privilegio.

PRIVILEGIOS SOBRE OBJETOS.

	TABLA	VISTA	SECUENCIA	PROCEDIMIENTO
ALTER	X		X	
DELETE	X	X		
EXECUTE				X
INDEX	X			
INSERT	X	X		
REFERENCES	X	X		
SELECT	X	X	X	
UPDATE	X	X		

OTORGAR PRIVILEGIOS SOBRE OBJETOS.

- Sintaxis:

  GRANT <privilegio>/ALL_PRIVILEGES ON <esquema>.objeto TO <usuario>/<rol>/PUBLIC;

  GRANT <privilegio>/ALL_PRIVILEGES ON <esquema>.objeto TO <usuario>/<rol>/PUBLIC WITH GRANT OPTION;

- Con *ALL PRIVILEGES* se otorgan todos los privilegios sobre el objeto (¡Peligro!).

- Con *PUBLIC* otorga el privilegio a todos los usuarios (¡Peligro!).

- La cláusula *GRANT OPTION* permite a aquel a quien se le concede el privilegio poder otorgarlo (¡Peligro!).

REVOCAR PRIVILEGIOS SOBRE OBJETOS.

- Sintaxis:

  REVOKE *<privilegio>/ALL_PRIVILEGES ON <esquema>.objeto FROM <usuario>/<rol>/PUBLIC <CASCADE CONSTRAINTS>;*

- *CASCADE CONSTRAINTS* elimina cualquier cláusula de integridad referencial que aquel a quien se retiran los permisos haya definido usando *REFERENCES* o *ALL PRIVILEGES*.

- Quien otorgo privilegios solo puede revocarlos a aquellos usuarios a quienes se los han concedido.

- En el caso de *GRANT OPTION* hay un efecto en cascada cuando se retira un privilegio.

ROLES.

- Es un grupo de privilegios, de sistema o sobre objetos, a los que se les da un nombre y pueden ser asignados a otros usuarios y roles.

- Características:

  - Pueden otorgarse a cualquier usuario o rol, pero no a si mismo y tampoco de forma circular.

  - Pueden habilitarse o deshabilitarse.

  - Pueden tener contraseña.

  - Su nombre es único.

  - No pertenecen a ningún esquema.

ROLES. BENEFICIOS.

- Simplifican el manejo de privilegios. Los permisos pueden asignarse a un rol y este a los diferentes usuarios.

- Manejo de privilegios dinámico. Si se modifican los privilegios asociados al rol, todos los usuarios que lo posean los adquieren de forma inmediata.

- Disponibilidad de privilegios selectiva. Roles pueden ser activados o desactivados temporalmente.

- Mejora de la productividad. El uso de roles disminuye el numero de "grants" almacenados en el diccionario de datos. Si se desactivan roles, hay menos privilegios que verificar durante la ejecución de una sentencia.

ROLES PREDEFINIDOS.

- Oracle proporciona roles predefinidos como ayuda a la administración de base de datos. Entre otros:
  - ***CONNECT***. Incluye los privilegios *ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE y CREATE VIEW*.
  - ***RESOURCE***. Incluye *CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER y CREATE TYPE.*
  - ***DBA***. Todo privilegio de sistema *WITH ADMIN OPTION*.
  - ***EXP_FULL_DATABASE***. Privilegios para exportaciones completas e incrementales de la base de datos.

ROLES PREDEFINIDOS.

- ***IMP_FULL_DATABASE***. Idem para importaciones completas.
- *DELETE_CATALOG_ROLE*. Privilegio de borrado en la tabla de auditoría de sistema (AUD$).
- *EXECUTE_CATALOG_ROLE*. Privilegio de ejecución sobre objetos en el diccionario de datos.
- *SELECT_CATALOG_ROLE*. Privilegio de consulta sobre objetos del diccionario de datos.

- Los roles *CONNECT, RESOURCE* y *DBA* se mantienen por compatibilidad con versiones anteriores de Oracle. No se asegura que sigan existiendo en un futuro.

CREACIÓN DE ROLES.

- Debe poseerse el privilegio CREATE ROLE.
- El nombre debe ser diferente a cualquier nombre de rol o usuario existente.
- Sintaxis:

  CREATE ROLE <rol> IDENTIFIED BY <contraseña>;

  CREATE ROLE <rol> NOT IDENTIFIED/<>;

- La cláusula IDENTIFIED BY indica como debe ser autorizado antes de usarse por un usuario al que se la ha otorgado (innecesario si es un rol por defecto y activo en el momento de conexión del usuario).

MODIFICACIÓN DE ROLES.

- Un rol solo puede modificarse para cambiar su método de autentificación.
- Debe poseerse el privilegio de sistema *ALTER ANY ROLE* o haber sido sido otorgado el rol con la opción *ADMIN.*
- Sintaxis:

ALTER ROLE <rol> NOT IDENTIFIED/ IDENTIFIED BY <contraseña>;

ASIGNAR ROLES A USUARIOS.

- *Sintaxis:*

  GRANT <rol> TO <usuario>/<rol>/PUBLIC;

  GRANT <rol> TO <usuario>/<rol>/PUBLIC WITH ADMIN OPTION;

- El usuario que crea el rol implícitamente lo tiene asignado con ADMIN OPTION.

- Si a un usuario no se le otorga un rol con ADMIN OPTION; necesita el privilegio GRANT ANY ROLE para otorgarlo, o haberlo creado.

PARÁMETRO MAX_ENABLED_ROLES.

- EL numero máximo de roles de base de datos activos, incluyendo aquellos contenidos dentro de otros roles, que un usuario puede poseer viene dado por el parámetro de inicialización *MAX_ENABLED_ROLES*.

ROLES POR DEFECTO.

- Un rol por defecto es aquel que automáticamente se activa al conectarse.
- Con la sentencia ALTER USER se limitan los roles por defecto asignados a un usuario.
- Sintaxis:

```
ALTER USER <usuario> DEFAULT ROLE <rol1>,…<roln>/
ALL [EXCEPT rol1 [,role2]... ] /          NONE;
```

ROLES POR DEFECTO.

- La cláusula DEFAULT ROLE se aplica solo a los roles otorgados de forma directa, y no para roles no asignados al usuario o asignados a través de otros roles.

- ALL hace que todos los roles sean por defecto excepto aquellos indicados en la cláusula EXCEPT.

- EXCEPT indica que los roles que le siguen no serán por defecto.

- NONE hace que ninguno de los roles sea por defecto, y los únicos privilegios al efectuarse la conexión serán aquellos asignados directamente.

DEASIGNACIÓN DE ROLES.

- Puede hacerlo cualquier usuario con la opción *ADMIN OPTION* para un rol, también aquellos usuarios con el privilegio *GRANT ANY ROLE*.

- Sintaxis:

```
REVOKE <rol1>, …<roln>
FROM <usuario>|<rol>|PUBLIC
[, <usuario>|<rol>} ]...
```

- Con *PUBLIC* se deasigna el rol de todos los usuarios.

BORRADO DE ROLES.

- Debe poseerse el privilegio *DROP ANY ROLE* o haber sido concedido el rol con *ADMIN OPTION*.
- Sintaxis:

  DROP ROLE <rol>;

- Al borrar un rol se deasigna de todos los usuarios y roles, y se elimina de la base de datos.

ACTIVACIÓN Y DESACTIVACIÓN DE ROLES.

- Durante una sesión, el usuario o una aplicación puede utilizar la sentencia *SET ROLE* para modificar los roles activos en la sesión.
- Previamente los roles deben haber sido asignados al usuario.
- No podrá hacerse uso de los privilegios otorgados a través del rol inactivo a menos que también se hayan otorgado de forma directa o a través de otros roles.
- En la siguiente sesión, los roles activos vuelven a ser los roles por defecto.

ACTIVACIÓN Y DESACTIVACIÓN DE ROLES.

- La vista *SESSION_ROLES* informa de aquellos roles que, para el usuario actual, están activados en un momento determinado.
- El número de roles activos a la vez está limitado por el parámetro *MAX_ENABLED_ROLES*.
- Al crear un usuario, todos los roles asignados son por defecto; a menos que se limite con *ALTER USER*.

ACTIVACIÓN Y DESACTIVACIÓN DE ROLES.

- *Sintaxis:*

  SET ROLE <rol> [IDENTIFIED BY <contraseña>]
  [, <rol> [IDENTIFIED BY <contraseña>]]…/
  ALL [EXCEPT <rol1> , … ,<roln>] ...]
  /NONE

- *IDENTIFIED BY* indica la contraseña del rol al activarlo.

- *ALL* activa todos los roles excepto los que aparecen en la cláusula *EXCEPT* (no puede usarse esta opción para activar roles con contraseña).

- *NONE* desactiva todos los roles en la sesión (solo son activos los privilegios otorgados directamente).

VISTAS ESTÁTICAS Y DINÁMICAS.

DBA_USERS	*Usuarios de la base de datos.*
DBA_ROLES	*Roles existentes en la base de datos.*
DBA_ROLE_PRIVS	*Roles concedidos a usuarios y roles.*
DBA_SYS_PRIVS	*Privilegios de sistema a usuarios y roles.*
DBA_TAB_PRIVS	*Permisos sobre objetos en la base de datos.*
DBA_TS_QUOTAS	*Cuotas de espacio para usuarios.*
DBA_PROFILES	*Perfiles en la base de datos.*
ROLE_ROLE_PRIVS	*Roles concedidos a otros roles.*

VISTAS ESTÁTICAS Y DINÁMICAS.

ROLE_SYS_PRIVS	*Privilegios de sistema concedidos a roles.*
ROLE_TAB_PRIVS	Privilegios sobre objetos concedidos a roles.
USER_PASSWORD_LIMITS	Parámetros de contraseña asignados al usuario.
USER_RESOURCE_LIMITS	Limites de recursos para el usuario activo.
SESSION_PRIVS	Privilegios disponibles en la sesión.
SESSION_ROLES	Roles activos en la sesión.
V$SESSION	Información de sesión.

TEMA 8.

TRABAJOS (JOBS).

TEMA 8.
TRABAJOS (JOBS).

- Trabajos y procesos.
- Procesos.
- Paquete dbms_job.
- Paquete dbms_job. Procedimiento “submit”.
- Trabajos. Ejecución y bloqueos. Errores de ejecución.
- Paquete dbms_job. Procedimiento “remove”.
- Paquete dbms_job. Modificación de un trabajo.

TEMA 8.
TRABAJOS (JOBS).

- Paquete dbms_job. Procedimiento “change”.
- Paquete dbms_job. Procedimiento “broken”.
- Paquete dbms_job. Procedimiento “run”.
- ¿Cómo matar un trabajo en ejecución?.
- Vistas.

TRABAJOS Y PROCESOS.

- Los trabajos son código PL/SQL a ejecutar de forma periódica. Se lanzan a la cola de trabajos, especificando la frecuencia con que deben ser ejecutados.

- Existe un proceso coordinador en "background" (ora_cjq0_xxxx), que selecciona los trabajos a ejecutar, los ordena y genera de forma dinámica los procesos de "job", Jnnn, en número suficiente para ejecutar los trabajos en cola.

- Cada trabajo es ejecutado en un momento dado por un sólo proceso.

PROCESOS.

- Se regulan por el parámetro (init.ora) siguientes:

  - *JOB_QUEUE_PROCESSES*. Parámetro dinámico que indica el máximo número de procesos (J000 ... J999) que pueden crearse para la ejecución de trabajos, - también se usan para el refresco de datos en entornos con réplica.

    Si su valor es cero, no existe proceso coordinador.

- EL máximo número de procesos Jnnn puede variarse de forma dinámica con la sentencia:

  ALTER SYSTEM SET JOB_QUEUE_PROCESSES = <numero>;

PAQUETE DBMS_JOB.

- Para gestionar la cola de trabajos se dispone del paquete *DBMS_JOB*.
- Cualquier usuario que pueda ejecutar los procedimientos contenidos en el paquete *DBMS_JOB* puede gestionar trabajos en la cola.
- No hay privilegios de base de datos asociados con el uso de trabajos.
- *DBMS_JOB* no permite a un usuario gestionar ningún trabajo excepto los propios.

PAQUETE DBMS_JOB. PROCEDIMIENTO "SUBMIT".

- Usado para lanzar un nuevo trabajo a la cola; el usuario que lo hace es identificado como su propietario.

- Se guarda la información de entorno siguiente:

  - *Usuario actual.*
  - *Usuario que lanza o modifica el trabajo.*
  - *Esquema actual (puede ser diferente si se ha ejecutado ALTER SESSION SET CURRENT_SCHEMA).*
  - *NLS_LANGUAGE - NLS_TERRITORY*
  - *NLS_CURRENCY - NLS_ISO_CURRENCY*
  - *NLS_DATE_FORMAT - NLS_DATE_LANGUAGE*
  - *NLS_SORT - NLS_NUMERIC_CHARACTERS*

- El entorno se restaura al ejecutar el trabajo.

PAQUETE DBMS_JOB. PROCEDIMIENTO "SUBMIT".

- Sintaxis:

```
DBMS_JOB.SUBMIT (:numero_trabajo,
                 código_del_trabajo,
                 next_date,
                 intervalo,
                 no_parse);
```

PAQUETE DBMS_JOB. PROCEDIMIENTO "SUBMIT".

JOB	Parámetro de salida (numérico). Es el identificador del trabajo creado, se usará para cualquier operación posterior con el mismo y no cambia. Se genera automáticamente a partir de la secuencia *SYS.JOBSEQ* y no cambia una vez asignado.
WHAT	Código PL/SQL a ejecutar (el cuerpo del trabajo). Generalmente es una llamada a un procedimiento con un numero indeterminado de parámetros. Las cadenas deben ir entrecomilladas (dos comillas simples) y la definición terminar con punto y coma. No puede ejecutarse un"job" dentro de otro "job", se genera el error ORA-32317.

PAQUETE DBMS_JOB. PROCEDIMIENTO "SUBMIT".

NEXT_DATE	Fecha siguiente en que se ejecutara el trabajo. Por defecto *SYSDATE*.
INTERVAL	Función de fecha que calcula la siguiente vez en que se ejecutara el trabajo. Por defecto *NULL*.
NO_PARSE	Parámetro booleano. Si su valor es *FALSE*, por defecto, se analiza el procedimiento asociado al trabajo. Si es *TRUE* se hará la primera vez que se ejecute. Por ejemplo, cuando se envia un trabajo antes de crear las tablas asociadas.

PAQUETE DBMS_JOB. PROC."SUBMIT": INTERVALO.

- Parámetro *INTERVAL* (intervalo de ejecución).
  - Se evalúa antes de ser ejecutado un trabajo; si este termina de forma satisfactoria, la fecha que se obtiene a partir de *INTERVAL* se convierte en el siguiente valor para *NEXT_DATE*.
  - Si su valor es *NULL* y el trabajo termina sin problemas, el trabajo se elimina de la cola de trabajos.

PAQUETE DBMS_JOB. PROC."SUBMIT": INTERVALO.

Expresión	*Significado*
'SYSDATE + 7'	Siete días después de la ultima ejecución.
'SYSDATE + 1/48'	Cada media hora.
'NEXT_DAY(TRUNC(SYSDATE), ' 'MONDAY' ') + 15/24'	Todos los lunes a las 15:00 horas.
'SYSDATE + 30/1440'	Cada treinta minutos.
'SYSDATE + 15/1440'	Cada quince minutos.
ADD_MONTHS(SYSDATE,1)	Un mes después.

TRABAJOS. EJECUCION.

- Los procesos Jnnn son los encargados de crear la sesión oportuna para ejecutar un trabajo. El entorno de ejecución es el mismo al existente cuando se lanzó y con los privilegios por defecto del propietario.

- Al propietario deben haberse asignado explícitamente los privilegios necesarios para los objetos referenciados en el trabajo (no a través de roles).

TRABAJOS. BLOQUEOS.

- Oracle sólo permite que un trabajo se ejecute en una sesión en un momento determinado. Al ejecutarse, su sesión adquiere un "queue lock" para el mismo (identificado con *JQ*).

- Mediante la vista V$LOCK puede identificarse las sesiones con este tipo de bloqueo.

TRABAJOS.
ERRORES DE EJECUCION.

- Cuando un trabajo falla se registra el error en el fichero de alertas y en ficheros de traza, incluye el número de trabajo:

> *ORA-12012 error on auto execute of job string*
>
> *Cause: Some kind of error was camught while doing an automatic execute of a job.*
>
> *Action: Look at the accompanying errors for details on why the execute failed.*

- Puede deberse a un fallo de red, de instancia, o excepción al ejecutarse.
- Si falla, se intenta ejecutar de forma repetida (intervalos de 1, 2, 4, 8 ... minutos). Si se alcanzan 16 intentos fallidos, el trabajo se marca como "broken" y no se ejecuta de nuevo.

PAQUETE DBMS_JOB. PROCEDIMIENTO "REMOVE".

- Se usa para eliminar un trabajo de la cola de trabajos.
- Puede eliminarse un trabajo que este en ejecución pero no se vera interrumpido hasta su terminación.
- Solo pueden eliminarse trabajos de los que se sea propietario. Si se intenta eliminar un trabajo que no se posee, se recibe un mensaje de error.
- Sintaxis:

DBMS_JOB.REMOVE (numero_trabajo);

PAQUETE DBMS_JOB. MODIFICACION DE UN TRABAJO.

- Solo pueden modificarse trabajos de los que se sea propietario. Si se intenta modificar un trabajo que no se posee, se recibe un mensaje de error.

- Con el procedimiento *DBMS_JOB.CHANGE* pueden modificarse cualesquiera de los parámetros asociados con un trabajo.

- Si se indica *NULL* para los campos código, next_date e interval permanecen sin cambios.

PAQUETE DBMS_JOB. PROCEDIMIENTO "CHANGE".

- Sintaxis:

DBMS_JOB.CHANGE (numero_trabajo,
código_del_trabajo,
next_date,
intervalo);

- Existen procedimientos para modificar parámetros específicos asociados a un trabajo:

DBMS_JOB.WHAT (numero_trabajo, código_del_trabajo);
DBMS_JOB.NEXT_DATE (numero_trabajo, next_date);
DBMS_JOB.INTERVAL (numero_trabajo, intervalo);

PAQUETE DBMS_JOB. PROCEDIMIENTO "CHANGE".

- Al modificar el código de un trabajo mediante el procedimiento *DBMS_JOB.CHANGE*, o mediante *DBMS_JOB.WHAT*, se registra el entorno actual, y pasa a ser el nuevo entorno para el trabajo modificado.

PAQUETE DBMS_JOB. PROCEDIMIENTO "BROKEN".

- Un trabajo deshabilitado o "broken" no es ejecutado. Este estado se adquiere tras fallar en su ejecución un total de 16 intentos, o tras marcarlo como "broken" mediante el procedimiento *DBMS_JOB.BROKEN*.

- Solo pueden marcarse trabajos de los que se es propietario.

  DBMS_JOB.BROKEN (numero_trabajo, "broken", next_date);

- EL valor de "broken" puede ser *TRUE* o *FALSE*.

PAQUETE DBMS_JOB. PROCEDIMIENTO "RUN".

- Una vez marcado como "broken" puede volver a ejecutarse:
  - Marcándolo como no "broken" y esperando que sea ejecutado.
  - Forzando su ejecución con *DBMS_JOB.RUN*. En este caso se ejecuta de forma immediate, si tiene éxito se marca como "no broken" y se pone a cero la cuenta de fallos.
- Sintaxis:

```
DBMS_JOB.RUN (numero_trabajo);
```

PAQUETE DBMS_JOB. PROCEDIMIENTO "RUN".

- Solo puede forzarse la ejecución de trabajos de los que se es propietario. En caso contrario se recibe un error.
- El procedimiento contiene un "commit" implícito, de forma que no es posible hacer "rollback".
- Al forzar la ejecución el trabajo se comporta de acuerdo a los intervalos de ejecución fijados para el trabajo.

¿CÓMO MATAR UN TRABAJO EN EJECUCION?.

- Un trabajo en ejecución puede finalizarse de la siguiente forma:
  - Marcarlo como “broken”.
  - Identificar la sesión que ejecuta el trabajo (vistas *V$SESSION* o *V$LOCK*).
  - Eliminar la sesión con la sentencia “*ALTER SYSTEM KILL SESSION …*”

VISTAS.

- DBA_JOBS. Trabajos en la base de datos.
- USER_JOBS. Trabajos que pertenecen al suuario actual.
- DBA_JOBS_RUNNING. Trabajos en ejecución.
- V$LOCK. Bloqueos mantenidos por el servidor.
- V$SESSION. Información de sesiones actuales.

TEMA 9.

AUDITORÍA.

TEMA 9.
AUDITORÍA.

- Auditoría.
- Tipos de auditoría.
- Registros de auditoría. “Audit trail”.
- Auditoría de usuarios administradores. “Audit trail” de sistema operativo.
- Auditoría. Sentencia “audit”.
- Auditoría de sentencias. Opciones.
- Auditoría de privilegios. Privilegios auditables.

TEMA 9.
AUDITORÍA.

- Auditoría de esquema. Opciones.
- Desactivación. Sentencia noaudit.
- Desactivación de la auditoría.
- Control del “audit trail”.
- Protección del “audit trail”.
- Auditoría. Recomendaciones.
- Interpretación del “audit trail”. Vistas estáticas.

AUDITORÍA.

- Mediante la auditoría se intenta monitorizar y registrar acciones en la base de datos con el fin de :
  - Investigar actividades maliciosas (borrado de tablas, ..)
  - Recoger datos sobre actividades concretas (tablas que se actualizan, usuarios concurrentes, …)
- Puede ser más o menos general, permitiendo auditar:
  - Ejecuciones de sentencias exitosas, fallidas o ambas.
  - Ejecución de sentencias por sesión o por lanzamiento.
  - Usuarios concretos o todos los usuarios.

TIPOS DE AUDITORÍA.

- Existen tres tipos:

  - De sentencias. Seleccionando un tipo concreto de las mismas, que afectan a una determinada clase de objetos de base de datos (por ejemplo, *AUDIT TABLE*).

  - De privilegios. Auditoría de privilegios de sistema (por ejemplo, *AUDIT CREATE TABLE*).

  - De esquema. Sentencias específicas sobre objetos de un esquema concreto (p. ej. *AUDIT SELECT ON <nombre_tabla>*).

REGISTROS DE AUDITORÍA. "AUDIT TRAIL".

- La información de auditoría se almacena en los registros de auditoría, que incluyen datos tales como:
  - Usuario.
  - Identificador de sesión y terminal.
  - Nombre del esquema accedido.
  - Operación.
  - Código de operación.
  - Fecha y hora.
- Los registros se guardan en la tabla SYS.AUD$ ("audit trail"). Está codificada y no es legible; existen diferentes vistas que permiten usar la información almacenada.

REGISTROS DE AUDITORÍA. "AUDIT TRAIL".

- El registro de información puede estar habilitado o deshabilitado. Aquellos usuarios autorizados de la base de datos pueden determinar las opciones de auditoría, pero el decidir si se graba o no información pertenece al administrador.

- Cuando la auditoría está habilitada se genera un registro durante la fase de la ejecución de la sentencia.

- La generación e inserción de un registro de auditoría es independiente de la transacción del usuario; si esta es deshecha ("rollback"), el registro de auditoría permanece ("commit").

REGISTROS DE AUDITORÍA. "AUDIT TRAIL".

- El "audit trail" no almacena información sobre los valores de los datos que pudieran estar involucrados en una determinada sentencia que está siendo auditada. Por ejemplo, los valores actuales y anteriores de una fila modificada no se guardan cuando se audita la sentencia *UPDATE (*puede hacerse usando disparadores de base de datos, "*triggers*").

REGISTROS DE AUDITORÍA. "AUDIT TRAIL".

- Independientemente de si la auditoría esta habilitada, siempre se registran algunos tipos de acciones que son escritos a ficheros de sistema operativo (ficheros en *$ORACLE_HOME/rdbms/audit*):
  - "Startup" de la instancia. Se almacena el usuario de sistema operativo que lo hace, el identificador de terminal del usuario, fecha y hora, y estado de la auditoría (activa o no, ayuda a detectar cuando un administrador rearranca la base de datos con la auditoría deshabilitada).
  - "Shutdown" de la instancia. Almacena el usuario de sistema operativo, el identificador de terminal del usuario, y fecha y hora.
  - Conexiones a la base de datos con privilegios de administrador.

AUDITORÍA.
PARAMETRO "AUDIT_TRAIL".

- Para comenzar a auditar debe asignarse al parámetro de inicialización *AUDIT_TRAIL* el valor *DB* (auditoría en base de datos). El valor *NONE*, valor por defecto, deshabilita la auditoría.

  El valor *OS* indica que la auditoría debe llevarse a sistema operativo. El parámetro *AUDIT_FILE_DEST* señala donde se guardan los ficheros, así como los registros de auditoría para SYS (por defecto *$ORACLE_HOME/rdbms/audit).*

- Para auditar una sentencia SQL o privilegio debe poseerse el privilegio de sistema "*AUDIT SYSTEM*".

- Para auditar operaciones sobre un objeto, debe pertenecer al esquema o tener privilegio "*AUDIT ANY*".

AUDITORÍA.
PARAMETRO "AUDIT_TRAIL".

- El almacenamiento de la auditoría en base de datos tiene diferentes ventajas:
  - Pueden usarse ciertas vistas predefinidas, existentes en el diccionario de base de datos, para consultar de forma sencilla el "audit trail".
  - Pueden usarse herramientas Oracle de generación de informes, como Oracle Reports.
- Por otra parte, la auditoría en sistema operativo puede ayudar a examinar la actividad global del sistema con mayor facilidad; al estar todos los registros de auditoría (de Oracle y de otras herramientas) en un mismo lugar.

AUDITORIA DE USUARIOS ADMINISTRADORES.

- El parámetro de inicialización *AUDIT_SYS_OPERATIONS* permite especificar la auditoría de aquellas sesiones de usuarios conectados como SYS. Los registros generados son escritos en el "audit trail" de sistema operativo.

- Si su valor es TRUE (*AUDIT_SYS_OPERATIONS=TRUE)*, se auditan dichas operaciones.

- Si su valor es FALSE, valor por defecto, no son auditadas (*AUDIT_SYS_OPERATIONS=FALSE).*

INFORMACION "AUDIT TRAIL" DE SISTEMA OPERATIVO.

- Los registros de auditoría escritos a ficheros de sistema operativo pueden contener información codificada que debe descifrarse:
  - Código de operación ("action code"). Debe consultarse la tabla *AUDIT_ACTIONS*.
  - Privilegios. Privilegios de sistema usados para realizar la acción. Consultar la tabla *SYSTEM_PRIVILEGE_MAP*.
  - Terminación ("completion code"). Describe el resultado; si hubo éxito se devuelve un valor cero, en caso contrario un código de error.

AUDITORÍA DE SENTENCIAS.

- Sintaxis:

  AUDIT <sentencia1, … sentencian>/ALL
  BY <usuario1, … usuarion>
  BY SESSION/ACCESS
  WHENEVER SUCESSFUL/NOT SUCCESSFUL;

- La cláusula *BY <usuario>* permite restringir la auditoría sólo a aquellas sentencias ejecutadas por los usuarios indicados.

- La cláusula *BY SESSION*, clausula por defecto, indica que se desea un sólo registro para todas las sentencias SQL y operaciones del mismo tipo ejecutadas en la misma sesión.

AUDITORÍA DE SENTENCIAS.

- La cláusula *BY ACCESS* indica que se desea un registro para cada sentencia SQL y operación auditadas. Si se auditan sentencias de definición de datos del lenguaje (DDL) siempre se audita por acceso.

- La cláusula *WHENEVER SUCESSFUL,* permite auditar sólo sentencias SQL y operaciones que surten efecto.

- La cláusula *WHENEVER NOT SUCESSFUL,* permite auditar sólo sentencias SQL y operaciones que fallan o generan errores.

- Si se omiten las dos opciones anteriores, se realiza la auditoría independientemente del éxito o fallo de la sentencia.

AUDITORÍA DE SENTENCIAS. OPCIONES.

Opcion	Sentencias SQL auditadas.
Database link	Create database link / drop database link
Index	Create index / alter index /drop index
Not exists	Todas las sentencias SQL que fallan por no existir un determinado objeto
Procedure	Create function / create package / create package body / create procedure / drop function / drop package / drop procedure
Public database link	Create public database link / drop public database link
Public synonym	Create public synonym / Drop public synonym
Role	Create role / alter role / drop role / set role

AUDITORÍA DE SENTENCIAS. OPCIONES.

Opción	Sentencias SQL auditadas.
Rollback Statement	Create rollback segment/ alter rollback segment / drop rollback segment
Sequence	Create sequence / drop sequence
Session	Conexiones - *valor por defecto y único BY SESSION* -
Synonym	Create synonym / drop synonym
System audit	Audit sentencias_sql / Noaudit sentencias_sql
System grant	Grant y revoke privilegios_sistema y roles
Table	Create table / drop table / truncate table
Tablespace	Create tablespace / drop tablespace / alter tablespace

AUDITORÍA DE SENTENCIAS. OPCIONES.

Opción	Sentencias SQL auditadas.
Trigger	Create trigger / drop trigger /alter trigger
User	Create user / alter user /drop user
View	Create view /drop view
Alter table	Alter table
Delete table	Delete from <tabla>, <vista>
Grant procedure	Grant / revoke <privilegio> on <procedimiento, funcion, paquete>
Grant sequence	Grant / revoke <privilegio> on <secuencia>
Grant table	Grant / revoke <privilegio> on <tabla, vista, vista materializada>

AUDITORÍA DE SENTENCIAS. OPCIONES.

Opción	Sentencias SQL auditadas.
Insert table	Insert into <tabla, vista>
Lock table	Lock table <tabla, vista>
Select sequence	Cualquier sentencia que contenga sequence.CURRVAL o sequence.NEXTVAL
Select table	Select from <tabla, vista, vista materializada>
Update table	Update <tabla, vista>

AUDITORÍA DE PRIVILEGIOS.

- Sintaxis:

AUDIT <priv_sistema1, … priv_sisteman>
/ALL PRIVILEGES
BY <usuario1, … usuarion>
BY SESSION/ACCESS
WHENEVER SUCESSFUL/NOT SUCCESSFUL;

- La cláusula *ALL PRIVILEGES,* indica que debe auditarse todo privilegio de sistema.

- Es posible indicar los roles *CONNECT, RESOURCE o DBA*, para auditar los privilegios de sistema asociados.

PRIVILEGIOS AUDITABLES.

ALTER DATABASE	*CREATE PROFILE*
ALTER SYSTEM	*ALTER PROFILE*
AUDIT SYSTEM	*DROP PROFILE*
CREATE DATABASE LINK	*CREATE ROLE*
CREATE PUBLIC DATABASE LINK	*ALTER ANY ROLE*
DROP PUBLIC DATABASE LINK	*DROP ANY ROLE*
CREATE PROCEDURE	*CREATE ROLLBACK SEGMENT*
CREATE ANY PROCEDURE	*ALTER ROLLBACK SEGMENT*
ALTER ANY PROCEDURE	*DROP ROLLBACK SEGMENT*
DROP ANY PROCEDURE	
EXECUTE ANY PROCEDURE	*CREATE SESSION*
	ALTER SESSION

PRIVILEGIOS AUDITABLES.

CREATE ANY TABLE / ANY INDEX	*CREATE USER*
ALTER ANY TABLE / ANY INDEX	*ALTER USER*
DELETE ANY TABLE	*DROP USER*
DROP ANY TABLE / ANY INDEX	
INSERT ANY TABLE	*CREATE VIEW*
UPDATE ANY TABLE	*CREATE ANY VIEW*
SELECT ANY TABLE	*DROP ANY VIEW*
	ANALYZE ANY
	AUDIT ANY
	COMMENT ANY TABLE

AUDITORÍA DE ESQUEMA.

- Sintaxis:

  AUDIT <clausula_objeto1, … clausula_objeton>/ALL
  ON <esquema>.objeto_auditado/DEFAULT
  BY SESSION/ACCESS
  WHENEVER SUCESSFUL/NOT SUCCESSFUL;

- La cláusula *ALL* indica todas las opciones posibles sobre un tipo de objeto concreto.

- Mediante la cláusula *ON DEFAULT* se establece por defecto las opciones indicadas para todo objeto creado con posterioridad. Al establecerlas permanecen aquellas definidas por defecto para objetos creados previamente (pueden cambiarse indicando explícitamente el objeto).

AUDITORÍA DE ESQUEMA. OPCIONES.

	TABLA	VISTA	SECUENCIA	PROCEDIMIENTO
ALTER	X		X	
AUDIT	X	X	X	X
COMMENT	X	X		
DELETE	X	X		
EXECUTE				X
GRANT	X	X	X	X
INDEX	X			
INSERT	X	X		

AUDITORÍA DE ESQUEMA. OPCIONES.

	TABLA	VISTA	SECUENCIA	PROCEDIMIENTO
LOCK	X	X		
RENAME	X	X		X
SELECT	X	X	X	
UPDATE	X	X		

DESACTIVACIÓN. SENTENCIA NOAUDIT.

- Para desactivar la auditoría de una sentencia es necesario poseer el privilegio de sistema "*AUDIT SYSTEM*".

- Para detener la auditoría sobre un objeto, debe pertenecer al esquema o tener el privilegio "*AUDIT ANY*".

- La sintaxis de *NOAUDIT* es igual a la de *AUDIT* (para cada sentencia de auditoría es necesaria una sentencia *NOAUDIT* que la deshabilite).

DESACTIVACIÓN. SENTENCIA NOAUDIT.

- Sintaxis:

NOAUDIT <sentencia1, … sentencian>/ALL
BY <usuario1, … usuarion>
WHENEVER SUCESSFUL/NOT SUCCESSFUL;

NOAUDIT <priv_sistema1, … priv_sisteman>
/ALL PRIVILEGES
BY <usuario1, … usuarion>
WHENEVER SUCESSFUL/NOT SUCCESSFUL;

NOAUDIT <clausula_objeto1, … clausula_objeton>/ALL
ON <esquema>.objeto_auditado/DEFAULT
WHENEVER SUCESSFUL/NOT SUCCESSFUL;

DESACTIVACIÓN DE LA AUDITORÍA.

- Mediante la sentencia *NOAUDIT* sólo se indican opciones de auditoría. Para deshabilitar la auditoría debe modificarse el parámetro de inicialización *AUDIT_TRAIL*.

CONTROL DEL “AUDIT TRAIL”.

- Si el “audit trail” se llena de forma que no pueden insertarse más registros, las sentencias auditadas no pueden ejecutarse correctamente hasta que se vacía, y se generan errores.

- El tamaño máximo de SYS.AUD$ depende de los parámetros de almacenamiento, “default storage”, del espacio SYSTEM. Pueden modificarse los parámetros para SYS.AUD$.

- No es posible crear la tabla SYS.AUD$ en un espacio de almacenamiento distinto al SYSTEM.

CONTROL DEL “AUDIT TRAIL”.

- La gestión del “audit trail”, el único objeto de SYS directamente modificable, es similar a la de otra tabla; pueden borrarse registros con la sentencia *DELETE*:

  - *DELETE FROM SYS.AUD$;*

  - *DELETE FROM SYS.AUD$ WHERE OBJ$NAME= <nombre_objeto>;*

CONTROL DEL “AUDIT TRAIL”.

- Después del borrado de registros, las extensiones adquiridas para esta tabla permanecen. Este espacio puede reducirse:

  - Copiar el “audit trail” a otra tabla o exportarlo (utilidad *EXPORT*).

  - Conectarse como usuario administrador y truncar SYS.AUD$.

  - Cargar los datos anteriormente salvados que interese mantener accesibles.

PROTECCIÓN DEL "AUDIT TRAIL".

- Debe protegerse de borrados no autorizados:

  - Otorgar el privilegio *DELETE ANY TABLE* solo a usuarios administradores.

  - Auditar cualquier cambio que se realice en el "audit trail" mediante la sentencia

AUDIT INSERT, UPDATE, DELETE ON SYS.AUD$ BY ACCESS;

AUDITORÍA. RECOMENDACIONES.

- Es necesario seguir una serie de reglas a la hora de auditar la actividad de la base de datos:

  - Limitar el numero de acciones auditadas y el tiempo durante el que se hará. Así disminuye el impacto de la auditoría sobre las sentencias supervisadas y se limita el tamaño del "audit trail" (¿qué debo o quiero auditar?).

  - Evaluar el propósito y planear una estrategia (¿para qué y por qué audito?, ¿qué actividad maliciosa he detectado?).

  - Si se audita debido a la sospecha de alguna acción maliciosa; debe comenzarse por auditar acciones de tipo general para, una vez analizada la información, pasar a auditar acciones mas concretas.

AUDITORÍA. RECOMENDACIONES.

- Proteger el “audit trail”, de forma que la infomacion de auditoría no pueda ser añadida, modificada o borrada sin ser registrada la operación.

- Controlar de forma estricta quien tiene derecho a auditar.

- En caso de que se desee recoger información histórica sobre determinadas operaciones debe auditarse sólo aquellas acciones que sean pertinentes; y preocuparse de guardar los registros de auditoría de interés y eliminar periódicamente del “audit trail” esta informacion.

INTERPRETACIÓN DEL "AUDIT TRAIL". VISTAS ESTÁTICAS.

- *AUDIT_OPTIONS. Descripciones para los códigos de tipos de acción.*
- *ALL_DEF_AUDIT_OPTS. Opciones por defecto de auditoría de objetos que serán aplicadas al crearlos.*
- *DBA_AUDIT_EXISTS. Registros producidos por AUDIT NOT EXISTS.*
- *DBA_AUDIT_OBJECT. Registros para todos los objetos en el sistema.*
- *DBA_AUDIT_SESSION. Registros relativos a conexiones y desconexiones.*

INTERPRETACIÓN DEL "AUDIT TRAIL". VISTAS ESTÁTICAS.

- *DBA_AUDIT_STATEMENT. Registros para las sentencias GRANT, REVOKE, AUDIT, NOAUDIT, y ALTER SYSTEM.*
- *DBA_AUDIT_TRAIL. Registros de "audit trail".*
- *DBA_OBJ_AUDIT_OPTS. Opciones de auditoría para todos los objetos.*
- *DBA_PRIV_AUDIT_OPTS. Privilegios de sistema auditados.*
- *DBA_STMT_AUDIT_OPTS. Opciones de auditoría por sentencia.*

TEMA 10.

COPIAS DE SEGURIDAD.

TEMA 10.
COPIAS DE SEGURIDAD.

- Copia física.
- Optimal flexible architecture (O.F.A.).
- O.F.A. En entornos unix.
- Copia física. Sistemas de ficheros.
- Copia física. Comando tar (unix).
- Utilidades export/import.
- Export. Modo directo.

TEMA 10.
COPIAS DE SEGURIDAD.

- Casos prácticos export.
- Utilidad import.
- Casos prácticos “import”.
- Sql*loader.
- Sql*loader. Fichero de control y ficheros de datos.
- Sql*loader. Ejemplos.
- Sql*loader. Ejecución.

COPIA FISICA.

- Realizar una copia física implica copiar los sistemas de ficheros asociados a la base de datos y aquellos donde se ha instalado la misma.
- La copia de los ficheros de base de datos se realizará de forma diaria.
- La copia del "software" de base de datos se realizará con una frecuencia semanal, dado su menor nivel de actualización.
- Previamente a la copia física se efectuará una parada de la base de datos, de forma que su contenido sea íntegro y coherente.

OPTIMAL FLEXIBLE ARCHITECTURE (O.F.A.).

- Oracle recomienda usar la "*Optimal Flexible Architecture*" (O.F.A.) en las instalaciones de base de datos.

- O.F.A. tiene los siguientes beneficios:

  - Organiza grandes cantidades de "software" y datos en disco evitando cuellos de botella y bajas productividades.

  - Facilita las tareas administrativas rutinarias.

  - Gestiona de forma adecuada el crecimiento y administración de la base de datos.

O.F.A. EN ENTORNOS UNIX.

$ORACLE_BASE			Por defecto */u01/app/oracle*
	$ORACLE_HOME		Por defecto */u01/app/oracle/product/<v_bd >*
		/bin	Binarios.
		/network	Ficheros *NET8. Comunicaciones.*
		/dbs	Enlaces a donde residen los ficheros de inicialización.
		/rdbms	Ficheros de servidor y librerías bd requeridas.

O.F.A. EN ENTORNOS UNIX.

$ORACLE_BASE				Por defecto */ u01/app/oracle*
	admin			
		/<nombre_BD>		Nombre base datos.
			adhoc	"Scripts" SQL Ad hoc.
			arch	Fich. archivados de "redo log".
			bdump	Ficheros de traza procesos "background".

O.F.A. EN ENTORNOS UNIX.

$ORACLE_BASE				Por defecto */ u01/app/oracle*
	admin			
		/<nombre_BD>		Nombre base datos.
			cdump	Ficheros “core dump”.
			create	Progr. creación bd.
			exp	Fich. Export bd.
			pfile	init.ora
			udump	Fich. traza usuario.

COPIA FISICA.
SISTEMAS DE FICHEROS.

- Los sistemas de ficheros a copiar **diariamente** serán:

  - Sistemas de ficheros de administración de la bd, bajo *$ORACLE_BASE/admin.*

  - Sistemas de ficheros asociados a la base de datos. Aquellos de la forma */uxx/oradata/<nombre_bd>;* contendrán los ficheros de datos (.dbf), de control (.ctl) y de "redo log" (.log) de la base de datos.

  - Sistemas de ficheros asociados a la exportación de base de datos (en caso de que se haya optado por realizarla). En nuestra bd de la forma */export/<nombre_bd>*.

  - Sistema de ficheros */etc.*

COPIA FISICA.
SISTEMAS DE FICHEROS.

- Los sistemas de ficheros a copiar **semanalmente** serán:

  - Sistemas de ficheros que integran la estructura O.F.A. Bajo *$ORACLE_BASE (/u01/app/oracle).*

  - Todos los sistemas de ficheros que entran en la copia diaria de la base de datos.

COPIA FISICA.
COMANDO TAR (UNIX).

- La copia se realizará tras:
  - Haber hecho la exportación correspondiente de la base de datos.
  - Haber cerrado cualquier comunicación con la base de datos (proceso "*listener*").
  - Cerrar la base de datos (*shutdown*).
- Se usará la sentencia *TAR*: Copia los sistemas de ficheros indicados a un dispositivo físico.
- El proceso debe ser automatizado y el cese de actividad de la base de datos afectar lo menos posible al servicio.

UTILIDADES EXPORT/IMPORT.

- Son usadas para:
  - Transferir objetos de datos entre bases de datos Oracle (independientemente de la plataformas en que residan).
  - Proporcionar copia de seguridad **lógica** para objetos de la base de datos, un espacio de almacenamiento o la base de datos al completo.
  - Migración entre versiones de base de datos (¡Peligro!).
- La copia lógica realizada con "export" se almacena en un fichero binario; este fichero sólo puede ser leído por la utilidad "import". Son utilidades complementarias.

UTILIDAD EXPORT.

- Para usar la utilidad debe tenerse el privilegio *CREATE SESSION*.

- Para exportar objetos pertenecientes a otro esquema de usuario debe tenerse asignado el rol *EXP_FULL_DATABASE.*

- Como resultado se obtiene un fichero *.dmp* (fichero de exportación) y, opcionalmente, un fichero *.log* con el informe de incidencias.

- Es conveniente que todos los ficheros generados para usar en la exportación o como resultado de ella estén en un sistema de ficheros independiente (p.ej. */export/<nombre_bd>*).

UTILIDAD EXPORT.

- Existen cuatro modos de realizar "export":

  - "Tabla". Cualquier usuario puede exportar tablas indicadas del esquema de usuario.

  - "Usuario". Cualquier usuario puede exportar todos los objetos del esquema.

  - "Tablespace". Permite mover "tablespaces" entre bases de datos. Sólo posible con el privilegio *EXP_FULL_DATABASE (como SYSDBA)*.

  - "Full database". Exporta todos los objetos de la BD (no se exportan disparadores pertenecientes al esquema SYS). Sólo posible con el privilegio *EXP_FULL_DATABASE*.

UTILIDAD EXPORT.

- Aunque puede ejecutarse la utilidad e ir indicando en el "prompt" diferentes opciones, se recomienda usar un fichero de parámetros.

- Sintaxis:

exp HELP=Y *Proporciona ayuda en línea.*

exp *Modo interactivo (sin modo directo).*

exp PARFILE=<fichero_parametros>

UTILIDAD EXPORT.

Parámetro	Descripción
BUFFER	Tamaño "bufer" de datos (bytes). No tiene efecto al usar "direct path".
COMPRESS (**Y**/N)	Incluir todos los datos en una extensión (no para LOB).
CONSISTENT (Y/**N**)	Asegura la consistencia de los datos exportados cuando pueden estar siendo actualizados (p.ej. con la bd abierta). Incompatible con copias incrementales.
CONSTRAINTS (**Y**/N)	Exportar o no las restricciones sobre tablas.
DIRECT (Y/**N**)	Modo directo o no -convencional- de exportación.
FILE	Nombre del fichero de "export", por defecto "*expdat.dmp*".
FULL (Y/**N**)	Exportación de la base de datos completa o no.

UTILIDAD EXPORT.

Parámetro	Descripción
GRANTS (**Y**/N)	Exportar o no privilegios sobre objetos.
INDEXES (**Y**/N)	Exportar o no índices.
LOG	Indica fichero donde se guardarán los mensajes.
OWNER	Indica los usuarios de los que se realizara la copia.
ROWS (**Y**/N)	Exportar o no los datos de las tablas.
TABLES	Relación de tablas a exportar (modo tabla).
TABLESPACES	Lista de "tablespaces" a exportar.
TRANSPORT_TABLESPACE (Y/**N**)	Permite exportación de "tablespaces".

UTILIDAD EXPORT.

Parámetro	Descripción
TRIGGERS (**Y**/N)	Exportar o no disparadores.
USERID	Usuario y contraseña del usuario que realiza la exportación (¡Peligro!).

EXPORT. MODO DIRECTO.

- Existen dos métodos prar realizar la exportación:
  - "*Conventional path Export*". Utiliza la sentencia SQL SELECT para extraer datos de las tablas. Se leen los datos desde disco a una "buffer cache" y, después de ser evaluados, se transfieren al cliente "export", que los escribe en el fichero de "export".
  - "*Direct path Export*". Extrae los datos de disco a la "buffer cache" y las filas son transferidas directamente al cliente "export", que las escribe en el fichero de salida. Los datos se hallan en el formato que "export" espera y no hay conversión.

    La extracción de datos es mucho más rápida.

    Se debe especificar "*DIRECT=Y*" en el fichero de parámetros.

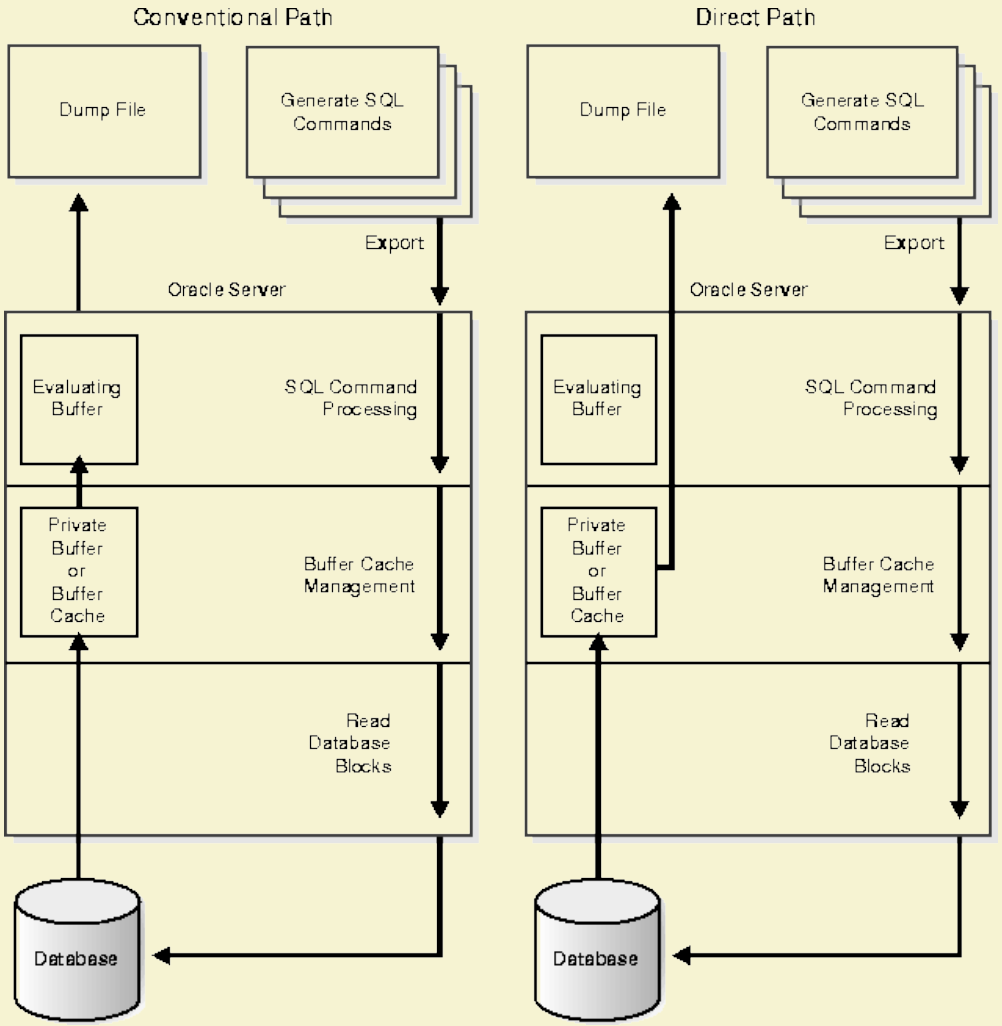
EXPORT. MODO DIRECTO.

- Mediante el parámetro *RECORDLENGTH* se especifica en el fichero de parámetros el tamaño del "buffer" que se usa para escribir en el fichero de "export".

- Se recomienda que *RECORDLENGTH* sea múltiplo del tamaño de bloque E/S y múltiplo de *DB_BLOCK_SIZE,* de forma que cada lectura en tabla devuelva un bloque de datos. Si los datos leídos no caben en el "buffer", se harán múltiples escrituras en el fichero de "export" para cada bloque.

- En modo interactivo no puede usarse el modo directo de exportación.

EXPORT. MODO DIRECTO.



CASOS PRACTICOS.

- La utilidad *"export"* se aconseja en los siguientes casos:

  - Realización de una copia de seguridad de la base de datos.

    FILE=<nombre_bd>.dmp
    FULL=Y
    LOG=<nombre_bd>.log

  - Copias de seguridad de esquema de usuario (al borrarlo y necesitar guardar su esquema de forma temporal).

    FILE=<nombre_esquema_bd>.dmp
    OWNER=<propietario del esquema>
    LOG=<nombre_esquema_bd>.log
    COMPRESS=Y

CASOS PRACTICOS.

- Copias de seguridad de tablas antes de realizar operaciones delicadas como borrado masivo de datos, cargas de datos, actualizaciones, …

FILE=<nombre_tabla>.dmp
TABLES=(<esquema>.<nombre_tabla>, …)
LOG=<nombre_esquema_tabla>.log
COMPRESS=Y

UTILIDAD IMPORT.

- La utilidad “import” es complementaria de “export”.
- Los objetos se importan en el orden en que están en el fichero de exportación:
  - Definición de tipos.
  - Definiciones de tablas.
  - Datos de tablas.
  - Índices.
  - Restricciones de integridad, vistas, procedimientos y disparadores.
  - Índices bitmap, funcionales y de dominio.
- En tablas que ya existen es aconsejable deshabilitar las restricciones de integridad referenciales temporalmente.

UTILIDAD IMPORT.

- Para usar la utilidad debe tenerse el privilegio *CREATE SESSION*. Por supuesto, deben tenerse los privilegios necesarios para crear o trabajar con los objetos a importar.

- Un usuario puede importar un fichero de "export" no creado por él. Si el fichero de exportación fue creado con el privilegio EXP_FULL_DATABASE, debe tenerse asignado el rol *IMP_FULL_DATABASE.*

- Como resultado se obtiene un fichero *.log* con el informe de incidencias.

UTILIDAD IMPORT.

- Existen cuatro modos de realizar “import”:

  - “Tabla”. Cualquier usuario puede importar tablas indicadas del esquema de usuario.

  - “Usuario”. Cualquier usuario puede importar todos los objetos del esquema.

  - “Tablespace”. Permite mover “tablespaces” entre bases de datos. Sólo usuarios privilegiados.

  - “Full database”. Sólo posible con el privilegio *IMP_FULL_DATABASE*.

UTILIDAD IMPORT.

- Aunque puede ejecutarse la utilidad e ir indicando en el "prompt" diferentes opciones, se recomienda usar un fichero de parámetros.

- Sintaxis:

imp HELP=Y *Proporciona ayuda en línea.*

imp *Modo interactivo.*

imp PARFILE=<fichero_parametros>

UTILIDAD IMPORT.

Parámetro	Descripción
BUFFER	Tamaño "bufer" de datos (bytes) usado en la transferencia.
COMMIT (Y/**N**)	Hacer "commit" o no después de cada inserción. Útil para evitar que los segmentos de "rollback" crezcan demasiado. Si ocurre un error y no hay clave única se producirán filas duplicadas.
CONSTRAINTS (**Y**/N)	Importar o no las restricciones sobre tablas.
DESTROY (Y/**N**)	Indica si los ficheros de datos existentes debe ser o no reutilizados.
FILE	Nombre del fichero de "export" a importar, por defecto "*expdat.dmp*".
FROMUSER	Lista de esquemas a importar (usado normalmente con TOUSER).

UTILIDAD IMPORT.

Parámetro	Descripción
FULL (Y/**N**)	Importar o no el fichero completo.
GRANTS (**Y**/N)	Importar o no privilegios sobre objetos.
IGNORE (Y/**N**)	Indica la forma de tratar los errores generados en la creación de objetos. Si *IGNORE=Y* se ignoran los errores de creación y se continua, en caso contrario se muestran. En el caso de tablas, si *IGNORE=Y* se importan los datos en las tablas existentes. Si *IGNORE=N* se genera un error y no se inserta.
INDEXES (**Y**/N)	Importar o no índices.
INDEXFILE	Fichero de descarga para sentencias de creación de índices, no son creados.

UTILIDAD IMPORT.

Parámetro	Descripción
LOG	Indica fichero donde se guardaran los mensajes.
ROWS (**Y**/N)	Incluir datos en la importación.
SHOW (Y/**N**)	Si *SHOW=Y*, el contenido del fichero de "export" se lista pero no se importa. Solo puede usarse con los parámetros FULL=Y, FROMUSER, TOUSER, o TABLES.
TABLES	Lista de tablas a importar.
TABLESPACES	Lista de espacios de almacenamiento a importar.
TO_USER	Lista de esquemas donde importar. Se requiere *IMP_FULL_DATABASE.*
TRANSPORT_TABLESPACE	Importar datos de espacios de almacenamiento.

CASOS PRACTICOS.

- Se hará uso de la utilidad *"import"* en los casos:

  - Recuperación de la base de datos a un punto en el tiempo.

    FILE=<nombre_fichero_export>.dmp
    FULL=Y
    LOG=<nombre_bd>.log

  - Recuperación de un esquema de usuario.

    FILE=<nombre_fichero_export>.dmp
    FROMUSER=<propietario del esquema>
    TOUSER=<esquema_importación>
    LOG=<nombre_esquema_bd>.log

CASOS PRACTICOS.

- Recuperación de tablas tras un borrado accidental (si son muy voluminosas emplear opción *COMMIT=Y*). Es aconsejable en este caso importar a un esquema diferente a aquel al que pertenece la tabla y posteriormente hacer un "*create table … as select * from …;*" o un "*insert into … select * from …;*".

FILE=<nombre_fichero_export>.dmp
FROMUSER=<esquema_origen>
TOUSER=<esquema_destino>
TABLES=(<nombre_tabla1>, …)
LOG=<nombre_esquema_tabla>.log

CASOS PRACTICOS.

- Recuperación de procedimientos, disparadores y paquetes. En este caso no es posible la importación, es necesario crear un fichero con el contenido del fichero de exportación, editarlo, seleccionar el texto buscado para confeccionar el "script" adecuado y ejecutar este ultimo.

  FILE=<nombre_fichero_export>.dmp
  FROMUSER=<esquema_origen>
  SHOW=Y
  GRANTS=N
  ROWS=N
  INDEXES=N
  LOG=<nombre_esquema_tabla>.log

- En el caso de índices se usa el parámetro INDEXFILE.

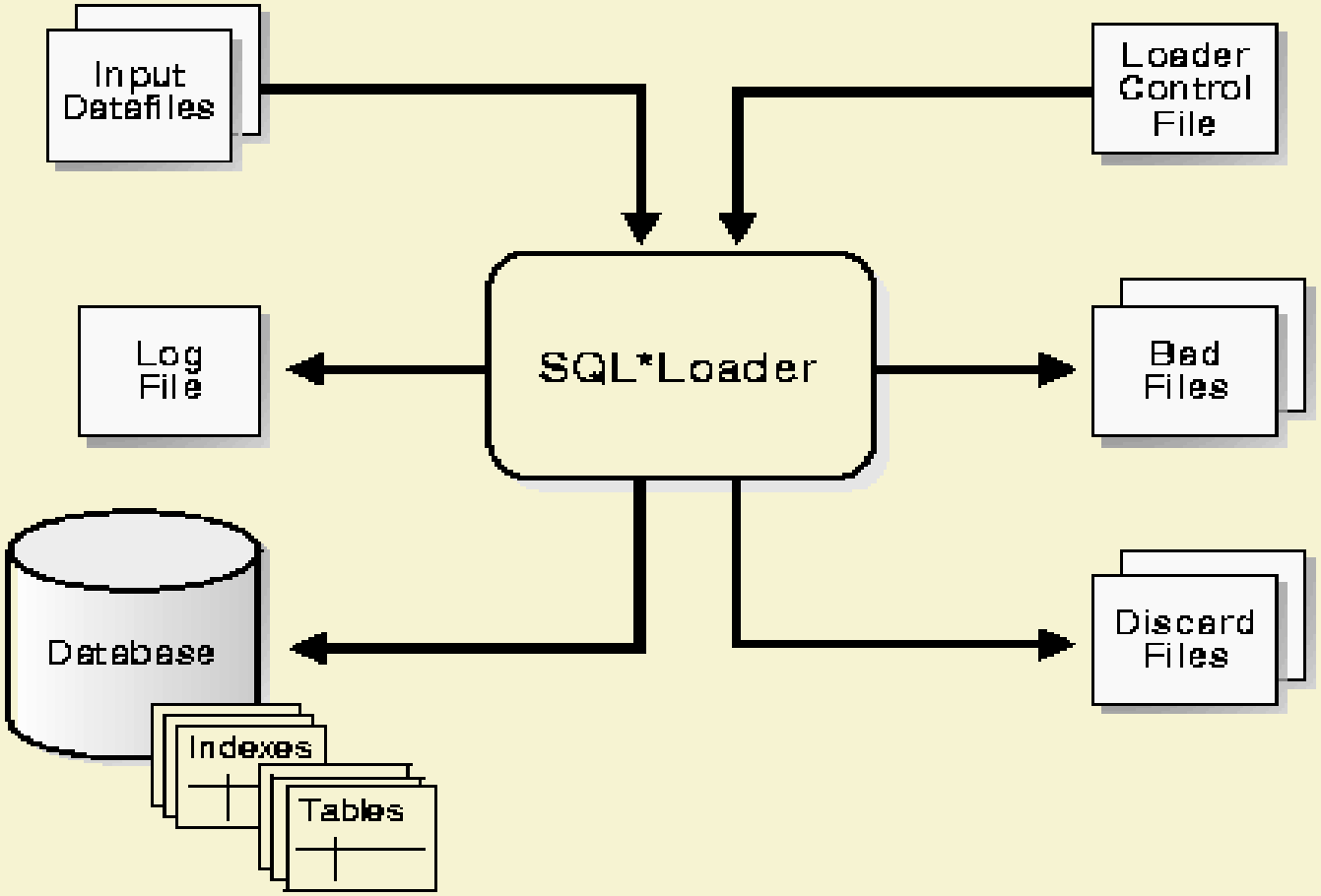
SQL*LOADER.

- SQL*Loader carga datos que proceden de otro tipo de ficheros distintos a tablas Oracle.

- Tiene como entrada un *fichero de control* y uno o más *ficheros de datos*.

- Su salida es una base de datos donde se cargan los datos, un fichero de "*log*", un fichero donde se guardan los registros rechazados ("*bad file*") y un fichero donde se almacenan los registros descartados por no cumplir los criterios especificados en el fichero de control ("*discard file*").

SQL*LOADER.



SQL*LOADER.
FICHERO DE CONTROL.

- Es un fichero de texto con instrucciones que indica donde encontrar los datos a cargar y su formato, la configuración del SQL*Loader al cargar los datos y como analizar e interpretar los datos.

- Se considera dividido en tres secciones:

  - Información de la sesión (opciones y cláusula *INFILE* que indica donde están los datos).

  - Uno o mas bloques "*INTO TABLE*" con información sobre la tabla en la que cargar los datos (nombre y columnas).

  - Datos de entrada, es opcional.

SQL*LOADER. FICHEROS DE DATOS.

- Lee uno o más ficheros de datos indicados en el fichero de control.
- Los datos se consideran organizados en registros.
- Un fichero de datos puede estar en tres formatos:
  - "*Fixed-record*". Cuando todos los registros tiene la misma longitud.
  - "*Variable-record*". Se indica la longitud del registro al principio del mismo.
  - "*Stream-record*". No se indica longitud, la marca un "terminador".

SQL*LOADER. EJEMPLOS.

```
-- Ejemplo 1 de fichero de control

load data
infile 'example.dat' "fix 10"
into table example
fields terminated by ',' optionally enclosed by '"'
(col1, col2)
```

example.dat:

```
0001, abcd,
0002,fghi,
0003,klmn,
```

SQL*LOADER. EJEMPLOS.

-- Ejemplo 2 de fichero de control (incluye datos en el fichero)

```
load data
infile *
into table dept
fields terminated by ',' optionally enclosed by '"'
(deptno, dname, loc)
begindata
12,research,"saratoga"
10,"accounting",cleveland
11,"art",salem
13,finance,"boston"
21,"sales",phila.
22,"sales",rochester
42,"int'l","san fran"
```

SQL*LOADER. EJECUCION.

- Desde la línea de sentencias debe ejecutarse:

sqlldr userid=<usuario>/<contraseña>
control=<nombre_fichero_control>
log=<nombre_fichero_log>

APENDICE A.
Recursos Oracle en Internet.

- www.oracle.com
  - Portal oficial de Oracle.
- metalink.oracle.com
  - Soporte técnico para usuarios con contrato de mantenimiento.
- otn.oracle.com
  - Portal para desarrolladores. Interesante registrarse.
- otn.oracle.com/oramag
  - Revista Oracle Magazine.
- www.orafaq.org
  - Sitio no oficial sobre Oracle.