Asignatura

Base de datos

UNIVERSAE
Instituto Superior de FP

Asignatura

Base de datos

UNIDAD 6

Construcciones de guiones

UNIVERSAE
Instituto Superior de FP

PL/SQL

¿Es SQL?

- Procedural Language/Structured Query Language
- Lenguaje procedimental
- Ejecución desde el servidor

Características

- Uso de variables
- Estructuras de control
- Estructuras funcionales
- Control de excepciones

Estructuras funcionales

- Scripts anonimos
- Procedimientos
- Funciones
- Disparadores

Compatibilidad de PL/SQL

MySQL o MariaDB

URL: *https://dev.mysql.com/doc/refman/8.0/en/create-procedure.html*
https://mariadb.com/kb/en/stored-procedures/

- No existe PL/SQL
- Dispone de algunas instrucciones para crear bloques
- Posibilidad de admitir instrucciones de Oracle.
  SET SESSION sql_mode=ORACLE;

```
DELIMITER //

CREATE PROCEDURE bloque ( OUT param1 CHAR(10) )
 BEGIN
   SELECT "Hola mundo" INTO param1 ;
 END;
//

DELIMITER ;
```

Oracle

URL: *https://docs.oracle.com/cd/A97630_01/appdev.920/a96624/toc.htm*
https://blogs.oracle.com/connect/post/building-with-blocks

- Nace para este tipo de base de datos

```
BEGIN
    dbms_output.put_line('Hola Mundo');
END;
/
```

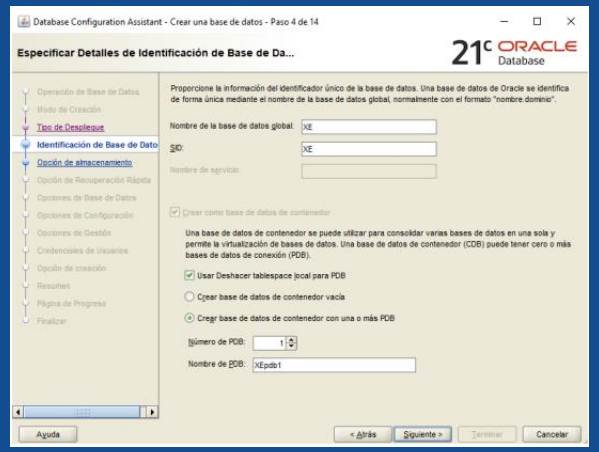
Tipos según la base de datos

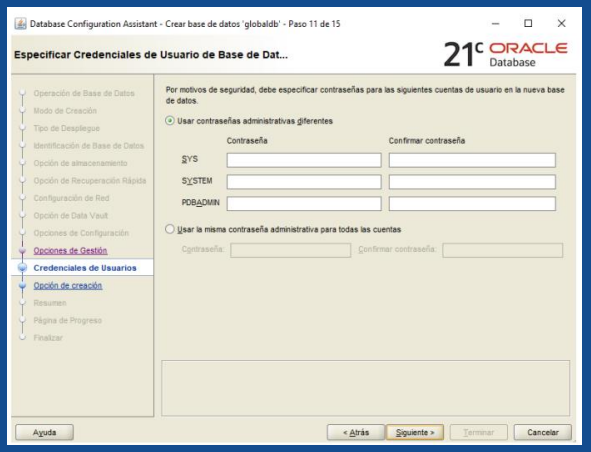
Lenguaje	Tipo base de datos
PL/SQL	Oracle, DB2
T/SQL	Microsoft SQL Server
PL/pgSQL	PostgreSQL
SPL	Informix

Instalación de oracle

Procedimiento

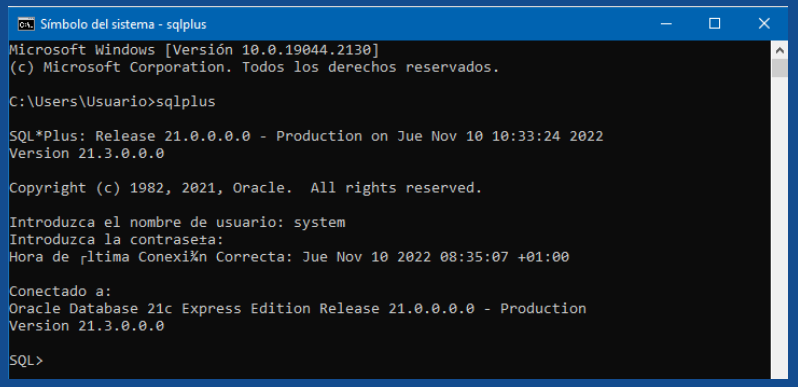
- Descargar la versión **Oracle express edition (XE)**
- *https://www.oracle.com/es/database/technologies/appdev/xe.html*
- Seguir el asistente de instalación por defecto en cada opción
- Parámetros a configurar:
  - Indicar el nombre de la base de datos, por defecto XE
  - Cambiar el puerto, por defecto 5500
  - Establecer la contraseña de los usuarios SYS, SYSTEM, PDADMIN



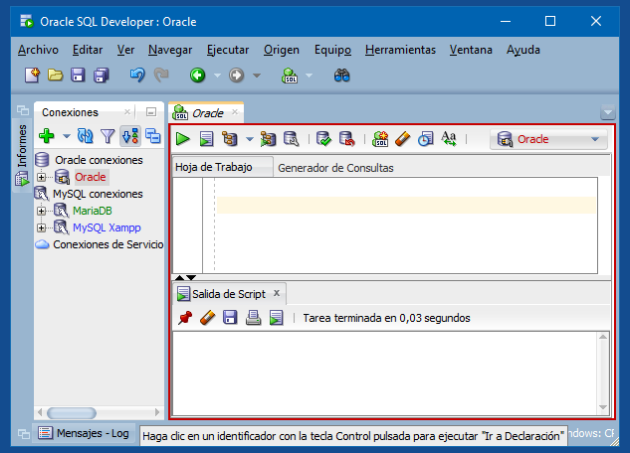


Clientes

- SQL/PLUS
  - ❑ Incluido con la instalación de Oracle XE
  - ❑ Abrir un terminal y ejecutar **sqlplus**



- SQL Developer
  - ❑ *Interfaz gráfica adaptada para Oracle*
  - ❑ *https://www.oracle.com/database/sqldeveloper/technologies/download/*



Bloques de código

Estructura

```
[DECLARE]
--Variables, cursores, excepciones
 BEGIN
        -- sentencia SQL deseadas
        -- sentencias de control PL/SQL deseadas;
[EXCEPTION]
        -- Excepciones implementadas
END;
/
```

Apartados

- **DECLARE**. Declaración de variables, cursores
- **BEGIN**. Inicio del bloque
- **EXCEPTION**. Uso de excepciones
- **END**. Fin del bloque

Variables

Contenedores de valores

- Almacenan valores
- Es necesario especificar un nombre identificativo
- Y el tipo de datos que va almacenar
- Se indican en la parte del bloque DECLARE
- Las **constantes** almacenan valores fijos que no cambian

Sintaxis

Nombre [CONSTANT] tipo_de_dato [NOT NULL] [:= | DEFAULT | Expresión];

Tipos de datos

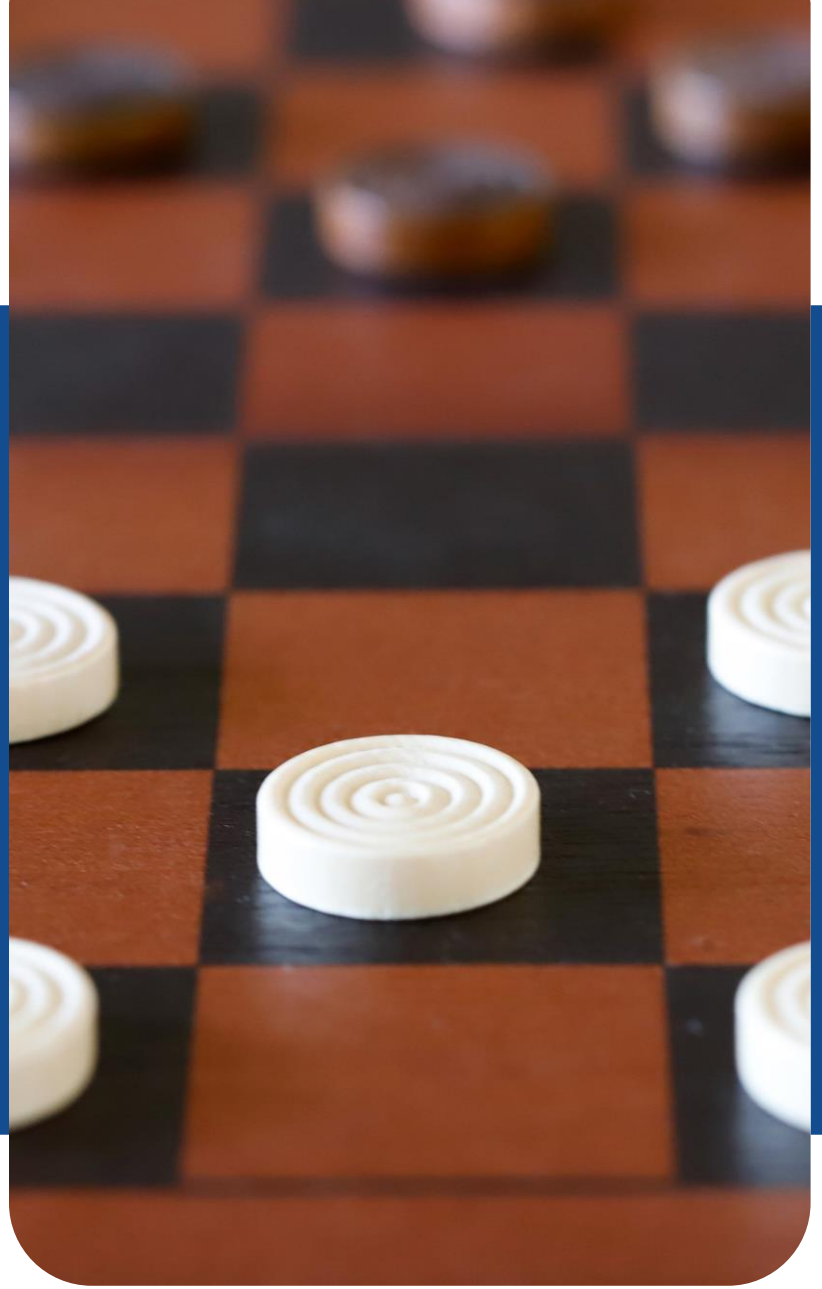
- Los propios de la base de datos (NUMBER, CHAR, VARCHAR2, BOOLEAN, DATE)
- Definición de nuevo tipos
- Obtener información %TYPE o %ROWTYPE

```
DECLARE
    numero NUMBER;
    fecha DATE;
    cadena VARCHAR2(100) := 'TEXTO';
    pi CONSTANT NUMBER := 3.141592;

    CURSOR c_cursos IS (SELECT  *  FROM CURSOS);
    r_cursos c_cursos%ROWTYPE;
BEGIN
    dbms_output.put_line(numero);
    dbms_output.put_line(cadena);
    dbms_output.put_line(pi);

    SELECT fecha INTO fecha FROM cursos;
    dbms_output.put_line(fecha);

    OPEN c_cursos;
    LOOP
      dbms_output.put_line(r_cursos.nombre);
      FETCH c_cursos INTO r_cursos;
      EXIT WHEN c_cursos%NOTFOUND;
    END LOOP;
END;
/
```



Operadores

Operador	Acción
**	Potencia
+- (unarios)	Signo positivo o negativo
*	Multiplicación
/	División
+	Suma
-	Resta
||	Concatenación
=,<,>,<=	Comparaciones: igual, menor, mayor, menor o igual
>=,<>,!=	Mayor o igual, distinto, distinto
IS NULL, LIKE	Es nulo, como
BETWEEN, IN	Entre, in
NOT	Negación lógica (boolean)
AND	Operador AND lógico entre datos boolean
OR	Operador OR lógico entre datos boolean

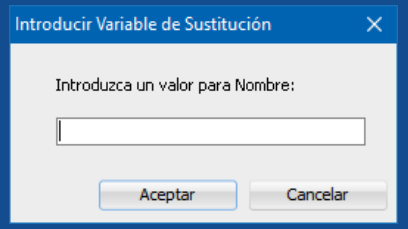
Entrada y salida

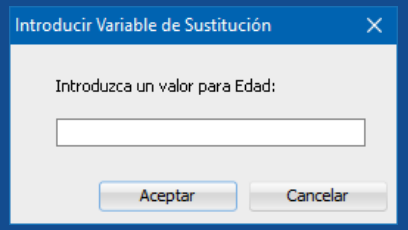
Entrada

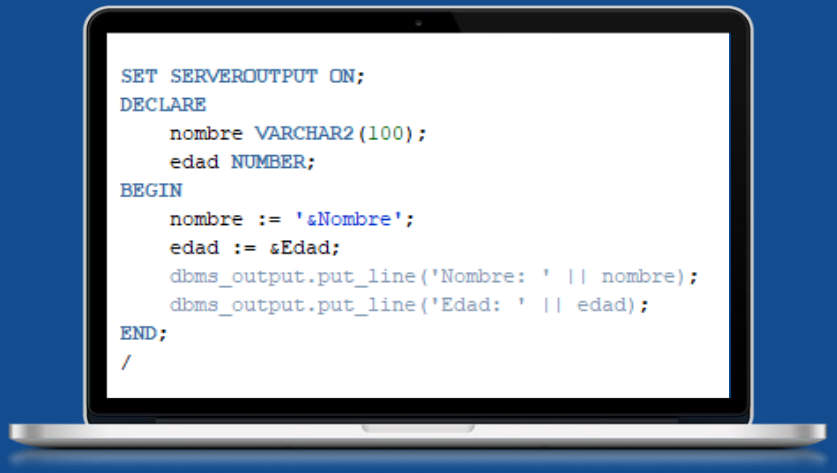
- Por consola.
    - Cada entrada se solicita mediante &
    - Seguido de la descripción, sin espacios
- Mediante parámetros en procedimiento y funciones

Salida

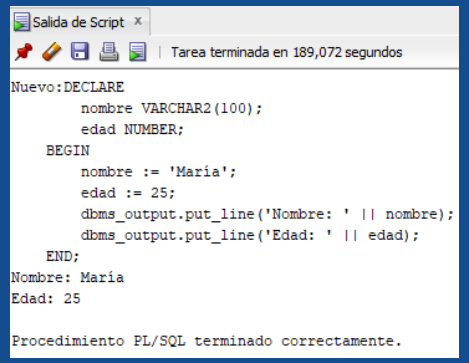
- Por defecto, mostrar los datos por consola esta desactivado.
  SET SERVEROUTPUT ON
- Sintaxis
  dbms_output.put_line(valor o variable);







```
SET SERVEROUTPUT ON;
DECLARE
    nombre VARCHAR2(100);
    edad NUMBER;
BEGIN
    nombre := '&Nombre';
    edad := &Edad;
    dbms_output.put_line('Nombre: ' || nombre);
    dbms_output.put_line('Edad: ' || edad);
END;
/
```



Estructuras de control. Selección o alternativa

Selección o alternativa

- IF
- IF/ELSE
- ELSEIF
- CASE

```
DECLARE
    nombre VARCHAR2(100) := '&nombre';
BEGIN
    IF nombre IS NULL THEN
        dbms_output.put_line('Hola desconocido');
    ELSE
        dbms_output.put_line('Hola ' || nombre);
    END IF;
END;
```

```
DECLARE
    nombre VARCHAR2(100) := '&nombre';
BEGIN
    IF (nombre IS NULL) THEN
        dbms_output.put_line('Hola desconocido');
    ELSIF (nombre='Pablo') THEN
        dbms_output.put_line('Pablo mi mejor amigo');
    ELSE
        dbms_output.put_line('Hola ' || nombre);
    END IF;
END;
```

```
DECLARE
    nota NUMBER := '&nota';
BEGIN
    CASE
        WHEN nota >= 5 THEN
            dbms_output.put_line('Aprobado');
        WHEN nota < 5 THEN
            dbms_output.put_line('Suspenso');
        ELSE
            dbms_output.put_line('La nota introducida es erronea');
    END CASE;
END;
```

Estructuras de control. Iterativa o repetición

Iterativa o repetición

- LOOP
- WHILE
- FOR

```
DECLARE
    numero NUMBER := &numero;
    temp NUMBER := 1;
    suma NUMBER := 0;
BEGIN
    LOOP
        suma := suma+temp;
        temp := temp+1;
        EXIT WHEN temp>numero;
    END LOOP;
    dbms_output.put_line('SUMA = ' || suma );

    suma := 0;
    temp := 1;
    WHILE temp <= numero LOOP
        suma := suma+temp;
        temp := temp+1;
    END LOOP;
    dbms_output.put_line('SUMA = ' || suma );

    suma := 0;
    temp := 1;
    FOR temp IN 1..numero LOOP
        suma := suma+temp;
    END LOOP;
    dbms_output.put_line('SUMA = ' || suma );
END;
/
```

Procedimiento

¿Qué son?

- Son bloques de código que se guardan permanentemente.
- Contiene un nombre descriptivo
- Puede recibir parámetros
  - ❑ Entrada IN
  - ❑ Salida OUT
  - ❑ Entrada salida IN OUT
- Al tener un nombre, se pueden volver a usar en cualquier momento

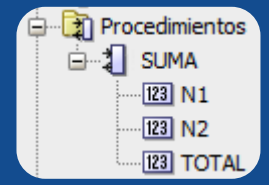
Forma de invocar

- Directamente con el nombre del procedimiento
- Usando CALL o EXEC

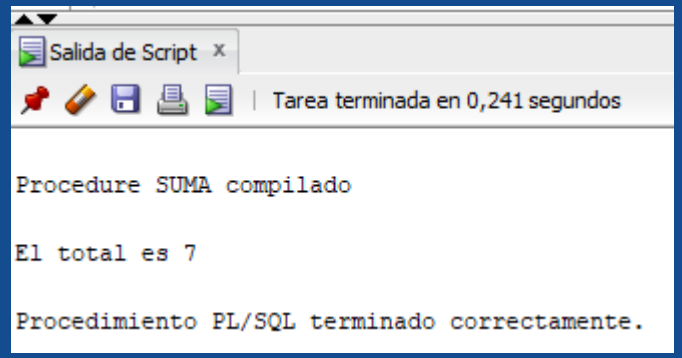
Sintaxis

CREATE (OR REPLACE) PROCEDURE Nombre del proceso
(Parámetros IN | OUT | IN OUT)
IS | AS
-- Declaración de variables internas
BEGIN
...
END;

```
CREATE OR REPLACE PROCEDURE suma(n1 IN NUMBER,
                                 n2 IN NUMBER,
                                 total OUT NUMBER)
AS
BEGIN
    total := n1 + n2;
END;
/
```




```
DECLARE
    resultado NUMBER(8) := 0;
BEGIN
    suma(2, 5, resultado);
    DBMS_OUTPUT.PUT_LINE('El total es ' || RESULTADO);
END;
/
```

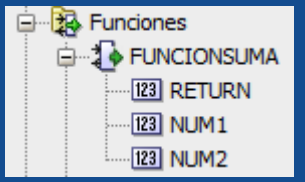


Funciones

¿Qué son?

- Parecido a un procedimiento
- Siempre devuelve algo. Tiene que haber RETURN
- No es necesario definir el tipo de parámetro. Todos son de entrada (IN)

```
CREATE OR REPLACE FUNCTION FuncionSuma(
   num1 NUMBER, num2 NUMBER
)
RETURN NUMBER
IS
    total NUMBER;
BEGIN
    total := num1 + num2;
    RETURN total;
END;
/
```



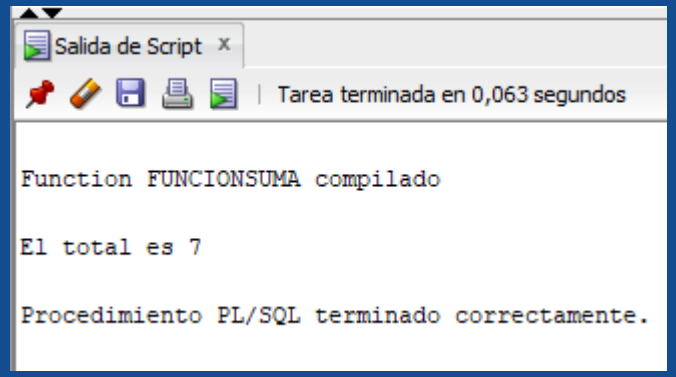
Forma de invocar

- Directamente con el nombre de la función
- Usando CALL o EXEC

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('El total es ' ||  FuncionSuma(2, 5));
END;
/
```

Sintaxis

CREATE (OR REPLACE) FUNCTION Nombre función (Parámetros)
RETURN tipo
IS | AS
-- Declaración de variables internas
BEGIN
...
END;



Sentencias SQL

Uso de SQL

- Se puede utilizar instrucciones de SQL dentro de un bloque
- Soporta cualquier instrucción, menos DDL y DCL

Consultas (*SELECT*)

- Se puede hacer cualquier tipo de SELECT
- Determinar la cantidad de resultados de la consulta
  - Si devuelve solo un valor de un campo, se puede guardar en una variable
  - Si devuelve un registro con varios valores. Uso de variable con %ROWTYPE. Se puede acceder a una parte mediante *VARIABLE.NOMBRE_CAMPO. Uso de **cursor implícito.***
  - Si devuelve un conjunto de registros. Necesario uso de ***cursor explícitos*** y **bucles** para recorrer cada fila.

Insertar, actualizar o modificar (*INSERT, UPDATE, DELETE*)

- Oracle funciona de forma transaccional
- Cuidado con insertar, actualizar y eliminar.
- Uso de COMMIT o ROLLBACK

```
DECLARE
    CURSOR c_cursos IS (SELECT  *  FROM CURSOS);
    r_cursos c_cursos%ROWTYPE;
BEGIN
    OPEN c_cursos;
    LOOP
        dbms_output.put_line(r_cursos.nombre);
        FETCH c_cursos INTO r_cursos;
        EXIT WHEN c_cursos%NOTFOUND;
    END LOOP;
    CLOSE c_cursos;
END;
/
```

```
DECLARE
  importe NUMBER;
  cuentaOrigen VARCHAR2(23);
  cuentaDestino VARCHAR2(23);
BEGIN
  importe := 50;
  cuentaOrigen := '2444 15 3110 1234567890';
  cuentaDestino := '2023 10 3005 9876543210';
  UPDATE CUENTAS SET saldo = saldo - importe WHERE cuenta = cuentaOrigen;
  UPDATE CUENTAS SET saldo = saldo + importe WHERE cuenta = cuentaDestino;
  INSERT INTO MOVIMIENTOS VALUES (cuentaOrigen, cuentaDestino, importe*(-1), SYSDATE);
  INSERT INTO MOVIMIENTOS VALUES (cuentaDestino,cuentaOrigen, importe, SYSDATE);
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
  dbms_output.put_line('Error: '||SQLERRM);
  ROLLBACK;
END;
```

Cursores

¿Qué son?

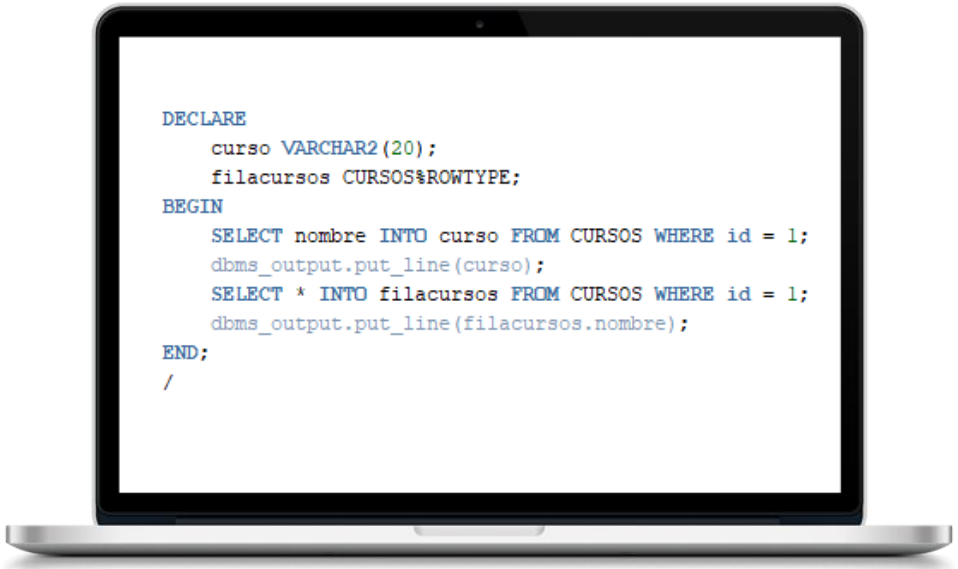
- Reserva de memoria para almacenar ejecuciones de sentencias SQL.
- Sirve para consultas (SELECT)

Tipos

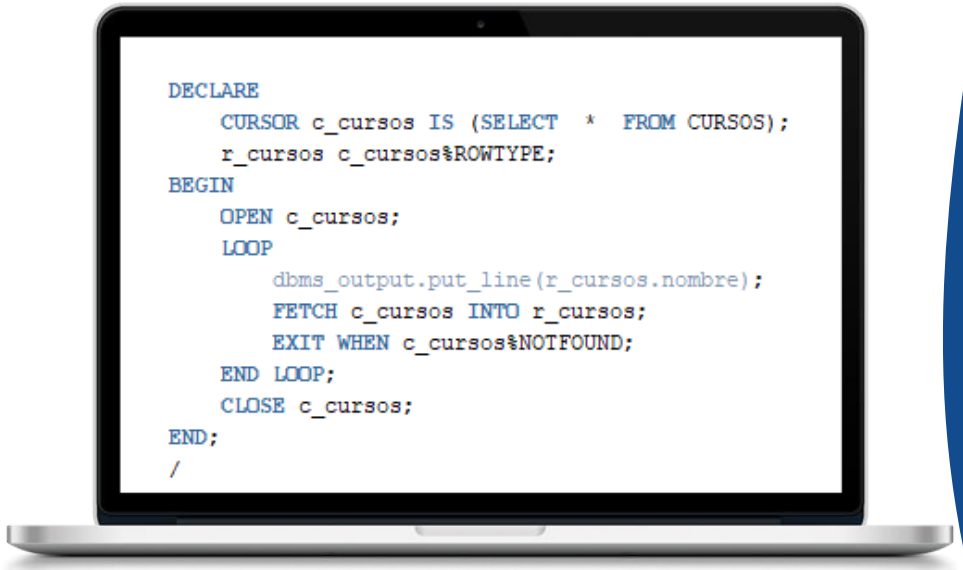
- Implícitos
  - Es necesario usar una variable
  - SELECT campo INTO variable
- Explícitos
  - Es necesario utilizar un cursor
  - Sintaxis: CURSOR nombre IS (SQL)

Acceso

- Implícitos
  - Directamente sobre la variable
  - Si es de tipo fila. *variable.campo*
- Explícitos
  - Abrir el cursor OPEN nombre
  - Recórrelo:
    - LOOP y WHILE. *FETCH nombre INTO variable*
    - FOR. *FOR variable IN nombre_cursor LOOP*
  - Cerrar el cursor



```
DECLARE
    curso VARCHAR2(20);
    filacursos CURSOS%ROWTYPE;
BEGIN
    SELECT nombre INTO curso FROM CURSOS WHERE id = 1;
    dbms_output.put_line(curso);
    SELECT * INTO filacursos FROM CURSOS WHERE id = 1;
    dbms_output.put_line(filacursos.nombre);
END;
/
```



```
DECLARE
    CURSOR c_cursos IS (SELECT  *  FROM CURSOS);
    r_cursos c_cursos%ROWTYPE;
BEGIN
    OPEN c_cursos;
    LOOP
        dbms_output.put_line(r_cursos.nombre);
        FETCH c_cursos INTO r_cursos;
        EXIT WHEN c_cursos%NOTFOUND;
    END LOOP;
    CLOSE c_cursos;
END;
/
```

Excepciones

¿Qué son?

- Errores producidos
- Se producen en tiempo de ejecución
- Pueden ser **Controlados** o **No controlados**
- Se gestionan en el bloque EXCEPTION

Declaración

- En el bloque DECLARE
  nombre EXCEPTION.
- Adicionalmente
  PRAGMA EXCEPTION_INIT (nombre, código)

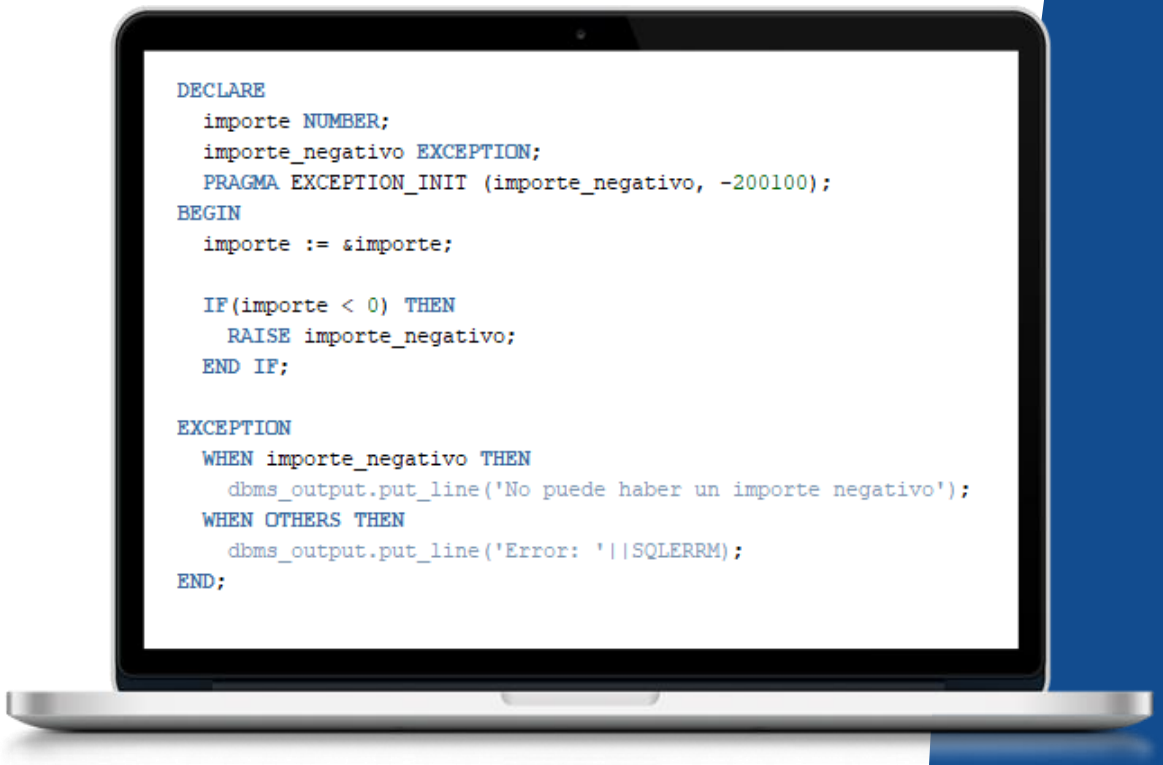
Lanzamiento

- En el bloque BEGIN
  RAISE nombre

Gestión

- En el bloque EXCEPTION
- Sintaxis

WHEN (nombre excepción | OTHERS) THEN tratamiento



```
DECLARE
  importe NUMBER;
  importe_negativo EXCEPTION;
  PRAGMA EXCEPTION_INIT (importe_negativo, -200100);
BEGIN
  importe := &importe;

  IF(importe < 0) THEN
    RAISE importe_negativo;
  END IF;

EXCEPTION
  WHEN importe_negativo THEN
    dbms_output.put_line('No puede haber un importe negativo');
  WHEN OTHERS THEN
    dbms_output.put_line('Error: '||SQLERRM);
END;
```

Disparadores (Triggers)

¿Qué son?

- Un tipo de procedimiento especial
- Se usa cuando se produce un INSERT, UPDATE o DELETE
- Se puede producir:
    - Antes de hacer la acción BEFORE
    - Después de la acción AFTER
- Se puede consultar los datos de los registros:
    - :new. Valores nuevos (INSERT O UPDATE)
    - :old. Valores antiguos (UPDATE, DELETE)

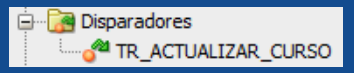
Forma de invocar

- Solo se invoca cuando se produce un INSERT, UPDATE o DELETE sobre una tabla

Sintaxis

CREATE (OR REPLACE) TRIGGER Nombre del disparador
(BEFORE | AFTER) (INSERT|UPDATE|DELETE) (OF campo) (OR …)
ON table
(FOR EACH ROW (WHEN Condición))
DECLARE
BEGIN
…
END;

```
CREATE OR REPLACE TRIGGER tr_actualizar_curso
BEFORE UPDATE
ON CURSOS
FOR EACH ROW
BEGIN
 INSERT INTO CURSOS_COPIA (idAnterior, idNuevo, nombreAnterior, nombreNuevo, fechaAnterior, fechaNueva)
   VALUES(:old.id, :new.id, :old.nombre, :new.nombre, :old.fecha, :new.fecha);
END;
/
```



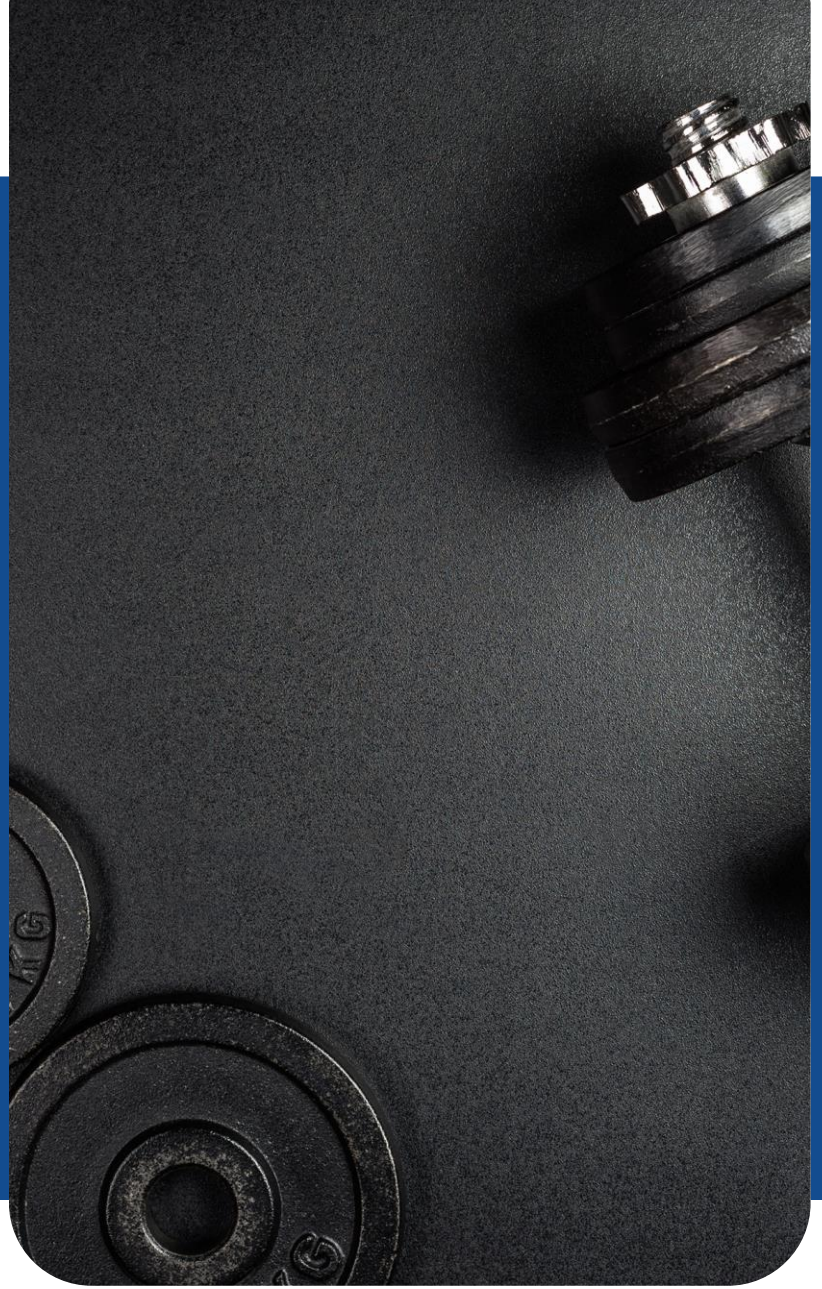

```
UPDATE CURSOS SET nombre = 'RRHH' WHERE id = 1;

SELECT * FROM CURSOS_COPIA;
```

Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 1 en 0,004 segundos

	IDANTERIOR	IDNUEVO	NOMBREANTERIOR	NOMBRENUEVO	FECHAANTERIOR	FECHAN…
1	1	1	Informatica	RRHH	(null)	(null)



Planteamiento de ejercicios

1. Crea un procedimiento o función que realice una división.

2. Crea un procedimiento o función que pida un nombre y edad. Muestre el nombre y si es mayor de edad o no.

3. Crea un procedimiento o función que muestre todos los campos de una tabla. (Podéis crear la tabla que queráis)

4. Crea una tabla *LINEA_FACTURA (numero, nombre_producto, cantidad, precio_unitario y precio_total)*
   Se pide un trigger que cuando se inserte un valor, calcule el precio total.
   El precio total es cantidad * precio_unitario

Resumen

1. PL/SQL
2. Compatibilidad de PL/SQL
3. Instalación Oracle
4. Bloques de código
5. Variables
6. Operadores
7. Entrada y salida
8. Estructuras. Selección o alternativa
9. Estructuras. Iterativa o repetición
10. Procedimientos
11. Funciones
12. Sentencias SQL
13. Cursores
14. Excepciones
15. Disparadores (Triggers)
16. Planteamiento de ejercicios

UNIVERSAE

CHANGE YOUR WAY