Master Universitario en Inteligencia Artificial Aplicada

Bases de Datos e Infraestructuras

uc3m | Universidad Carlos III de Madrid

Estructura del Curso

0.- Introducción

1.- Diseño de BB.DD. Relacionales (estática)

2.- Operación de BB.DD. Relacionales (dinámica básica)

3.- SQL para consultas analíticas

4.- Otros elementos de las BBDD Relacionales

Tema 4: Ampliando el SQL

Estructura del Tema

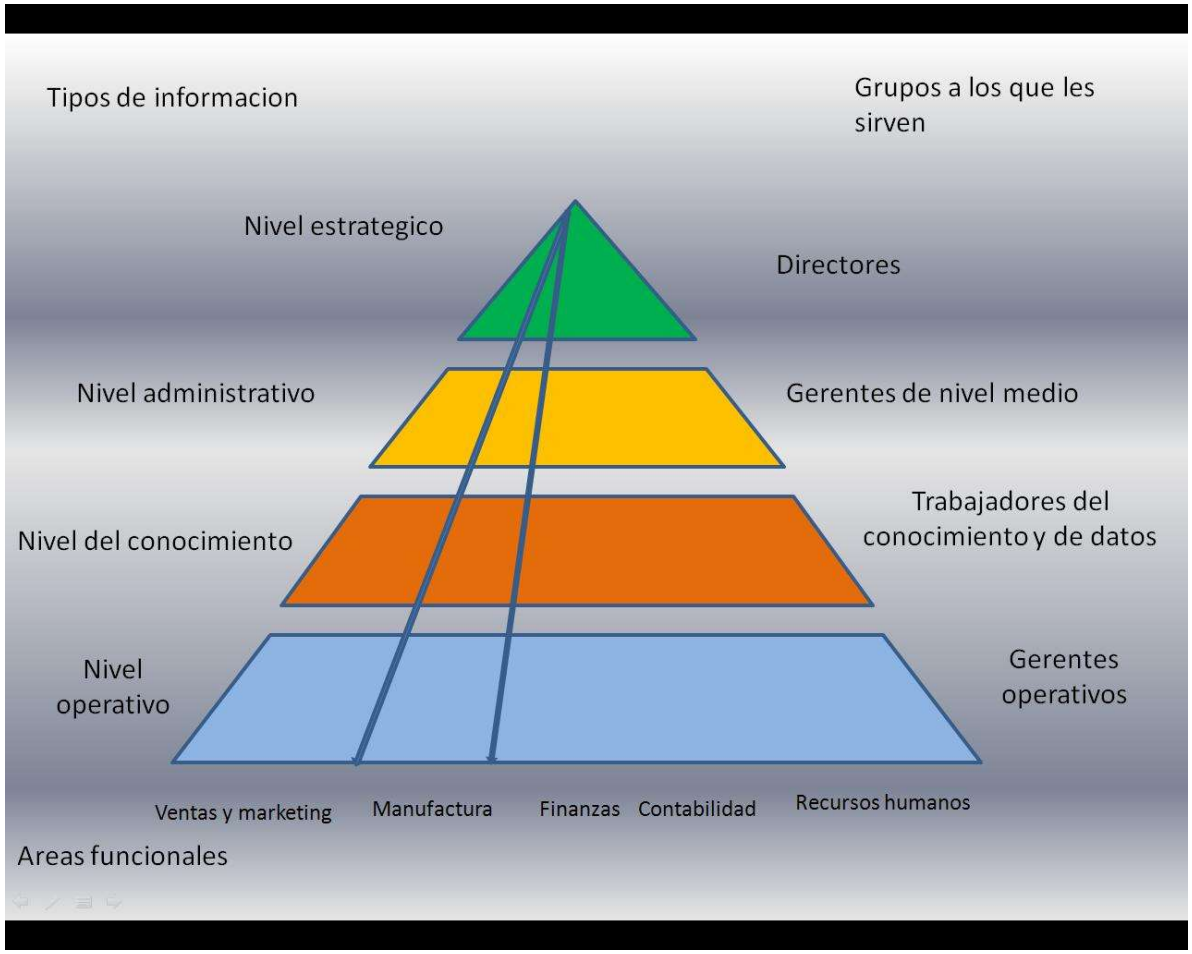
1.- Paradigmas de Almacenamiento: transaccional vs analítico

2.- Introducción a elementos **SQL** "avanzados"

- usuarios y esquemas de relación
- concepto de *vista*
- *bloque* y procedimiento
- concepto de *disparador* y de *job*

Tema 4.1: Paradigmas Almacenamiento

- Distintas Necesidades según sea el **USO** (niveles de la organización)



OLTP

- **Bases de Datos** Estructuradas: orientación operacional, independencia dato-uso…
  ACID: **A**tomicidad
  Consistencia
  a**I**slamiento
  Durabilidad

OLAP

- **Data Analytics:** Síntesis de información, volúmenes masivos, soporta toma de decisiones, Data Ware House & …

Tema 4.1: ¡¡DEMASIADOS DATOS!!

- Si ofrezco demasiados productos en un supermercado, ¿Cómo facilitar al consumidor que obtenga lo que necesita?

  → Organización por "secciones"

Tema 4.1: MERCADOS de DATOS

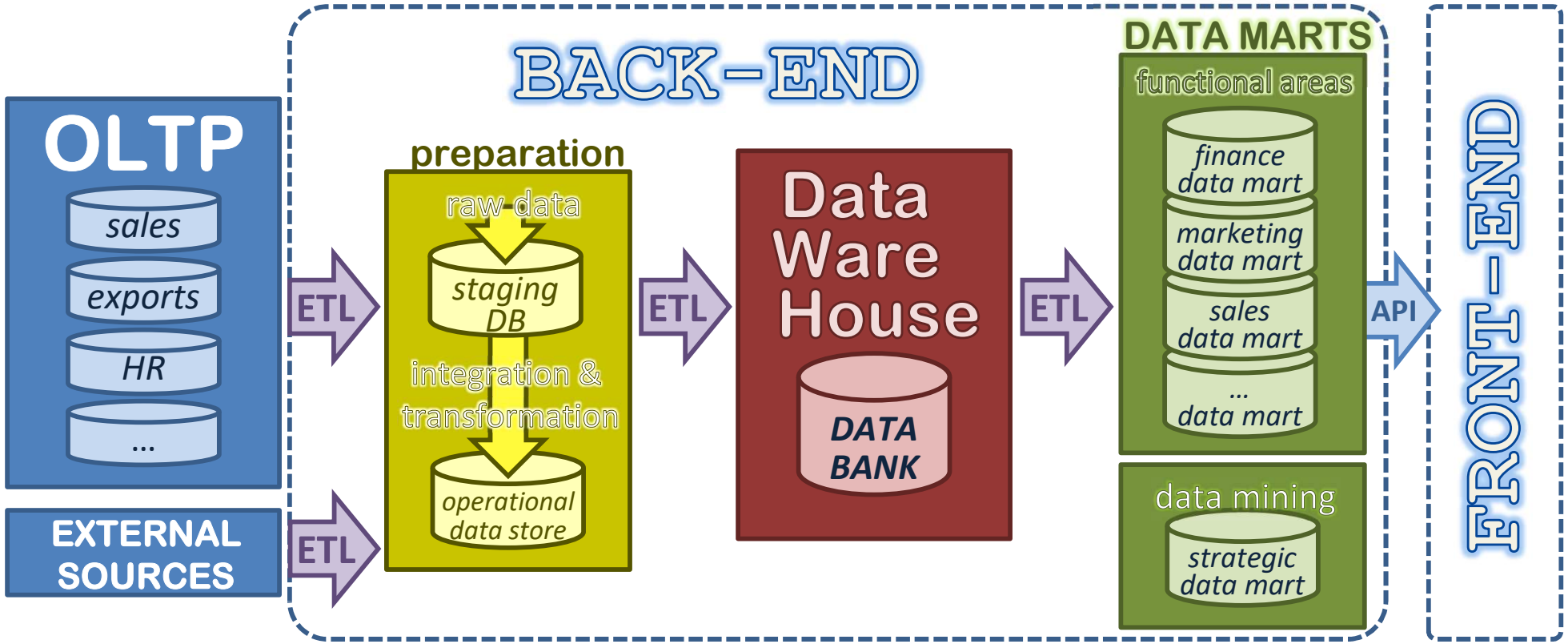
- Subsistemas especializados en un área de la compañía/organización

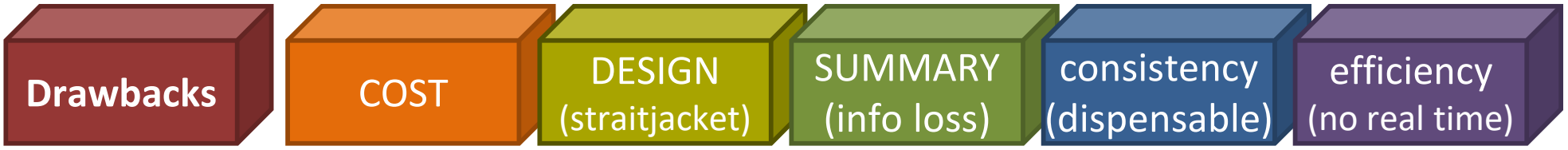
. . .

Tema 4.1: Acopio de Datos

- ¿De dónde salen los datos? De TODAS partes (hasta de la web…)
- Fuentes propias y ajenas → hay que reciclar, limpiar, integrar…

Tema 4.1: DataWareHouse

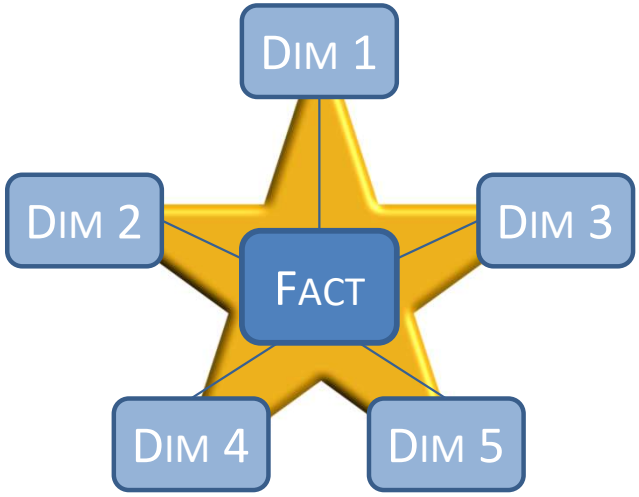




Tema 4.1: Enfoque ROLAP

- **ROLAP**: escalabilidad, coste, mantenimiento (no tanta eficiencia)
  - Esquema en **estrella**: una tabla central (facts) relacionada con varias tablas satélite (normalizadas) acorde con un diseño previo de dimensiones relevantes
  - Por ejemplo, si el hecho es 'ventas' las dimensiones podrían ser clientes, productos, tiendas, horas, …

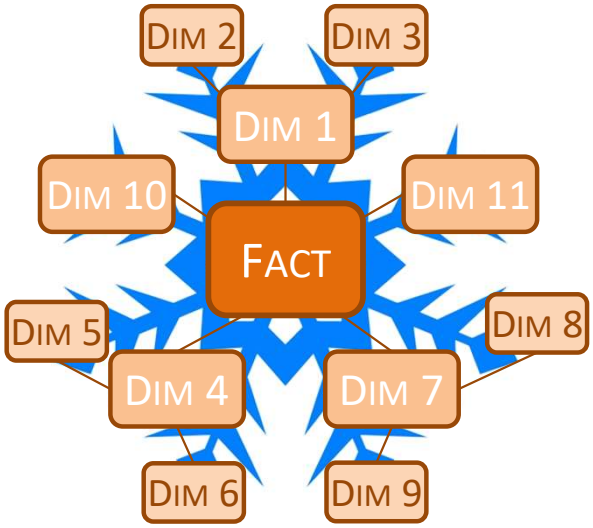
→ *ventajas: diseño simple, baja redundancia, coste mantenimiento.*



Tema 4.1: Enfoque ROLAP

- **ROLAP**: escalabilidad, coste, mantenimiento (no tanta eficiencia)
  - Esquema en **copo de nieve**: una o más tablas centrales (hechos) relacionadas con tablas satélite que a su vez son desarrolladas en otras tantas tablas satélite según se diseñan las dimensiones, subdimensiones…
  - *Las tiendas tienen empleados, vehículos… estos tienen potencia, capacidad…*

→ *ventajas: simplicidad de consulta (query), agilidad.*



Tema 4.2.1: Instancias, Usuarios y Esquemas

En **Oracle DB**, un servidor puede albergar varias instancias:

- **Instancia de BD**: conjunto de servicios, procesos, y estructuras en memoria (principal y secundaria), que implementan un Sistema de BD Completo
  - Listener: servicio que permite a un cliente conectarse con una instancia (sesión)
  - Sesión: intervalo desde la autenticación de usuario hasta la desconexión; conjunto de recursos dedicados a ese usuario durante ese intervalo

- **Esquema de BD**: no existe este concepto propiamente dicho en Oracle, todos los objetos pertenecen a la instancia. Sin embargo, la nomenclatura de objetos y la gestión de privilegios se organiza por usuarios, y por esto se suele decir que cada usuario define un esquema relacional (subconjunto de objetos de ese usuario)

- **Privilegios**: capacidad de realizar una operación sobre un objeto (crear tablas en mi espacio, consultar mis tablas, consultar otras tablas, ...). Se distinguen privilegios de usuario y privilegios de sistema (reservados para usuarios administradores).

Tema 4.2.1: Usuarios y Roles (LCD)

- **Usuario**: objeto (configurable) que permite la conexión con ciertos privilegios
- Perfil: configuración de usuario (restricciones sobre recursos: espacio, tiempo,...)
- Rol: conjunto de privilegios

```
CREATE USER <username> IDENTIFIED BY <password>;
```

- opciones:

```
[DEFAULT TABLESPACE <tablespace>]
[QUOTA <size> ON <tablespace>]
[PROFILE <profilename>]
[PASSWORD EXPIRE]
[ACCOUNT {LOCK|UNLOCK}]
```

```
CREATE PROFILE <profilename> LIMIT <resources>;

CREATE ROLE <rolename>;
```

Tema 4.2.1: Control de Privilegios (LCD)

- **Concesión**: de un privilegio (o cjto. privs.) a un usuario (o cjto. privs.)

- privilegios de sistema: {CREATE|ALTER|DROP} { session | table | role | user | …}

```
GRANT { <rolename> | <sys_privileges> | ALL PRIVILEGES }
      TO <users/roles> [WITH ADMIN OPTION];
```

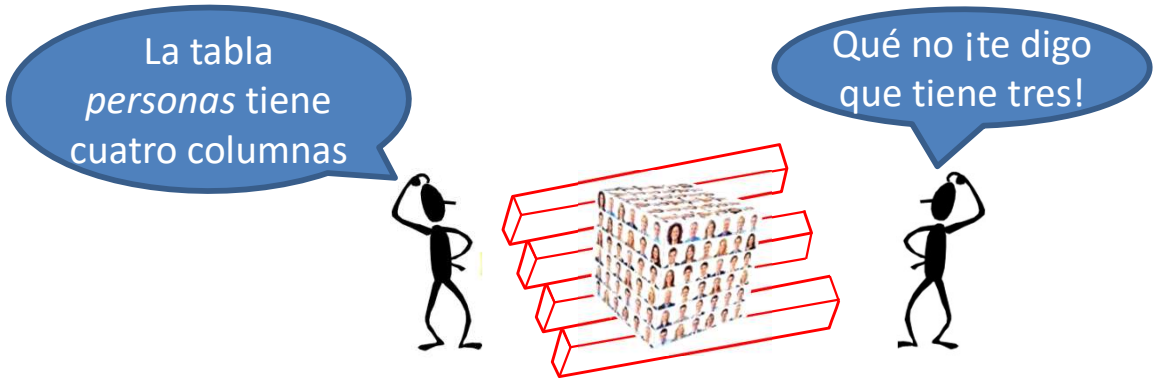
- privilegios de objeto: {INSERT|DELETE|UPDATE|SELECT} { table | view | …}

```
GRANT { <object_privileges> | ALL PRIVILEGES }
     ON [schema.]<object> TO <users/roles>
     [WITH HIERARCHY OPTION] [WITH GRANT OPTION];
```

- **Revocación**: de un privilegio (o cjto. privs.) de un usuario (o cjto. privs.)

```
REVOKE <privileges> [ON <object>] FROM <users/roles>;
```

Tema 4.2.2: Concepto de Vista



- Un solo objeto (tabla) admite varias definiciones para distintos tipos de usuario
- La *relación base* será la conjunción de todas ellas (esq. lógico global)
- Se definirá aparte la versión específica de cada tipo de usuario
- Existirán por tanto diversos esquemas (externos) y uno sólo global.
- La vista puede reducir el grado y/o la cardinalidad de la relación base, e incluso fusionar varias relaciones en una sola → **¡es como una consulta!**

Tema 4.2.2: Creación de Vistas

```
CREATE [MATERIALIZED] VIEW <nombre de tabla>
       [(<nombre_columna> [,<nombre_columna>]…)
       AS (SELECT ... FROM ...) [WITH CHECK OPTION]
```

- La *vista materializada* tiene sus propios datos (redundancia física).
  La *vista lógica* es virtual (se computa cuando el usuario accede a ella).
- Si no se definen nombres de columna, toma las etiquetas de la consulta.
- Las vistas son operables: si inserto/borro/modifico una vista, la operación debería ejecutarse sobre la tabla base.
  - al insertar, las columnas no presentes adquirirán el valor por defecto.
  - 'check option' comprueba que la nueva fila pertenecerá a la vista

Tema 4.2.2: Ejemplos de uso de Vistas

- **Diseño Externo**: existen dos tipos de usuario: *oficina* y *RRHH*…

```
CREATE ROLE oficina;
CREATE ROLE rrhh;
CREATE TABLE empleados_ALL
       (DNI NUMBER(8) PRIMARY KEY,
        nombre VARCHAR2(25) NOT NULL,
        tlf NUMBER(9) UNIQUE,
        salario NUMBER (8,2), …);
...
CREATE VIEW empleados(nombre, telefono) AS
        SELECT nombre, tlf FROM empleados_ALL;
GRANT ALL PRIVILEGES ON empleados TO oficina;
...
CREATE MATERIALIZED VIEW asalariados AS
        SELECT nombre, DNI, salario FROM empleados_ALL;
GRANT ALL PRIVILEGES ON asalariados TO rrhh;
...
```

Tema 4.2.2: Ejemplos de uso de Vistas

- **Control de Versiones:** (vigente / borrador / histórico)

```
CREATE TABLE documentos ( … ,
      fecha_ini DATE,
      fecha_fin DATE,      …);

CREATE VIEW borrador AS SELECT * FROM documentos
      WHERE fecha_ini IS NULL OR fecha_ini>sysdate;

CREATE VIEW historico AS SELECT * FROM documentos
      WHERE fecha_fin<sysdate;

CREATE VIEW vigente AS SELECT * FROM documentos
      WHERE (fecha_fin IS NULL OR fecha_fin>sysdate)
            AND fecha_ini<=sysdate ;
```

Tema 4.2.2: Ejemplos de uso de Vistas

- **Fusión** de relaciones y atributos **derivados**

```
CREATE TABLE refs(ref NUMBER(8) PRIMARY KEY,
                  nombre VARCHAR2(25) NOT NULL,
                  tipo VARCHAR2(5),
                  coste NUMBER (8,2) NOT NULL, …);

CREATE TABLE vats(tipo VARCHAR2(5) PRIMARY KEY,
                  iva NUMBER(2,2) NOT NULL, …);

CREATE VIEW productos(nombre, ref, precio) AS
       SELECT a.nombre, a.ref, a.coste*(1+b.iva)
              FROM refs a NATURAL JOIN vats b;
```

Tema 4.2.3: SQL procedimental, el Bloque

- El *bloque* es un *programa* desarrollado para procesar datos
- Puede ser anónimo o nominado (procedimientos, funciones, ...)
- ***Estructura:*** tiene tres partes: declaración de variables, cuerpo del bloque, y gestión de excepciones

```
[DECLARE
      varname type; [...] ]
BEGIN
      <código procedimental>
[EXCEPTION
      WHEN … THEN …; […] ]
END;
```

Tema 4.2.3: Bloques nominados

- En Oracle DB, los bloques nominados (*function*, *procedure*, …) son objetos que se almacenan en la BD.
- Al definirlos, se omite la keyword DECLARE
- Los procedimientos sólo se pueden invocar dentro de otro bloque
- Las funciones también se pueden invocar en una SELECT

```
CREATE OR REPLACE PROCEDURE name(params) IS
  <declaraciones>
BEGIN
  <código>
END;

CREATE OR REPLACE FUNCTION name(params) RETURN CHAR IS
  <declaraciones>
BEGIN
    <código>
    RETURN <valor>;
END;
```

Tema 4.2.3: SQL procedimental, el Bloque

- En la parte declarativa, se pueden incluir variables, tipos de datos, cursores (consultas), excepciones, etc.
- Si el control del flujo de programa llega al control de excepciones, este no puede regresar; para conseguirlo, se puede hacer un bloque anidado:

```
... Begin
    ... Begin ... Exception ... End;
... End;
```

- el código procedimental admite
  - Asignaciones: *variable*:=valor;
  - Consultas: select *columna* into *variable* from...;
  - Bucles: ... for *var* in (1..10) loop ... end loop;
  - Cursores sobre consulta: ...for *fila* in (select ...) loop ... end loop;
  - Secuencia condicional: ... if (*condición*) then ... else... end if;
  - Llamadas a bloques almacenados (procedimientos/funciones)
  - Instrucciones LMD actualización (insert/delete/update);
  - etc.

Tema 4.2.3: PL/SQL procedimental: Paquetes

- Es una colección de variables y procedimientos almacenados
- Su creación requiere dos pasos: descripción e implementación (cuerpo)

- Descripción del paquete: enumeración de elementos públicos

```
CREATE OR REPLACE PACKAGE my_package AS
   <declaración variables, cabeceras proc., etc>
END my_package;
```

- Cuerpo del paquete: Implementación completa de cada elemento

```
CREATE OR REPLACE PACKAGE BODY my_package AS
   <implementación completa de cada elemento>
END my_package;
```

Tema 4.2.4: Concepto de Disparador

- Es un bloque (o procedimiento) que se ejecuta cuando le ocurre algo a un objeto.
- Los eventos que captura son operaciones de actualización (insert/delete/update)
- El tiempo del 'disparo' es elegible: antes o después (de realizar la actualización).
- La *granularidad* define si el disparo se realiza una vez para toda la instrucción (opción por defecto en Oracle) o una vez para cada fila operada (for each row)
- La acción se define con un bloque o una llamada a procedimiento: *call proc(...)*

```
CREATE OR REPLACE TRIGGER [<nombre>]
[before|after] {INSERT|DELETE|UPDATE} ON tabla
[for each row]
<bloque definiendo la acción disparada>
```

Tema 4.2.4: Trigger, usos y extensiones

- Si es de temporalidad 'before' y causa un error, ya sea natural o forzado por la instrucción raise_application_error(...), la actualización no llega a producirse.
  → disparador de rechazo: permitirá implementar las aserciones
- Si es de inserción/modificación, temporalidad 'before', y granularidad 'cada fila' el disparador puede sustituir los valores de la nueva fila por otros que se calculen.
- Una vez ejecutada la actualización, se puede registrar la descripción de la operación, la identidad del usuario, y el momento en que la hizo (auditoría).
- Se pueden programar "Reglas de integridad no nativas" (DSD / UC / USN / ...)
- Algunos gestores soportan "disparadores DDL y DCL" para acciones como *drop*, *grant*, etc.; y "disparadores DB" para eventos *logon*, *logoff*, *startup*, *shutdown*, ...
- Algunos gestores soportan objetos que implementan disparadores con eventos relacionados con el tiempo (*dbms_jobs*, en Oracle DB) → activación periódica