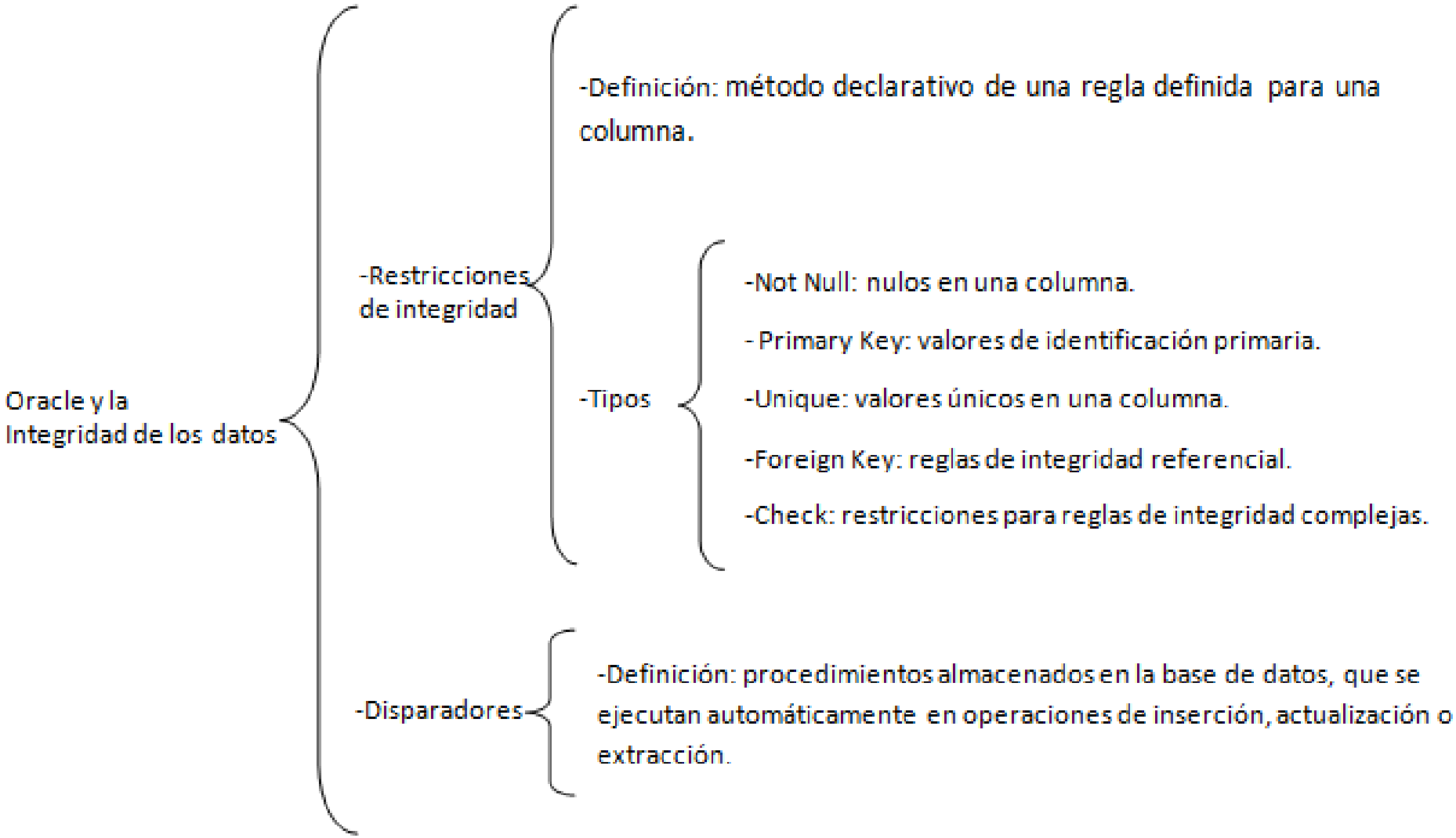
Tema 10

Integridad.

Disparadores.

Oracle y la integridad de los datos



Restricciones de integridad sobre los datos

- Regla de NO nulos:
  - Prohíbe inserciones o actualizaciones de registros con valor nulo en una columna determinada.

- Valores únicos:
  - Permite inserciones o actualizaciones de un registro, sólo si contiene un valor único en esa columna.

- Valores de clave primaria:
  - Una regla de clave primaria definida en una columna especifica que cada registro de la tabla puede ser identificado por el valor contenido en dicha columna y que no puede tener el valor nulo.

Restricciones de integridad sobre los datos

- Reglas de integridad referencial:
  - Reglas definidas en una clave de una tabla que garantiza que los valores de esa clave concuerdan con los valores referenciados.

- Restricciones de validación de reglas de integridad complejas:
  - Es una regla definida por el usuario que permite comprobar, en una fila, la validez de la condición expresada.

Ventajas del sistema de restricciones de integridad

- Creación de manera declarativa.
- Restricciones centralizadas.
- Máxima productividad en el desarrollo de aplicaciones.
- Retroalimentación inmediata al usuario.
- Mayor rendimiento.
- Flexibilidad en la identificación de las violaciones de integridad durante la carga de datos.

Repercusión del sistema de restricciones de integridad en el rendimiento

- Las ventajas del sistema de las reglas de integridad vienen acompañadas de algunas pérdidas de rendimiento.

- En general, este coste es como mucho el mismo que el de ejecutar una sentencia SQL que evalúa la restricción.

No nulidad (NOT NULL)

- Por defecto, todas las columnas de una tabla permiten nulos.
- Obliga a que una columna de una tabla no contenga valores nulos.
- Nulo significa la ausencia de valor.

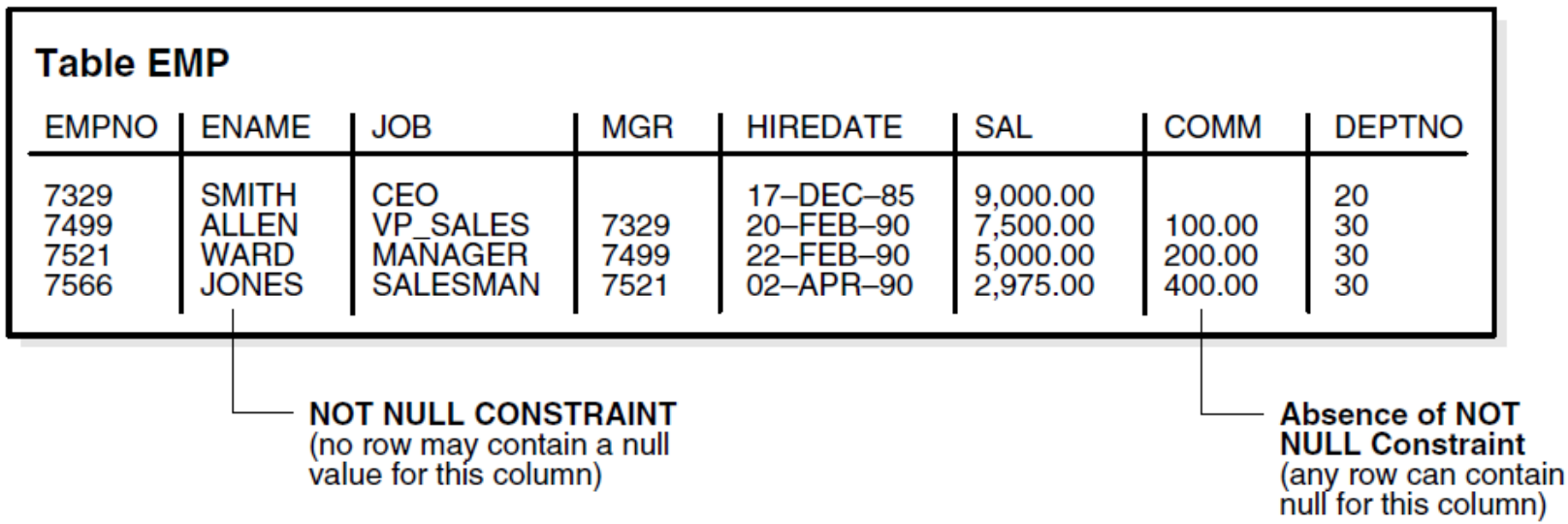
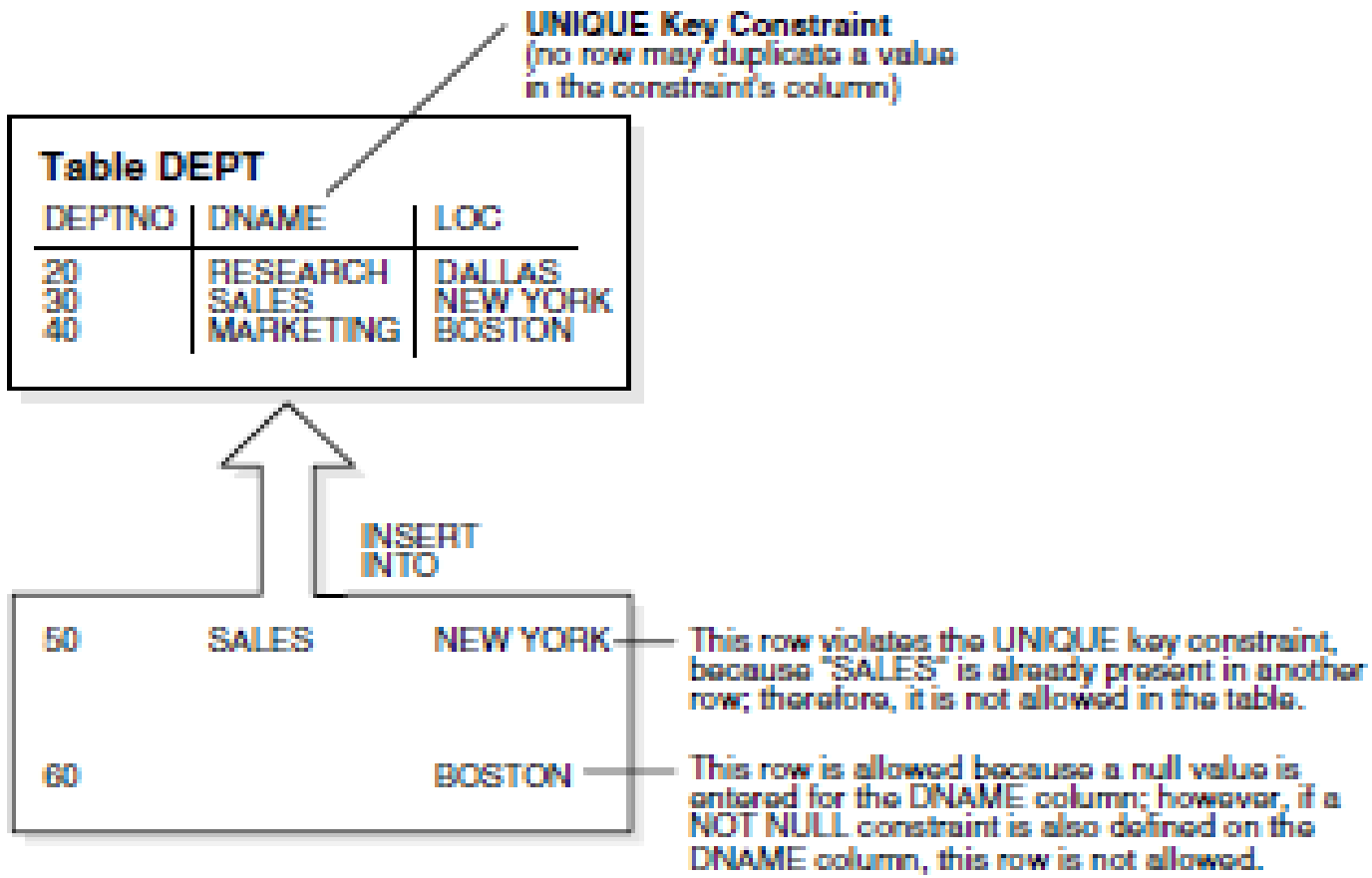


Table EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7329	SMITH	CEO		17–DEC–85	9,000.00		20
7499	ALLEN	VP_SALES	7329	20–FEB–90	7,500.00	100.00	30
7521	WARD	MANAGER	7499	22–FEB–90	5,000.00	200.00	30
7566	JONES	SALESMAN	7521	02–APR–90	2,975.00	400.00	30

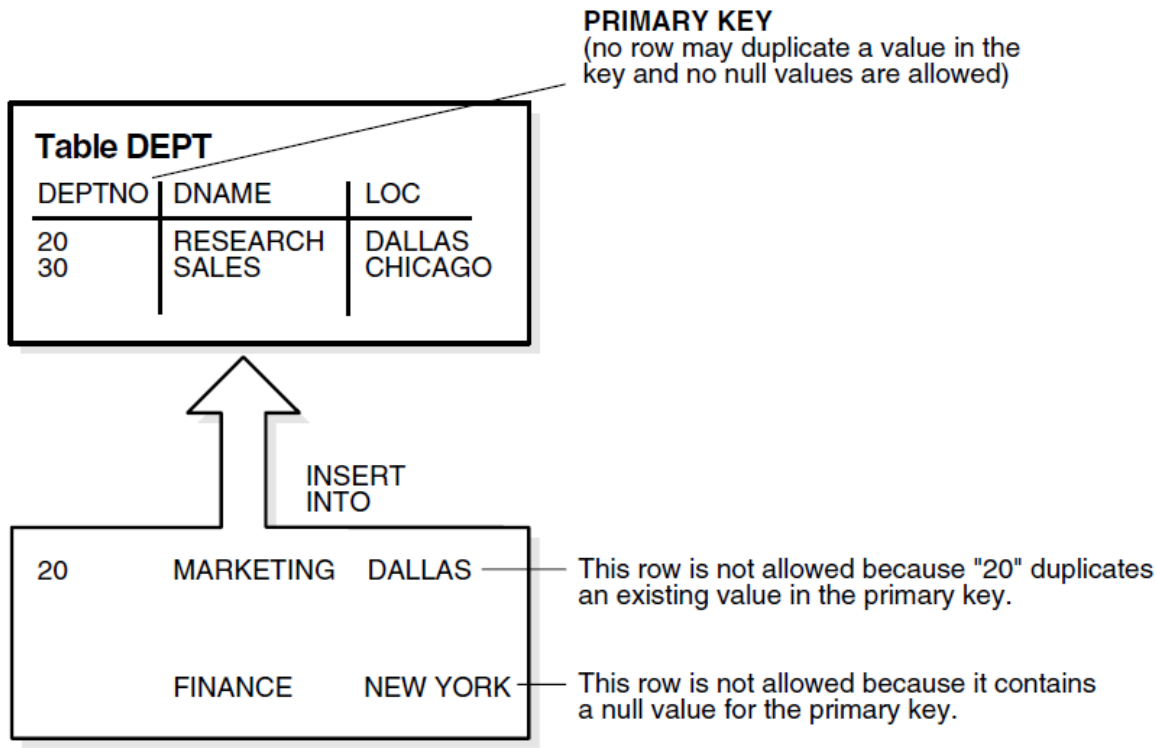
Unicidad (UNIQUE)

- Cada valor en una columna debe ser único.
- No puede haber dos registros de una tabla que tengan un valor duplicado en la columna.



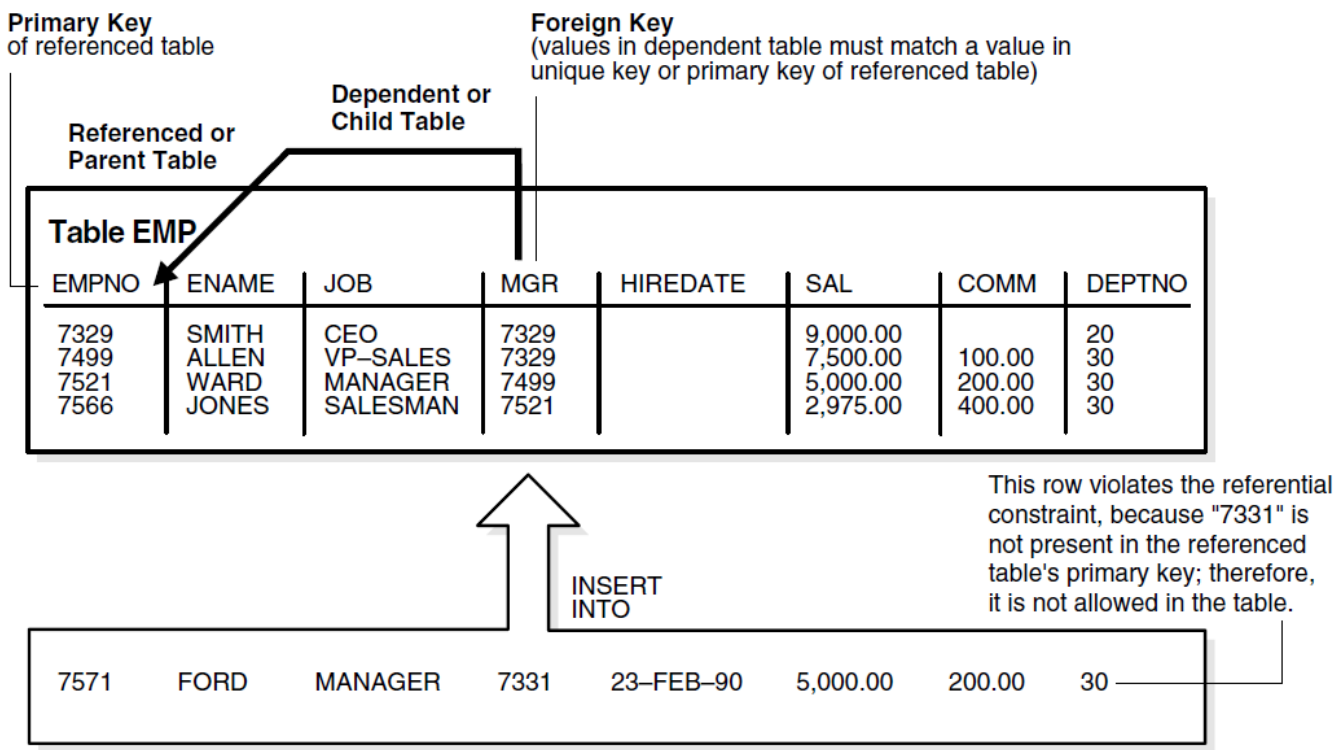
Clave primaria (PRIMARY KEY)

- Cada tabla puede tener como mucho una restricción de clave primaria.
- El valor de la columna sujeta a esta restricción constituye el identificador único del registro.



Restricciones de integridad referencial

- Las tablas de una BD relacional pueden estar relacionadas a través de columnas basadas en el mismo dominio.
- Se deben mantener las reglas que gobiernan las relaciones de esas columnas.



Acciones definidas por restricciones de integridad referencial

- DELETE:
  - CASCADE.
  - SET NULL.
  - Si no se especifica regla de borrado, es RESTRICTED.

- UPDATE:
  - No admite regla de actualización.
  - Siempre es RESTRICTED.

Restricciones de validación (CHECK)

- Restricción en una columna o conjunto de columnas que requieren que una determinada condición sea verdadera o desconocida para cada fila de la tabla.
  - Tiene que ser una expresión booleana que se evalúa utilizando los valores de la fila que se insertan o actualizan.
  - No pueden contener:
    - Subconsultas.
    - Secuencias.
    - Funciones de SQL:
      - SYSDATE.
      - UID.
      - USER.
      - USERENV.
    - Pseudocolumnas:
      - LEVEL.
      - ROWNUM.

Comprobación diferida de una restricción

- Una restricción es diferida (DEFERRED) si el sistema intenta verificar que se cumple en la confirmación (COMMIT).
  - Si es ilegal, el intento de COMMIT provoca la operación de ROLLBACK.
- Si una restricción es inmediata (NONDEFERRED), se comprueba al final de cada acción.
  - Si es ilegal, la acción se deshace inmediatamente.

Atributos de restricción

- Se especifican con palabras clave en la cláusula de restricción:
  - DEFERRABLE o NOT DEFERRABLE.
  - INITIALLY DEFERRED o INITIALLY IMMEDIATE.
- Las restricciones se pueden:
  - Añadir.
  - Eliminar.
  - Modificar.
- Y pueden ser :
  - Activadas.
    - Con validación.
    - Sin validación.
  - Desactivadas.
    - Con validación.
    - Sin validación.

SET CONSTRAINTS

- La sentencia SET CONSTRAINTS puede modificar el momento de comprobación de las restricciones:
  - SET CONSTRAINTS ... IMMEDIATE causa que las restricciones especificadas se verifiquen inmediatamente después de cada sentencia.

- La declaración ALTER SESSION también tiene cláusulas de SET CONSTRAINTS IMMEDIATE o DEFERRED.
  - Son equivalentes a ejecutar una instrucción SET CONSTRAINTS al comienzo de cada transacción en la sesión actual.

Estados de una restricción

- ENABLE:
  - Habilita la restricción.
  - Se garantiza para las futuras filas.
    - VALIDATE:
      - Es la opción por defecto para ENABLE.
      - Se comprueba la restricción para las filas existentes.
    - NOVALIDATE:
      - No se comprueba la restricción para las filas existentes.

Estados de una restricción

- DISABLE:
  - Deshabilita la restricción.
  - Elimina, en su caso, el índice creado por la restricción.
    - VALIDATE:
      - No permite ninguna modificación de las columnas afectadas por la restricción.
    - NOVALIDATE:
      - Es la opción por defecto para DISABLE.
      - La restricción no se comprueba.

Triggers

Triggers en ORACLE

- Triggers (disparadores) en Oracle:
  - Bloques de código que son implícitamente invocados cuando algo sucede.

- Triggers vs. Procedimientos Almacenados:
  - Ejecución implícita: Disparar
  - No admiten argumentos

- PL/SQL:
  - Lenguaje de programación estructurado en bloques.

- Bloques:
  - Unidad mínima en PL/SQL
    - Soportan DML y DDL
    - Anónimos o con nombre

Triggers en ORACLE

- Se utilizan para:
  - Mantener la integridad semántica,
  - Auditar cambios en los datos,
  - Evitar la ejecución de transacciones inválidas,
  - …
- La estructura básica de un trigger es:
  - Sentencia de activación:
    - Es la sentencia que permite disparar el código a ejecutar.
  - Restricción:
    - Es la condición que se debe cumplir para que se ejecute el código.
  - Acción a ejecutar:
    - Es la secuencia de instrucciones a ejecutar una vez que se ha cumplido las condiciones iniciales.
- Los triggers se almacenan en la BD (user_triggers) y están asociados a una tabla.
- Pueden afectar a n filas.
- Se compilan cada vez que se ejecutan.

Estructura general de un trigger

CREATE [OR REPLACE] TRIGGER nombre

{BEFORE | AFTER | INSTEAD OF}

{INSERT | DELETE | UPDATE [OF <atributo>]} ON <tabla>

[FOR EACH ROW | STATEMENT]

[WHEN condición]

```
DECLARE
    /* Declaraciones de uso local:
      variables, cursores y excepciones de usuario */
BEGIN
    /* Proceso: conjunto de sentencias ejecutables */
EXCEPTION
    /* Excepciones: zona de control de errores */
END;
```

Temporalidad del evento: AFTER/BEFORE

- BEFORE: Ejecutan la acción asociada antes de que la sentencia sea ejecutada.

```
CREATE TRIGGER NombreTrigger
BEFORE Insert ON NombreTabla …
```

- AFTER: Ejecutan la acción asociada después de que se haya ejecutado la sentencia.

```
CREATE TRIGGER NombreTrigger
AFTER Insert ON NombreTabla …
```

Condición: WHEN

•Expresa una condición que debe cumplirse en el momento de producirse el evento, para que la acción sea ejecutada.

•Se evalúa para cada fila.

WHEN old.nombre = ‘pepe’ OR old.edad > 35

•Debe ser una expresión booleana y no puede contener subconsultas.

•Se puede utilizar cualquier combinación de operadores lógicos (AND, OR, NOT) y relacionales (< <= > >= = <>).

•No se puede especificar una condición para los disparadores a nivel de sentencia (STATEMENT) ni en los disparadores INSTEAD OF.

Ámbito del evento:
FOR EACH ROW / STATEMENT

- A NIVEL DE FILA: ROW TRIGGERS
  - Ejecutan la acción asociada tantas veces como filas se vean afectadas por la sentencia que lo dispara.
  - Si ninguna fila se ve afectada, no se dispara.

```
CREATE TRIGGER NombreTrigger
BEFORE UPDATE ON NombreTabla
FOR EACH ROW
[WHEN …]…
```

  - Se utiliza cuando la acción depende de la sentencia que produjo el evento o de las filas afectadas.

Ámbito del evento:
FOR EACH ROW / STATEMENT

- A NIVEL DE SENTENCIA: STATEMENT TRIGGERS
  - Ejecutan una única vez la acción asociada, independientemente del número de filas que se vean afectadas por la sentencia (incluso si no hay filas afectadas).

```
CREATE TRIGGER NombreTrigger
BEFORE INSERT ON NombreTabla
[STATEMENT] …
```

  - Utilizar cuando la acción NO depende de la sentencia que produjo el evento o de las filas afectadas.
    - Comprobaciones de seguridad acerca del usuario o del momento concreto.
    - Generar registros de auditoría.

Orden de ejecución

- Una sentencia SQL puede disparar varios TRIGGERS.

- La activación de un trigger puede disparar la activación de otros triggers (Triggers en cascada).

    1. Triggers Before (a nivel de sentencia)
    2. Para cada fila:
        1. Triggers Before (a nivel de fila)
        2. Ejecuta la Sentencia
        3. Triggers After (a nivel de fila)
    3. Triggers After (a nivel de sentencia)

- Se compromete o se deshace toda la transacción.

- El orden de ejecución de disparadores del mismo tipo es indeterminado.

Tipos de triggers: INSTEAD OF

- Podemos crear triggers que no se ejecutan antes ni después de una instrucción sino en lugar de (instead of).

- Sólo podemos utilizar estos triggers si están asociados a vistas, además actúan siempre a nivel de fila.

```
CREATE VIEW existenciasCompleta (tipo,modelo, precio,almacen, cantidad) AS
SELECT p.tipo, p.modelo, p.precio_venta, e.n_almacen, e.cantidad
FROM PIEZAS p, EXISTENCIAS e
WHERE p.tipo=e.tipo AND p.modelo=e.modelo
ORDER BY p.tipo, p.modelo, e.n_almacen;
```

- Esta instrucción daría lugar a error:

```
INSERT INTO existenciasCompleta VALUES ('ZA',3,4,3,200);
```

ORA-01732: operación de manipulación de datos no válida en esta vista

Tipos de triggers: INSTEAD OF

- Esta situación se puede solucionar con un trigger que inserte primero en la tabla piezas y luego en la tabla existencias.

- Esta operación se puede realizar a través de un trigger del tipo INSTEAD OF, que sustituirá el INSERT original por el código indicado en el trigger:

```
CREATE OR REPLACE TRIGGER ins_piezas_exis INSTEAD OF INSERT ON
existenciasCompleta
BEGIN
 INSERT INTO piezas(tipo,modelo,precio_venta)
      VALUES(:NEW.tipo,:NEW.modelo,:NEW.precio);
 INSERT INTO existencias(tipo,modelo,n_almacen,cantidad)
      VALUES(:NEW.tipo,:NEW.modelo,:NEW.almacen,:NEW.cantidad);
END;

 INSERT INTO existenciasCompleta VALUES ('ZA',3,4,3,200);
                              1 fila creada.
```

Tipos de triggers:
de sistema y de usuario.

- Disparados por:
  - Eventos del sistema
    - Arranque y parada de BD
    - Transacciones
    - Errores
  - Eventos relacionados con las acciones de los usuarios.
    - Conexión / Desconexión
    - Sentencias DDL:
      - CREATE
      - ALTER
      - DROP
      - …

Tipos de triggers: de sistema y de usuario.

- Ejemplo:

```
CREATE OR REPLACE TRIGGER LogCreations
AFTER CREATE ON SCHEMA
BEGIN
    INSERT INTO LogCreate (user_id, object_type,
      object_name, object_owner, creation_date)
    VALUES (USER, ORA_DCIT_OBJ_TYPE,
      ORA_DICT_OBJ_NAME, ORA_DICT_OBJ_OWNER,
      SYSDATE)
END LogCreations;
```

Activar/Desactivar/Borrar Triggers

- Todos los triggers asociados a una tabla:

  ALTER TABLE *nombre_tabla* ENABLE ALL TRIGGERS;

  ALTER TABLE *nombre_tabla* DISABLE ALL TRIGGERS;

  - Por defecto todos están activados al crearse.

- Un trigger específico:

  ALTER TRIGGER *nombre_trigger* ENABLE;

  ALTER TRIGGER *nombre_trigger* DISABLE;

- Borrar un trigger:

  DROP TRIGGER *nombre_trigger*;

Consultar información sobre los triggers

- Ver todos los triggers y su estado

  SELECT TRIGGER_NAME , STATUS FROM USER_TRIGGERS;

TRIGGER_NAME	STATUS
INS_PIEZAS_EXIS	ENABLED
REPCATLOGTRIG	ENABLED
DEF$_PROPAGATOR_TRIG	ENABLED

- Ver el cuerpo de un trigger

  SELECT TRIGGER_BODY FROM USER_TRIGGERS WHERE

  TRIGGER_NAME=‘nombre_disparador’;

- Ver la descripción de un trigger

  SELECT DESCRIPTION FROM USER_TRIGGERS WHERE

  TRIGGER_NAME= ‘nombre_disparador’;