TEMA 6.

ESPACIO DE "UNDO".

TEMA 6.
ESPACIO DE "UNDO".

- Concepto de Transacción.
- Espacio de "UNDO" (DESHACER).
  - Caídas del sistema. "Redo log".
  - Consistencia en lectura.
  - Retroceso de transacción. "Flashback".
- Gestión de Espacio de "UNDO" (DESHACER).
  - Parámetros obsoletos.
- Modo automático de "undo".
  - Parámetros de inicialización.
  - Cuota de "UNDO".
  - Espacio almac. Creación, modificación, borrado e intercambio.
  - Vistas.
  - Dimensionado manual y automático.
  - Periodo de permanencia. "Retention guarantee".

TEMA 6.
ESPACIO DE "UNDO".

- Modo manual de "UNDO" (DESHACER).
  - 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").
  - Análisis de datos previos usando Oracle Flashback Query.
  - Recuperación lógica usando Oracle Flashback.

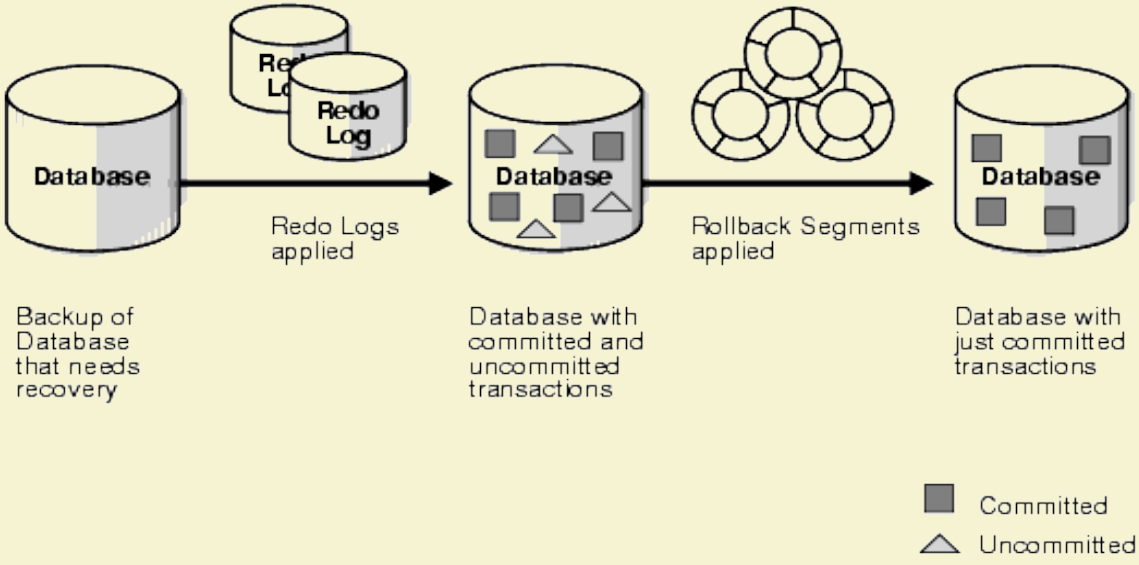
ESPACIO DE "UNDO" Y CAÍDAS DEL SISTEMA.

- 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 CAÍDAS DEL SISTEMA.



ESPACIO DE "UNDO" Y "REDO LOG".

- En el caso del "redo log", al igual que con "undo", el sistema recoge datos estadísticos que ayudan a determinar su tamaño óptimo.

- En general los "redo" deben ser lo bastante grandes de como para que los "checkpoint" no ocurran demasiado frecuentemente; si se produce "log switch" con frecuencia superior a 20 minutos, el rendimiento decrece. Por otra parte, los "redo" excesivamente grandes afectan al rendimiento en disco y ocupan espacio.

- La columna *OPTIMAL_LOGFILE_SIZE* de la vista *V$INSTANCE_RECOVERY* indica el valor mínimo recomendado para el tamaño de los "redo log".

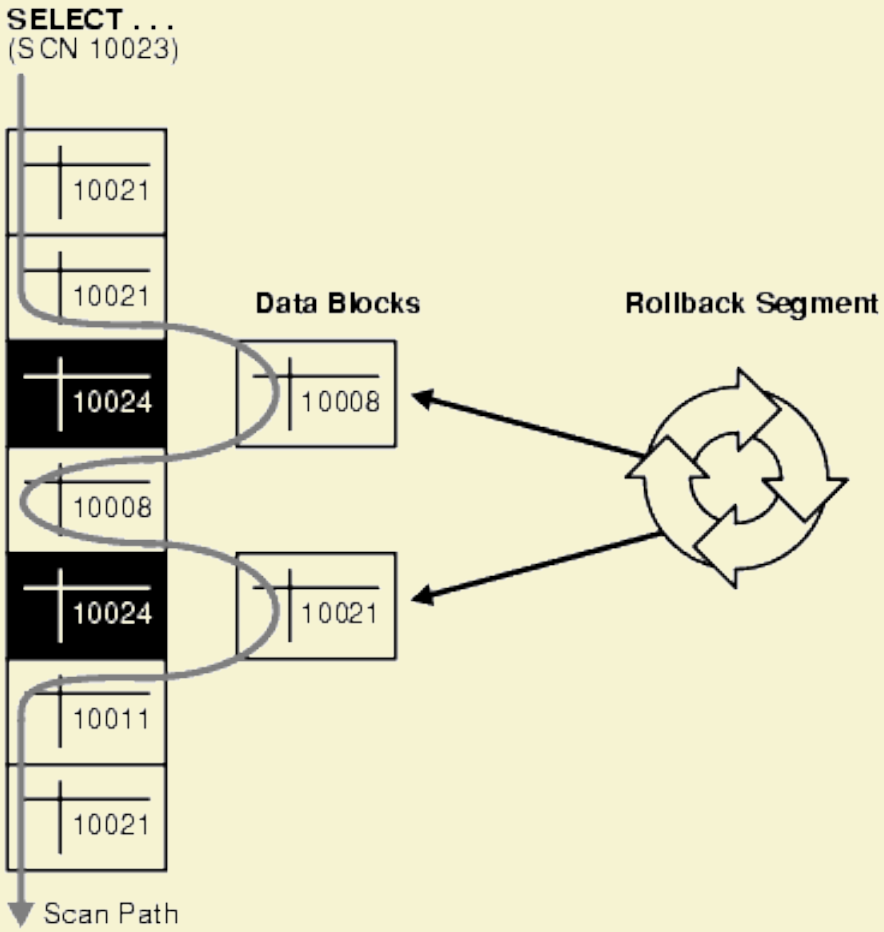
ESPACIO DE "UNDO" Y CONSISTENCIA EN LECTURA.

- 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.



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). A desaparecer.

  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*.

PARAMETROS OBSOLETOS.

- Existen distintos parámetros usados en versiones anteriores, relacionados con "undo" -manual y automático-, que han quedado obsoletos:
  - *UNDO_SUPPRESS_ERRORS*
  - *MAX_ROLLBACK_SEGMENTS*
  - *ROW_LOCKING*
  - *SERIALIZABLE*
  - *TRANSACTION_AUDITING*

MODO AUTOMATICO DE "UNDO"

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), el primero disponible.

- Si no existe, se usa el segmento de "rollback" *SYSTEM*, y se genera un mensaje de error en el fichero de alertas. ¡Error!.

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, al menos, la informacion de "undo" disponible -importante en largas transacciones, consistencia en lectura, y utilidades Flashback -.

    Sólo en ciertas circunstancias es necesario fijar este parámetro:

    - Esp. de alm. con la opción *AUTOEXTEND*.
    - Fijar el periodo de retención para *LOB*.
    - Se desea especificar la "retention guarantee".

    El sistema calcula automáticamente el periodo de permanencia, para satisfacer los requerimientos de las consultas, basándose en el uso y tamaño del esp.alm. de "undo" e ignorando el valor *UNDO_RETENTION*, si es necesario, a menos que "*retention guarantee*" esté activa (si se necesita espacio para las transacciones activas y no hay suficiente , se reutiliza el existente y puede provocar el fallo de consultas muy largas).

MODO AUTOMATICO “UNDO”. PARAMETROS INICIALIZACION.

- *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.

  Si se omite, se elige el primer espacio de almacenamiento de “undo” disponible, si no hay ninguno se arranca sin esp.alm. de “undo” y las transacciones se ejecutan en el segmento de “rollback” SYSTEM ¡Error!.

- **Al arrancar en modo automático, cualquier parámetro relativo al modo manual usado en el fichero de parámetros es ignorado.**

MODO AUTOMATICO “UNDO”. ESPACIO DE ALMACENAMIENTO.

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

- A cada instancia se le asigna sólo un esp.alm. de “undo” -puede ser de tipo “bigfile”-. Los datos de “undo” se manejan usando segmentos de “undo” que se crean y mantienen por el sistema.

- Si no hay ningún espacio de “undo” se emplea el segmento de “rollback” SYSTEM. ¡Error!.

MODO AUTOMATICO “UNDO”. ESPACIO DE ALMACENAMIENTO.

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

  - Creación.
  - Modificación.
  - Borrado.
  - Intercambio entre distintos espacios de “undo”.
  - Modificar la clausula de retención.

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 RETENTION
      GUARANTEE;
```

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”.

ALTER DATABASE DATAFILE
'/u03/oradata/CURSO25/undo01.dbf' RESIZE 100M;

- Renombrar un fichero de datos.

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

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

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

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

- Modificar claúsula “RETENTION”.

  - NO preservar los datos de “undo” que son aun válidos.

    ALTER TABLESPACE UNDOTBS
    RETENTION NOGUARANTEE;

  - Preservar los datos de “undo” que son aun válidos.

    ALTER TABLESPACE UNDOTBS
    RETENTION GUARANTEE;

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.

- Si se indica *ALTER SYSTEM SET UNDO_TABLESPACE='';* se deasigna el espacio de “undo” actual y se pasa al siguiente disponible.

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$TRANSACTION*. Transacciones activas en el sistema.
- *DBA_TABLESPACES*.
- *DBA_UNDO_EXTENTS*. Extensiones en el espacio de “undo”.
- *DBA_HIST_UNDOSTAT*. Estadísticas acerca de consumo de espacio de “undo”, concurrencia de transacciones, longitud de las consultas ejecutas en la instancia, ... (contiene vistas de v$UNDOSTAT).

MODO AUTOMATICO “UNDO”. DIMENSIONANDO EL ESPACIO “UNDO”.

- Puede dimensionarse de tres formas:
  - Empleando esp. de almacenamiento autoextensibles, de forma que incrementa su tamaño conforme es necesario (claúsula *AUTOEXTEND*).
  - Cálculo usando método manual.
  - Cálculo usando método automático: *Undo Advisor*.

MODO AUTOMATICO "UNDO". DIMENSIONADO MANUAL.

- 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 número máximo de bloques "undo" por segundo (columna *UNDOBLKS* de *V$UNDOSTAT*).

- Es preciso añadir al valor obtenido entre un 10% y un 20% de forma que pueda hacerse frente a situaciones inesperadas.

MODO AUTOMATICO "UNDO". DIMENSIONADO AUTOMATICO.

- Si el espacio NO es autoextensible puede estimarse su valor óptimo usando el "*Undo Advisor*", empleando el paquete *DBMS_ADVISOR*; este depende del repositorio *Automatic Workload Repository (AWR)* que contiene instantáneas de todas las estadísticas clave y de la carga de trabajo de la bd, a intervalos de 30 minutos (registro histórico de uso de la bd).
- Operaciones a seguir:

a) Determinar el periodo de tiempo a analizar (seleccionando los identificadores de "snap" apropiados). La vista *DBA_HIST_SNAPSHOT* muestra la información sobre instantáneas en el AWR (número de instantánea y fecha).

```
SQL >Select snap_id, begin_interval_time, end_interval_time
from DBA_HIST_SNAPSHOT
where begin_interval_time > '...' and
end_interval_time < '...' order by end_interval_time desc;
```

MODO AUTOMATICO “UNDO”. DIMENSIONADO AUTOMATICO.

b) Invocar *Undo Advisor* mediante el siguiente bloque para determinar el número de tarea.

```
DECLARE
        tid NUMBER;
        tname VARCHAR2(30);
        oid NUMBER;
BEGIN
        DBMS_ADVISOR.CREATE_TASK('Undo Advisor', tid, tname, 'Undo Advisor Task');
        DBMS_ADVISOR.CREATE_OBJECT(tname, 'UNDO_TBS', null, null, null, 'null', oid);
        DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'TARGET_OBJECTS', oid);
        DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'START_SNAPSHOT',
           snap_id_inicial);
        DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'END_SNAPSHOT', snap_id_final);
        DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'INSTANCE', 1);
        DBMS_ADVISOR.execute_task(tname);
        DBMS_OUTPUT.PUT_LINE ('Identificador de trabajo es: ' || tid ||'  '|| tname);
end;
/
```

MODO AUTOMATICO “UNDO”. DIMENSIONADO AUTOMATICO.

c) Consultar la vista DBA_ADVISOR_FINDINGS para averiguar las recomendaciones.

SQL >Select owner, task_id, task_name, type, message, more_info from dba_advisor_findings where task_id= <numero_tarea -tid->

- *Puede generarse un informe de texto o HTML mediante AWR ejecutando*

*SQL> @$ORACLE_HOME/rdbms/admin/**awrrpt.sql***

MODO AUTOMATICO “UNDO”. PERIODO DE PERMANENCIA “UNDO”.

- El cálculo del valor óptimo para *UNDO_RETENTION* se realiza a partir de información estadística de uso y se ajusta para la consulta más larga, la información sobre duración de las consultas se obtiene cada 30 segundos.

- La bd analiza automáticamente el uso de “undo” para determinar el tamaño del esp.alm. de “undo” de forma que pueda soportar la consulta más larga.

- Esta característica está activada por defecto (arquitectura) y NO puede desactivarse. El valor calculado será tal que permita evita el error “snapshot too old”.

MODO AUTOMATICO "UNDO". PERIODO DE PERMANENCIA "UNDO".

- En el cálculo, automático, del periodo de permanencia es necesario considerar:
  - El valor por defecto para UNDO_RETENTION son 900 segundos.

    Si el parámetro se ajusta a cero o no se indica valor, se ajusta automáticamente empleando el valor 900 como el mínimo.

    Si UNDO_RETENTION se ajusta a un valor distinto de cero, se ajusta automáticamente empleando el valor indicado como mínimo.
  - Para un esp.alm. de "undo" con AUTOEXTEND. El sistema lo ajusta para ser algo superior que el mayor tiempo correspondiente a la ejecución de una consulta, si hay espacio. Si hay espacio libre, el periodo no es inferior al UNDO_RETENTION asignado.
  - Para un esp.alm. de "undo" de tamaño fijo, la bd lo ajusta al máximo posible (hasta que haya espacio). Se ignora el valor de UNDO_RETENTION a menos que "*RETENTION GUARANTEE*" esté activa.

MODO AUTOMATICO "UNDO". PERIODO DE PERMANENCIA "UNDO".

  - El ajuste automático del periodo de permanencia no es soportado por LOB. Para estos objetos, el valor es el fijado por el parámetro UNDO_RETENTION.

  - En operaciones DML de gran carga no se garantiza UNDO_RETENTION. Para asegurar que el espacio de "undo" necesario siempre estará disponible durante el periodo indicado se emplea la claúsula *RETENTION GUARANTEE*.

- El tiempo que el sistema retiene los datos en "undo" para el esp.alm. actual puede obtenerse consultando la columna *TUNED_UNDORETENTION* de la vista *V$UNDOSTAT*.

  - La vista presenta estadísticas en periodos de diez minutos, una fila de datos por periodo, para los últimos 4 días (más allá han de consultarse en la vista *DBA_HIST_UNDOSTAT*).

MODO AUTOMATICO “UNDO”. PERIODO DE PERMANENCIA “UNDO”.

```
select to_char(begin_time, 'DD-MON-RR HH24:MI') begin_time,
to_char(end_time, 'DD-MON-RR HH24:MI') end_time,
tuned_undoretention
from v$undostat order by end_time;
```

BEGIN_TIME	END_TIME	TUNED_UNDORETENTION
04-FEB-06 00:01	04-FEB-06 00:11	12100
...		
07-FEB-06 23:21	07-FEB-06 23:31	86700
...		

El valor de “*tuned_undoretention*” se muestra en segundos.

MODO AUTOMATICO "UNDO". "RETENTION GUARANTEE".

- La claúsula "RETENTION GUARANTEE" permite garantizar el éxito de consultas de larga duración u operaciones "Flashback". Puede usarse al crear el esp.almacenamiento de "undo" (CREATE UNDO TABLESPAE o CREATE DATABASE) o usando la sentencia ALTER TABLESPACE.

- Al activar la claúsula se garantiza el mínimo "undo_retention" especificado, de forma que la bd nunca sobeescribe datos (las transacciones pueden fallar por falta de espacio).

- Para desactivar la garantía de retención debe usarse la claúsula "RETENTION NOGUARANTEE".

- El valor actual puede consultarse en la vista *DBA_TABLESPACES* (columna *RETENTION*, con valores *GUARANTEE, NOGUARANTEE* o *NOT APLY*).

MODO MANUAL DE "UNDO"

MODO MANUAL "UNDO". PARAMETROS INICIALIZACION.

- El parámetro de inicialización UNDO_MANAGEMENT debe tener el valor MANUAL, o bien no indicarse (es el valor por defecto).

UNDO_MANAGEMENT = MANUAL

MODO MANUAL "UNDO". PARAMETROS INICIALIZACION.

Parametros de inicialización relacionados son:

- *ROLLBACK_SEGMENTS*. Asigna segmentos a 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”).

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".

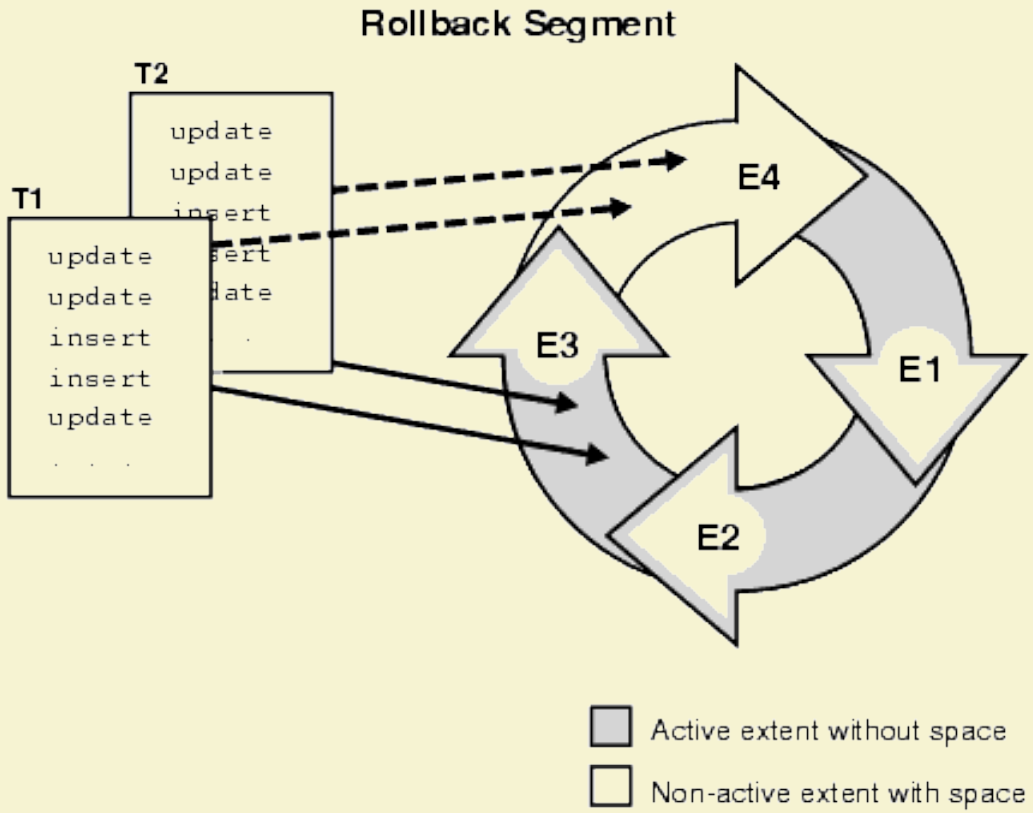
- Cada segmento de "rollback" debe tener al menos dos extensiones asignadas.

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

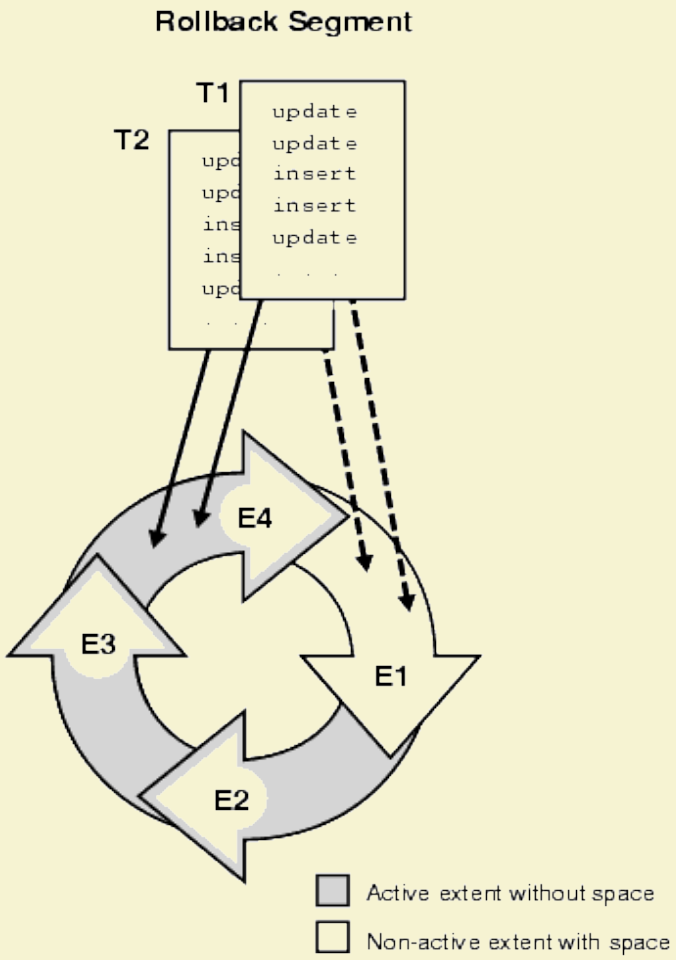
  - Bien se reutiliza una extensión ya asignada al segmento. Se comprueba la siguiente extensión y si no contiene información de una transacción activa, se convierte en la extensión actual.

  - O bien se asigna una nueva extensión al segmento.

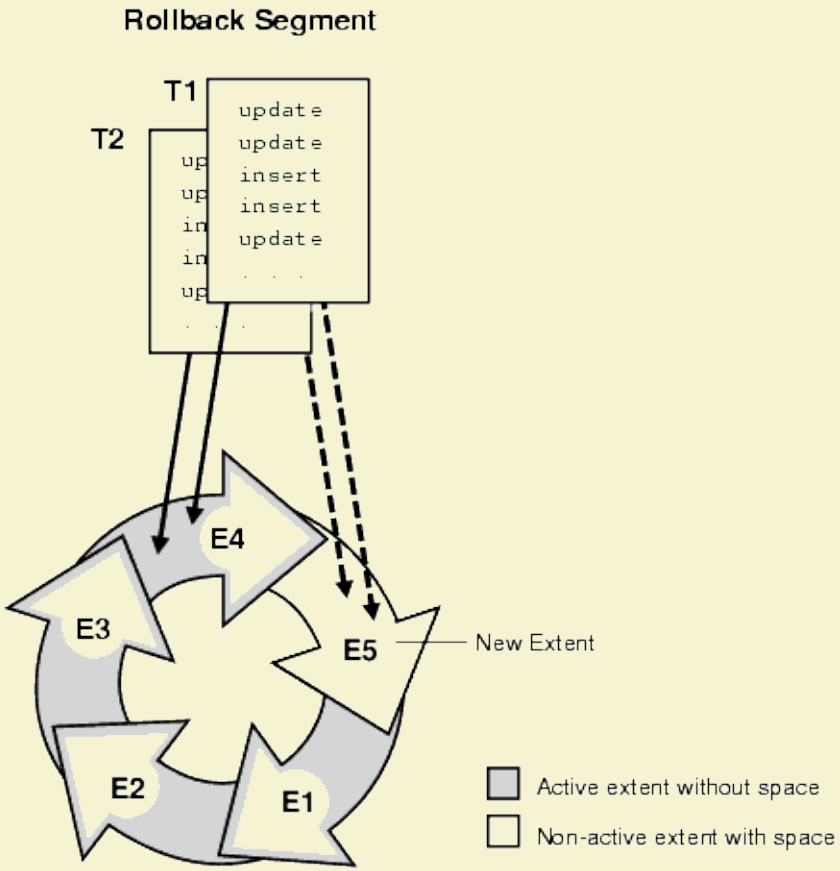
ESCRITURA EN UN SEGMENTO DE "ROLLBACK".



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 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.
- Grupos de espacios temporales.
- Espacio de almacenamiento SYSAUX.
- Perfiles. Parámetros. Función de verificación.
- 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. Beneficios de los 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.

USUARIOS

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. Tras la autentificación, puede autorizarse, o no, el acceso a determinados elementos y la ejecución de ciertas acciones.
- 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 de la autentificación por sistema operativo es los usuarios ops$ o "identified externally". El prefijo a usar 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.
- No es posible la conexión del usuario creado a menos que posea el privilegio de sistema *CREATE SESSION*.
- Sintaxis:

```
CREATE USER <usuario>
IDENTIFIED BY <contraseña>/EXTERNALLY
DEFAULT TABLESPACE <espacio>
TEMPORARY TABLESPACE <espacio>/<grupo_espacios>
QUOTA <xx>/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 crearán los objetos del esquema del usuario cuando al hacerlo no se indica ninguno en particular.

  Si se omite la clausula, los objetos se crean en el esp. alm. por defecto de la bd (default user tablespace) que se indica mediante la sentencia: ***alter database default tablespace ...;*** *si este no se ha especificado el espacio por defecto es SYSTEM (¡Error!).*

- TEMPORARY TABLESPACE.

  Indica el espacio o grupo de espacios de almacenamiento para los segmentos temporales requeridos por el usuario.

  No debe indicarse cuota. Si se omite la clausula, el espacio temporal por defecto es el SYSTEM (¡Error!) a menos que se haya fijado el valor del esp. temporal por defecto (default temporary tablespace): ***alter database default temporary tablespace ...;*** .

CREACIÓN DE USUARIOS.

- QUOTA.

  Indica la cantidad máxima de espacio que un usuario puede utilizar en un determinado espacio de almacenamiento. Puede indicarse cuota en múltiples espacios al tiempo.

  El creador del usuario puede indicar cuota sobre espacios de almacenamiento aunque él no las posea.

  Por defecto no se tiene cuota en ningún espacio de almacenamiento. Indicando *UNLIMITED*, es ilimitado el espacio a usar. Pueden usarse distintas abreviaturas para indicar el tamaño: kilobytes (K), megabytes (M), gigabytes (G), terabytes (T), petabytes (P), o exabytes (E).

  Puede revocarse el acceso a un espacio de almacenamiento asignando cuota cero en 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 (especifica limitaciones en recursos del sistema y restricciones). Si se omite se asigna el perfil *DEFAULT*.

- 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.

GRUPOS ESPACIOS TEMPORALES.

- Un grupo de espacios de alm. temporales (“*temporary tablespace group”)* es un sinónimo que engloba a un conjunto de espacios de almacenamiento temporales.

- El grupo se crea cuando se añade el primer espacio temporal al mismo (no puede estar vacio, por lo que, al menos, tiene un miembro). Si se eliminan todos sus componentes, el grupo deja de existir.

- Creación de grupos:

  alter tablespace <nombre_temporal> tablespace group <nombre_grupo_temporales>;

  create temporary tablespace <nombre_temporal> tempfile '/.../... .dbf' size ... tablespace group <nombre_grupo>;

GRUPOS ESPACIOS TEMPORALES.

- Asignar grupos por defecto:

  alter database default temporary tablespace <nombre_grupo>;

- Eliminar un temporal de un grupo:

  alter tablespace <nombre_temporal> tablespace group '';

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>/<grupo_espacios>
QUOTA <xx>/UNLIMITED ON <espacio>
DEFAULT 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 (con la sentencia *GRANT*).

  Oracle activa los roles indicados sin necesidad de especificar sus contraseñas.

  Al crear el usuario los roles por defecto son todos los asignados, se limitan posteriormente mediante *ALTER USER*.

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.

- Oracle no borra esquemas de usuario no vacios a menos que se indique CASCADE, realiza un borrado de objetos previo, o se hayan eliminado con anterioridad los objetos.

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

  - Se invalida vistas o sinónimos para objetos en el esquema borrado.
  - Se invalidan procedimientos almacenados, funciones, o paquetes que consulten objetos pertenecientes al esquema eliminado.
  - Las vistas materializadas en otros esquemas basados en tablas pertenecientes al esquema borrado no podrán refrescarse.
  - Se borran todos los disparadores, "triggers", del esquema.
  - No se eliminan roles creados por el usuario.

- Sintaxis:

  DROP USER <usuario> <CASCADE>;

ESPACIO DE ALMACENAMIENTO SYSAUX.

- El espacio de almacenamiento *SYSAUX* acompaña a SYSTEM, ayudando a incrementar la disponibilidad de la bd al descargar datos de aplicaciones que utilizaban el espacio SYSTEM u otros con anterioridad a esta versión de bd.

  Tiene las mismas características de almacenamiento que SYSTEM.

- La vista dinámica *V$SYSAUX_OCCUPANTS* indica, entre otras cosas, las aplicaciones que usan *SYSAUX*, el espacio utilizado, el nombre del esquema propietario de la mismas y el nombre del procedimiento que debe emplearse para desplazar determinado contenido desde *SYSAUX* hasta otra lozalización:

```
Select occupant_name, schema_name, space_usage_kbytes,
  move_procedure from V$SYSAUX_OCCUPANTS;
```

PERFILES

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 (no aplicable a los parámetros de contraseña que siempre están habilitados):

  - 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 él tienen valor *UNLIMITED*, por lo que es conveniente modificarlo (sentencia *ALTER PROFILE*).
- Un usuario al que no se le asigna perfil posee el perfil *DEFAULT*.
- Aquellos recursos para los que en el perfil asignado no se ha definido un valor, o se ha indicado *DEFAULT*, toman el valor designado en el perfil por defecto.
- Sintaxis:

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

CREACIÓN DE PERFILES. PARÁMETROS DE RECURSOS.

- ***Parámetros de recursos:***
  - *SESSIONS_PER_USER.* Número 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 continua 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.

- ***Parámetros de contraseña:***

  - *FAILED_LOGIN_ATTEMPTS. Número de intentos fallidos de conexión antes del bloqueo.*

  - *PASSWORD_LIFE_TIME. Número de días en que la clave es válida para autentificación.*

    Si se indica un valor para PASSWORD_GRACE_TIME, la clave expira si no se cambia en este periodo. Si no se indica valor para PASSWORD_GRACE_TIME, por defecto UNLIMITED, se genera un aviso pero el usuario puede seguir conectándose.

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

  - *PASSWORD_REUSE_TIME . Número de días en los cuales la contraseña no puede reutilizarse.*

CREACIÓN DE PERFILES. PARÁMETROS DE CONTRASEÑA.

- *PASSWORD_REUSE_MAX . Número de cambios de clave necesarios antes de poder reutilizar la clave actual.*

NOTA: PASSWORD_REUSE_TIME y PASSWORD_REUSE_MAX deben usarse conjuntamente. Si se indica un entero para ambos parámetros, el usuario no puede reutilizar la contraseña hasta que ha cambiado el número de veces indicado en PRM durante el periodo indicado por PRT. Si alguno de los dos tiene valor UNLIMITED, nunca se podrá reutilizar la contraseña. Si ambos tiene valor UNLIMITED, la bd los ignora.

- *PASSWORD_LOCK_TIME. Número de días que la cuenta estará bloqueada después de un cierto número de fallos de conexión indicado.*

- *PASSWORD_VERIFY_FUNCTION. Permite indicar como argumento un "script" PL/SQL que verifica la complejidad de la clave. Si se indica NULL no se usa función alguna. Oracle proporciona una función por defecto: "verify_function".*

CREACIÓN DE PERFILES.

- Valor *UNLIMITED.* Si es un parámetro de recurso 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.
- **Ejemplo de creación de perfil.**

```
CREATE PROFILE perfil_2 LIMIT
FAILED_LOGIN_ATTEMPTS 5
PASSWORD_LIFE_TIME 60
PASSWORD_REUSE_TIME 60
PASSWORD_REUSE_MAX 5
PASSWORD_VERIFY_FUNCTION verify_function
PASSWORD_LOCK_TIME 1/24
PASSWORD_GRACE_TIME 10;
```

FUNCIÓN DE VERIFICACIÓN.

- La función de verificación de contraseña debe pertenecer al usuario SYS. Realiza las comprobaciones:
  - La contraseña satisface un mínimo de longitud.
  - La contraseña no coincide con el nombre de usuario.
- Puede modificarse, siempre en el esquema SYS y conectando como CONNECT SYS/*password* AS SYSDBA

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*. El perfil *DEFAULT* no puede borrarse.
- 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.
- Sintaxis:

DROP PROFILE <perfil>;

DROP PROFILE <perfil> CASCADE;

VISTAS. USUARIOS Y PERFILES.

DBA_USERS	*Usuarios de la base de datos.*
ALL_USERS	*Usuarios visibles al usuario actual.*
USER_USERS	*Describe el usuario actual.*
DBA_TS_QUOTAS ***USER_TS_QUOTAS***	*Cuotas de espacio para usuarios.*
USER_PASSWORD_LIMITS	*Parámetros de contraseña asignados al usuario.*
USER_RESOURCE_LIMITS	*Parámetros de recursos asignados al usuario.*
V$SESSION	*Información sobre sesiones.*
V$SESSTAT	*Estadísticas de sesión (ver también V$STATNAME).*

PRIVILEGIOS

PRIVILEGIOS.

- Derecho a ejecutar un tipo determinado de sentencia SQL o a acceder a un objeto de otro usuario. Pueden asignarse a usuarios o, preferiblemente, a roles. Es importante no excederse en la concesión de privilegios.

- Se distinguen dos tipos:

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

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

PRIVILEGIOS DE SISTEMA.

- DATABASE.
  - *ALTER DATABASE*
  - *ALTER SYSTEM*
  - *AUDIT SYSTEM*
- DATABASE LINKS
  - *CREATE DATABASE LINK*
  - *CREATE PUBLIC DATABASE LINK*
  - *DROP PUBLIC DATABASE LINK*
- TABLAS – INDICES
  - *CREATE TABLE*
  - *CREATE ANY TABLE / CREATE ANY INDEX*
  - *ALTER ANY TABLE / ALTER ANY INDEX*
  - *DROP ANY TABLE / DROP ANY INDEX*
  - *DELETE ANY TABLE / DROP ANY TABLE / INSERT ANY TABLE / UPDATE ANY TABLES/ SELECT ANY TABLE*

PRIVILEGIOS DE SISTEMA.

- PROCEDURE
  - *CREATE PROCEDURE*
  - *CREATE ANY PROCEDURE*
  - *ALTER ANY PROCEDURE*
  - *DROP ANY PROCEDURE*
  - *EXECUTE ANY PROCEDURE*
- PROFILES
  - *CREATE PROFILE*
  - *ALTER PROFILE*
  - *DROP PROFILE*
- ROLES
  - *CREATE ROLE*
  - *ALTER ANY ROLE*
  - *DROP ANY ROLE*
  - *GRANT ANY ROLE*

PRIVILEGIOS DE SISTEMA.

- ROLLBACK SEGMENTS
  - *CREATE ROLLBACK SEGMENT*
  - *ALTER ROLLBACK SEGMENT*
  - *DROP ROLLBACK SEGMENT*
- SESSIONS
  - *CREATE SESSION*
  - *ALTER SESSION*
- TABLESPACES
  - *CREATE TABLESPACE*
  - *ALTER TABLESPACE*
  - *DROP TABLESPACE*
  - *MANAGE TABLESPACE*
  - *UNLIMITED TABLESPACE*

PRIVILEGIOS DE SISTEMA.

- USUARIO
  - *CREATE USER*
  - *ALTER USER*
  - *DROP USER*
- OTROS
  - *ANALYZE ANY*
  - *AUDIT ANY*
  - *COMMENT ANY TABLE*
  - *...*

Nota: Consultar "Oracle Database SQL Reference" para una lista completa de privilegios de sistema.

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. Este privilegio permite usar una cantidad ilimitada de espacio en cualquier espacio de almacenamiento de la bd y se antepone a cualquier cuota explícita asignada al usuario (¡Error!).

OTORGAR PRIVILEGIOS DE SISTEMA.

- Para que un usuario pueda otorgar un priv.de sistema bien debe haberse otorgado con *ADMIN OPTION*, permite a aquel a quien se le concede el privilegio poder otorgarlo (¡peligro!), o haber sido concedido el privilegio *GRANT ANY PRIVILEGE*.

- 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!).

RESTRICCIONES EN PRIVILEGIOS DE SISTEMA.

- El parámetro de inicialización, fichero init.ora, *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 (privilegios *ANY*).

- Por defecto su valor es *FALSE*, el acceso a objetos en este esquema está entonces restringido a SYS y aquellos usuarios que se conectan como SYSDBA. 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).

- Si su valor es *TRUE*, se permite el acceso a los objetos del esquema *SYS (¡Error!)*.

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 referente a operaciones DDL (por ej. CREATE TABLE); si lo hay cuando se revoca un privilegio referente a operaciones DML (por ejemplo SELECT ANY TABLE).

- Si se retira un privilegio de sistema de *PUBLIC*, pero existen usuarios a los que se ha otorgado aquel directamente o a través de roles, estos siguen pudiendolo usar.

PRIVILEGIOS SOBRE OBJETOS.

- TABLAS
  - *ALTER* - *DELETE* - *FLASHBACK*
  - *INDEX* - *INSERT* - *REFERENCES*
  - *SELECT* - *UPDATE*
- VISTAS
  - *DELETE* - *FLASHBACK* - *INSERT*
  - *REFERENCES* - *SELECT* - *UPDATE*
- SECUENCIAS
  - *ALTER* - *SELECT*
- FUNCIONES, PAQUETES Y PROCEDIMIENTOS
  - *EXECUTE*

Nota: Consultar "Oracle Dat.SQL Reference" para lista completa privilegios sobre objetos.

OTORGAR PRIVILEGIOS SOBRE OBJETOS.

- Sintaxis:

GRANT <privilegio>/ALL PRIVILEGES <columna>/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!).

OTORGAR PRIVILEGIOS SOBRE OBJETOS.

- Ejemplos:

GRANT ALL PRIVILEGES ON nomina.retenciones
TO gestor WITH GRANT OPTION;

GRANT REFERENCES (dni_empleado), UPDATE (dni_empleado, sueldo_base) ON nomina.empleados TO gestor;

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 ha concedido.

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

ROLES

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 tener contraseña.

  - Su nombre es único en la bd, distinto a cualquier otro nombre de usuario o rol.

  - 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 asignados a un usuario pueden ser activados o desactivados temporalmente.
- *Mejora aplicaciones*. Cuando un usuario ejecuta una determinada aplicación puede activarse, o desctivarse, selectivamente roles en función de nuestro interés. Los roles también pueden protegerse con claves y estos activarse sólo si la aplicación suministra la correcta.
- *Mejora de la productividad*. El uso de roles disminuye el numero de "grants" almacenados en el diccionario de datos.

ROLES PREDEFINIDOS.

- Oracle proporciona roles predefinidos como ayuda a la administración de base de datos, entre los que se encuentran:
  - ***CONNECT***. Incluye sólo el privilegio *CREATE SESSION*.
  - ***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 realizar exportaciones completas e incrementales de la base de datos.
  - *IMP_FULL_DATABASE*. Idem para importaciones completas.

ROLES PREDEFINIDOS.

  - *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.
- Se recomienda crear roles específicos en cada bd y asignarles los permisos necesarios, evitando el uso de roles predefinidos, con lo que no surgirán problemas si estos quedan obsoletos en futuras versiones.

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.

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.*
- No se ven afectadas las sesiones en las que el rol está ya activo.
- 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;

- Para que un usuario pueda otorgar un rol debe habérsele concedido con *ADMIN OPTION*, poseer el privilegio *GRANT ANY ROLE*, o haberlo creado.

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

PARÁMETRO MAX_ENABLED_ROLES.

- Este parámetro está obsoleto y sólo se mantiene por compatibilidad. Es preferible NO usarlo.

- Es un parámetro de inicialización que define el numero máximo de roles de base de datos activos concurrentemente, incluyendo aquellos contenidos dentro de otros roles, que un usuario puede poseer.

- Un usuario puede activar como máximo 2+MAX_ENABLED_ROLES puesto que cada usuario tiene dos roles adicionales (PUBLIC y y el propio rol del usuario).

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. La claúsula puede sólo indicar roles otorgados directamente al usuario con una sentencia *GRANT*.
- 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* (pueden revocar cualquier rol).

- 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. Las sesiones en las que el rol está activo no se ven afectadas, pero ninguna otra lo podrá usar.

ACTIVACIÓN Y DESACTIVACIÓN DE ROLES.

- Durante una sesión, el usuario o una aplicación puede usar la sentencia *SET ROLE* para modificar los roles activos en la sesión. Previamente los roles deben haber sido asignados al usuario.

  Al crear un usuario todos los roles asignados son por defecto, a menos que se limite con *ALTER USER.*

- 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.

- La vista *SESSION_ROLES* informa de aquellos roles que, para el usuario actual, están activos en un momento determinado.

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.

- ***DATABASE_PROPERTIES*** *... Propiedades de la bd.*
- ***DBA_USERS*** *... Usuarios de la bd.*
- ***DBA_ROLES*** *... Roles existentes en la bd.*
- ***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 bd.*
- ***DBA_TABLESPACE_GROUPS*** *... Grupos de espacios temporales.*
- ***DBA_COL_PRIVS*** *... Permisos sobre columnas de objetos en bd.*
- ***DBA_TS_QUOTAS*** *... Cuotas de espacio para usuarios.*
- ***DBA_PROFILES*** *... Perfiles en la bd.*

VISTAS.

- ***ROLE_ROLE_PRIVS*** *... Roles concedidos a otros roles.*
- ***ROLE_SYS_PRIVS*** *... Privilegios de sistema concedidos a roles.*
- ***ROLE_TAB_PRIVS*** *... Privilegios sobre objetos concedidos a roles.*
- ***DBA_CONNECT_ROLE_GRANTEES*** *... Usuarios con privilegio CONNECT.*
- ***USER_PASSWORD_LIMITS*** *... Parámetros contraseña usuario.*
- ***USER_RESOURCE_LIMITS*** *... Límites de recursos por usuario.*
- ***SESSION_PRIVS*** *... Privilegios disponibles en la sesión.*
- ***SESSION_ROLES*** *... Roles activos en la sesión.*
- ***V$SESSION*** *... Información de sesión.*

TEMA 8.

TRABAJOS.

TEMA 8.
TRABAJOS (JOBS).

- DBMS_JOB vs. DBMS_SCHEDULER.
- MIGRACION A DBMS_SCHEDULER.
  - CREACION DE TRABAJOS.
  - MODIFICACION DE TRABAJOS.
  - BORRADO DE TRABAJOS.
- PLANIFICADOR (SCHEDULER).
- ARQUITECTURA DEL PLANIFICADOR.
- NOMENCLATURA OBJETOS.
- PRIVILEGIOS PLANIFICADOR.

TEMA 8.
TRABAJOS (JOBS).

- PROCEDIMIENTOS PLANIFICADOR.
  - "CREATE_JOB". INTERVALO EJECUCION.
  - "SET _ATTRIBUTE".
  - "SET _ATTRIBUTE_NULL".
  - "COPY".
  - "ENABLE".
  - "DISABLE".
  - "RUN_JOB".
  - "STOP_JOB".
  - "DROP_JOB".

- VISTAS.

DBMS_JOB vs. DBMS_SCHEDULER.

- Los trabajos son el resultado de la combinación de una planificación y un programa, p.ej. código PL/SQL, junto con los argumentos requeridos por dicho programa. Se lanzan a la cola de trabajos, especificando la periodicidad con que deben ser ejecutados.

- Las funciones de planificación se ofrecen a través del paquete *DBMS_SCHEDULER*, este reemplaza al paquete *DBMS_JOB*.

- Para gestionar la cola de trabajos se dispone, por tanto, del paquete *DBMS_SCHEDULER*. Aunque *DBMS_JOB* sigue estando disponible, no es probable que lo esté en un futuro.

- Dentro del paquete *DBMS_SCHEDULER* existen diversos procedimientos como *CREATE_JOB, DROP_JOB, STOP_JOB,* ... que permiten planificar los trabajos automatizados.

DBMS_JOB vs. DBMS_SCHEDULER.

- *DBMS_JOB* permite ejecutar sólo programas almacenados o bloques PL/SQL. *DBMS_SCHEDULER* puede ejecutar también ejecutables de S.O.

- En *DBMS_JOB* sólo hay un componente, el trabajo o "job". En *DBMS_SCHEDULER* hay múltiples componentes que incrementan la capacidad de planificación.

- Usando *DBMS_SCHEDULER* los intervalos de planificación pueden definirse en lenguaje natural y de forma más compleja que con *DBMS_JOB*, el cual sólo acepta expresiones tipo fecha.

- *DBMS_SCHEDULER* proporciona mayor detalle en cuanto al estado del trabajo y sus fallos, esta información puede consultarse en el diccionario de datos.

MIGRACION A DBMS_SCHEDULER (I). CREACION DE TRABAJOS.

- Usando DBMS_JOB:

```
VARIABLE jobno NUMBER;
BEGIN
        DBMS_JOB.SUBMIT(:jobno, 'INSERT INTO employees VALUES (7935, ''SALLY'', ''DOGAN'',
            ''sally.dogan@xyzcorp.com'', NULL, SYSDATE, ''AD_PRES'', NULL,NULL, NULL, NULL);',
            SYSDATE, 'SYSDATE+1');
        COMMIT;
END;
/
```

- Usando DBMS_SCHEDULER:

```
BEGIN
        DBMS_SCHEDULER.CREATE_JOB(
                job_name => 'job1',
                job_type => 'PLSQL_BLOCK',
                job_action => 'INSERT INTO employees VALUES (7935, ''SALLY'', ''DOGAN'',
                    ''sally.dogan@xyzcorp.com'', NULL, SYSDATE,''AD_PRES'', NULL, NULL, NULL,
                    NULL);');
                start_date => SYSDATE,
                repeat_interval => 'FREQ = DAILY; INTERVAL = 1');
END;
/
```

MIGRACION A DBMS_SCHEDULER (II). MODIFICACION DE TRABAJOS.

- Usando DBMS_JOB:

```
BEGIN
        DBMS_JOB.WHAT(31, 'INSERT INTO employees VALUES (7935, ''TOM'', ''DOGAN'',
            ''tom.dogan@xyzcorp.com'', NULL, SYSDATE,''AD_PRES'', NULL, NULL, NULL, NULL);');
        COMMIT;
END;
/
```

- Usando DBMS_SCHEDULER:

```
BEGIN
        DBMS_SCHEDULER.SET_ATTRIBUTE(
        name => 'JOB1',
        attribute => 'job_action',
        value => 'INSERT INTO employees VALUES (7935, ''TOM'', ''DOGAN'', ''tom.dogan@xyzcorp.com'',
            NULL, SYSDATE, ''AD_PRES'', NULL, NULL, NULL, NULL);');
END;
/
```

MIGRACION A DBMS_SCHEDULER (III). BORRADO DE TRABAJOS.

- Usando DBMS_JOB:

```
BEGIN
        DBMS_JOB.REMOVE(14144);
        COMMIT;
END;
/
```

- Usando DBMS_SCHEDULER:

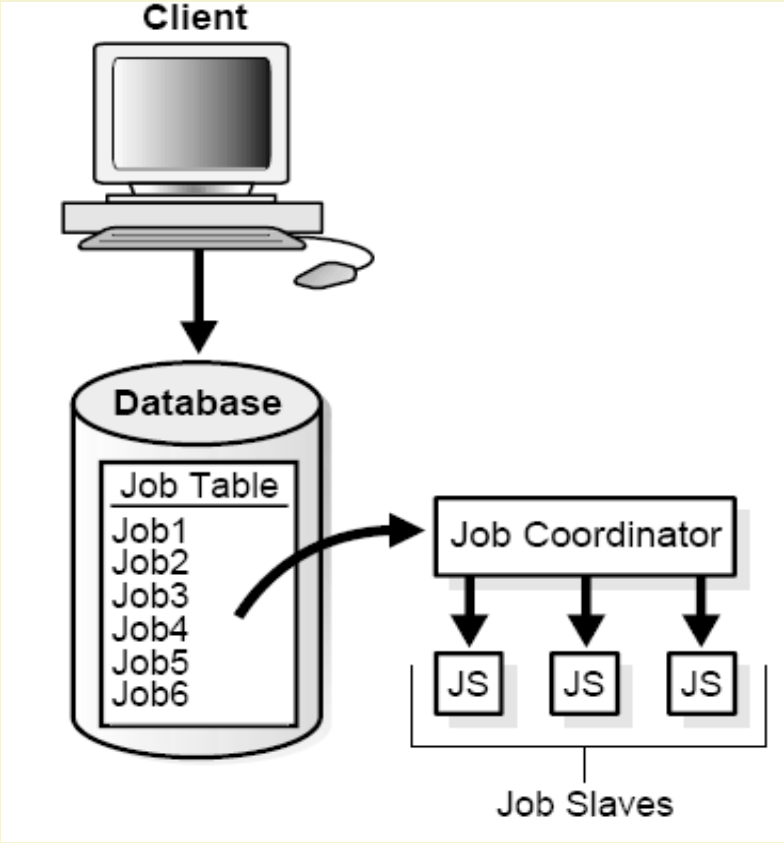
```
BEGIN
        DBMS_SCHEDULER.DROP_JOB('myjob1');
END;
/
```

PLANIFICADOR (SCHEDULER).

- Mediante el planificador puede controlarse cuándo y dónde se ejecutarán los trabajos automatizados. Permite, por ejemplo:
  - Planificar trabajos de mantenimiento de la bd como copias de seguridad u otras tareas a realizar durante horas de menor actividad.
  - Planificar la ejecución de trabajos en una fecha determinada o basándose en eventos.
  - Agrupar trabajos en clases y priorizar entre las mismas (incluso modificar la priorización en función del tiempo).
  - Gestionar y monitorizar trabajos.

ARQUITECTURA DEL PLANIFICADOR.

- En la arquitectura del planificador se distinguen distintos componentes:
  - Tabla de trabajos.
  - Proceso coordinador.
  - Procesos esclavos.



ARQUITECTURA DEL PLANIFICADOR.

- Tabla de trabajos.

  Hay una por base de datos y almacena informacion tal como el propietario, nivel de "log" o si el trabajo se elimina al finalizar – opción por defecto – (visible mediante la vista *DBA_SCHEDULER_JOBS* o las análogas *ALL_...* y *USER_...).*

- Proceso coordinador.

  Proceso "background", *cjqNNN*, que arranca automáticamente cuando un trabajo debe ejecutarse y se desactiva tras un periodo de inactividad. Se encarga de:

  – Controlar y crear los procesos esclavos.
  – Consultar la tabla de trabajos.
  – Coger los trabajos de la tabla y colocarlos en memoria cache. Darlos a los esclavos para su ejecución.

ARQUITECTURA DEL PLANIFICADOR.

- Limpiar el conjunto de esclavos cuando no son necesarios.
- Desactivarse cuando no hay trabajos planificados.
- Despertar cuando debe ejecutarse un nuevo trabajo o se ha creado un trabajo.
- Recuperar los trabajos en ejecución tras el arranque posterior a un cierre anormal de la bd (p.ej. caida del sistema o "shutdown abort").

Existe sólo un proceso coordinador por instancia.

- Procesos esclavos.

Su número es ajustado automáticamente por el planificador. Son despertados por el coordinador cuando hay trabajos por ejecutar. Se encargan de:

- Ejecutar el trabajo y realizar las operaciones asociadas

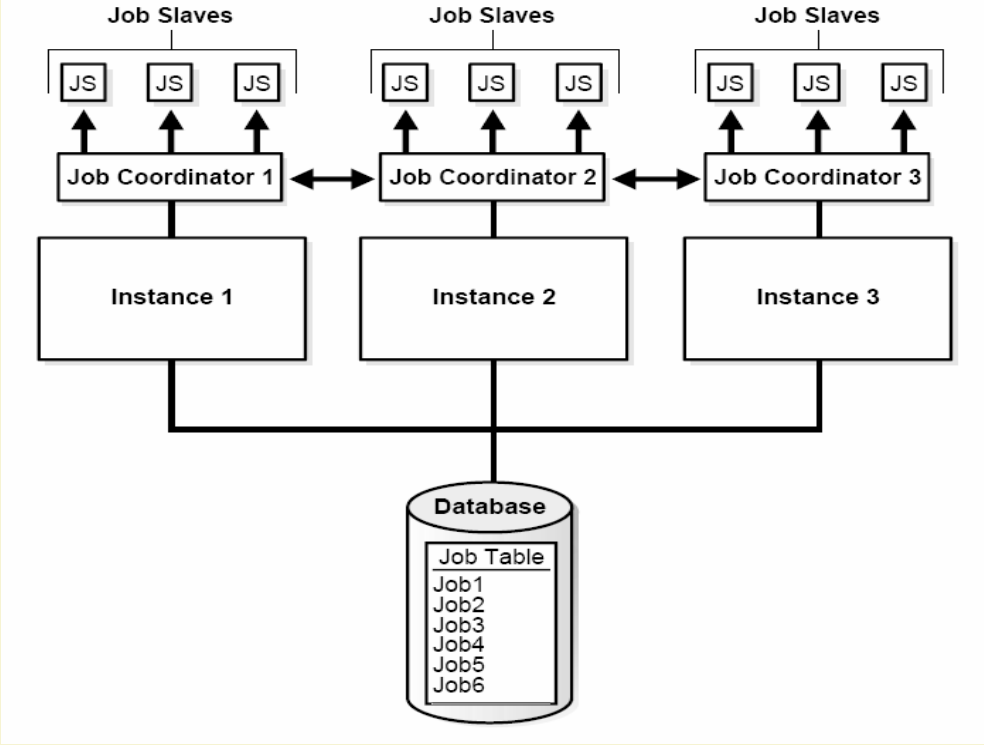
ARQUITECTURA DEL PLANIFICADOR.

(abrir sesión, comenzar la transacción, comenzar la ejecución, finalizar la transacción y cerrar sesión).

- Actualizar el estado del trabajo en la tabla de trabajos reflejando el estado del trabajo.
- Actualizar la cuenta de ejecuciones o fallos del trabajo.
- Buscar nuevos trabajos a ejecutar (o ir a dormir si no hay ninguno).
- ...

PLANIFICADOR. RAC.

- El planificador usa una tabla de trabajos por cada base de datos y un proceso coordinador para cada instancia. Los coordinadores comunican entre sí para guardar información actualizada.



NOMENCLATURA OBJETOS.

- Los objetos del planificador se nombran de la misma forma que cualquier otro objeto de la base de datos: "*[esquema_usuario].nombre"*. Los objetos deben ser únicos en el espacio de nombres.

- Por defecto, los nombres de objetos se almacenan en mayúsculas a menos que estén encerrados en dobles comillas.

PRIVILEGIOS PLANIFICADOR.

- Para administrar el planificador debe poseerse el rol *SCHEDULER_ADMIN* (generalmente sólo administradores).

- Es necesario ser restrictivos con la concesión de estos permisos, siendo aconsejable la concesión de privilegios concretos, por ejemplo:

  GRANT CREATE JOB TO scott; -crear trabajos,... en su esquema-

  GRANT ALTER myjob1 TO scott;

- Una alternativa para administrar el planificador es el privilegio *MANAGE SCHEDULER*. Por ejemplo:

  GRANT MANAGE SCHEDULER TO <usuario>;

  El usuario podrá crear, modificar, o borrar ventanas de ejecucion, clases de trabajo o grupos de ventanas.

PRIVILEGIOS PLANIFICADOR.

JOB SCHEDULER OBJECTS:	The following privileges are needed to execute procedures in the DBMS_SCHEDULER package.
CREATE JOB	Create jobs, schedules, or programs in the grantee's schema.
CREATE ANY JOB	Create, alter, or drop jobs, schedules, or programs in any schema. Note: This extremely powerful privilege allows the grantee to execute code as any other user. It should be granted with caution.
CREATE EXTERNAL JOB	Create in the grantee's schema an executable scheduler job that runs on the operating system.
EXECUTE ANY PROGRAM	Use any program in a job in the grantee's schema.
EXECUTE ANY CLASS	Specify any job class in a job in the grantee's schema.
MANAGE SCHEDULER	Create, alter, or drop any job class, window, or window group.

PRIVILEGIOS PLANIFICADOR.

Privilege Name	Operations Authorized
System Privileges:	
CREATE JOB	This privilege enables you to create jobs, schedules, and programs in your own schema. You will always be able to alter and drop jobs, schedules and programs in your own schema, even if you do not have the CREATE JOB privilege. In this case, the job must have been created in your schema by another user with the CREATE ANY JOB privilege.
CREATE ANY JOB	This privilege enables you to create, alter, and drop jobs, schedules, and programs in any schema. This privilege is very powerful and should be used with care because it allows the grantee to execute code as any other user.
EXECUTE ANY PROGRAM	This privilege enables your jobs to use programs from any schema.
EXECUTE ANY CLASS	This privilege enables your jobs to run under any job class.
MANAGE SCHEDULER	This is the most important privilege for administering the Scheduler. It enables you to create, alter, and drop job classes, windows, and window groups. It also enables you to set and retrieve Scheduler attributes and purge Scheduler logs.
Object Privilege:	
EXECUTE	This privilege enables you to create a job which runs with the program or job class. It also enables you to view object attributes. It can only be granted for programs and job classes.

PRIVILEGIOS PLANIFICADOR.

ALTER	This privilege enables you to alter or drop the object it is granted on. Altering includes such operations as enabling, disabling, defining or dropping program arguments, setting or resetting job argument values and running a job. For programs and jobs, this privilege enables you to view object attributes. This privilege can only be granted on jobs, programs and schedules. For other types of Scheduler objects, you can grant the MANAGE SCHEDULER system privilege. This privilege can be granted for: jobs (DROP_JOB, RUN_JOB, SET_JOB_ARGUMENT_VALUE, RESET_JOB_ARGUMENT_VALUE, SET_JOB_ANYDATA_VALUE) and (STOP_JOB without the force option) programs (DROP_PROGRAM, DEFINE_PROGRAM_ARGUMENT, DEFINE_ANYDATA_ARGUMENT, DEFINE_METADATA_ARGUMENT, DROP_PROGRAM_ARGUMENT, GET_ATTRIBUTE, SET_ATTRIBUTE, SET_ATTRIBUTE_NULL) schedules (DROP_SCHEDULE)
ALL	This privilege authorizes operations allowed by all other object attributes possible for a given object. It can be granted on jobs, programs, schedules and job classes.
SCHEDULER_ADMIN:	
All Pre-Defined Roles	The SCHEDULER_ADMIN role is created with all of the preceding system privileges (with the ADMIN option). The SCHEDULER_ADMIN role is granted to dba (with the ADMIN option).

PROCEDIMIENTOS PLANIFICADOR.

Task	Procedure	Privilege Needed
Create a job	CREATE_JOB	CREATE JOB or CREATE ANY JOB
Alter a job	SET_ATTRIBUTE	ALTER or CREATE ANY JOB or be the owner
Run a job	RUN_JOB	ALTER or CREATE ANY JOB or be the owner
Copy a job	COPY_JOB	ALTER or CREATE ANY JOB or be the owner
Drop a job	DROP_JOB	ALTER or CREATE ANY JOB or be the owner
Stop a job	STOP_JOB	ALTER or CREATE ANY JOB or be the owner
Disable a job	DISABLE	ALTER or CREATE ANY JOB or be the owner
Enable a job	ENABLE	ALTER or CREATE ANY JOB or be the owner

DBMS_SCHEDULER. PROCEDIMIENTO "CREATE_JOB".

- Debe indicarse la acción, planificación y atributos del trabajo.
- El propietario del trabajo es el usuario en cuyo esquema se ha creado, el creador del trabajo es el usuario que ha creado el mismo. Un trabajo puede crearse en otro esquema indicando "*esquema.nombre_trabajo*".
- Los trabajos se ejecutan con los privilegios del esquema en el cual se crean. El entorno del trabajo cuando se ejecuta es aquel que existía en el momento de su creación.
- Cualquier trabajo puede consultarse una vez creado usando las vistas **_SCHEDULER_JOBS*. Por defecto los trabajos se crean deshabilitados y necesitan activarse para ser ejecutados.

DBMS_SCHEDULER. PROCEDIMIENTO "CREATE_JOB".

- Sintaxis:

```
DBMS_SCHEDULER.CREATE_JOB (
    job_name IN VARCHAR2,
    job_type IN VARCHAR2,
    job_action IN VARCHAR2,
    number_of_arguments IN PLS_INTEGER DEFAULT 0,
    start_date IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
    repeat_interval IN VARCHAR2 DEFAULT NULL,
    end_date IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
    job_class IN VARCHAR2 DEFAULT 'DEFAULT_JOB_CLASS',
    enabled IN BOOLEAN DEFAULT FALSE,
    auto_drop IN BOOLEAN DEFAULT TRUE,
    comments IN VARCHAR2 DEFAULT NULL);
```

DBMS_SCHEDULER. PROCEDIMIENTO "CREATE_JOB".

- *job_name*

  Identificador unívoco del trabajo. Si el trabajo reside en otro esquema debe indicarse el nombre de esquema. Para generar el nombre puede usarse el procedimiento *"GENERATE_JOB_NAME"*

  DBMS_SCHEDULER.GENERATE_JOB_NAME (prefijo IN VARCHAR2 DEFAULT 'JOB$_') RETURN VARCHAR2;

- *job_type*

  Tipo de trabajo creado. Algunos de los tipos soportados son:

  a) '*PLSQL_BLOCK*'. Bloque PL/SQL. En este caso no pueden indicarse argumentos.

  b) '*STORED_PROCEDURE*'. Procedimiento almacenado o subprograma C externo (sólo procedimientos, no funciones con valor de retorno, son soportados).

DBMS_SCHEDULER. PROCEDIMIENTO "CREATE_JOB".

- *job_action*

  Para un bloque PL/SQL, la acción es es ejecutar un código PL/SQL code. Estos bloques deben acabar con un "*;*" ("*my_proc();*" o "*BEGIN my_proc(); END;*" ...)

  Para un procedimiento almacenado la acción es el nombre del mismo.

- *number_of_arguments*. Número de argumentos para el trabajo (0-255, por defecto 0).

- *program_name*. Nombre del programa asociado altrabajo.

- *start_date*. Fecha en que el trabajo arrancará. Si "start_date" y "repeat_interval" son nulos, el trabajo se inicia tan pronto sea habilitado.

- *event_condition*. Expresión de eventos.

DBMS_SCHEDULER. PROCEDIMIENTO "CREATE_JOB".

- *queue_spec.* Cola de eventos.
- *repeat_interval.* Intervalo de ejecución. Si no se indica el trabajo sólo se ejecuta una vez.
- *schedule_name.* Nombre de la planificación, window, o window group asociada al trabajo.
- *end_date.* Fecha tras la que el trabajo no se ejecutará más (el STATE del trabajo se asigna a COMPLETED, y se deshabilita). Si no se indica fecha, el trabajo se repetirá indefinidamente a menos que se alcance "max_runs" o "max_failures" en cuyo caso el trabajo para.
- *job_priority.* Prioridad del trabajo entre los que integran la clase (de 1 -máxima- a 5 -mínima-, por defecto 3).
- *comments.* Comentarios.

DBMS_SCHEDULER. PROCEDIMIENTO "CREATE_JOB".

- *enabled.* Indica si el trabajo se crea habilitado o no (TRUE o FALSE). Por defecto su valor es FALSE, por tanto el planificador lo ignora y no se envía a ejecución. Para ser ejecutado el argumento debe tener valor TRUE o ser activado con el *procedimiento ENABLE*.

- *auto_drop*. Si su valor es TRUE, el trabajo será borrado tras ser deshabilitado o completarse su ejecución (cuando llega a su fecha final, alcanza el número de ejecuciones indicadas por "*max_runs*" -fijadas con *SET_ATTRIBUTE*- o sólo debe ejecutarse una vez). Un trabajo se deshabilita cuando falla las veces indicadas por "*max_failures* times" (fijadas con *SET_ATTRIBUTE*).

  Si el valor es FALSE, el trabajo no se borra hasta hacerlo explícitamente con el *procedimiento DROP_JOB*.

  Por defecto los trabajos se crean con "*auto_drop*" igual a TRUE.

DBMS_SCHEDULER. PROCEDIMIENTO "CREATE_JOB".

- Ejemplo:

```
BEGIN
    DBMS_SCHEDULER.CREATE_JOB (job_name => 'my_emp_job1',
    job_type => 'PLSQL_BLOCK',
    job_action => 'INSERT INTO sales VALUES( 7987, ''SALLY'',
       ''ANALYST'', NULL, NULL, NULL, NULL, NULL);',
    start_date => '28-APR-03 07.00.00 PM Australia/Sydney',
    repeat_interval => 'FREQ=DAILY;INTERVAL=2',
    end_date => '20-NOV-04 07.00.00 PM Australia/Sydney',
    comments => 'My new job');
END;
/
```

INTERVALO EJECUCION.

- Está definido por el valor del atributo "*repeat_interval*" o por el de la planificación que referencia el trabajo. Si no se indica valor para "*repeat_interval*" el trabajo sólo se ejecuta una vez (indicada en "*start date*").

- Puede indicarse de dos formas:

  - Mediante una expresión acorde a "Scheduler Calendaring Syntax" (véase *PL/SQL Packages and Types Reference* para una descripción detallada). Por ejemplo:

    a) Ejecución todos los viernes (los ejemplos son equivalentes).

    FREQ=DAILY; BYDAY=FRI;
    FREQ=WEEKLY; BYDAY=FRI;
    FREQ=YEARLY; BYDAY=FRI;

INTERVALO EJECUCION.

b) Ultimo día del mes.

FREQ=MONTHLY; BYMONTHDAY=-1;

c) Ejecución el 10 de marzo.

FREQ=YEARLY; BYMONTH=MAR; BYMONTHDAY=10;

FREQ=YEARLY; BYDATE=0310;

d) Cada 10 dias.

FREQ=DAILY; INTERVAL=10;

e) Diariamente a las 4, 5 y 6 de la tarde.

FREQ=DAILY; BYHOUR=16,17,18;

f) Cada 50 horas.

FREQ=HOURLY; INTERVAL=50;

g) Cada hora durante los tres primeros días del mes.

FREQ=HOURLY; BYMONTHDAY=1,2,3;

INTERVALO EJECUCION.

- Mediante una expresión PL/SQL . Por ejemplo:

```
BEGIN
      DBMS_SCHEDULER.CREATE_JOB (
      job_name => 'scott.my_job2',
      start_date => '15-JUL-04 01.00.00 AM Europe/Warsaw',
      repeat_interval => 'SYSTIMESTAMP + INTERVAL '30' MINUTE',
      end_date => '15-SEP-04 01.00.00 AM Europe/Warsaw',
      comments => 'Comentarios');
END;
/
```

Supone la ejecución por primera vez el 15 de julio y luego cada 30 minutos hasta el 15 de septiembre.

DBMS_SCHEDULER. PROCEDIMIENTO "SET _ATTRIBUTE".

- Sintaxis:

DBMS_SCHEDULER.SET_ATTRIBUTE (
name IN VARCHAR2,
attribute IN VARCHAR2,
value IN [VARCHAR2, TIMESTAMP WITH TIMEZONE,
PLS_INTEGER, BOOLEAN, INTERVAL DAY TO SECOND],
<value2 IN VARCHAR2 ***DEFAULT NULL****>);*

Name ... nombre del objeto.

Attribute ... atributo a modificar.

Value ... nuevo valor (no puede ser NULL). Para fijar un atributo a *NULL* debe usarse el procedimiento *SET_ATTRIBUTE_NULL*.

Value2 ... Usado en atributos que pueden tener dos valores asociados.

Cualquier parámetro, excepto job_name, puede modificarse.

DBMS_SCHEDULER. PROCEDIMIENTO "SET _ATTRIBUTE".

- Algunos de los valores que pueden modificarse son:
  - *logging_level*. Indica qué información se guarda: *DBMS_SCHEDULER.LOGGING_OFF* -ninguna-, *DBMS_SCHEDULER.LOGGING_RUNS* -información sobre ejecuciones- y *DBMS_SCHEDULER.LOGGING_FULL* -información sobre creación, habilitación, modificación, ...-.
  - *restartable*. Indica si un trabajo se rearranca en caso de fallo. Por defecto su valor es FALSE.
  - *max_failures*. Número de veces que un trabajo puede fallar consecutivamente antes de ser deshabilitado (en este caso su STATE es puesto a BROKEN). Por defecto NULL, nuevas instancias del trabajo arrancan independientemente del número de fallos previos.
  - *max_runs*. Número máximo de ejecuciones consecutivas. Al alcanzarlas, el trabajo se deshabilita y su estado pasa a ser COMPLETED. Por defecto es NULL, con lo que el trabajo se repite indefinidamente o hasta alcanzar "end_date" o "max_failures".

DBMS_SCHEDULER. PROCEDIMIENTO "SET _ATTRIBUTE".

- *job_action*
- *job_type*
- *repeat_interval*
- *start_date*
- *end_date*
- *comments*
- *auto_drop*. Indica si el trabajo debe ser borrado después de completar su ejecución.

DBMS_SCHEDULER. PROC. "SET _ATTRIBUTE_NULL".

- Sintaxis:

```
DBMS_SCHEDULER.SET_ATTRIBUTE_NULL (
    name IN VARCHAR2,
    attribute IN VARCHAR2);
```

Este procedimiento asigna el valor NULL a un atributo.

DBMS_SCHEDULER. PROCEDIMIENTO "COPY".

- Sintaxis:

```
DBMS_SCHEDULER.COPY_JOB (
    old_job IN VARCHAR2,
    new_job IN VARCHAR2);
```

- Copia todos los atributos de un trabajo existente a un nuevo trabajo, este es creado deshabilitado -el estado del trabajo original no varía-.

- Esta operación requiere tener privilegios para crear un trabajo en el esquema del nuevo trabajo (CREATE JOB para el propio esquema o CREATE ANY JOB en otro caso). Si el trabajo original no está en el propio esquema de usuario se necesita también privilegio ALTER en el mismo o
- poseer el privilegio CREATE ANY JOB.

DBMS_SCHEDULER. PROCEDIMIENTO "ENABLE".

- Sintaxis:

DBMS_SCHEDULER.ENABLE (name IN VARCHAR2);

- Permite activar un trabajo, programa, ... todos los cuales, por defecto, son creados deshabilitados. A partir de la activación el proceso coordinador puede llevarlo a ejecución.

- Se llevan a cabo operaciones de validación previas a la habilitación. Si fallan, no se habilita el objeto y se genera un error.

- Se debe ser el propietario del objeto, tener privilegio *ALTER* en el mismo o poseer el privilegio *CREATE ANY JOB*.

DBMS_SCHEDULER. PROCEDIMIENTO "DISABLE".

- Sintaxis:

```
DBMS_SCHEDULER.DISABLE (
    name IN VARCHAR2,
    force IN BOOLEAN DEFAULT FALSE);
```

- Permite desactivar un trabajo, programa, ... puede indicarse en "name" una lista de nombres separados por comas. Deshabilitar un objeto ya deshabilitado no genera error.

- Se debe ser el propietario del objeto, tener privilegio *ALTER* en el mismo o poseer el privilegio *CREATE ANY JOB*.

- Cuando un trabajo se deshabilita su estado se modifica a "*disabled*". Si "*force*" es *FALSE* y el trabajo está ejecutándose se develve un error; si "*force*" es *TRUE*, se deshabilita el trabajo pero se permite finalizar la instancia del mismo que está ejecutándose.

DBMS_SCHEDULER. PROCEDIMIENTO "RUN_JOB".

- Sintaxis:

```
DBMS_SCHEDULER.RUN_JOB (
    job_name IN VARCHAR2,
    use_current_session IN BOOLEAN DEFAULT TRUE);
```

Ejecuta un trabajo de forma inmediata. Usar *RUN_JOB* requiere bien ser el propietario del trabajo, bien tener privilegio *ALTER* en dicho trabajo. También pueden ejecutarse trabajos con el privilegio *CREATE ANY JOB*.

- Si *use_current_session* es *TRUE*, el trabajo se ejecuta en la sesion que llama a *RUN_JOB*. Esta forma de ejecutarse no hace que se modifique los contadores "*failure_count*" y "*run_count*", el trabajo se reflejará en el "log" del mismo.

- Si *use_current_session* es *FALSE*: es preciso chequear el "log" para obtener información sobre la ejecución del trabajo; "*run_count*", "*last_start_date*", "*last_run_duration*" y "*failure_count*" son actualizados y *RUN_JOB* falla si hay un trabajo planificado ejecutándose.

DBMS_SCHEDULER. PROCEDIMIENTO "STOP_JOB".

- Sintaxis:

```
DBMS_SCHEDULER.STOP_JOB (
    job_name IN VARCHAR2
    force IN BOOLEAN DEFAULT FALSE);
```

- Permite parar trabajos en ejecución. Tras la parada de un trabajo, el estado de este pasa a ser *STOPPED* si sólo se ejecutaba una vez; mientras que el de uno programado para repetirse múltiples veces pasa a ser *SCHEDULED* o *COMPLETED* (en función de si la siguiente ejecución estaba planificada).

- Si el argumento "force" de la llamada es *FALSE*, el planificador intenta parar el trabajo ordenadamente y genera un error si falla en su intento. Si su valor es *TRUE*, se interrumpe inmediatamente el proceso esclavo (Oracle sólo recomienda esta opción si ha fallado la parada ordenada).

DBMS_SCHEDULER. PROCEDIMIENTO "DROP_JOB".

- Sintaxis:

```
DBMS_SCHEDULER.DROP_JOB (
    job_name IN VARCHAR2,
    force IN BOOLEAN DEFAULT FALSE);
```

- Permite borrar un trabajo (desaparece de cualquier vista, por tanto, y no vuelve a ejecutarse).
- Requiere ser el propietario del trabajo, tener el privilegio *ALTER* sobre el mismo, o poseer el privilegio de sistema *CREATE ANY JOB*.
- Si el argumento "*force*" es *FALSE*, y una instancia del trabajo está ejecutándose se genera un error. Si su valor es *TRUE*, el planificador intenta parar la instancia del trabajo y después lo borra.

VISTAS.

- ***DBA_SCHEDULER_JOB_ARGS***. Argumentos de trabajos.
- ***DBA_SCHEDULER_JOB_LOG***. Cambios de estado de trabajos.
- ***DBA_SCHEDULER_JOB_RUN_DETAILS***. Detalles sobre la ejecución de trabajos.
- ***DBA_SCHEDULER_JOBS***. Información sobre trabajos en la bd.
- ***DBA_SCHEDULER_RUNNING_JOBS***. Trabajos en ejecución.
- ***DBA_SCHEDULER_SCHEDULES***. Planificación de trabajos.
- **SESSION_PRIVS**. Privilegios de sistema actuales.
- ***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”.
- Parámetro “audit_trail”.
- Auditoría de usuarios administradores.
- Información “audit trail” de sistema operativo.
- Auditoría de sentencias. opciones.
- Auditoría de privilegios. Privilegios auditables.

TEMA 9.
AUDITORÍA.

- Auditoría de esquema. Opciones.
- Auditoría y “flashback”.
- Auditoría de grano fino. Paquete dbms_fga. Procedimientos.
- Desactivación. Sentencia noaudit.
- Desactivación de la auditoría.
- Control y protección del “audit trail”.
- Auditoría. Recomendaciones.
- Vistas.

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, ..)
  - Detectar privilegios incorrectamente otorgados a usuarios (que permiten realizar acciones inapropiadas, las cuales son detectadas)
  - Recoger datos sobre actividades concretas (tablas que se actualizan, usuarios concurrentes, …)
  - Detectar problemas con la implementación de políticas de seguridad (puntos débiles que generan registros)
- Puede ser más o menos general, permitiendo auditar:
  - Ejecuciones de sentencias exitosas, fallidas o ambas.
  - Ejecución de sentencias por sesión o por lanzamiento de sentencia.
  - Usuarios concretos o todos los usuarios.

TIPOS DE AUDITORÍA.

- Existen varios 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* que audita *create table, alter table* y *drop 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>*).
  - De grano fino ("fine grained"). Acceso a datos concretos y cambios en los mismos a nivel columna.

TIPOS DE AUDITORÍA.

- Si se habilitan opciones de auditoría semejantes de sentencia y de privilegio, sólo se genera un registro; por ej. la auditoría de *TABLE* -sentencia- y *CREATE TABLE* -privilegio- sólo genera un registro al crear una tabla. La auditoría de privilegios es más restringida que la de sentencias pues audita sentencias específicas.

REGISTROS DE AUDITORÍA. "AUDIT TRAIL".

- La información de auditoría se almacena en los registros de auditoría, que incluyen datos como:
  - Usuario. Identificador de sesión y terminal
  - Nombre del esquema accedido
  - Operación y código de operación
  - Fecha y hora
- En la auditoría básica (sentencias, privilegios, esquemas) los registros se guardan en la tabla de diccionario de datos SYS.AUD$ ("db audit trail") o en ficheros de sistema operativo ("operating system audit trail"). Está codificada y no es legible. En el primer caso, existen diferentes vistas que permiten usar la información almacenada como *DBA_AUDIT_TRAIL*.
- En la auditoría de grano fino, los registros se escriben en *DBA_FGA_AUDIT_TRAIL* (tabla *SYS.FGA_LOG$*).

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 como acabe la transacción del usuario; si esta es deshecha ("rollback"), el registro de auditoría permanece ("commit").

- Las opciones de auditoría para sentencias y privilegios en vigor cuando un usuario se conecta permanecen así durante la sesión (auque se modifiquen o establezcan nuevas opciones, que sólo tendrán efecto en una nueva sesión). Por el contrario, cambios en las opciones sobre esquema se aplican inmediatamente para la sesión en curso.

REGISTROS DE AUDITORÍA. "AUDIT TRAIL".

- La auditoría de grano fino se aplica a objetos individuales y, por tanto, no se ve afectada por la auditoría básica que está habilitada o no para la bd al completo.
- 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 al auditar la sentencia *UPDATE (*puede hacerse usando disparadores de base de datos, "*triggers*").

```
CREATE TRIGGER audit_emp_salaries
AFTER INSERT OR DELETE OR UPDATE ON employee_salaries
for each row
      begin
      if (:new.salary> :old.salary * 1.10) then insert into emp_salary_audit
         values (:employee_no,:old.salary,:new.salary,user,sysdate);
      endif;
end;
```

REGISTROS DE AUDITORÍA. "AUDIT TRAIL".

- Independientemente de si la auditoría esta habilitada o no, 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. El "audit trail" de bd no está disponible hasta que se ha arrancado satisfactoriamente esta.
  - "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, estático, *AUDIT_TRAIL* el valor *DB* (auditoría en base de datos excepto para los valores que siempre se escriben a s.o.). 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 *$ORACLE_BASE/$DB_UNIQUE_NAME/adump* y, en segundo lugar, *$ORACLE_HOME/rdbms/audit).*
- El valor *XML* indica que los registros se escriben como ficheros xml en el s.o.. Pueden consultarse mediante la vista V*$XML_AUDIT_TRAIL.*
- 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*".
- Habilitar la auditoría de grano fino o la del SYS no precisa indicar valor para *AUDIT_TRAIL*.

AUDITORIA DE USUARIOS ADMINISTRADORES.

- El parámetro de inicialización *AUDIT_SYS_OPERATIONS* - estático- permite especificar la auditoría de aquellas sesiones de usuarios conectados como SYS (privilegio SYSDBA). Los registros generados son escritos en el "audit trail" de sistema operativo, no depende del valor del parámetro AUDIT_TRAIL.

- 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).*

AUDITORÍA BD vs SO.

- 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" (como *dba_audit_trail*).
  - Pueden usarse herramientas Oracle de generación de informes, como Oracle Reports.
  - Evita la ejecución de eventos si el "audit trail" no puede grabar los registros que se generan.
- Por otra parte, la auditoría en sistema operativo podría 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. También podría ser más seguro al requerir permisos concretos para acceder a los ficheros.

INFORMACION "AUDIT TRAIL" DE SISTEMA OPERATIVO.

- Los registros de auditoría escritos a ficheros de sistema operativo contienen información codificada que incluye los campos:
  - Código de operación ("action code"). Debe consultarse la tabla *AUDIT_ACTIONS*, esta muestra la descripción de los códigos de operación.
  - 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.

- Nota general: Si se configura la auditoría por s.o., la cláusula *BY SESSION* equivale a *BY ACCESS*. Se genera un registro cada vez que se realiza un acceso.

- Nota general: Para cualquier tipo de auditoría (sentencia, privilegio o sistema), si se opta por auditar "NOT SUCCESSFUL" se generan registros sólo si el fallo se produce por alguna razón relacionada con la opción auditada. Por ejemplo, no se produce registro si una sentencia falla por no tener cuota estando auditando CREATE TABLE.

AUDITORÍA DE SENTENCIAS. OPCIONES.

Opcion	Sentencias SQL auditadas.
Database link	Create database link / drop database link
Index	Create / alter / analyze /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.

- Toda la auditoría de privilegios sobre sentencias DDL se hace por acceso. Por defecto es por sesión.

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 *ALTER ANY TABLE / ANY INDEX* *DELETE ANY TABLE* *DROP ANY TABLE / ANY INDEX* *INSERT ANY TABLE* *UPDATE ANY TABLE* *SELECT ANY TABLE*	*CREATE USER* *ALTER USER* *DROP USER* *CREATE VIEW* *CREATE ANY VIEW* *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.

- Puede auditarse todas las sentencias SELECT y DML permitidas por los privilegios de objetos.
- No pueden auditarse procedimientos incluidos en paquetes, aunque sí procedimientos y funciones individuales y también paquetes.
- Si se habilita afecta a todos los usuarios de la bd.

AUDITORÍA DE ESQUEMA. OPCIONES.

- Tablas.
  - Alter / audit / comment / delete / grant / index / insert / lock rename / select /update / flashback (sólo a consultas flashback)
- Vistas.
  - Audit / comment / delete / grant / insert / lock rename / select /update / flashback (sólo a consultas flashback)
- Secuencias.
  - Alter / audit / grant / select
- Procedimientos.
  - Audit / execute / grant / rename
- Vistas materializadas.
  - Alter / audit / comment / delete / index / insert / lock / select /update

AUDITORÍA Y "FLASHBACK".

- Para usar las prestaciones "flashback", la bd debe estar en modo automático de "undo".
- La vista *FLASHBACK_TRANSACTION_QUERY* muestra información acerca de las transacciones realizadas en la bd. Si se han modificado datos en una tabla, la vista muestra información sobre los mismos y señala la sentencia precisa (campo *UNDO_SQL*) que permite deshacerlos.
- "Flashback Version Query" hace posible ver todos los cambios producidos en una fila de una tabla en un periodo de tiempo concreto. FVQ permite añadir una cláusula *VERSIONS* a la sentencia *SELECT* que indique un rango SCN o temporal en el que se desea consultar las modificaciones. La sentencia también puede devolver información sobre la transacción responsable del cambio (*VERSIONS_XID*).

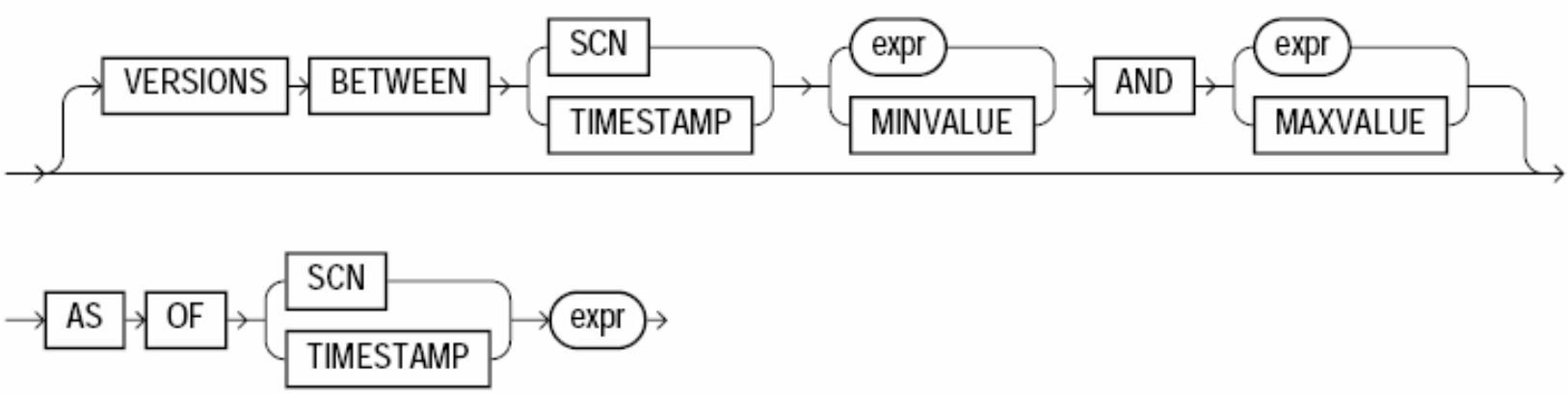
AUDITORÍA Y "FLASHBACK".

- Para ejecutar "flashback query" es necesario el privilegio *SELECT* sobre los objetos, además bien el privilegio *FLASHBACK* sobre los objetos, bien *FLASHBACK ANY TABLE* (permite ejecutarla sobre cualquier tabla, vista o vista materializada en cualquier esquema).

- Puede recuperarse información a través de distintas pseudocolumnas :
  - VERSIONS_STARTTIME. Fecha de la primera versión de las filas devueltas por la consulta.
  - VERSIONS_STARTSCN. Idem para SCN.
  - VERSIONS_ENDTIME. Fecha de la última versión de las filas devueltas por la consulta.
  - VERSIONS_ENDSCN. Idem para SCN
  - VERSIONS_XID. Transacción que generó la versión.
  - VERSIONS_OPERATION. Operación que dió lugar a la versión (I -insert-, U -update- o D -delete-).

AUDITORÍA Y "FLASHBACK".

- Clausula *FLASHBACK:*



- ***AS OF*** pemite recuperar la version de las filas a un momento en el tiempo o SCN particular.

- ***VERSIONS*** permite recuperar múltiples versiones validades de las filas entre dos SCN o momentos en el tiempo. MINVALUE y MAXVALUE se refieren a los datos más antiguo y más reciente respectivamente.

AUDITORÍA Y RECUPERACION DE TABLAS. "FLASHBACK TABLE".

- Es posible restaurar una tabla a un estado anterior a aquel donde tuvieron lugar una serie de cambios mediante la sentencia *FLASHBACK TABLE.*

Nota: Esta posibilidad será analizada en relacion a los mecanismos de copias de seguridad.

AUDITORÍA DE GRANO FINO.

- Para llevarla a cabo NO es preciso habilitar la auditoria básica ("audit trail").
- Permite monitorizar accesos a datos basándose en su contenido y auditar sentencias SELECT y DML en tablas y vistas mediante la creación de una política de auditoría al efecto.
- Se usa el paquete *DBMS_FGA* y sus procedimientos asociados, generándose apuntes en el "audit trail" de grano fino (*SYS.FGA_LOG$,* accesible a través de la vista *DBA_FGA_AUDIT_TRAIL*).
- Este tipo de auditoría permite indicar columnas singulares, relevantes, que precisan ser auditadas (por ejemplo: dni, ayudas sociales, datos de salud, ...) -caso de no hacerlo se aplica a todas las columnas-; así como establecer acciones a llevar a cabo si se produce el acceso conforme a la política de auditoría implementada.

AUDITORÍA DE GRANO FINO.

- Proporciona las siguientes funcionalidades:
  - Usar políticas diferenciadas para INSERT, UPDATE, DELETE, y SELECT; y poder tener varias de ellas asociadas a cada tabla.
  - Activar la auditoria sólo cuando es necesaria (por ejemplo, con información crítica como el salario o datos de salud) y sólo si es referenciada una columna concreta.

AUDITORÍA DE GRANO FINO.

- En el ejemplo se auditan las sentencias INSERT, UPDATE, DELETE, y SELECT en la tabla "*hr.emp*", controlando cualquier acceso a la columna "*salary*" de empleados pertenecientes al departamento "sales":

```
DBMS_FGA.ADD_POLICY(
    object_schema => 'hr',
    object_name => 'emp',
    policy_name => 'chk_hr_emp',
    audit_condition => 'dept = ''SALES'' ',
    audit_column => 'salary'
    statement_types => 'insert,update,delete,select');
```

- Cualquiera de las sentencias siguientes genera un registro:

```
SELECT count(*) FROM hr.emp WHERE dept = 'SALES' and salary > 10000000;
SELECT salary FROM hr.emp WHERE dept = 'SALES';
DELETE from hr.emp where salary >1000000;
```

PAQUETE DBMS_FGA. PROCEDIMIENTO ADD_POLICY.

- Procedimiento ADD_POLICY.

  Permite crear una política de auditoría (hasta un máximo de 256 sobre una tabla o vista).

```
DBMS_FGA.ADD_POLICY(
        object_schema VARCHAR2,
        object_name VARCHAR2,
        policy_name VARCHAR2,
        audit_condition VARCHAR2,
        audit_column VARCHAR2,
        handler_schema VARCHAR2,
        handler_module VARCHAR2,
        enable BOOLEAN,
        statement_types VARCHAR2,
        audit_trail BINARY_INTEGER IN DEFAULT,
        audit_column_opts BINARY_INTEGER IN DEFAULT);
```

PAQUETE DBMS_FGA. PROCEDIMIENTO ADD_POLICY.

Parameter	Description	Default Value
object_schema	The schema of the object to be audited. (If NULL, then the current login user schema is assumed.)	NULL
object_name	The name of the object to be audited.	-
policy_name	The unique name of the policy.	-
audit_condition	A condition in a row that indicates a monitoring condition. NULL is allowed and acts as TRUE.	NULL
audit_column	The columns to be checked for access. These can include hidden columns. The default, NULL, causes audit if any column is accessed or affected.	NULL
handler_schema	The schema that contains the event handler. The default, NULL, causes the current schema to be used.	NULL
handler_module	The function name of the event handler includes the package name if necessary. This function is called only after the first row that matches the audit condition in the query is processed. If the procedure fails with an exception, then the user SQL statement will fail as well.	NULL
enable	Whether the policy is to be enabled: TRUE means enable it.	TRUE
statement_types	The SQL statement types to which this policy is applicable: INSERT, UPDATE, DELETE, or SELECT only.	SELECT
audit_trail	Both where to write the fine-grained audit trail and whether or not to populate LSQLTEXT and LSQLBIND.	DB+EXTENDED
audit_column_opts	Whether a statement is audited when the query references *any* column specified in the audit_column parameter or only when *all* such columns are referenced.	ANY_COLUMNS

PAQUETE DBMS_FGA. PROCEDIMIENTO ADD_POLICY.

- Ejemplo.

```
DBMS_FGA.ADD_POLICY(object_schema => 'scott', object_name=>'emp',
    policy_name => 'mypolicy1', audit_condition => 'sal < 100', audit_column
    =>'comm, credit_card, expirn_date', handler_schema => NULL,
    handler_module => NULL, enable => TRUE, statement_types=> 'INSERT,
    UPDATE', audit_trail => DBMS_FGA.DB+DBMS_FGA.EXTENDED,
    audit_column_opts => DBMS_FGA.ALL_COLUMNS);
```

PAQUETE DBMS_FGA. PROCEDIMIENTO DISABLE_POLICY.

- Procedimiento DISABLE_POLICY.

  Deshabilita una política de auditoría.

```
DBMS_FGA.DISABLE_POLICY(
        object_schema VARCHAR2,
        object_name VARCHAR2,
        policy_name VARCHAR2 );
```

Parameter	Description
object_schema	The schema of the object to be audited (If NULL, then the current login user schema is assumed.)
object_name	The name of the object to be audited
policy_name	The unique name of the policy

PAQUETE DBMS_FGA. PROCEDIMIENTO ENABLE_POLICY.

- Procedimiento ENABLE_POLICY.

  Habilita una política de auditoría.

```
DBMS_FGA.ENABLE_POLICY(
      object_schema VARCHAR2,
      object_name VARCHAR2,
      policy_name VARCHAR2
      enable BOOLEAN);
```

Parameter	Description
object_schema	The schema of the object to be audited. (If NULL, then the current log-on user schema is assumed.)
object_name	The name of the object to be audited.
policy_name	The unique name of the policy.
enable	Defaults to TRUE to enable the policy.

PAQUETE DBMS_FGA. PROCEDIMIENTO DROP_POLICY.

- Procedimiento DROP_POLICY.

  Borra una política de auditoría.

```
DBMS_FGA.DROP_POLICY(
      object_schema VARCHAR2,
      object_name VARCHAR2,
      policy_name VARCHAR2);
```

Parameter	Description
object_schema	The schema of the object to be audited. (If NULL, then the current log-on user schema is assumed.)
object_name	The name of the object to be audited.
policy_name	The unique name of the policy.

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 desactivan opciones de auditoría básica. Para deshabilitar la auditoría básica 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 (por ejemplo, si se está auditando CREATE SESSION no se podrán conectar usuarios a la bd).

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

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>;*

- Sólo el usuario *SYS*, un usuario al que se haya concedido el privilegio *DELETE* sobre *SYS.AUD$*, o un usuario con el privilegio *DELETE ANY TABLE* pueden efectuar el borrado.

- La información del "audit trail" puede archivarse copiándola a una tabla (*INSERT INTO <tabla> SELECT ... FROM SYS.AUD$*) o ser exportada.

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

  - Conectarse como usuario administrador y truncar *SYS.AUD$* (sentencia *TRUNCATE*).

  - 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 SELECT, 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.

AUDITORÍA. VISTAS.

- ***AUDIT_ACTIONS***. *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.*
- ***DBA_AUDIT_STATEMENT***. *Registros para las sentencias GRANT, REVOKE, AUDIT, NOAUDIT, y ALTER SYSTEM.*
- ***DBA_AUDIT_TRAIL***. *Registros de "audit trail".*

AUDITORÍA. VISTAS.

- ***DBA_OBJ_AUDIT_OPTS***. *Opciones de auditoría para todos los objetos.*
- ***DBA_COMMON_AUDIT_TRAIL***. *Incluye registros de auditoría básica y de grano fino, así como el contenido de V$XML_AUDIT_TRAIL*
- ***DBA_FGA_AUDIT_TRAIL***. *Registros de auditoría grano fino.*
- ***DBA_AUDIT_POLICIES***. *Políticas de auditoría de grano fino en el sistema.*
- ***DBA_PRIV_AUDIT_OPTS***. *Privilegios de sistema auditados.*
- ***DBA_STMT_AUDIT_OPTS***. *Opciones de auditoría por sentencia.*
- ***FLASBACK_TRANSACTION_QUERY.*** *Información sobre transacciones y operaciones realizadas.*

AUDITORÍA. VISTAS.

- ***STMT_AUDIT_OPTION_MAP****. Información códigos de auditoría.*

TEMA 10.

COPIAS DE SEGURIDAD.

TEMA 10.
COPIAS DE SEGURIDAD.

- MODOS DE OPERACION DE LA BD.
- COPIAS DE SEGURIDAD.
- COPIA FISICA.
- OPTIMAL FLEXIBLE ARCHITECTURE. O.F.A. Y UNIX.
- COPIA FISICA. SISTEMAS DE FICHEROS.
- COPIA FISICA. TAR (UNIX) Y RECOVERY MANAGER.
- RECUPERACION DE TABLAS. “FLASHBACK TABLE”.

TEMA 10.
COPIAS DE SEGURIDAD.

- RECUPERACION DE TABLAS. “RECYCLE BIN”.
- RECUPERACION DE TABLAS. “PURGE”. “FLASHBACK TO BEFORE DROP”.
- COPIA LOGICA. DATA PUMP EXPORT/IMPORT.
- ARQUITECTURA DATA PUMP.
- DATA PUMP. VENTAJAS. EJECUCION. INFORMACION. FICHEROS.
- DATA PUMP EXPORT. PARAMETROS.

TEMA 10.
COPIAS DE SEGURIDAD.

- DATA PUMP EXPORT. PARAMETROS. MODO INTERACTIVO.
- DATA PUMP IMPORT. PARAMETROS.
- DATA PUMP IMPORT. PARAMETROS. MODO INTERACTIVO.
- DATA PUMP Y PARAMETROS.
- COPIA LOGICA. UTILIDADES EXPORT/IMPORT.
- EXPORT/IMPORT vs. DATA PUMP
- UTILIDAD EXPORT. EXPORT MODO DIRECTO. CASOS PRACTICOS.

TEMA 10.
COPIAS DE SEGURIDAD.

- UTILIDAD IMPORT. CASOS PRACTICOS.
- RENOMBRAR ESP. DE ALMACENAMIENTO.
- SQL*LOADER. CARGA DE DATOS. FICHEROS DATOS Y CONTROL.
- SQL*LOADER. EJEMPLOS. EJECUCION.
- VISTAS.
- APENDICE A. RECURSOS ORACLE.

MODOS DE OPERACION DE LA BD.

- Existen dos modos de operación de la bd:
- **Modo NOARCHIVELOG.**
  - El archivado de los "redo log" está deshabilitado. Cuando un grupo de "redo" pasa a estar inactivo tras un "log switch", está disponible para ser nuevamente usado por el LGWR.
  - Este modo protege a la bd de fallos en la instancia pero no de fallos en los soportes ("media failure"). Sólo los cambios recientes en la bd, almacenados en el "redo" en línea pueden recuperarse; si ocurre un fallo en disco, la bd sólo puede recuperarse hasta el momento en que se realizó la copia más reciente.
  - Para la recuperación sólo pueden emplearse copias completas y coherentes realizadas con la bd cerrada consistentemente.

MODOS DE OPERACION DE LA BD.

- **Modo ARCHIVELOG.**
  - El archivado de los "redo log" está habilitado. Un grupo de "redo" no puede reutilizarse por LGWR hasta ser archivado tras el "log switch".
  - Se garantiza que todas las transacciones validadas pueden recuperarse en caso de fallo en el sistema o disco. Además pueden emplearse copias realizadas con la bd abierta y en uso normal.

Nota: durante este tema se asume que la base de datos está en modo noarchivelog.

COPIAS DE SEGURIDAD.

- Pueden distinguirsen dos tipos de copias:
  - Copias de seguridad **físicas**. Se realiza la copia de los ficheros que constituyen la base de datos. Se distinguen copias de seguridad en línea ("en caliente") y fuera de línea ("en frio").

    Se pueden hacer manualmente (comando "*tar*" de s.o.) o mediante la utilidad *RMAN* ("*recovery manager*") de Oracle.

  - Copias de seguridad **lógicas**. Implican la lectura de un conjunto de registros de base de datos y su escritura en un fichero especial.

    Se emplean utilidades propias de la base de datos: *export/import* y *data pump export/import*.

COPIAS DE SEGURIDAD.

- Las copias de seguridad lógicas se emplean 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!).
- Una buena estrategia de copias de seguridad incluirá ambos tipos de copias, físicas y lógicas, pues son complementarias.

COPIA FISICA

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 realiza de forma diaria.
  - La copia del "software" de base de datos se realiza con frecuencia semanal, dado su menor nivel de actualización.
- Dado que se asume que la base de datos está en modo noarchivelog, sólo es posible realizar copias "en frio" de la misma. Por tanto, 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 *NET. 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	Fich.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** son:

  - Sistemas de ficheros de administración de la bd, bajo *$ORACLE_BASE/admin.* Incluirá la copia del fichero de parámetros de inicialización "*init.ora*" y del fichero de parámetros "*spfile*" en su caso.

  - 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** son:

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

  - Todos los sistemas de ficheros que integran 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* de sistema operativo: 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.

COPIA FISICA. RECOVERY MANAGER.

- Es una utilidad que permite realizar copias y recuperaciones en la bd y automatiza la administración de las mismas. El entorno para RMAN incluye:
  - Bd a copiar.
  - Cliente RMAN. Permite ejecutar sentencias RMAN.
  - "Flash recovery area" (opcional). Localización en disco en que la bd puede almacenar y gestionar ficheros de copia y recuperación.
  - "Media management software" (opcional). SW necesario para que RMAN interactúe con dispositivos de copia como unidades de cinta y otros.
  - BD "recovery catalog" (opcional). Esquema de bd empleado para registrar la actividad de RMAN en una o más bbdd destino.

FLASHBACK

RECUPERACION DE TABLAS. "FLASHBACK TABLE".

- Es posible restaurar una tabla a un estado anterior a aquel donde tuvieron lugar cambios, usando el espacio de "undo", mediante la sentencia *FLASHBACK TABLE.* Se proporciona así una forma rápida de recuperar una tabla modificada o borrada, y se evita recurrir a métodos complejos.
- Características:
  - La operación se realiza en línea.
  - Se mantienen automáticamente todos los atributos de la tabla (índices, disparadores, "constraints") necesarios.
  - Se mantienen las restricciones de integridad.
  - No se pierden los datos originales, pudiendo volver al principio (no puede hacerse un "rollback" de la sentencia, pero puede realizarse una nueva recuperación a un momento justo anterior al actual -es conveniente anotar previamente el SCN actual-).

RECUPERACION DE TABLAS. "FLASHBACK TABLE".

- Durante la recuperación se adquiere un bloqueo DML que impide cualquier operación sobre la tabla. Esta se realiza en una transacción única, independientemente del número de tablas que comprenda la recuperación.

- Requisitos:
  - Debe poseerse el privilegio *FLASHBACK ANY TABLE* o tener el privilegio *FLASHBACK* sobre la tabla.
  - Deben poseerse los privilegios *SELECT, INSERT, DELETE,* y *ALTER* en la tabla.
  - La información de "undo" debe ser suficiente como para recuperar hasta el punto indicado en la operación.
  - Debe habilitarse en la tabla la característica "row movement" mediante la sentencia:

ALTER TABLE <nombre_tabla> ENABLE ROW MOVEMENT;

RECUPERACION DE TABLAS. "FLASHBACK TABLE".

- Restricciones:
  - Esta operación no puede realizarse en tablas que forman parte de un "cluster", vistas materializadas, tablas Advanced Queuing (AQ), tablas estáticas de diccionario, tablas de sistema, tablas remotas, "object" tablas, tablas anidadas o particiones individuales de tablas.
  - Las operaciones DDL que cambian la estructura de una tabla no permiten usar esta utilidad (actualización, "moving" o truncado de la tablas, adición de "constraint", adición de una tabla a un "cluster", modificación o borrado de una columna, adición, borrado, coalescencia, división, modificación o truncado de una partición -salvo la adición de una partición "range"-).

RECUPERACION DE TABLAS. "FLASHBACK DROP".

- Al borrar una tabla, la bd no borra inmediatamente el espacio ocupado por la misma. La bd renombra la tabla y la coloca junto con sus objetos asociados ("constraints", tablas anidadas y semejantes) en lo que denomina "*recycle bin*", de forma que si fue borrada por error es posible recuperarla usando la sentencia *FLASHBACK TABLE* ("*flashback drop*").

- El "*recycle bin*" es una tabla de diccionario con información sobre los objetos borrados y sus asociados, los cuales no se eliminan hasta indicarlo explicitamente o hasta haber necesidades de espacio (se eliminan en orden "first-in first-out" -FIFO-). Cada usuario, a menos que tenga privilegio *SYSDBA*, sólo puede acceder a sus propios objetos en esta tabla:

  *SELECT * FROM RECYCLEBIN;*

- El espacio libre en un espacio de almacenamiento puede consultarse en la vista *DBA_FREE_SPACE.*

RECUPERACION DE TABLAS. "RECYCLE BIN".

- Al borrar un espacio de almacenamiento incluyendo su contenido, los objetos en el mismo no se almacenan en el "recycle bin" y la bd elimina cualquier referencia en el mismo. Igualmente sucede cuando se borra un usuario, "cluster" o "type".

- Este modo de funcionamiento, habilitado por defecto, puede habilitarse o deshabilitarse mediante el parámetro de inicialización *RECYCLEBIN*, o, dinámicamente mediante las sentencias:

  ALTER SESSION/SYSTEM SET recyclebin = OFF;

  ALTER SESSION/SYSTEM SET recyclebin = ON;

  Deshabilitar "recycle bin" no afecta a los objetos que ya se hallan en el mismo.

RECUPERACION DE TABLAS. "RECYCLE BIN".

- También se obvia en ciertas ocasiones el "recycle bin":
  - En la sentencia *DROP TABLE ... PURGE* (ej. *SQL> drop table emp purge;*).
  - En la sentencia *DROP TABLESPACE ... INCLUDING CONTENTS*. Cada espacio de almacenamiento tiene su propio "recycle bin" por lo que borrar el espacio implica eliminar este y cualquier objeto en el mismo.
  - AL emplear *DROP USER...CASCADE*. Se elimina el usuario y todos sus objetos (incluidos aquellos en el "recycle bin".

RECUPERACION DE TABLAS. "RECYCLE BIN".

- A los objetos almacenados en el "recycle bin" se les asigna un nombre generado por el sistema de la forma *BIN$unique_id$version* , donde :
  - *unique_id* es un identificador único para el objeto compuesto por 26 caracteres, y
  - *version* es un número de version asignado a la bd.

- Es posible obtener información mediante las vistas:
  - *USER_RECYCLEBIN*. Permite a un usuario ver los objetos de su propiedad que han sido borrados, tiene un *sinónimo: RECYCLEBIN*.
  - *DBA_RECYCLEBIN*. Da información a los administradores sobre todos los objetos borrados.

  O mediante la sentencia SQL*Plus *SHOW RECYCLEBIN*.

RECUPERACION DE TABLAS. "PURGE". "FLASHBACK TO BEFORE DROP".

- Mediante la sentencia *PURGE* puede borrarse explicitamente el contenido del "recycle bin" y liberar el espacio asociado (son necesarios los mismos privilegios requeridos para borrar el elemento):

  PURGE TABLE/INDEX <nombre>;

  PURGE TABLESPACE <nombre_tbsp> USER <nombre_usuario>;

  Nota: Se eliminan todos los objetos de "nombre_usuario" para el esp.almacenamiento "nombre_tbsp".

  PURGE RECYCLEBIN/DBA_RECYCLEBIN;

- La sentencia *FLASHBACK TABLE ... TO BEFORE DROP* permite recuperar objetos del "recycle bin":

  FLASHBACK TABLE <nombre_tabla> TO BEFORE DROP
  RENAME TO <nombre_tabla>;

COPIA LOGICA: DATAPUMP

COPIA LÓGICA.
DATA PUMP EXPORT/IMPORT.

- Son utilidades integradas en la bd que permiten cargas y descargas de información a gran velocidad. Todo el proceso de exportación/importación se lleva a cabo en el servidor.

- No son compatibles con el export/import originales.

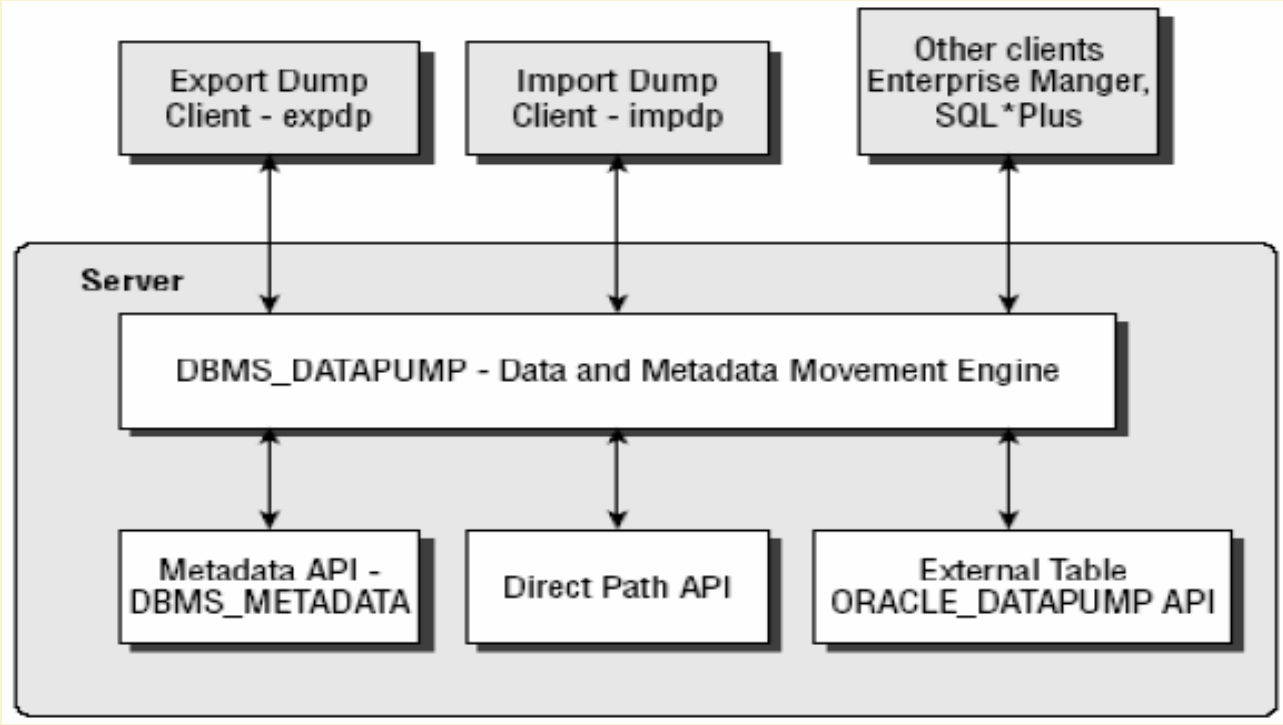
- Consta de los siguientes componentes:

  - Clientes *expdp* e *impdp*. Usan los procedimientos proporcionados por el paquete DBMS_DATAPUMP.

  - Paquete *DBMS_DATAPUMP*. Conocido como Data Pump API, permite crear y monitorizar los trabajos implicados.

  - Paquete *DBMS_METADATA*. Conocido como Metadata API, proporciona la definición de los objetos de la bd a Data Pump.

ARQUITECTURA DATA PUMP.



DATA PUMP. VENTAJAS.

- Algunas de las ventajas de Data Pump son:
  - La forma de acceso a los datos se decide automáticamente (habitualmente "direct path", "external" cuando no es posible).
  - La exportación puede realizarse en paralelo y puede escribirse a múltiples ficheros en diferentes discos (opciones *PARALLEL* y *DUMPFILE*).
  - Puede reiniciarse un trabajo fallido donde se interrumpió.
  - Permite filtrar metadatos con múltiples combinaciones (opciones *INCLUDE* y *EXCLUDE*).
  - Pueden filtrarse filas de datos durante la importación.
  - Permite estimar el espacio en disco necesario antes de realizar el trabajo (opción *ESTIMATE_ONLY*).
  - Pueden hacerse exportaciones e importaciones remotas usando "database link".

DATA PUMP. VENTAJAS.

- Los trabajos pueden asignar recursos dinámicamente según la carga de trabajo.
- Puede indicarse explícitamente la versión de bd, de forma que sólo los objetos soportados se exportan.
- Durante la importación, puede cambiarse los nombres de fichero, esquemas y espacios de almacenamiento destino.

DATA PUMP. EJECUCIÓN.

- En cada exportación o importación se crea un proceso maestro que controla la misma, incluyendo la comunicación con los clientes, la creación y control del conjunto de procesos de trabajo necesarios y la realización de las operaciones de conexión.

- En la transferencia de datos y metadatos, se emplea un tabla maestro, que se crea en el esquema del usuario que realiza la exportación/importación y tiene el nombre del trabajo -job- que realiza la operación, para realizar el seguimiento del proceso:

  - Durante la exportación la tabla registra la localización de los objetos en el conjunto de ficheros de descarga ("*dump file set*"). Al finalizar, el contenido de la tabla se escribe a fichero.

  - Durante la importación la tabla maestro se carga desde el conjunto de ficheros de descarga y se emplea para controlar la secuencia de localización de los ficheros a importar.

DATA PUMP. EJECUCIÓN.

- Esta tabla maestro también se emplea para reiniciar un trabajo.
- El destino final de la tabla maestro es diverso:
  - Si el trabajo finaliza correctamente, la tabla se borra.
  - Si el trabajo se para, usando la sentencia *STOP_JOB*, se retiene para poder reiniciarlo.
  - Si el trabajo se elimina, usando la sentencia *KILL_JOB*, la tabla se elimina y el trabajo no puede reiniciar.
  - Si un trabajo termina inesperadamente, se mantiene la tabla. Puede borrarse si no hay intención de reiniciar el trabajo.
- El progreso de un trabajo y los errores producidos pueden registrarse en un fichero de "log". El estado del mismo, en tiempo real, puede obtenerse usando la sentencia *STATUS* en modo interactivo de Data Pump.

DATA PUMP. INFORMACIÓN.

- Puede obtenerse información en las siguientes vistas:
  - *DBA_DATAPUMP_JOBS* y *USER_DATAPUMP_JOBS*. Identifican trabajos Data Pump activos.
  - *DBA_DATAPUMP_SESSIONS*. Identifica sesiones de usuario asociadas a un trabajo.
  - *V$SESSION_LONGOPS*. Información sobre el progreso del trabajo.

DATA PUMP. FICHEROS.

- Existen tres tipos de ficheros gestionados por Data Pump:
  - Ficheros de volcado ("dump"). Contienen datos y metadatos.

    Durante una exportación pueden indicarse al definir el trabajo así como posteriormente (por ejemplo, pueden añadirse ficheros de volcado con *ADD_FILE*).

    Durante un importación, todos los ficheros de volcado deben especificarse al definir el trabajo.
  - Ficheros "log". Contienen mensajes sobre la operación realizada.
  - Ficheros SQL. Registran la salida de una operación *SQLFILE* (parámetro *SQLFILE*) consistente en el volcado de todas las sentencias DDL que se hubieran ejecutado durante una importación a un fichero.

    Nota: Los ficheros de "log" y SQL pueden sobreescribirse si ya existen. Nunca se sobreescriben ficheros de volcado si ya existen, se genera un error.

DATA PUMP.
LOCALIZACION DE FICHEROS.

- Es necesario crear objetos tipo directorio (asocia un nombre a un directorio o sistema de ficheros) por un usuario administrador o con el privilegio *CREATE ANY DIRECTORY*. Al exportar/importar se indica mediante el parámetro *DIRECTORY*.

  SQL> CREATE DIRECTORY DPUMP_VOLCADO AS '/datapump/ficheros';

  Por defecto se asigna el valor para *DATA_PUMP_DIR:*

  SQL> CREATE DIRECTORY DATA_PUMP_DIR AS '/datapump/ficheros';

- Tras la creación de un directorio, su creador debe conceder el permiso de lectura (READ) o escritura (WRITE) a otros usuarios.

  SQL> GRANT READ, WRITE ON DIRECTORY <directorio> TO <usuario>;

  Nota: Implica tener acceso a través de bd pero no el poder acceder a través de s.o.

DATA PUMP.
LOCALIZACION DE FICHEROS.

- El orden de precedencia usado para determinar la localización es:

  1.- Objeto directorio formando parte de la especificación de un nombre de fichero (*DUMPFILE=<localización>:<fichero>.dmp*).

  2.- Objeto directorio indicado en el parámetro *DIRECTORY*.

  3.- Valor de la variable de entorno *DATA_PUMP_DIR* (definida en el entorno del cliente):

  ```
  SQL> CREATE DIRECTORY DUMP_FICHERO AS
    '/DATAPUMP/FICHEROS';

  # EXPORT DATA_PUMP_DIR=<VARIABLE_FICHERO_VOLCADO>
  ```

  4.- Si es un usuario privilegiado se emplea el valor por defecto en bd del directorio *DATA_PUMP_DIR* (que debe estar previamente creado). No confundir con la variable de entorno.

DATA PUMP EXPORT.

- Se emplea la utilidad *expdp*, indicando las características de la exportación en la línea de comandos o mediante un fichero de parámetros. Los ficheros resultantes de una exportación sólo pueden importarse con D.P. Import.

  Cualquier usuario puede exportar objetos de su propiedad o su esquema al completo.

  Los usuarios no privilegiados deben tener permiso *WRITE* en el objeto directorio y deben indicar éste en el parámetro DIRECTORY o junto al nombre de fichero de volcado.

  Modo ayuda en línea: *expdp HELP=y*

  Modo interactivo: *expdp directory=<localización>*

  Modo fichero de parámetros: *expdp PARFILE=<fichero_parametros>*

Nota: Si no se indica ningún otro parámetro se usan los valores por defecto en la exportación: *expdat.dmp* y *export.log* -ficheros-, exportación a nivel de esquema, *SYS_EXPORT_SCHEMA_01* -trabajo- y se exportan datos y metadatos.

DATA PUMP EXPORT.

- Existen cinco modos de trabajar con "datapump export":

  - "Full export" (parámetro *FULL*). Se exporta toda la bd. Es necesario el rol *EXP_FULL_DATABASE.*

  - "Esquema de usuario" (parámetro *SCHEMAS*). Si no se posee rol *EXP_FULL_DATABASE* sólo es posible exportar el esquema propio.

  - "Tabla" (parámetro *TABLES)*. Para exportar tablas no pertenecientes al propio esquema es necesario el rol *EXP_FULL_DATABASE.*

  - "Tablespace" (parámetro *TABLESPACES*). Debe tenerse el rol *EXP_FULL_DATABASE.*

  - "Transport Tablespace" (parámetro *TRANSPORT_TABLESPACES*). Permite exportar metadatos para tablas y objetos dependientes pertenecientes a un conjunto de espacios de almacenamiento. Es necesario el rol *EXP_FULL_DATABASE*.

DATA PUMP EXPORT. PARÁMETROS.

- *CONTENT =<**ALL** | DATA_ONLY | METADATA_ONLY>*

  Indica contenido a exportar: datos, metadatos o ambos.

- *DIRECTORY =<**DATA_PUMP_DIR**>*

  Localización para los ficheros de volcado y registro.

- *DUMPFILE = <directorio_objeto:>nombre_fichero*

  Fichero de volcado. Por defecto *expdat.dmp*

- *ESTIMATE = <**BLOCKS** | STATISTICS>*

  Método de estimación de espacio en disco necesario para cada tabla.

- *ESTIMATE_ONLY =<**N** | Y>*

  Permite estimar el espacio consumido en una exportación sin realizarla.

DATA PUMP EXPORT. PARÁMETROS.

- *EXCLUDE = tipo_objeto <:cláusula>*

  Filtrar metadatos exportados indicando objetos y tipos excluidos. Por defecto ninguno. Es excluyente con el parámetro INCLUDE.

- *FILESIZE*

  Tamaño máximo para ficheros de exportación (por defecto ilimitado).

- *FULL = <**N** | Y>*

  Realizar una exportación de la bd al completo. En una exportacion completa no se incluyen los esquemas SYS, ORDSYS, EXFSYS, MDSYS, DMSYS, CTXSYS, ORDPLUGINS, LBACSYS, XDB, SI_INFORMTN_SCHEMA, DIP, DBSNMP, y WMSYS. Nunca se exportan tampoco los permisos sobre objetos de SYS.

DATA PUMP EXPORT. PARÁMETROS.

- *INCLUDE = tipo_objeto <:cláusula>*

  Filtrar metadatos exportados indicando objetos y tipos incluidos (consultar las vistas DATABASE_EXPORT_OBJECTS, SCHEMA_EXPORT_OBJECTS y TABLE_EXPORT_OBJECTS).

- *JOB_NAME*

  Especificar un nombre para el trabajo, "job", de exportación; coincide tambien con la tabla maestro usada para controlar la exportación. Por defecto de las forma *sys_export_<modo>_xx*

- *LOGFILE = <directorio_objeto:>nombre_fichero*

  Fichero de registro, por defecto *export.log*

- *NOLOGFILE = <Y | **N**>*

  No crear fichero de registro.

DATA PUMP EXPORT. PARÁMETROS.

- *PARALELL = n*

  Grado de paralelismo en la ejecución (por defecto 1). Su número debe ser igual al número de ficheros de volcado indicados.

- *QUERY =<nombre_esquema.><nombre_tabla> | <cláusula>*

  Permite seleccionar filas a exportar según condiciones a cumplir. No compatible con *CONTENT=METADATA_ONLY, ESTIMATE_ONLY* o *TRANSPORT_TABLESPACES*.

- *SCHEMAS =<lista de esquemas>*

  Permite indicar que se realizará una exportación de esquemas, es el modo por defecto. Por defecto referencia el esquema de usuario.

- *STATUS = nn*

  Frecuencia con que muestra el estado del trabajo junto a una descripción de la operación actual (por defecto es cero y no muestra información hasta acabar la exportación de cada objeto).

DATA PUMP EXPORT. PARÁMETROS.

- *TABLES = <nombre_esquema.><nombre_tabla>, ...*

  Indica exportación en modo tabla (del propio esquema, por defecto).

- *TABLESPACES = <nombre_espacioalmac>, ...*

  Indica tbsp. a exportar en modo "tablespace".

- *TRANSPORT_FULL_CHECK =<**N** | Y>*

  Chequear o no las dependencias entre los objetos residentes los espacios de almacenamiento a "transportar" y aquellos situados fuera del conjunto (sólo en modo "transport tablespace").

- *TRANSPORT_TABLESPACES = <nombre_espacioalmac>, ...*

  Realizar una exportación en modo "transport tablespace" y espacios de almacenamiento a los que se aplica. Este modo tiene un grado de paralelismo igual a 1 y requiere disponer del rol *EXP_FULL_DATABASE*, sus trabajos no se pueden reiniciar.

DATA PUMP EXPORT. MODO INTERACTIVO.

- Tras iniciar un trabajo, puesto que se ejecuta en el servidor, es posible interrumpir la sesión cliente y asociarse al trabajo posteriormente. También es posible modificar ciertos parámetros de forma interactiva.

  Para pasar a modo interactivo puede interrumpirse la conexión actual con *Ctrl+C* y asociar posteriormente una sesión cliente (*ATTACH*):

  \/home/...> expdp attach=<nombre_trabajo>

  Los parámetros que pueden usarse en modo interactivo son:

  – *ADD_FILE = <directorio:><nombre_fichero>, ...*

    Permite añadir ficheros de volcado.

DATA PUMP EXPORT. MODO INTERACTIVO.

- *CONTINUE_CLIENT.* Salir de modo interactivo y entrar en modo registro.
- *EXIT_CLIENT.* Salir de sesión cliente. El trabajo continúa ejecutándose en el servidor.
- *HELP.* Ayuda en línea.
- *KILL_JOB.* Cancelar trabajo actual y desasociar las sesiones cliente relacionadas.
- *PARALLEL.* Indicar grado paralelismo o número de trabajos.
- *START_JOB.* Reiniciar el trabajo asociado.
- *STATUS.* Mostrar informe del estado del trabajo asociado.
- *STOP_JOB [INMEDIATE].* Detener el trabajo asociado, puede reiniciarse más tarde. Con "inmediate" los procesos acaban al instante, las tareas incompletas se volverán a realizar al reiniciarlo.

DATA PUMP IMPORT.

- Se emplea la utilidad *impdp*, indicando las características de la importación en la línea de comandos o mediante un fichero de parámetros. Sólo pueden usarse en la importación ficheros creados con D.P. Export.

  Los usuarios deben tener permiso *READ* en el objeto directorio dond resida el fichero de volcado y de escritura donde se cree el fichero de registro (log) y los SQL.

  Modo ayuda en línea: *impdp HELP=y*

  Modo interactivo: *impdp directory=<localización>*

  Modo fichero de parámetros: *impdp PARFILE=<fichero_parametros>*

Nota: Si no se indica ningún otro parámetro se usan los valores por defecto en la exportación: *expdat.dmp* e *import.log* -ficheros-, importación total del fichero y trabajo *SYS_IMPORT_xx_01*.

DATA PUMP IMPORT.

- Existen cinco modos de trabajar con "datapump import":
  - "Full import" (parámetro *FULL*). Se importa el fichero de volcado al completo (modo por defecto). Necesario el rol *IMP_FULL_DATABASE* si se hizo la exportación con *EXP_FULL_DATABASE.*
  - "Esquema de usuario" (parámetro *SCHEMAS*). Sólo se cargan objetos propiedad del usuario actual. Si se posee rol *IMP_FULL_DATABASE* pueden cargarse esquemas diferentes al propio.
  - "Tabla" (parámetro *TABLES)*. Para importar tablas no pertenecientes al propio esquema es necesario el rol *IMP_FULL_DATABASE.*
  - "Tablespace" (parámetro *TABLESPACES*).
  - "Transport Tablespace" (parámetro *TRANSPORT_TABLESPACES*). Permite importar metadatos para un conjunto de espacios de almacenamiento. Es necesario el rol *IMP_FULL_DATABASE*.

DATA PUMP IMPORT. PARÁMETROS.

- *CONTENT =<**ALL** | DATA_ONLY | METADATA_ONLY>*

  Indica contenido a importar: datos -filas, no se crean objetos-, metadatos -definiciones- o ambos.

- *DIRECTORY =<**DATA_PUMP_DIR**>*

  Por defecto *DATA_PUMP_DIR* para usuarios privilegiados, ninguno para el resto. Localización para los ficheros de volcado y registro.

- *DUMPFILE = <directorio_objeto:>nombre_fichero*

  Fichero de volcado. Por defecto *expdat.dmp*

- *ESTIMATE = <**BLOCKS** | STATISTICS>*

  Método de estimación de espacio en disco necesario para cada tabla (sólo válido si se indica también NETWORK_LINK).

DATA PUMP IMPORT. PARÁMETROS.

- *EXCLUDE = tipo_objeto <:cláusula>*

  Filtrar metadatos importados indicando objetos y tipos excluidos. Por defecto ninguno. Es excluyente con el parámetro INCLUDE. No puede usarse junto con *CONTENT=DATA_ONLY.*

- *FULL = <N | **Y**>*

  Realizar una exportación de la bd al completo, todo el fichero fuente se importa. Si la operación de exportación que dio lugar al fichero se hizo usando el rol *EXP_FULL_DATABASE*, es necesario el rol *IMP_FULL_DATABASE*.

- *INCLUDE = tipo_objeto <:cláusula>*

  Filtrar metadatos importados indicando objetos y tipos incluidos (consultar las vistas *DATABASE_EXPORT_OBJECTS, SCHEMA_EXPORT_OBJECTS y TABLE_EXPORT_OBJECTS*). No puede usarse junto con *CONTENT=DATA_ONLY.*

DATA PUMP IMPORT. PARÁMETROS.

- *JOB_NAME*

  Especificar un nombre para el trabajo, "job", de importación; coincide tambien con la tabla maestro usada para controlar la importación. Por defecto de las forma *sys_import_<modo>_xx*

- *LOGFILE = <directorio_objeto:>nombre_fichero*

  Fichero de registro, por defecto *import.log.*

- *NOLOGFILE = <Y | **N**>*

  No crear fichero de registro.

- *PARALELL = n*

  Grado de paralelismo en la ejecución (por defecto 1).

DATA PUMP IMPORT. PARÁMETROS.

- *QUERY =<nombre_esquema.><nombre_tabla:> <cláusula>*

  Permite seleccionar filas a importar según condiciones a cumplir. No compatible con *CONTENT=METADATA_ONLY, SQLFILE* o *TRANSPORT_TABLESPACES*.

- *REMAP_DATAFILE=fichero_fuente : fichero_destino*

  Cambia el nombre de fichero_fuente por el de fichero_destino en las sentencias SQL donde se referencie la fuente (*CREATE TABLESPACE, CREATE LIBRARY* y *CREATE DIRECTORY*). Debe tenerse el rol *IMP_FULL_DATABASE*.

- *REMAP_SCHEMA=esquema_fuente : esquema_destino*

  Carga objetos del esquema fuente al esquema destino. Debe tenerse el rol *IMP_FULL_DATABASE*. Si el esquema destino no existe, se crea si el fichero de volcado contiene la información necesaria (*CREATE USER*) y se importa con suficientes privilegios (en caso contrario debe crearse previamente).

DATA PUMP IMPORT. PARÁMETROS.

- *REMAP_TABLESPACE=tbsp_fuente : tbsp_destino*

  Carga todos los objetos, incluso el usuario, del esquema fuente en el esquema destino.

- *REUSE_DATAFILES=<Y | **N**>*

  Indica si la operación de importación debe reutilizar los ficheros de datos existentes o no en la creación de espacios de almacenamiento.

- *SKIP_UNUSABLE_INDEXES=<Y | **N**>*

  Permite indicar que no se carguen tablas que tengan índices en estado no disponible.

- *SQLFILE=<directorio_objeto:>nombre_fichero*

  Define un fichero en el que descargar todas las sentencias SQL tipo DDL que se hubieran ejecutado al importar (no son ejecutadas). Si se generan contraseñas en el fichero, aparecen comentadas.

DATA PUMP IMPORT. PARÁMETROS.

- *SCHEMAS =<lista de esquemas>*

  Permite indicar que se realizará una importación de esquemas. Por defecto referencia el esquema de usuario, para otros es necesario el rol *IMP_FULL_DATABASE*.

- *STATUS = nn*

  Frecuencia con que muestra el estado del trabajo junto a una descripción de la operación actual y el porcentaje que falta para acabar el trabajo (por defecto es cero).

- *TABLE_EXISTS_ACTION = <**SKIP** | APPEND |TRUNCATE |REPLACE>*

  Indica qué hacer si la tabla a importar ya existe. Si se indica *CONTENT=DATA_ONLY* la opción por defecto es *APPEND*):
  - *SKIP.* Deja la tabla tal como está e importa siguiente objeto.
  - *APPEND*. Carga filas, el contenido anterior no se modifica.
  - *TRUNCATE*. Borra las filas existentes antes de cargar.
  - *REPLACE*. Borra la tabla y la carga al completo.

DATA PUMP IMPORT. PARÁMETROS.

- *TABLES = <nombre_esquema.><nombre_tabla><:partición>, ...*

  Indica exportación en modo tabla (por defecto del propio esquema; para tablas de otro esquema debe tenerse el rol *IMP_FULL_DATABASE*).

- *TABLESPACES = <nombre_espacioalmac>, ...*

  Indica tbsp. a exportar en modo "tablespace".

- *TRANSFORM=<SEGMENT_ATTRIBUTES|STORAGE>:<N|**Y**><:TABLE|:INDEX>*

  Permite modificar parámetros en la sentencia de creación DDL a importar:

  - *SEGMENT_ATTRIBUTES*. Atributos físicos, "storage", "tablespaces", "logging".
  - *STORAGE.* Cláusula "storage".

  Por defecto se aplica a todos los tipos de objetos.

DATA PUMP IMPORT. PARÁMETROS.

- *TRANSPORT_DATAFILES*

  Conjunto de ficheros a importar en el destino en modo "transport tablespace".

- *TRANSPORT_TABLESPACES = <nombre_espacioalmac>, ...*

  Realizar una importación en modo "transport tablespace" y espacios de almacenamiento a los que se aplica.

DATA PUMP IMPORT. MODO INTERACTIVO.

- Al igual que con "data pump export", tras iniciar un trabajo, ya que es ejecutado en el servidor, es posible interrumpir la sesión cliente y asociarse al trabajo posteriormente. También es posible modificar ciertos parámetros de forma interactiva.

  Para pasar a modo interactivo puede interrumpirse la conexión actual con *Ctrl+C* y asociar posteriormente una sesión cliente (*ATTACH*):

  \/home/...> impdp attach=<nombre_trabajo>

  Los parámetros que pueden usarse en modo interactivo son:

  - *CONTINUE_CLIENT.* Salir de modo interactivo y entrar en modo registro.
  - *EXIT_CLIENT.* Salir de sesión cliente. El trabajo continúa ejecutándose en el servidor.

DATA PUMP IMPORT. MODO INTERACTIVO.

- *HELP.* Ayuda en línea.
- *KILL_JOB.* Cancelar trabajo actual y desasociar las sesiones cliente relacionadas.
- *PARALLEL.* Indicar grado paralelismo o número de trabajos.
- *START_JOB.* Reiniciar el trabajo asociado.
- *STATUS.* Mostrar informe del estado del trabajo asociado.
- *STOP_JOB [INMEDIATE].* Detener el trabajo asociado, puede reiniciarse más tarde. Con "inmediate" los procesos acaban al instante, las tareas incompletas se volverán a realizar al reiniciarlo.

DATA PUMP Y PARAMETROS.

- Ciertos parámetros de inicialización pueden afectar al rendimiento de "data pump". Puede probarse a usar las siguientes asignaciones:
  - *DISK_ASYNCH_IO = TRUE*
  - *DB_BLOCK_CHECKING = FALSE*
  - *DB_BLOCK_CHECKSUM = FALSE*
- Además los parámetros siguientes deben tener valores lo suficientemente altos como para permitir el máximo grado de paralelismo:
  - *PROCESSES*
  - *SESSIONS*
  - *PARALLEL_MAX_SERVERS*
  -

COPIA LOGICA: EXPORT/IMPORT

COPIA LÓGICA. UTILIDADES EXPORT/IMPORT.

- Generalmente se recomienda emplear Data Pump export/import y no usar las utilidades originales export/import; sin embargo estas son necesarias en los siguientes casos:
  - Al importar ficheros creados con la utilidad "export". Por ejemplo, aquellos procedentes de bases de datos con versiones anteriores a la 10g.
  - Si se desea exportar ficheros que serán importados con la utilidad "import". Por ejempo, si se van a transferir datos a bases con una versión anterior a la 10g.

EXPORT/IMPORT vs. DATA PUMP

- Data Pump trabaja con un grupo de ficheros de volcado ("dump file set"), no con uno único.
- D.P. basa su funcionamiento en el servidor no en el cliente.
- D.P. emplea ejecución paralela.
- D.P. almacena los metadatos como documentos XML no como sentencias DDL.
- D.P. se ajusta automáticamente (no son necesarios parámetros empleados en export/import originales como *BUFFER* y *RECORDLENGTH*).
- Usando D.P. no es posible realizar "commit" intermedio durante una importación (sí con import original -parámetro *COMMIT*-).
- Con D.P. no es posible comprimir extensiones al recrear tablas (sí con import original -parámetro *COMPRESS-)*.
- Al importar con D.P. en una tabla que ya existe, si una fila viola una restricción se interrumpe la operación y no se carga ningún dato. Con import original se registra el error y prosigue la carga.

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 rol *EXP_FULL_DATABASE.*

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

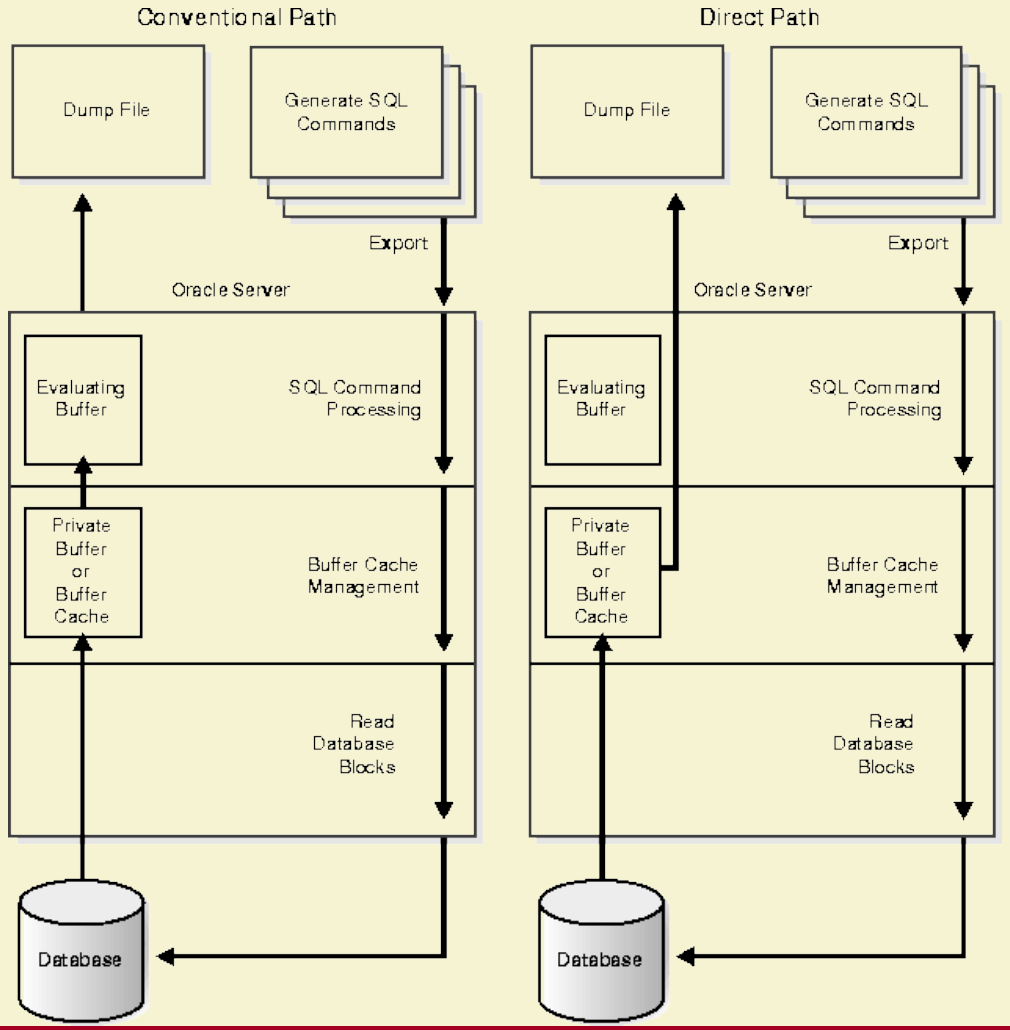
EXPORT. MODO DIRECTO.

- 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.

RENOMBRAR ESP. DE ALMACENAMIENTO.

- Util en casos donde se emplea la utilidad para “transportar” esp. de almacenamiento (por ejemplo, recuperación avanzada TSPITR).

ALTER TABLESPACE user RENAME to u1;

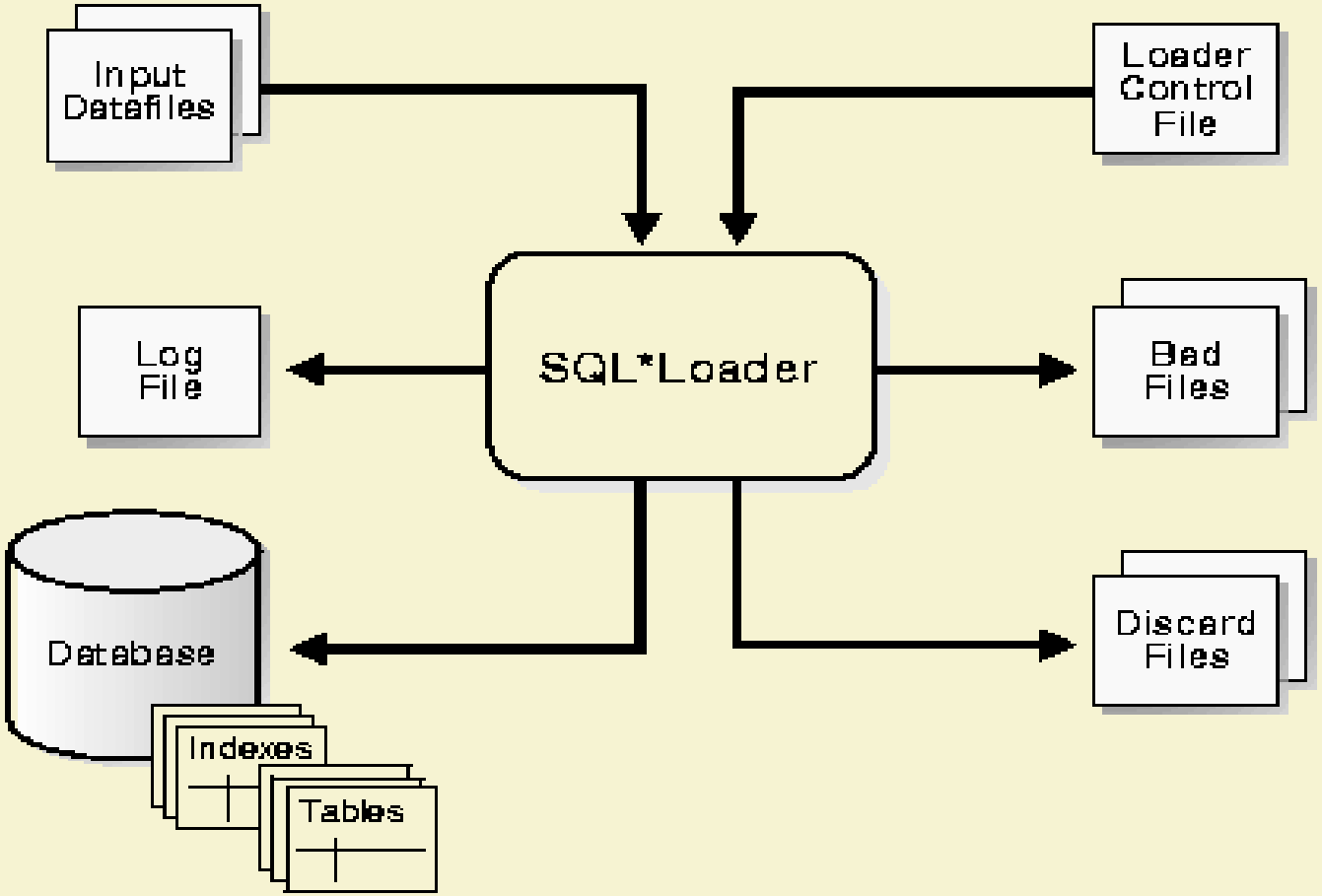
  - No puede renombrarse ni SYSTEM ni SYSAUX
  - Los esp. de almacenamiento y todos los ficheros de datos asociados deben estar en línea.
  - Pueden rebnombrarse también esp. alm. en sólo lectura (READ ONLY).

SQL*LOADER

SQL*LOADER. CARGA DE DATOS.

- 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>

VISTAS.

- ***DBA_DATAPUMP_JOBS***. Trabajos Data Pump activos.
- ***DBA_DATAPUMP_SESSIONS***. Sesiones de usuario asociadas a un trabajo.
- ***DBA_DIRECTORIES***. Directorios definidos en bd.
- ***V$SESSION_LONGOPS***. Información sobre desarrollo de trabajos.

APENDICE.
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.