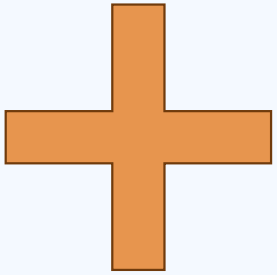
Bases de datos

PL-SQL

PL-SQL

Estructurado

- Variables
- Sentencias:
  - Composición
  - Condicionales
  - Bucles
- Procedimientos/funciones



SQL

- DML:
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
- Cursores
- Transacciones:
  - COMMIT
  - ROLLBACK
  - SAVEPOINT

Restricciones de integridad

- Dominio: tipo del atributo impide meter datos
  - Dominios: **CREATE DOMAIN** NombreDominio tipoDominio. (**DROP, ALTER**)
  - CHECK: **CONSTRAINT CK_**nombreRestricción **CHECK** condiciónBooleana;
  - Unicidad: **UNIQUE** admite nulos, **PRIMARY KEY** no admite nulos
  - Nulos: Por defecto se admiten, **NOT NULL para no admitirlos**
- Integridad referencial: impide meter datos que no existan
  - **FOREIGN KEY / REFERENCE**

  en Oracle se pude ON DELETE CASCADE, ON DELETE SET NULL
- Disparadores/Triggers: resto de acciones

Disparadores: Sintaxis

Si se omite `OR REPLACE` *y el trigger existe dará error*

```
CREATE [OR REPLACE] TRIGGER nombreDisparador
{BEFORE | AFTER}
{INSERT | DELETE | UPDATE [OF columnas]} ON <nombreTabla>
```

EVENTO

```
[FOR EACH ROW | STATEMENT]
[WHEN condición]
```

CONDICIÓN

```
[DECLARE
  Variables]
BEGIN
  -- Cuerpo del trigger
END;
/
```

ACCIÓN

Disparadores. Orden de ejecución

1. Se ejecuta, si existe, el disparador de tipo BEFORE con nivel de sentencia
2. Para cada fila a la que afecte la sentencia SQL:
   i. Se ejecuta, si existe, el disparador de tipo BEFORE con nivel de fila
   ii. Se ejecuta la sentencia SQL
   iii. Se ejecuta, si existe, el disparador de tipo AFTER con nivel de fila
3. Se ejecuta, si existe, el disparador de tipo AFTER con nivel de sentencia

Disparadores. Funciones del cuerpo del disparador

```
CREATE OR REPLACE TRIGGER Ejemplo
BEFORE INSERT OR UPDATE OR DELETE ON tabla
BEGIN
   acciones comunes antes
   IF DELETING THEN
        Acciones asociadas al borrado
   ELSIF INSERTING THEN
        Acciones asociadas a la inserción
   ELSIF UPDATING('COL1')
        Acciones asociadas a la modificación
   ELSIF UPDATING('COL2')
        Acciones asociadas a la modificación
   END IF;
 acciones comunes después
END Ejemplo;
```

Instrucción condicional: If

```
IF condicion THEN sentencias

ELSIF condicion THEN sentencias

ELSE sentencias

END IF;
```

Instrucción condicional: If

```
IF condicion THEN sentencias

ELSIF condicion THEN sentencias

ELSE sentencias

END IF;
```

SELECT... INTO...

SELECT columna1, ..., columnaN

INTO variable1, ..., variableN o variableRegistro

FROM <tablas/Vistas>

WHERE

<Predicados>;

- NO_DATA_FOUND. Se lanza si la consulta devuelve un resultado vacío.
- TOO_MANY_ROWS. Se lanza si la consulta devuelve más de una fila.

CURSORES

```
CURSOR cmotos (cilindrada INTEGER DEFAULT 125) IS
    SELECT marca, modelo
    FROM motos
    WHERE cc>=cilindrada;
```

CURSORES: Bucles Loop

****

```
OPEN cmotos;
LOOP
        FETCH cmotos INTO xmarca, xmodelo;
        EXIT WHEN cmotos%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(xmarca || ' ' || xmodelo);
END LOOP;
CLOSE cmotos;
```

****

CURSORES: Bucles While

****

```
OPEN cmotos;
FETCH cmotos INTO xmarca, xmodelo;
WHILE cmotos%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(xmarca || ' ' || xmodelo);
        FETCH cmotos INTO xmarca, xmodelo;
END LOOP;
CLOSE cmotos;
```

****

CURSORES: Bucles For

****

```
FOR moto IN cmotos LOOP
        DBMS_OUTPUT.PUT_LINE(moto.marca || ' ' || moto.modelo);
END LOOP;
-- No hace falta ni abrir ni cerrar el cursor.
```

****

CURSORES: Variables

- %FOUND. Desde que el cursor es abierto hasta que se obtiene la primera fila el valor de %FOUND es NULL. Tras obtener una fila %FOUND devuelve TRUE si realmente se ha podido obtener una fila FALSE en otro caso.

- %ISOPEN. Devuelve TRUE si el cursor se ha abierto y FALSE en otro caso.

- %NOTFOUND. Es el opuesto de %FOUND.

- %ROWCOUNT. Antes de que se obtenga alguna fila %ROWCOUNT devuelve el valor 0, después de que se haya obtenido alguna fila devuelve el número de filas que se han obtenido hasta el momento (es decir el número de FETCH exitosas hasta el momento).

Procedimientos

CREATE [OR REPLACE] PROCEDURE <nombre> [(<parámetrol>, ... <parámetroN>)] IS

    [<declaraciones de variables, constantes, ..>]

BEGIN

    sentencias;

[EXCEPTION

    manejadores de excepciones;]

END [<nombre>];

Funciones

CREATE [OR REPLACE] FUNCTION <nombre> [(<parámetrol>, ... <parámetroN>)] RETURN <tipo> IS

    [<declaraciones de variables, constantes, ..>]

BEGIN

    sentencias;

    RETURN v; -- La función devuelve el valor de la variable v.

[EXCEPTION

    manejadores de excepciones;]

END [<nombre>];

Excepciones

RAISE_APPLICATION_ERROR(-20000, 'Usted no debería borrar filas');