Seguridad en BD

Diseño de Bases de Datos
Grado en Ingeniería Informática

Departamento de
Lenguajes y Sistemas Informáticos

Universitat d'Alacant
Universidad de Alicante

Seguridad en BD

- Contexto
- Objetivos
- ¿Por qué protegernos?
- ¿De quién protegernos?
- ¿Cómo protegernos?
- ¿Algo más? Sí auditar.

Etapas en el diseño físico de una BD

1. Traducir el esquema lógico para el SGBD específico.
2. Diseñar la representación física.
3. **Diseñar los mecanismos de seguridad.**
4. Monitorizar y afinar el sistema.

Objetivos

- **Confidencialidad: sólo los usuarios autorizados pueden tener información a la información que les corresponde y con los permisos que les corresponde.**
- **Integridad: asegurar que lo que los usuarios tratan de hacer es correcto y evitar la pérdida accidental de la consistencia**
- **Disponibilidad: asegurarse que la información estará disponible cuando sea necesaria**

¿por qué protegernos?

- **Importancia estratégica de la información:** La información es un bien muy valioso. El 40% de las compañías que pierden completamente su sistema informático desaparecen.
- **Evitar robos y/o sabotajes.**
- **Un robo o pérdida tiene un coste de imagen elevado.**
- **Y además … LEY DE PROTECCIÓN DE DATOS.**

¿de quién protegernos?

- **De agentes externos**
  - Control de acceso a la BD
- **De agentes internos: usuarios autorizados. Se debe controlar:**
  - qué usuarios tienen acceso a qué datos
  - qué operaciones pueden realizar sobre dichos datos
  - que no se permitan operaciones que vulneren la integridad de los datos
- **De catástrofes y fallos**
  - Política de copias de seguridad

¿cómo protegernos?

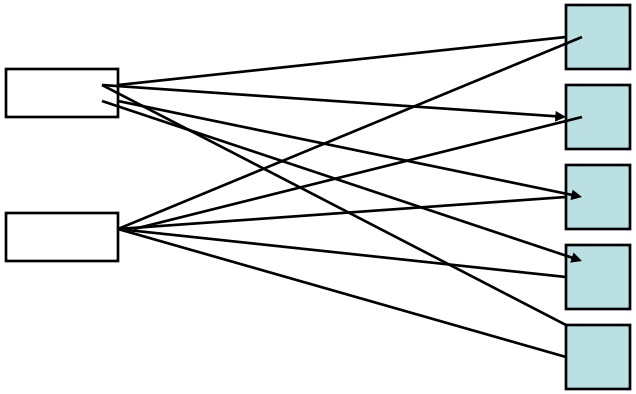
- **De agentes externos**
  - **A nivel de red**
    - Sólo acceden al servidor los ordenadores autorizados
  - **A nivel de SO**
    - Sólo los usuarios con privilegios acceden a los ficheros de la BD
  - **A nivel de BD**
    - Política de contraseñas

¿cómo protegernos?

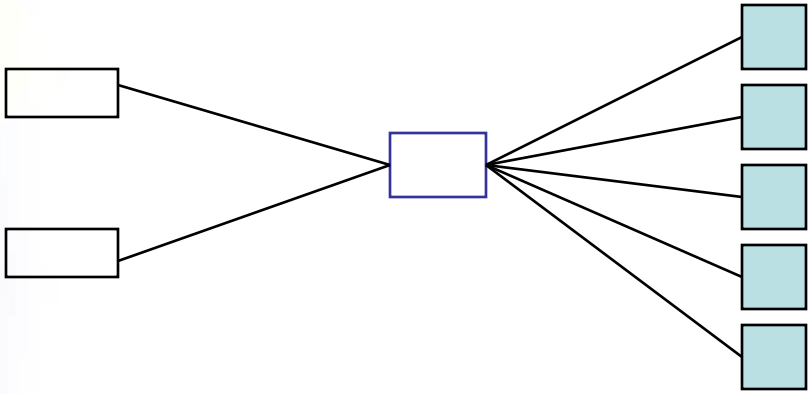
De usuarios conocidos: Controlar acceso a datos

- **Seguridad por niveles**
  - Asigna a cada usuario y cada objeto (tabla, fila, columna, vista…) se le asigna un nivel
  - Control de acceso:
    - un sujeto S puede **leer** el objeto O si nivel(S) ≥ nivel(O)
    - un sujeto S puede **escribir** el objeto O si nivel(S) = nivel(O)
- **Seguridad por privilegios**
  - *Privilegio:* derecho a ejecutar un tipo de SQL o de acceder a objetos de otros usuarios
  - Tipos de privilegios
    - » de sistema: permite ejecutar cierto tipo de acciones como crear tablas, índices, usuarios, procedimientos …
    - » de objetos: permite ejecutar ciertas acciones sobre objetos específicos (hacer un insert en una tabla, etc.) Se permite poner privilegios de objetos incluso a nivel de campos (un usuario puede hacer un update sobre unos campos pero no sobre otros).
  - La mayoría de las BD permiten la gestión de privilegios por ROLES para facilitar la administración
  - *Roles:* Agrupaciones de privilegios
  - A un usuario se le puede otorgar un privilegio concreto o todos los incluidos en un role (mejora administración)

¿cómo protegernos?



Sin ROLES:
Usuarios con mismos privilegios hay que definir los permisos para cada uno



Con roles:
Usuarios con mismos privilegios tienen mismo role

¿cómo protegernos?

De usuarios conocidos: para asegurar integridad en datos

- **Constraints :** ***son propiedades de la base de datos que se deben satisfacer en cualquier momento. Si la constraint está activa es porque se cumple la restricción que define.***
  - ***Tratamiento de valores nulos.***
  - ***Valores por defecto.***
  - ***Integridad de clave primaria.***
  - ***Claves alternativas.***
  - ***Integridad referencial.***
  - ***Restricciones de integridad estáticas (check)***

- **Disparadores: NO GARANTIZAN la integridad ya que sólo actúan cuando están activos y al activarse NO garantizan que cumpla aquello que controlan.**
- **Control de transacciones, principio ACID**

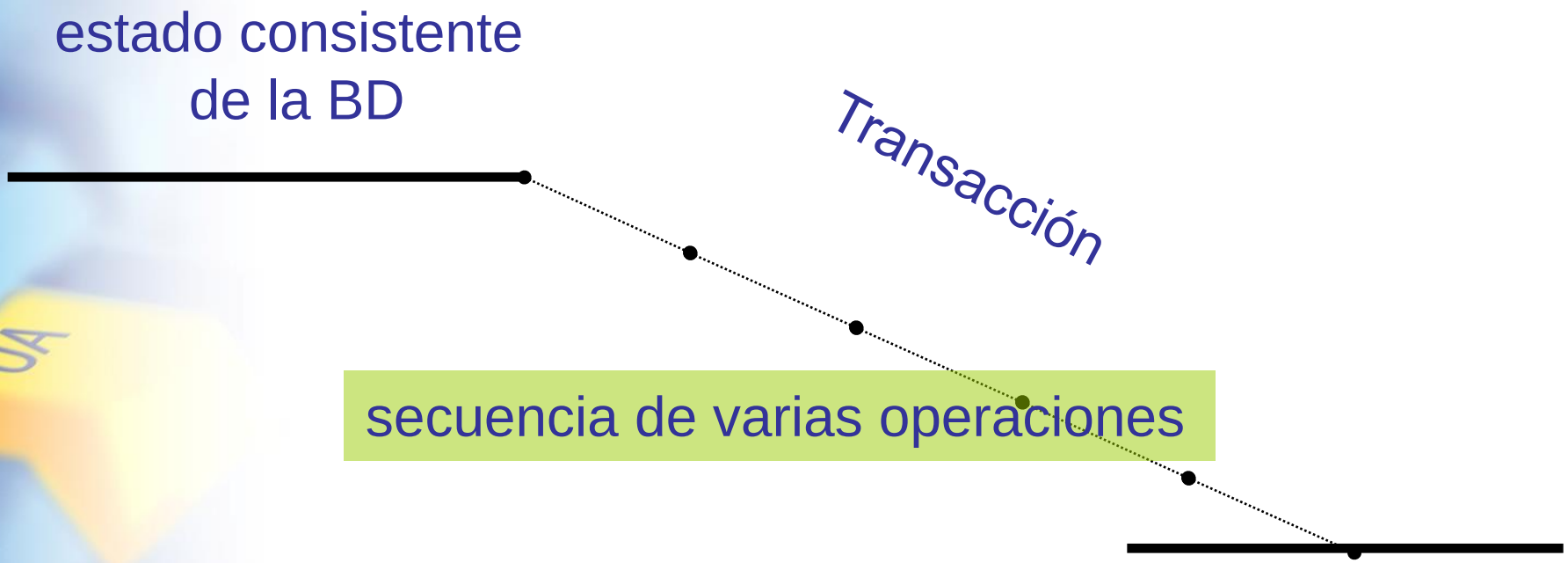
¿cómo protegernos?

- **De usuarios conocidos: para asegurar integridad en datos**

  - Control de transacciones → Garantizar el principio ACID
    - **ATOMICIDAD: todas las operaciones que la componen son tomadas como u todo. O todas las acciones de la transacción se realizan de forma válida o ninguna de ellas se considera correcta**
    - **CONSISTENCIA: cualquier transacción llevará a la base de datos desde un estado válido a otro también válido, La integridad debe quedar garantizada**
    - **DURABILIDAD: una vez que se ha validado la transacción, no se debe perder.**
    - **AISLAMIENTO: una transacción en ejecución no puede revelar sus resultados a otras transacciones concurrentes antes de finalizar**

¿cómo protegernos? Control de transacciones

Una TRANSACCIÓN es una unidad lógica de trabajo



Sin conservar, por fuerza, la consistencia en los puntos intermedios

¿cómo protegernos? Control de transacciones

Ejemplo (suponiendo una tabla PIEZA con una columna para los totales suministrados)

SIN CONTROL DE TRANSACCIONES

INSERTAR
en SUMINISTRAR
(codprov, codpie, cantidad)
valores (“S5”, “P1”, 1000)

. . . otras instrucciones

ACTUALIZAR
PIEZA canttotal=canttotal+1000
donde código=“P1”

Si en este momento se produce algún tipo de error y se detienen la ejecución, la inserción se ha realizado y por tanto hay inconsistencia: Se viola que canttotal sea la suma de todos los suministros (CONSISTENCIA)

CON CONTROL DE TRANSACCIONES

INICIO TRANSACCION

INSERTAR
en SUMINISTRAR
(codprov, codpie, cantidad)
valores (“S5”, “P1”, 1000)

. . . otras instrucciones

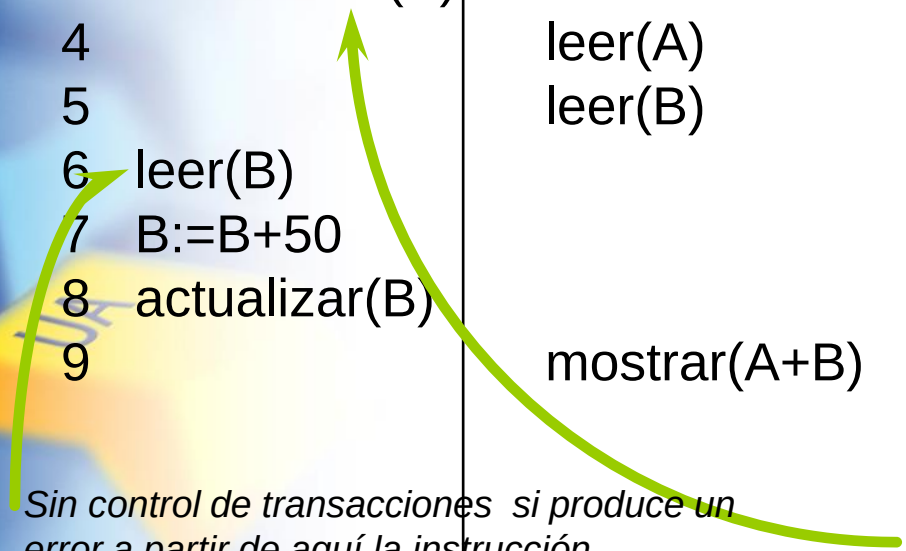
ACTUALIZAR
PIEZA canttotal=canttotal+1000
donde código=“P1”

COMMIT

Si en este momento se produce algún tipo de error y se detienen la ejecución, la inserción se cancela (ROLLBACK)y por tanto se mantiene *la consistencia*

¿cómo protegernos? Control de concurrencia

	T0	T1
1	leer(A)	
2	A:=A-50	
3	actualizar(A)	
4		leer(A)
5		leer(B)
6	leer(B)	
7	B:=B+50	
8	actualizar(B)	
9		mostrar(A+B)



Situación en un banco:
- Por un lado una transacción T0
transfiere 50 euros de una cuenta bancaria A a otra B de un cliente y
- por otro lado otra transacción T1
consulta los saldos de las cuentas y muestra el saldo total del cliente.

Si el cliente tiene 200 euros en A y 300 en B y las dos transacciones concurren como se muestra ¿Qué podría ocurrir sin control de transacciones?

NECESARIO CONTROL DE CONCURRENCIA

Sin control de transacciones si produce un error a partir de aquí la instrucción escribir(A) esta validada por lo que !!!FALTARIAN 50 EN SALDO de B !!!!! Con control de transacciones, si hay un error se anulan automáticamente TODAS las instrucciones de la transacción (ATOMICIDAD)

Sin control de transacciones en este momento la cuenta A ya muestra el saldo actualizado, con control de Transacciones (AISLAMIENTO) el saldo de A no se verá modificado por otras transacciones hasta que finalice.

¿cómo protegernos? Control de concurrencia

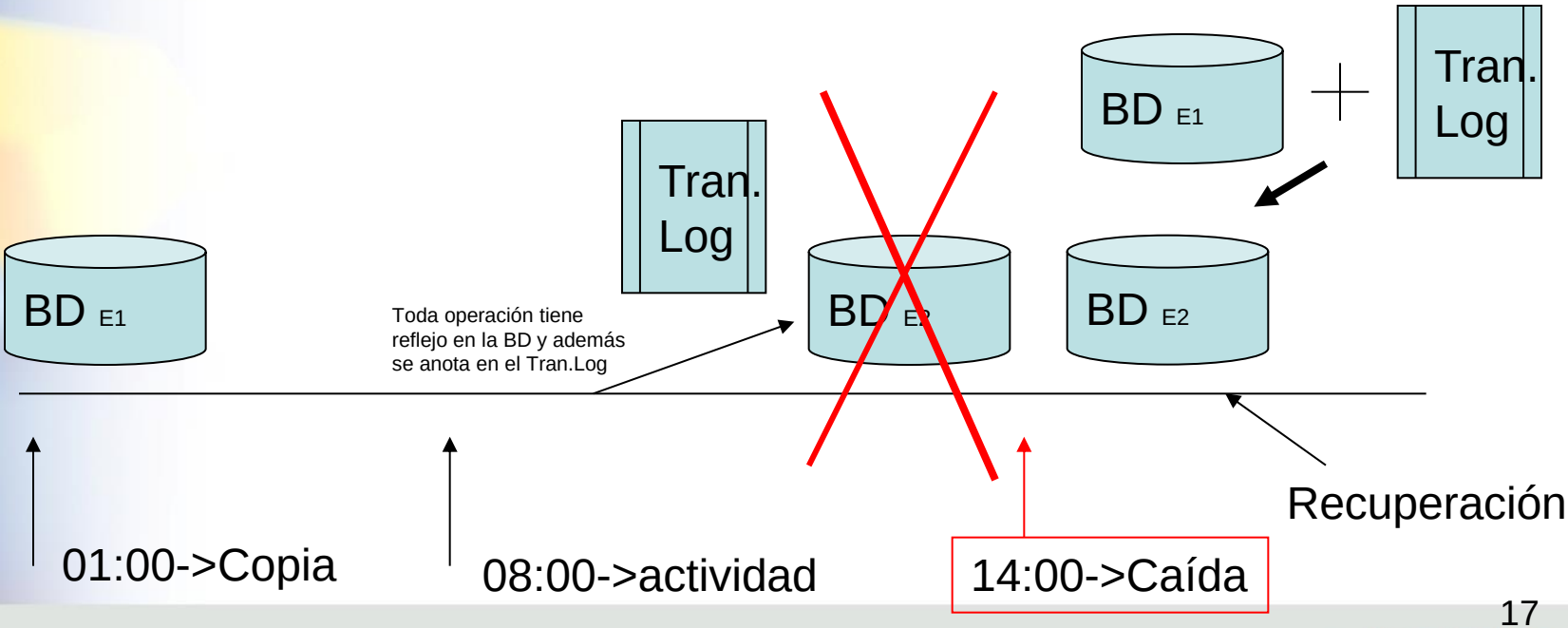
- El control ACID lo hace el propio SGBD
- Hay que llevar especial cuidado en lenguajes como php, .net, jsp., etc, ya que por defecto llevan “commit implícito (después de cada instrucción se hace un commit)”, es decir NO hay control de transacciones a no ser que el programador lo especifique. El programador solo debe definir el inicio y el final de una transacción
- Cualquier error o fallo en alguna de las instrucciones produce un rollback automático

¿cómo protegernos?

- **Catástrofes y fallos.**
  - **Se debe garantizar la Recuperación ante fallos de los usuarios (por ejemplo borrado accidental de información) y ante fallos del sistema (por ejemplo rotura de un disco). Pare ello se dispone de los mecanismos de copias de seguridad**
  - **Se debe de garantizar la recuperación ante grandes catástrofes, para ello se dispone de los centros de respaldo.**

¿cómo protegernos?

– De errores y fallos: Copias de seguridad
  – Las copias de seguridad permiten recuperar la base de datos, pero sólo la información que había en la misma hasta el momento de hacer la copia
  – Los SGBD emplean LOG DE TRANSACCIONES (en ORACLE modo ARCHIVELOG) para garantizar la recuperación total



¿cómo protegernos?

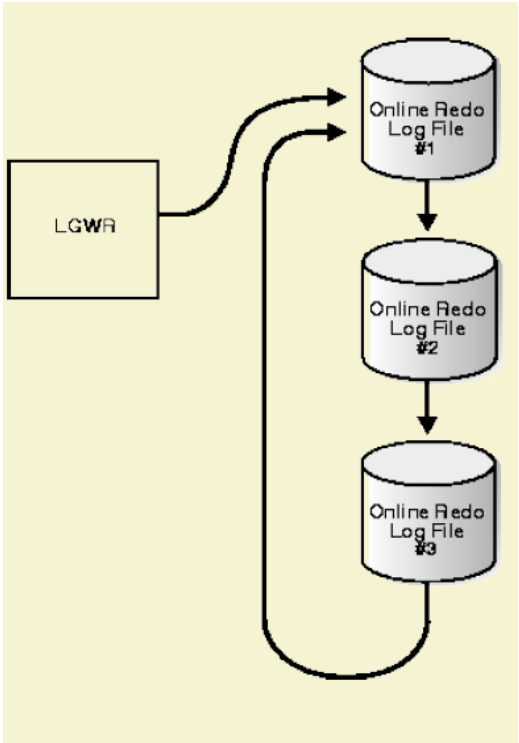
- **Caso de ORACLE: uso de los REDO LOG en MODO ARCHIVER**
  - **Los ficheros redo log guardan todos los cambios hechos en los datos y permiten volver a aplicarlos en caso de caída de la BD.**
  - **Los ficheros redo log se organizan en grupos. Una BD requiere al menos dos grupos. Cada fichero redo log dentro de un grupo se llama miembro. Lo usual es tener 3 grupos de redo con 2 miembros cada uno.**
  - **Los ficheros redo log se usan de manera circular: cuando uno se llena, el LGWR comienza a escribir en el siguiente grupo ("log switch"), hasta volver al primero (machacando la información)**

```
create database PRUEBA
datafile '/oracle/u06/oradata/ORAPROD/Tablespaces/system01.dbf'     SIZE
2000M AUTOEXTEND ON NEXT 50M MAXSIZE 5000M
    sysaux   datafile
'/oracle/u07/oradata/ORAPROD/Tablespaces/system_aux.dbf'  SIZE 1500M
AUTOEXTEND ON NEXT 25M MAXSIZE 3000M
   logfile
      GROUP 1
('/oracle/u03/oradata/ORAPROD/RedoLogs/Gr1/redoORAPROD01.log',

'/oracle/u02/oradata/ORAPROD/RedoLogs/Gr2/redoORAPROD01.log' )
      size 10M,
      GROUP 2
('/oracle/u03/oradata/ORAPROD/RedoLogs/Gr1/redoORAPROD02.log',

'/oracle/u02/oradata/ORAPROD/RedoLogs/Gr2/redoORAPROD02.log' )
      size 10M,
      GROUP 3
('/oracle/u03/oradata/ORAPROD/RedoLogs/Gr1/redoORAPROD03.log',

'/oracle/u02/oradata/ORAPROD/RedoLogs/Gr2/redoORAPROD03.log' )
      size 10M;
```



¿cómo protegernos?

- En ORACLE por defecto la BD se crea en modo NOARCHIVELOG (con CREATE DATABASE).
- Si activamos el modo ARCHIVELOG se irán archivando los ficheros redo log conforme se llenan (cada vez que ocurre un “log switch”).
- Si disponemos de ARCHIVE LOG, en caso de caída, podemos recuperar la BD hasta el instante mismo de la caída aplicando los REDO LOGS archivados a la última copia de seguridad *(realizada con RMAN)*

¿Algo más? Sí auditar.

- **Auditar -> registrar de forma automática los :**
  - **Accesos a la BD**
  - **Accesos y operaciones sobre los objetos de la BD.**
- **El tener procesos de auditoría en marcha pueden afectar al rendimiento de la BD**
- **¿Cómo auditar? Depende del SGBD, en ORACLE:**
  - **Con disparadores**
  - **Con utilidad AUDIT**

- Con disparadores (ejemplos)

Control de acceso a la BD:.

```
CREATE OR REPLACE TRIGGER SYSTEM."CONTROLCONEXION"
AFTER LOGON ON DATABASE

BEGIN
   insert into LOG_ACCESOS (usuario, IP, fecha) values (USER, SYS_CONTEXT ('USERENV', 'IP_ADDRESS'), sysdate);

END;
/
```

Modificación de datos:

```
CREATE OR REPLACE TRIGGER audit_modif
BEFORE UPDATE ON EMPLE
FOR EACH ROW
DECLARE
v_cad_inser auditaremple.col1%TYPE;
BEGIN
v_cad_inser := USER || '->'|| TO_CHAR(sysdate,'DD/MM/YY*HH24:MI*') ||:OLD.EMP_NO ||'* MODIFICACION *';
IF UPDATING ('APELLIDO') THEN
v_cad_inser := v_cad_inser
||:OLD.APELLIDO|| '*'||:NEW.APELLIDO;
END IF;
IF UPDATING ('SALARIO') THEN
v_cad_inser := v_cad_inser
||:OLD.SALARIO|| '*'||:NEW.SALARIO;
END IF;

INSERT INTO AUDITAREMPLE VALUES(v_cad_inser);
END;
```

¿Algo más? Sí auditar.

- **Utilidad AUDIT de ORACLE .**
  - **Hay que configurarla**
  - **Tiene muchas opciones y configuraciones posibles**
  - **La información de la auditoría puede quedar en tablas propias de la BD (SYS.AUD$, o *SYS.FGA_LOG$ en caso de ser de grano fino)* o en ficheros del S.O.**

¿Algo más? Sí auditar.

- **Varios niveles de auditoría**
  - **De sentencias. Seleccionando un tipo concreto de las mismas, que afectan a una determinada clase de objetos de base de datos.**

    ejemplo: ***AUDIT SELECT TABLE BY PEPE, JUAN BY ACCESS ;***

    genera un registro de log cada vez que los usuarios PEPE o JUAN hagan un select sobre una tabla o vista .
  - **De privilegios. Auditoría de privilegios de sistema**

    ejemplo, ***AUDIT CREATE TABLE BY ACCESS genera un registro de log cada vez que se otorge un privilegio de create table*****.**

¿Algo más? Sí auditar.

- **Varios niveles de auditoría**
  - **De esquema. Sentencias específicas sobre objetos de un esquema concreto (p. ej. audit insert,update on pepe.tabla1 by access o por ejemplo audit select, delete, insert,update on pepe.tabla1 by access). Afectan a TODOS los usuarios de la BD**
  - **De grano fino ("fine grained"). Acceso a datos concretos y cambios en los mismos a nivel columna. 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*).**

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;
```

Seguridad en BD

Diseño de Bases de Datos
Grado en Ingeniería Informática

Departamento de
Lenguajes y Sistemas Informáticos

Universitat d'Alacant
Universidad de Alicante