irenchuchu
www.wuolah.com/student/irenchuchu



# Resumen-Tema-4.pdf

*Apuntes Examen Final (Parcial 2)*

**3° Administración de Bases de Datos**

**Grado en Ingeniería Informática**

**Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación**
**Universidad de Granada**

# TEMA 4: Seguridad y fiabilidad de los datos.

El SGBD almacena información sensible, se debe garantizar que cada usuario llegue sólo a la información que necesita, y que si el sistema sufre algún fallo se pueda recuperar de éste.

## Seguridad de los datos

- Autorización de usuarios.
    - Característica del conocimiento: contraseñas.
    - También se autorizan mediante huellas dactilares, retina, etc.
- Gestión y privilegios de roles
    - De sistema.
    - De objetos.
- Privilegios en el catálogo (metainformación).

A cualquier S. I. se le pide:

- Que garantice la exclusividad en el acceso a la información a quién tiene permisos.
- Que garantice que se puede recuperar cuando ocurra un fallo.

**Seguridad:**

- Se puede garantizar a distintos niveles:
    - Nivel físico: acceso a ubicación.
    - Nivel humano: confianza en usuarios con permisos.
    - Nivel del SO: si permite conexión remota y proporciona acceso local desde conexión remota.
    - A nivel del SBD: a nivel de información (quién accede a qué) y de operación (qué puede hacer).

- Con información sensible es mejor cifrar: el cifrado afecta a la velocidad de respuesta del sistema, que aumenta. Requiere sistemas más potentes, tanto en almacenamiento como en procesamiento.
- En la actualidad hay dispositivos cifrados, es decir, el propio disco duro es el que cifra la información cuando la almacena.

**Aspectos de seguridad:**

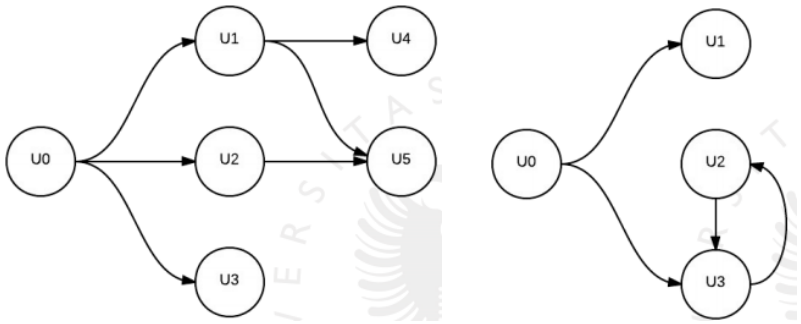
- Autorización de usuarios.
- Gestión de privilegios y roles:
    - De sistema.
    - De objeto
    - ¿dónde encontrarlos en el catálogo?

**Autorización de usuarios:**

- Roles de usuario: te permiten asignar conjuntos de privilegios a los usuarios.
- Dos niveles de autorización:
    - Par usuario/contraseña: La autorización de usuarios suele hacerse usando el par usuario-contraseña, ya que es la solución más barata. No obstante, esta solución es la más débil.
    - Según el uso de los datos, hay posibilidades de cerrar el acceso de los usuarios a los datos:
        - En sistemas centralizados, el administrador lo crea todo y concede permisos.
        - En sistemas descentralizados, hay una jerarquía de usuarios con permisos que pueden concederse o cederse, según determine el propietario.

- Algunos permisos posibles:
    - Consultar una tabla: necesario para leer o crear una vista sobre la tabla.
    - Insertar tuplas: se puede hacer insert pero no select, update ni delete.
    - Borrar tuplas: se puede hacer delete pero no select, update ni insert.
    - Modificar tuplas en algunos o todos los atributos: el permiso update puede estar solo en algunos atributos de la relación.
    - Modificar tabla.
    - Borrar tabla.
    - Crear índices sobre la tabla.
    - Concesión y cesión de permisos a otros usuarios.

- Grafo de autorización:




**Gestión de privilegios y roles:**

- Privilegio: derecho a ejecutar una determinada sentencia o acceder a un determinado objeto.
- Es responsabilidad del administrador el concederlo
- Pueden concederse:
    - Directamente.
    - A través de un rol.

- Rol: conjunto de privilegios con un nombre, que pueden cederse o concederse en grupo.

- Existen algunos roles predefinidos:
    - CONNECT: Permite conectar a la BD, consultar tablas públicas, crear vistas y exportar tablas. El usuario PUBLIC representa a todos los usuarios con este permiso.
    - RESOURCE: Permite crear tablas, índices y objetos dentro de la base de datos (hay que limitar al usuario el espacio que pueda consumir en el *tablespace*).
    - DBA: Es el rol del administrador de Oracle, y puede realizar cualquier acción ya que es el *sysdba*. Incluye a los anteriores y permite acceder a datos de todos los usuarios, conceder y revocar privilegios, realizar mantenimiento de sistema y *backups,* crear índices, gestionar cuotas, etc. También permite exportar la BD total o parcialmente.

**Privilegios de sistema:**

- Derecho a realizar una acción genérica sobre el sistema (creación de objetos, operaciones de la BD en conjunto, …)

    - Sobre TABLE: CREATE..., ALTER ANY..., BACKUP ANY..., DROP ANY..., SELECT ANY..., UPDATE ANY...
    - Sobre INDEX: CREATE..., CREATE ANY..., DROP ANY...
    - Sobre VIEW: CREATE ANY..., DROP ANY...
    - Sobre PROCEDURE: CREATE/EXECUTE ANY...
    - Sobre PRIVILEGE: GRANT ANY...
    - Sobre ROLE: CREATE ROLE..., CREATE ANY...
    - Sobre SESSION: CREATE...
    - Sobre USER: CREATE...
    - Sobre TABLESPACE: CREATE..., ALTER..., DROP...
    - Sobre DATABASE: ALTER...

*Exportfulldatabase: ese privilegio permite exportar la BD a otra BD.*

*GRANT: Permite conceder permisos a los usuarios, GRAN ANY.*

*REVOQUE: Retira privilegios a un usuario o a un rol.*

Se puede hacer un *GRANT* de un rol a otro, para construir conjuntos de permisos cada vez más grandes.

```
GRANT <priv / role>
TO <user / role / PUBLIC>
[WITH ADMIN OPTION];


REVOKE <priv / role>
FROM <user / role / PUBLIC>;
```

**Privilegios de objeto:**

- Derecho a realizar una acción particular sobre un objeto concreto (insertar tuplas, borrar tuplas, consultar tuplas, …)
- Ejemplos:
  - ALL (todos los permisos a un usuario sobre un objeto)
  - DELETE (Borrar de tabla)
  - EXECUTE (ejecución)
  - INDEX (crear índices)
  - INSERT (insertar)
  - UPDATE on (y poner lista de atributos)
  - WITH GRANT OPTION (ceder privilegio a otros usuarios).
  - REVOKE ON (permiso que quieras requisar al usuario, todo especificado)

```
GRANT <priv / role>
ON <objeto>
TO <user / role / PUBLIC>
[WITH ADMIN OPTION];


REVOKE <priv / role>
ON <objeto>
FROM <user / role / PUBLIC>;
```

**Roles y privilegios en el catálogo:**

| Vista | Descripción |
|---|---|
| DBA_ROLES | Lista de roles existentes |
| DBA_ROLE_PRIVS | Roles asignados a usuarios o a otros roles |
| ROLE_ROLE_PRIVS | Roles asignados a roles |
| DBA_SYS_PRIVS | Privilegios de sistema asignados a usuarios o a roles |
| ROLE_SYS_PRIVS | Privilegios de sistema asignados a roles |
| ROLE_TAB_PRIVS | Privilegios de tablas asignados a roles |
| DBA_TAB_GRANTS_MADE | Privilegios de tablas asignados a usuarios |

(palabras clave: *dba* y *privs*)

## Fiabilidad y recuperación frente a fallos.

- **Transacciones:** mantener esa consistencia mediante varias reglas (autosalvado).
- Gestión de bloques y buffers.
- Recuperación de transacciones: cuando hay un problema (se ha ido la luz) y el sistema vuelve, tiene que volver a un estado de consistencia anterior de forma que se restaure la semántica anterior.
  - La tabla de modificaciones.
  - Modificación diferida de la BD.
  - Puntos de Verificación.

- Las transacciones de Oracle.
- **Fiabilidad:** Consistente. Que concuerda con una semántica especifica (serie de normas que deben cumplir los datos de la BD.)
    - Invariante de representación: condición lógica que hay que cumplir. Ejemplo: Una fecha.
- Consistencia es un concepto relacionado con una BD. En un SGBD hay muchas bases de datos y cada una con su semántica/consistencia.

### Fallos en el SGBD:

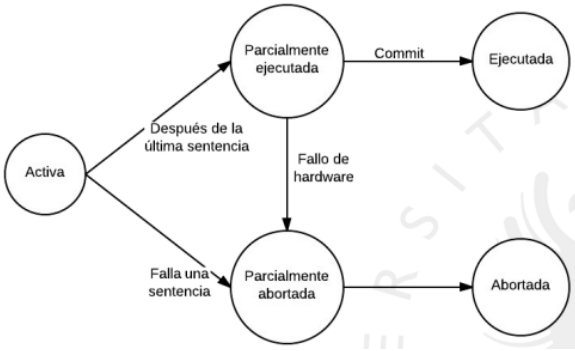
- Errores lógicos: problemas internos.
- Errores del sistema: acceso concurrente/violación/exceso de procesos. La transacción se ejecuta cuando se recupere el sistema
- Caída: fallo hardware o eléctrico.
- Fallo en almacenamiento externo: El disco(s) que almacena la información falla, y automáticamente el sistema muere como mecanismo para guardar todos los discos no afectados.

¿Por qué es importante la consistencia? Por ejemplo, en una cuenta bancaria. Si en un momento determinado no cuadran los sueldos de las cuentas, se pueden recuperar los movimientos, sumarlos y se recupera los saldos.

**Transacciones:**

- Definición: Unidad lógica de procesamiento constituida por varias sentencias que deben ejecutarse en bloque o no ejecutarse.
- Para ejecutarse entero debe tener todos los recursos.
- Las transacciones son algo que tiene que ejecutarse en bloque.
- Caso cuentas bancarias: si retiro dinero, me ha de dar el dinero y actualizar el banco, y anotar el movimiento y actualizar el saldo tiene que ser un movimiento que se haga entero o que no se haga. Si hay fallo que deshaga el movimiento.
- Propiedades (ACID):
    - Atómica (Atomicity): se hace entero o no se hace.
    - Consistente (Consistency): antes y después la consistencia se cumple.
    - Aislada (Isolation): no hay problemas.
    - Persistente (Durability): cuando hayas terminado de hacer operaciones, todo se queda.

- Estados:
    - Activa: empezado.
    - Parcialmente ejecutada: ha llegado a un punto de la ejecución que ha terminado las operaciones. Si se consigue hacer datos consistentes se pasa a ejecutada. Datos persistentes en el sistema. Si no lo conseguimos, pasa a parcialmente abortada.
    - Parcialmente abortada: alguna operación no ha salido bien, hay que prepararse para volver atrás.
    - Ejecutada: Ejecutadas las sentencias y es firme (datos persistentes).
    - Abortada: Ha habido fallo y hay que retroceder.



## Gestión de bloques y buffers.

- Para prevenir fallos es necesario:
  - Mantener información para permitir recuperar.
  - Garantizar la consistencia de la BD y la atomicidad de las transacciones.

- La ejecución de transacciones produce intercambios de información entre memoria volátil y la no volátil.
- Las transferencias a discos pueden tener estados:
  - Realizada con éxito.
  - Parcialmente fallida: se reintenta.
  - Totalmente fallida.

- Cualquier fallo tiene que se ha detectado y, si ocurre, el bloque destino debe quedar intacto.
- Para hacer una operación de lectura se ha de leer de disco. Cuando accedemos a datos hay que pedir que se cargue ese dato.
- La escritura se hace al contrario. El dato se mantiene en memoria todo el tiempo que se pueda. Si no se va a usar en un tiempo se devuelve a disco, pero eso no es lo que realmente se hace. Se necesita un mecanismo para solucionarlo, ya que esto ralentiza mucho el sistema.
- Alternativa:
  - Usar dos bloques físicos por cada lógico, cuando escribo, escribo en dos bloques y si falla el primer bloque intento leer el otro y entonces la transacción es correcta.
    - Se escribe en uno,
    - Si no hay fallo, entonces se escribe el otro.
    - Será exitosa cuando se hayan realizado las dos.

  - Para la recuperación de errores:
    - Si los dos son iguales y no se detectan errores, no hay fallos.
    - Si se produce fallo durante la lectura uno, se copia el otro bloque encima del fallo.

- Operaciones entre buffers y disco (escritura):
  - Lee_bloque (X): transfiere el bloque físico que contiene el dato X en buffer.
    - Edad de una persona.
  - Escribe_bloque(X): transfiere el buffer que contiene el dato X al disco, reemplazando el bloque físico.

- Operaciones entre programa y buffers(lectura): (si los datos no están en la memoria del programa no puedo hacer cálculos, ya que no se puede hacer cálculos en un bloque)
  - Lee (X, x): lee el dato X del buffer sobre la variable X, y si es necesario, se ejecuta lee_bloque(X) si el dato no está en memoria.
  - Escribe (X, x): asigna el valor de la variable local x al dato X del buffer, si es necesario, porque el dato no está en memoria, se ejecuta lee_bloque(X):
    - Si el bloque que tiene X se modifica el valor x
    - Si no está en memoria ejecuta lee_bloque(X) (INSERT)

- La lectura se realiza por necesidad del dato.
- La escritura se realiza por necesidad de espacio.

**Recuperar transacciones:**

- Si hay datos modificados en los buffers y ocurre una caída del sistema, parte de los datos pueden haber sido escritos en el disco. ¿Cómo recuperar valores antiguos?
- Problemas:
    - Determinar que valores han sido modificados es costoso en operaciones E/S.
    - Encontrar los valores antiguos de los datos es aún más costoso (cláusulas WHERE sobre varias tuplas).

- Recuperación de fallos activa: garantiza que guarda suficiente información para poder recuperarse.

**Tabla de modificaciones/bitácora**: mantiene la información recuperable.

- Definición: Mantienen una tabla en memoria con las modificaciones que se quieren realizar (*log*) mediante inserciones, actualizaciones o borrados.
- Almacena: código de transacción, estado, operación, marca de tiempo, nombre del dato, valor antiguo y valor nuevo, aunque algunos sistemas pueden incluir más.
- DML: operaciones totalmente recuperables.
- *Create_table*: no se puede mantener en memoria.
- *Drop_Table*: imposible de recuperar.
- Copia de seguridad es la única forma de recuperar esas acciones.
- Cada vez que comienza la transacción se divide en operaciones más simples y un usuario no puede acceder.
- Al comenzar una transacción, se escribe una entrada <Ti, *start*>, con el resto de valores vacíos, simplemente se indica que se ha iniciado.
- Antes de hacer la operación, anoto que la he hecho (*redulog*).
- Cualquier operación de una transacción, va precedida por su correspondiente entrada.
- Cuando termina una transacción, se escribe una entrada <Ti, *commit*> (*commit* = cometida o finalizada), con el resto de valores vacíos. Marca el final de la transacción. Se hace cuando consideras que está totalmente ejecutada, si no está el *commit* entonces está parcialmente ejecutada.

| Instante | T1 | T2 | T3 |
|---|---|---|---|
| 1 | lee(A,r) | | |
| 2 | r:=r*2 | lee(W,n) | |
| 3 | Escribe(A,r) | lee(X,t) | |
| 4 | | t:=t+n | lee(A,s) |
| 5 | | escribe(X,t) | |
| 6 | | lee(Y,t) | s:=s*5 |
| 7 | | t:=t-n | escribe(A,s) |
| 8 | | escribe(Y,t) | lee(B,s) |
| 9 | | | s:=s*5 |
| 10 | | | escribe(B,s) |

- ¿A qué sentencia de sql está asociada la primera transacción?: UPDATE o modificación. (lee, hace cambio y escribe).
- Transacción dos (T2): SELECT, se lee de memoria (W). Sobre la X se hace un UPDATE. Y sobre la Y otro UPDATE. Se consiguen modificaciones.
- Más o menos, se ejecuta al mismo tiempo, aunque en informática nada se ejecuta al mismo tiempo (pero aproximadamente sí). Contenido de la tabla de transacciones cambia dependiendo de cuando se ejecute uno u otro. La T2 empieza detrás de la T1 y la T3 después de la mitad del T2.

- ¿Cuál creéis que sería la primera entrada a la tabla de modificaciones?: T1-marca de tiempo-activa (y el resto vacío).
- TABLA DE MODIFICACIONES:

| T_id | Hora | Estado | Operación | Dato | Vantiguo | Vnuevo |
|---|---|---|---|---|---|---|
| T1 | I1 | start | | | | |
| T2 | I2 | start | | | | |
| T1 | I3 | | update | A | 2 | 4 |
| T1 | I4 | commit | | | | |
| T3 | I5 | start | | | | |
| T2 | I6 | | update | X | 10 | 15 |
| T3 | I7 | | update | A | 4 | 20 |
| T2 | I8 | | update | Y | 20 | 15 |
| T2 | I9 | commit | | | | |
| T3 | I10 | | update | B | 16 | 80 |
| T3 | I11 | commit | | | | |

### Modificación diferida de la BD:

- La tabla de modificaciones tiene que escribirse frecuentemente (sobre todo, después de un *commit*).
- De hecho, hasta que la transacción no está guardada en la tabla y está en el disco, no comienza la ejecución real de la transacción (modificaciones en el buffer de este al disco), es decir, se anota que voy a cambiar los valores, pero hasta no haber hecho el *commit* no se hacen los cambios.
- De este modo, la ejecución real se aplaza hasta que la transacción esté parcialmente ejecutada.
- Cuando ocurre una caída:
  - Se mira la última copia de la tabla de modificaciones.
  - Las transacciones que no estén parcialmente ejecutadas no han tenido ejecución real y pueden olvidarse (borrándolas de la tabla): *UNDO(T).*
  - Las transacciones parcialmente ejecutadas pueden no haberse ejecutado realmente y es mejor rehacer la transacción volviendo a escribir los valores nuevos: *REDO(T).*

- En resumen:
  - *UNDO(T):* se ejecuta cuando hay un *start* pero no un commit.
  - *REDO(T):* se ejecuta cuando hay un *start* y un *commit*.

### Puntos de verificación:

- Hay dos problemas:
  - La tabla de modificaciones puede ser enorme.
  - Muchas transacciones ya fueron escritas, pero no lo sabemos.

- Para solucionarlos, se introducen los *checkpoints* (punto de verificación). El proceso es:
  - Grabar la tabla de modificaciones en disco.
  - Guardar los bloques modificados en los *datafiles.*
  - Escribir un registro *checkpoint* en la tabla.
  - Graba la tabla de modificaciones en disco.
  - Al final se pone un *checkpoint*.

- El algoritmo que incluye *checkpoints* es:
    - Se efectúa un *REDO* para aquellas transacciones con un *commit* tras el último *checkpoint*.
    - Se efectúa un *UNDO* para aquellas transacciones que tengan un *start* antes del último *checkpoint* y no tengan un *commit* tras el último *checkpoint*: hay que hacer un deshacer, retroceder la transacción ya que se no se considera hecha.

**Gestión de transacciones en ORACLE:**

- Oracle usa el mecanismo de *redo log buffer* y *redo log file* para implementar las transacciones.
- Usa dos buffers para que se pueda seguir escribiendo en uno mientras el otro se transfiere a disco.
- Una transacción comienza en cualquier operación DML, y no puedo meter sentencias de otro lenguaje porque se da por finalizada.
- Termina cuando:
    - Se hace una llamada a *COMMIT* o *ROLLBACK*.
    - Se ejecuta una sentencia *DDL*.
    - El usuario desconecta (*EXIT*, hace un *commit*), o
    - El proceso actual termina de forma anormal.

**Sentencia *COMMIT* de Oracle:**

- Graba el *redo log buffer* en el *redo log file*.
- Da por finalizada la transacción.
- Libera los recursos empleados o bloqueados por la transacción.

**Sentencia *ROLLBACK* de Oracle:**

- Aborta la transacción en curso.
- Deshace los cambios registrados por la transacción abortada.
- Libera los recursos empleados o bloqueados por la transacción, para que otros usuarios puedan usarla.
- *ROLLBACK*: Vuelve a cargar los bloques como estaban, vuelvo al principio, se deshace todo.

**Sentencia *SAVEPOINT* en Oracle:**

- Establece un punto de guardado de los cambios, con un identificador único. Permite hacer un *ROLLBACK* a un estado anterior al mismo pero posterior al comienzo de la transacción.
- Sintaxis: **ROLLBACK TO <savepoint name>;**
- *Savepoint*: deshacer cambios del *savepoint* en adelante, te permite volver a un estado intermedio de la transacción. Vuelves al punto donde todo estaba bien.

## Salvado y recuperación de una BD:

- Otra parte de la fiabilidad son las copias de seguridad: responsabilidad del administrador de sistemas.
- Esquemas de *backup* de Oracle:
    - Copia física: copiar físicamente los ficheros que están sin interpretar el contenido.
      Condiciones:
        - Salvado en frío: con el sistema apagado completamente.
        - Recuperación de una copia en frío.
        - Salvado en caliente (*on-line*): sistema encendido. El sistema se copia a sí mismo.
        - Recuperación de una copia en caliente.

    - Copia lógica: exportación. Copia los datos y no el sistema. Copia de seguridad que te permite importar en otro sistema. La copia lógica la realiza el propio SGBD y lo que se hace es o guardar toda la BD o esquemas de BD (conjunto de objetos asociados a un usuario). Para cada objeto se puede guardar o parte o toda su estructura (contenido, restricciones…)
        - Problema: se puede recuperar con la misma versión. (Si se instala ORACLE 11.2.0, solo se puede restaurar ORACLE 11.2.0).

    - Importación.
    - Criterios de selección: con que temporalidad hay que usarlos. Hay que usarlos todos.

- La persistencia puede verse comprometida por un fallo en el almacenamiento masivo.
- Es necesario realizar copias de seguridad.
- El estado de la base de datos puede recuperarse a partir de la copia de seguridad y rehaciendo las transacciones posteriores a dicha copia.
- Copia física:
    - Suelen usar aplicaciones del SO.
    - Debe ser completa y consistente, debe incluir todos los ficheros de BD (*datafiles*, ficheros de control, de instalación, de configuración, parámetros, contraseña, ficheros de *redu log files* de modo archivados…, se incluye el software de instalación, cualquier parche que se le haya pasado a la BD hasta ese momento)
    - Se realiza con la base de datos "derribada" (no levantada).
    - Ventaja de la copia física: es muy simple, lo único que se hace es usar herramientas del SO para copiar, nunca se hace a mano, sino que se crea un script de forma que garantice que ninguna copia de seguridad física que hagas, pierda ningún fichero.
    - No hay problema de que haya mutabilidad (acceso de usuarios) dentro de la BD, porque en ese caso ni siquiera la BD está funcionando.

- Copia lógica:
    - Realizada por el propio sistema SGBD.
    - Guarda total o parcialmente objetos de la BD pero sin su estructura interna. Puede guardar un usuario, una tabla, toda la base de datos, guarda las ordenes necesarias para restaurar ordenes de la BD.

**Copia física en ORACLE:** Salvado en frío:

- Se hace una copia de seguridad incluyendo todos los ficheros que están incluidos dentro del propio sistema.
- En este caso, una copia sólo podrá ser recuperada por otra instalación exacta a la que generó los datos.
- En el caso de Oracle hablamos de los ficheros de datos (*data file*), ficheros de control con todas sus copias (*control file*), y los ficheros de la tabla de modificaciones (*redu log files*), ficheros de parámetros, de contraseñas, de alertas y todos los ficheros necesarios para hacer la instalación desde cero y pasar todos los parches del software hasta que momento en que ocurrió el fallo para poder restablecer el sistema.
- Procedimiento:
    - Detener la instancia
    - Usar los comandos del SO para copiar los ficheros correspondientes al dispositivo de copia de seguridad.
    - Iniciar la instancia.
- Los usuarios no pueden acceder mientras tanto (problemático en algunas situaciones).
- Los "ficheros correspondientes" son:
    - *Datafiles.*
    - *Control file*.
    - *Redo log files*
- Se recomiendan también los ficheros *init.ora* y *config.ora* y el software de aplicaciones actualizado mediante parches.

**Copia física en Oracle:** recuperación de una copia en frío.

- Habrá que sustituir todos los ficheros actuales por los de la copia de seguridad.
- El reemplazo es necesario hacerlo con la base de datos "derribada".
- Cuando se intenta recuperar el sistema de un fallo de este tipo lo que se hace es sustituir los ficheros actuales, es decir, hacer una "suplantación" e instala una BD exactamente igual con los mismos *data files*, contenidos etc. y se reemplazan todos los ficheros (no puedes dejarte ni uno sin reemplazar) porque entonces no funciona.

**Copia física en Oracle:** Salvado en caliente:

- Partes del propio SGBD que se puede copiar sin apagar el sistema. Simplemente se pone offline una parte del SGBD, en concreto se puede hacer con los *data files* o con un *tablespace* completo, se pone off-line, se copia y lo vuelves a poner on-line.
- Se puede hacer una copia de seguridad con el sistema encendido.
- Cuando pones off-line los elementos de los que quieres hacer copia de seguridad, todas las transacciones que afecten a objetos que están dentro de esos elementos automáticamente tienen que volcarse, es decir, fuerza un volcado de los *redu log*, los guarda en el disco por si hay algún tipo de problema y esos elementos hay que guardarlos aparte al estar en off-line, se van escribiendo en el fichero "*Redu log archivados*" mientras hacemos la copia de seguridad, hasta que se desactive la copia de seguridad. Cuando se desactive la copia de seguridad todo lo que se ha hecho en los "*redu log files* archivados" que sería como una declaración de intenciones, se vuelca en el *redu log files* de la BD hasta que llega un momento que esos archivos se vacían.
- Permite hacer copia de seguridad con la base de datos en uso por parte de los usuarios.

- El sistema transfiere las actualizaciones de datos que cualquier transacción terminada (en la tabla de modificaciones).
- Las nuevas transacciones y transacciones en curso se almacenan en la tabla de modificaciones, pero no se transfieren bloques de datos a disco. Esto requiere muchos ficheros redo log file numerados de forma consecutiva.
- Cuando la copia termine, las transacciones se rehacen sobre los *datafiles* para almacenar los cambios.
- Este modo se llama *archive log mode* y debe estar habilitado.
- Procedimiento:

```
ALTER TABLESPACE <nombre> BEGIN BACKUP;
HOST xcopy <ruta>\<nombre>*.dbf
<destino>
ALTER TABLESPACE <nombre> END BACKUP;
```

  - El procedimiento consiste en pasar un *tablespace* al modo *backup*. Deja que los usuarios sigan leyendo, pero no escribiendo sobre él, pudiendo usar los comandos que quieras del SO para hacer la copia de seguridad de uno o varios de los *datafiles* del *tablespace*.
  - Se copia en la ubicación de la copia de seguridad.
  - Después, se desactiva la copia de seguridad para que el sistema vuelva a trabajar.
  - Cada vez que hacemos un *begin backup* se introduce una entrada en una tabla del sistema que puede consultarse a través de la vista *v$backup* (vista de rendimiento dinámico, están asociadas al estado actual de la BD, no están almacenadas en el sistema, existen ya que la BD está funcionando).

**Copia física en Oracle:** recuperación de una copia en caliente:

- Cuando un *tablespace* o uno de sus *datafiles* produce un fallo, se pueden reemplazar por una copia anterior.
- El proceso sustituye los *datafiles* y aplica todos los cambios posteriores a la copia a partir de los históricos de ejecución:

```
RECOVER TABLESPACE <nombre>;
RECOVER DATAFILE <ruta>;
RECOVER DATABASE;
```

- Cuando el sistema entra en un error, eso significa que el sistema se cae la mayor parte de las veces. Entonces intentamos recuperarnos del error. Lo suyo es si el sistema está funcionando, el *tablespace* se pone "offline" y podemos recuperar o iniciar el modo de recuperación. Si el sistema se ha caído, se intenta volver a levantarlo (*startup*) y el sistema te informa de que no se puede recuperar porque uno de los *datafiles* esta corrupto.
- ¿Qué se hace?: Se va a la ubicación de la copia de seguridad y se sustituye el fichero corrupto por el que se tiene la copia de seguridad y se ejecuta la orden *recover* si quieres recuperar todo el *tablespace*, que va a recuperar todos los *datafiles*, o pones que quieres recuperar un *datafile* especifico. Se abre el *datafile* y va a mirar los *redu log* a ver si la última copia de seguridad está dentro de los *redu log*, significa que cuando se hizo la copia de seguridad todavía está dentro de los *redu log*.

- Se modifica el *data file* aplicando a todos los bloques, a cada dato que haya modificado, todo lo que hay en los *redu log* y cuando haya terminado te informa de la recuperación.
- Y a continuación, se indica que abra la base de datos *alter database open* y eso pone el sistema a funcionar otra vez.
- Si se rompen todos los *datafiles*, puedes recuperar todo el *tablespace* o cada *datafile* uno a uno.
- Si solo se ha roto un *datafile* y se sustituyen todos los *datafiles* del *tablespace* por las versiones de la copia de seguridad y se recupera el *tablespace* completo tarda muchísimo más tiempo ya que se tiene que recuperar los datos de cada uno de los ficheros que no estaban rotos.
- Mucho mejor solo recuperar un *datafile*.
- PROBLEMA: Si el *redu log file* no cubre todo el tiempo no hay forma de recuperarlo, luego el sistema tiene un problema gordo.
- En el modo archivado no hay problema porque esos *redu log* archivados que se pueden mantener con estados en todo momento, es decir, tu puedes desactivar el modo *backup* de un *tablespace*, como tenga activado el modo archivado el sistema sigue escribiendo todas las modificaciones que se hacen en dos sitios, en los *redu log files* y en los *redu log files* del modo archivado, el problema es que los *redu log files* del modo archivado no son cíclicos, cuando se termina un fichero abre otro y al terminarse ese, abre otro y así consecutivamente. Hasta que se come todo el espacio del sistema.
- El *redu log* del modo archivado es muy útil, pero se come mucho almacenamiento.

**Copia lógica en Oracle:** exportación.

- Consulta la base de datos y el catálogo para crear un fichero binario con los objetos seleccionados de *extensión .dmp*
- Puede hacerse una exportación de la base de datos completa, esquema de un usuario concreto (o varios) o una aplicación, o una tabla concreta (o varias), o un objeto de un esquema o una aplicación.
- En el caso de los objetos se puede exportar la tabla solo con estructura, pero sin contenido, es decir, sin filas, pero con toda su estructura e incluso con sus restricciones (claves primarias, externas etc.).
- Una forma fácil de exportarlo sería mirando la consulta que se usó para crearlas y exportar dicha sentencia.
- Permite almacenar simultáneamente información de catálogo correspondiente al objeto u objetos salvados (privilegios, índices, restricciones).
- Las exportaciones de base de datos se llaman completas y las de tablas modificadas se llaman incrementales. (Se puede programar hacer una completa semanalmente y otra incremental diaria).

**Recuperación de una copia lógica en Oracle:** Importación.

- Permite leer un *fichero.dmp* (dump) recuperando su contenido sobre una base de datos y especificando el usuario que creará los objetos recuperados.
- *Oracle imp* para importación.

**Criterios de selección:**

- Hay que indicar un usuario con una contraseña que tengan los privilegios necesarios tanto para poder realizar la exportación o la importación, si se tienen privilegios para hacer la exportación, tu puedes hacer la exportación de tus propios objetos, aunque normalmente el administrador no te lo permita (ley de protección de datos). Luego el administrador debe garantizar las copias de seguridad.
- Todos los métodos de la copia de seguridad no son excluyentes, de hecho, hay que usar cuantos más mejor, el problema es que no se pueden usar en todo momento.

- La exportación te da otra herramienta de copia de seguridad que puedes realizar más de seguido.
- Se elige siempre un método primario, y otro en caso de que este falle.
- La copia en frio solo se usa cuando fallan los demás.
- En bases de datos orientadas a las transacciones, es mejor la copia en caliente.
- Si se usa exportación, tendremos que conformarnos con el estado de la base de datos cuando se exportó.
- Cualquier procedimiento debería incluir una exportación y una copia física.
- Problema: si ocurre un fallo integral, la única forma de recuperar la base de datos en un estado cualquiera, y el único sitio donde tenemos una base de datos es en la copia física en frio, luego tendría que recuperar el estado de la copia de seguridad de cuando sea. Lo único que se tendría que hacer después sería aplicarle todas las copias de seguridad en especial la última que tenga en caliente, y a partir de ahí aplicar las copias incrementales que tenga después, para llevarla al estado actual de la BD, pero recuperarse de un fallo integral es grave y llevaría mucho tiempo.
- Es decir, hay que integrarlos todos y crear una estructura de copias de seguridad a largo plazo que incluya todos los tipos y es un ciclo que vuelve a repetirse, de hecho, nos da tiempo a cambiar el ciclo para la siguiente iteración por el hecho de haber detectado algún tipo de disfuncionalidad en el que no nos haya pasado nada, pero habría que resolverlo.