



UNIVERSIDAD
DE GRANADA



Administración de Bases de Datos

Grado en Ingeniería Informática

Seguridad y fiabilidad de los datos



I. J. Blanco, A. G. López Herrera

**Departamento de Ciencias de la
Computación e Inteligencia Artificial**
<http://decsai.ugr.es>

- Seguridad de los datos
- Fiabilidad y recuperación frente a fallos
- Salvado y recuperación de BD



- Autorización de usuarios
- Gestión de privilegios y roles
 - de Sistema
 - de Objetos
- Privilegios en el catálogo



- A cualquier S. I. se le pide:
 - que garantice la exclusividad en el acceso a la información a quien tiene permisos
 - que garantice que se puede recuperar cuando ocurra un fallo

- Para garantizar accesos indebidos se puede actuar:
 - A **nivel físico**: acceso a ubicación
 - A **nivel humano**: confianza en usuarios con permisos
 - A **nivel del S. O.**: si permite conexión remota y proporciona acceso local desde conexión remota
 - A **nivel del S. B. D.**: a nivel de información (quién accede a qué) y de operación (qué puede hacer)

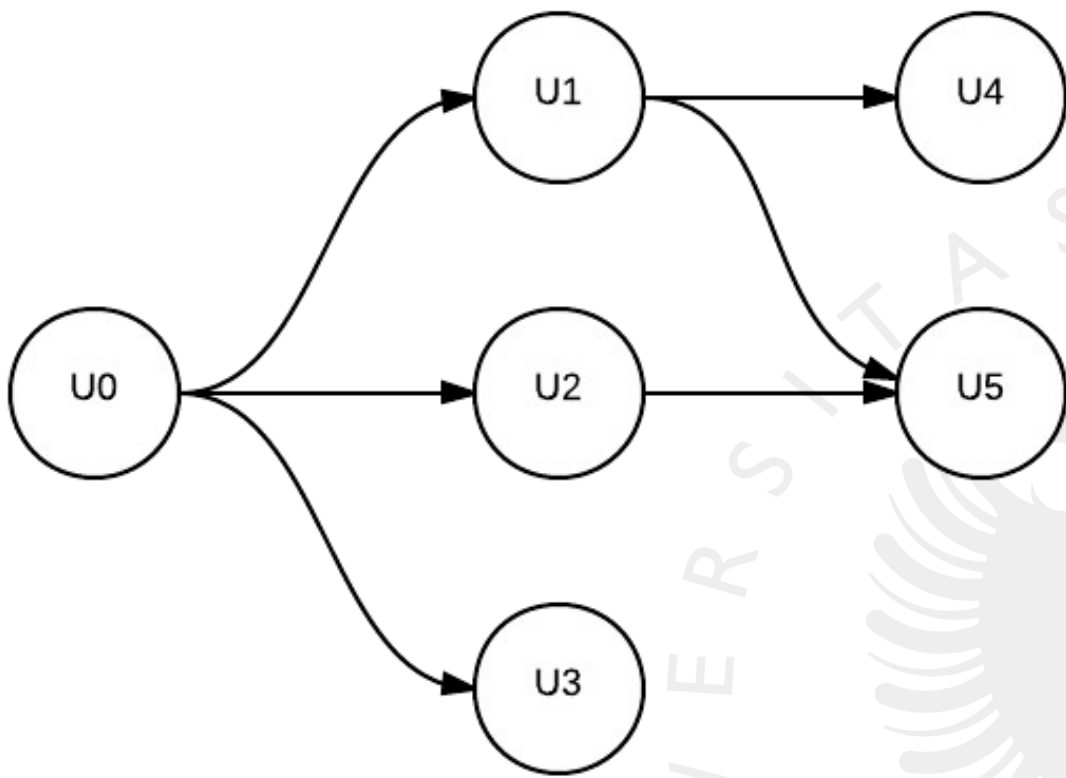
- Con información sensible, mejor cifrar los datos y las conexiones.

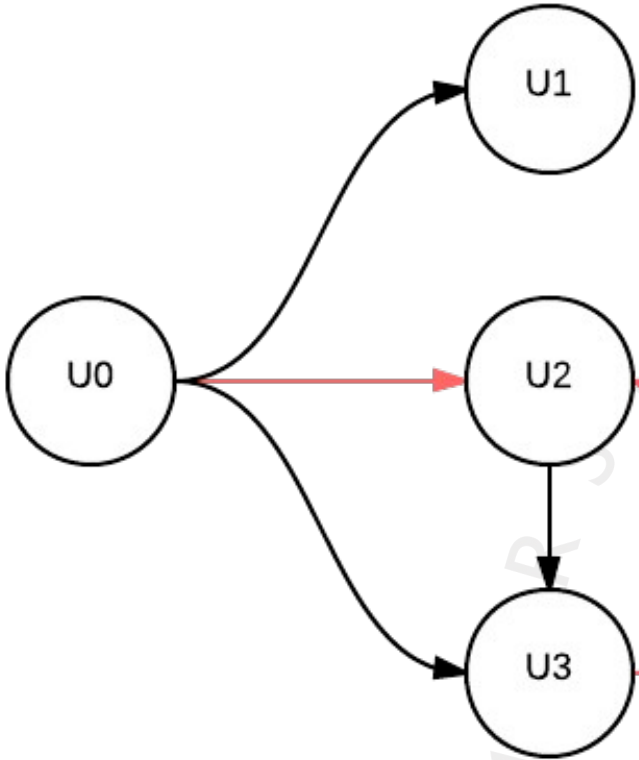
- Autorización de usuarios
- Gestión de privilegios y roles
 - de sistema
 - de objeto
 - ¿dónde encontrarlos en el catálogo?

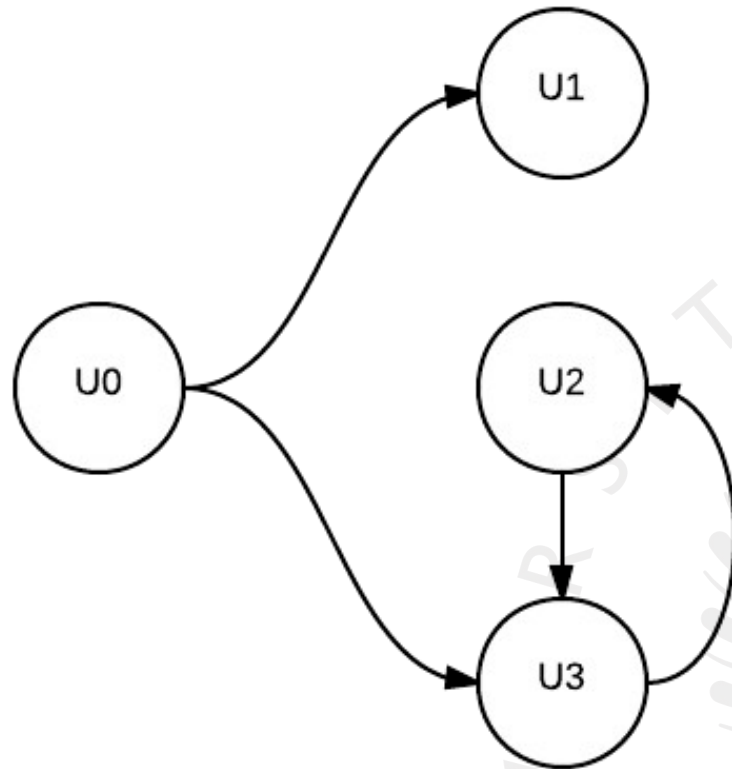


- Dos niveles de autorización:
 - Par usuario/contraseña
 - Según el uso de los datos:
 - En *sistemas centralizados*, el administrador lo crea todo y concede permisos.
 - En *sistemas descentralizados*, hay una jerarquía de usuarios con permisos que pueden concederse o cederse, según determine el propietario.

- Consultar tabla: para leer o crear vistas sobre ella
- Insertar tuplas (pero no modificar)
- Borrar tuplas (incluso todas)
- Modificar tuplas en algunos o todos los atributos
- Modificar la tabla
- Borrar la tabla
- Crear índices sobre la tabla (¡cuidado si no es tuya!)
- Concesión y cesión de permisos a otros usuarios







- **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 e índices.
 - *DBA*: 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.

- 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...




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

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

- Derecho a realizar una acción particular sobre un objeto concreto (insertar tuplas, borrar tuplas, consultar tuplas, ...)

- ALL (PRIVILEGES)
- DELETE
- EXECUTE
- INDEX
- INSERT
- REFERENCES
- SELECT
- UPDATE



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

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



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

- **Errores lógicos:** por problema interno (*overflow*, entrada inválida, etc.)
- **Errores del sistema:** acceso concurrente o exceso de procesos. La transacción se ejecuta cuando se recupere el sistema.
- **Caída:** fallo hardware o eléctrico.
- **Fallo en almacenamiento externo:** sólo salvable si tenemos copias de seguridad.

- El ejemplo de los movimientos y los saldos bancarios.

Cuenta	Saldo
1	50
2	20

Cuenta	Movimiento	Cantidad
1	1	100
2	1	20
1	2	-50

- Registro de un nuevo movimiento en la cuenta 2 (cantidad de -5)

Cuenta	Saldo
1	50
2	20

Cuenta	Movimiento	Cantidad
1	1	100
2	1	20
1	2	-50
2	2	-5

PASO 1

- Registro de un nuevo movimiento en la cuenta 2 (cantidad de -5)

Cuenta	Saldo
1	50
2	20 -15

PASO 2

Cuenta	Movimiento	Cantidad
1	1	100
2	1	20
1	2	-50
2	2	-5

- ¿Qué ocurre cuando hay una caída entre las dos operaciones?

Cuenta	Saldo
1	50
2	20

~~PASO 2~~

Cuenta	Movimiento	Cantidad
1	1	100
2	1	20
1	2	-50
2	2	-5

PASO 1

- Conclusión:
 - El contenido de la base de datos sólo es **consistente** si:
 - no se realiza ningún paso, o
 - se realizan los dos pasos.
 - El contenido de la base de datos es **inconsistente** en cualquier otro caso.

- Transacciones
- Gestión de bloques y *buffers*
- Recuperación de transacciones
 - La tabla de modificaciones
 - Modificación diferida de la Base de Datos
 - Puntos de Verificación
- Las transacciones en Oracle®

Transacción: unidad lógica de procesamiento constituida por varias sentencias que deben ejecutarse en bloque o no ejecutarse.

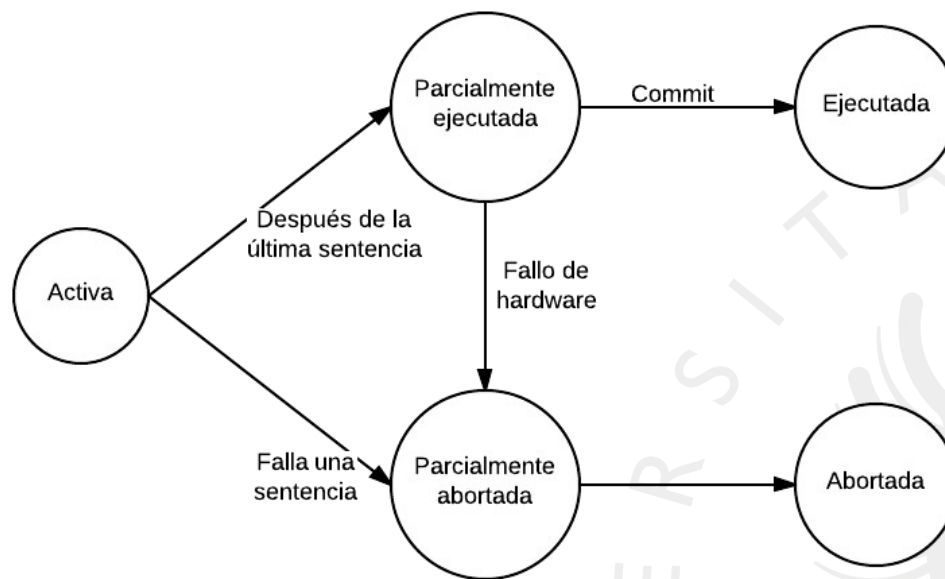
Propiedades (ACID):

- atómica (*Atomicity*)
- consistente (*Consistency*)
- aislada (*Isolation*)
- persistente (*Durability*)



- activa
- parcialmente ejecutada
- parcialmente abortada
- ejecutada
- abortada





- 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 la memoria volátil y la no volátil.
- Las transferencias pueden tener estados:
 - realizada con éxito
 - parcialmente fallida
 - totalmente fallida

- Cualquier fallo tiene que ser detectado y, si ocurre, el bloque destino debe quedar intacto.
- Alternativa: dos bloques físicos por cada bloque lógico:
 - se escribe primero uno,
 - si no hay fallo, entonces se escribe el otro;
 - será exitosa cuando se hayan realizado las dos.

Para la recuperación:

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

- Operaciones entre *buffers* y disco:
 - lee_bloque(X): transfiere el bloque físico que contiene el dato X al *buffer*
 - escribe_bloque(X): transfiere el *buffer* que contiene el dato X al disco, reemplazando el bloque físico
- Operaciones entre programa y *buffers*:
 - lee(X,x_i): lee el dato X del buffer sobre la variable local x_i - si es necesario, se ejecuta lee_bloque(X)-
 - escribe(X,x_i): asigna el valor de la variable local x_i al dato X del buffer -si es necesario, se ejecuta lee_bloque(X)-

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

- 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 qué 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)

- Los SGBD mantienen una tabla en memoria con las modificaciones que se quieren realizar (*log*) mediante inserciones, actualizaciones o borrados. Se llama **tabla de modificaciones** o **bitácora**.
- Almacena: código de transacción, estado, operación, marca de tiempo, nombre del dato, valor antiguo y valor nuevo, aunque algunos sistemas pueden incluir más.

T_id	Hora	Estado	Operación	Dato	Vantiguo	Vnuevo
...

- Al comenzar una transacción, se escribe una entrada $\langle T_i, \text{start} \rangle$, con el resto de valores vacíos.
- Cualquier operación de una transacción, va precedida por su correspondiente entrada.
- Cuando termina una transacción, se escribe una entrada $\langle T_i, \text{commit} \rangle$, con el resto de valores vacíos.

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)

Con valores iniciales A=2, B=16, W=5,X=10,Y=20

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				

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				

- 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 esta en el disco, no comienza la ejecución real de la transacción (modificaciones en el buffer de este al disco).
- 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_i)
 - Las transacciones parcialmente ejecutadas pueden no haberse ejecutado realmente y es mejor *rehacer* la transacción volviendo a escribir los valores nuevos: REDO(T_i)

- En resumen:
 - $\text{UNDO}(T_i)$ se ejecuta cuando hay un *start* pero no un *commit*
 - $\text{REDO}(T_i)$ se ejecuta cuando hay un *start* y un *commit*

- 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*. 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.

- 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*.

- 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.
- Termina cuando:
 - se hace una llamada a COMMIT o ROLLBACK,
 - se ejecuta una sentencia DDL,
 - el usuario se desconecta, o
 - el proceso actual termina de forma anormal.

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

- Aborta la transacción en curso
- Deshace los cambios registrados por la transacción abortada
- Libera los recursos empleados o bloqueados por la transacción.

- 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>;

- Esquemas de *backup* en Oracle®
 - Copia física
 - Salvado en frío
 - Recuperación de una copia en frío
 - Salvado en caliente (*on-line*)
 - Recuperación de una copia en caliente
 - Copia lógica: exportación
 - Importación
 - Criterios de selección

- 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 S. O.
 - Debe ser completa y consistente (*datafiles*, ficheros de control, de instalación, de configuración, ...)
 - Se realiza con la base de datos “derribada” (no levantada)

- Copia lógica:
 - Realizada por el propio SGBD
 - Guarda total o parcialmente objetos de la BD pero sin su estructura interna

- En este caso, una copia sólo podrá ser recuperada por otra instalación exacta a la que generó los datos.
- Procedimiento:
 - Detener la instancia
 - Usar los comandos del S. O. 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.

- Habrá que sustituir todos los ficheros actuales por los de la copia de seguridad.
- Es necesario hacerlo con la base de datos “derribada”.

- Permite hacer copia de seguridad con la base de datos en uso por parte de los usuarios.
- El sistema transfiere las actualizaciones de datos de 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 **archivelog mode** y debe estar habilitado.

- Procedimiento:

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

```
ALTER TABLESPACE <nombre> END BACKUP;
```

- La vista v\$backup nos da información sobre el estado de los backups.

- 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;
```

- Consulta la base de datos y el catálogo para crear un *fichero binario* con los objetos seleccionados, de extensión .dmp
- Puede hacerse de la base de datos completa, un usuario concreto (o varios) o una tabla concreta (o varias).
- Permite almacenar 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**.

- Permite leer un fichero .dmp recuperando su contenido sobre una base de datos y especificando el usuario que creará los objetos recuperados.

- Se elige siempre un método primario, y otro en caso de que este falle.
- La copia en frío sólo 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.