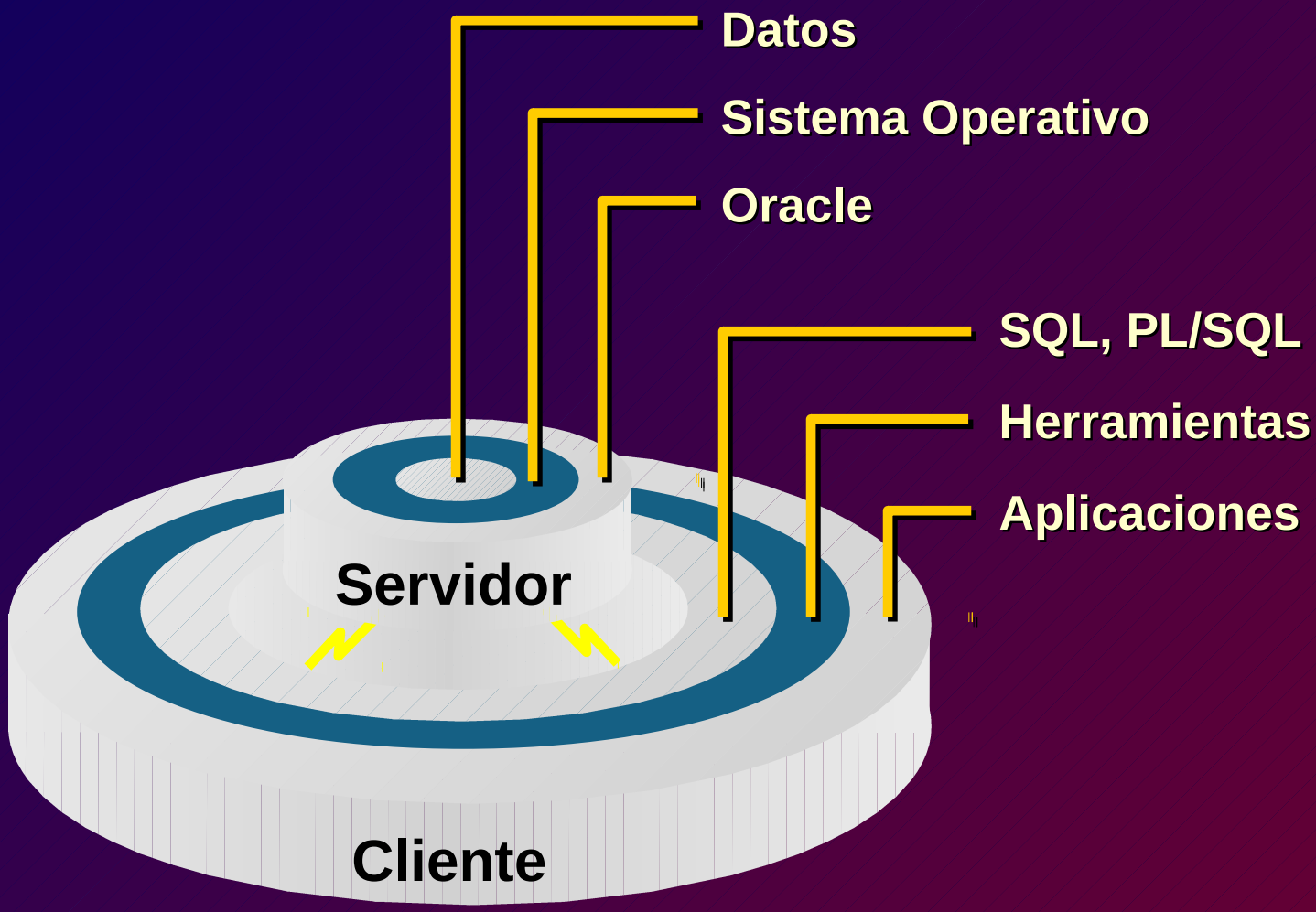
1

PL/SQL

Visión Global



ORACLE®

PL -SQL

Es una extensión procedimental del lenguaje SQL, es decir, se trata de un lenguaje creado para dar a SQL nuevas posibilidades. Esas posibilidades permiten utilizar condiciones y bucles al estilo de los lenguajes de tercera generación (como Basic, Cobol, C++, Java, etc.).

En otros sistemas gestores de bases de datos existen otros lenguajes procedimentales: SQL Server utiliza Transact SQL, Informix usa Informix 4GL,...

PL -SQL

Lo interesante del lenguaje PL/SQL es que integra SQL por lo que gran parte de su sintaxis procede de dicho lenguaje.

PL/SQL es un lenguaje pensado para la gestión de datos. La creación de aplicaciones sobre la base de datos se realiza con otras herramientas (Oracle Developer) o lenguajes externos como Visual Basic o Java.

Conceptos básicos PL/SQL

En PL/SQL existen distintos elementos de programación, que pueden ser:

- Bloque PL/SQL
- Programa PL/SQL
- Procedimiento
- Función
- Trigger (disparador)
- Paquete

Conceptos básicos PL/SQL

Bloque PL/SQL: se trata de un trozo de código que puede ser interpretado por Oracle. Se encuentra inmerso dentro de las palabras BEGIN y END.

Programa PL/SQL: conjunto de bloques que realizan una determinada labor.

Procedimiento: programa PL/SQL almacenado en la base de datos y que puede ser ejecutado si se desea con solo saber su nombre (y teniendo permiso para su acceso).

Función: programa PL/SQL que a partir de unos datos de entrada obtiene un resultado (datos de salida). Una función puede ser utilizada desde cualquier otro programa PL/SQL e incluso desde una instrucción SQL.

Conceptos básicos PL/SQL

Trigger (disparador) :programa PL/SQL que se ejecuta automáticamente cuando ocurre un determinado suceso a un objeto de la base de datos.

Paquete: colección de procedimientos y funciones agrupados dentro de la misma estructura. Similar a las bibliotecas y librerías de los lenguajes convencionales.

Estructura de Bloque

[DECLARE declaraciones]

BEGIN

Sentencias Ejecutables

[EXCEPTION manejo_de_errores]

END;

Este bloque es anónimo porque no tiene nombre

Estructura de Bloque

- DECLARE (opcional)

  Define objetos PL/SQL que serán utilizados dentro del mismo bloque
- BEGIN (obligatorio)

  Sentencias Ejecutables
- EXCEPTION (opcional)

  Qué hacer si la acción ejecutada causa un error
- END; (obligatorio)

Estructura de Procedimientos

Procedure nombre_procedure

IS

 bloque (sin declare)

Estructura de Funciones

Function nombre_funcion
RETURN tipo_de_datos
IS
 bloque (sin declare)

IMPORTANTE: Normas de escrituras

La mayor parte de las normas de escritura en PL/SQL proceden de SQL, por ejemplo:

- Las palabras clave, nombres de tabla y columna, funciones,... no distinguen entre mayúsculas y minúsculas
- Todas las instrucciones finalizan con el signo del punto y coma (;), excepto las encabezan un bloque y exception
- Los bloques comienzan con la palabra BEGIN y terminan con END
- Las instrucciones pueden ocupar varias líneas

IMPORTANTE: Normas de escrituras

Los comentarios pueden ser de dos tipos:

- Comentarios de varias líneas. Comienzan con /* y terminan con */
- Comentarios de línea simple. Son los que utilizan los signos -- (doble guión). El texto a la derecha de los guiones se considera comentario (el de la izquierda no)

Declaración de Variables PL/SQL

Sintaxis

```
identifier [CONSTANT] datatype [NOT NULL]
      [:= | DEFAULT expr];
```

Ejemplos

```
Declare
     v_hiredate     DATE;
  v_deptno          NUMBER(2) NOT NULL := 10;
     v_location     VARCHAR2(13) := 'Atlanta';
  c_comm            CONSTANT NUMBER := 1400;
```

Recomendaciones Declaración Variables

- Inicializar las ctes. y variables que no sean nulas
- Inicializar usando el parámetro de asignación := o la palabra DEFAULT
- Los identificadores de Oracle deben de tener 30 caracteres, empezar por letra y continuar con letras, números o guiones bajos (_) (también vale el signo de dólar ($) y la almohadilla (#). No debería coincidir con nombres de columnas de las tablas ni con palabras reservadas (como SELECT).
- **Declarar como máximo un identificador por línea**

Principales tipos

- VARCHAR2 (*maximum_length*)
- NUMBER [(*precision, scale*)]
- DATE
- CHAR [(*maximum_length*)]
- LONG
- LONG RAW
- BOOLEAN (true,false or NULL)
- BINARY_INTEGER

Inicialización de variables

USO:

- **:=** **Operador de asignación**
- **DEFAULT**
- **NOT NULL**

Declaración de variables

Ejemplos

```
DECLARE
  v_job              VARCHAR2(9);
  v_count          BINARY_INTEGER := 0;
  v_total_sal      NUMBER(9,2) := 0;
  v_orderdate      DATE := SYSDATE + 7;
  c_tax_rate       CONSTANT NUMBER(3,2) := 8.25;
  v_valid          BOOLEAN NOT NULL := TRUE;
  ...
```

Operadores

En PL/SQL se permiten utilizar todos los operadores de SQL: los operadores aritméticos (+ - * /), condicionales (> < != <> >= <= OR AND NOT) y de cadena (||).

A estos operadores, PL/SQL añade el operador de potencia **. Por ejemplo 4**3 es 4 elevado a 3.

Funciones

Se pueden utilizar las funciones de Oracle procedentes de SQL (TO_CHAR, SYSDATE, NVL, SUBSTR, SIN, etc.) excepto la función DECODE y las funciones de grupo (SUM, MAX, MIN, COUNT,...)

A estas funciones se añaden diversas procedentes de paquetes de Oracle o creados por los programadores y las funciones GREATEST y LEAST.

Instrucciones de control de flujo

IF condición THEN

instrucciones

[ELSE

instrucciones]

END IF;

Instrucciones de control de flujo

Se pueden anidar los if unos de otros, o con la siguiente estructura:

IF condición1 THEN

instrucciones1

ELSIF condición2 THEN

instrucciones3

[ELSIF....]

[ELSE

Instrucciones]

ENDIF;

Instrucciones de control de flujo

Sintaxis del Bucle:

```
LOOP
        instrucciones
        ...
EXIT [WHEN condición]
END LOOP;
```

El exit se puede colocar en cualquier parte del bucle.

Paquetes estándares

Oracle incorpora una serie de paquetes para ser utilizados dentro del código PL/SQL. Es el caso del paquete DBMS_OUTPUT que sirve para utilizar funciones y procedimientos de escritura como PUT_LINE.

Por ejemplo

DBMS_OUTPUT.PUT_LINE('hola');

Para que el resultado se vea hay que habilitar primero el paquete en el entorno de trabajo que utilicemos. En el caso de iSQL*Plus hay que colocar la orden interna (no lleva punto y coma):

SET SERVEROUTPUT ON

Paquetes estándares

El paquete DBMS_RANDOM contiene diversas funciones para utilizar número aleatorios. La más útil es la función DBMS_RANDOM.RANDOM que devuelve un número entero (positivo o negativo) aleatorio (y muy grande). Por ello si deseáramos un número aleatorio entre 1 y 10 se haría con la expresión:

MOD(ABS(DBMS_RANDOM.RANDOM),10)+1

Entre 20 y 50 sería:

MOD(ABS(DBMS_RANDOM.RANDOM),31)+20

Sentencias SQL en PL/SQL

- PL/SQL soporta:

Sentencias SQL, para extraer información o aplicar cambios a la base de datos

- PL/SQL no soporta:

El lenguaje de definición de datos (DDL), como CREATE TABLE, ALTER TABLE o DROP TABLE

Lenguaje de control de datos (DCL), como GRANT y REVOKE

Sentencias SQL en PL/SQL

- Extraer una fila de datos de la base de datos utilizando el comando SELECT. Sólo puede ser devuelta una fila
- Realizar cambios a las filas de una tabla de la base de datos, utilizando comandos DML (INSERT, UPDATE, DELETE)

Sentencias SELECT de PL/SQL

Recuperar datos de tablas de una base de datos mediante SELECT, y con la clausula INTO obligatoria

Sintaxis: (la sentencia select puede ser tan compleja como se quiera)

```
SELECT  select_list
INTO    variable_name[, variable_name]...
    | record_name
FROM    table
WHERE   condition;
```

Sentencias SELECT de PL/SQL

Ejemplo:

```
DECLARE
  v_deptno          NUMBER(4);
  v_location_id  NUMBER(4);
BEGIN
  SELECT  department_id, location_id
  INTO    v_deptno, v_location_id
  FROM    departments
  WHERE   department_name = 'Sales';
...
END;
/
```

Inserción de Datos

Añadir nuevos registros a una tabla de la base de datos

Ejemplo:

```
BEGIN
  INSERT INTO employees
  (employee_id, first_name, last_name, email,
   hire_date, job_id, salary)
  VALUES
   (employees_seq.NEXTVAL, 'Ruth', 'Cores', 'RCORES',
    sysdate, 'AD_ASST', 4000);
END;
/
```

Actualización de datos

Ejemplo:

```
DECLARE
  v_sal_increase   employees.salary%TYPE := 800;
BEGIN
  UPDATE      employees
  SET         salary = salary + v_sal_increase
  WHERE       job_id = 'ST_CLERK';
END;
/
```

Supresión de datos

Ejemplo:

```
DECLARE
  v_deptno   employees.department_id%TYPE := 10;

BEGIN
  DELETE FROM   employees
  WHERE         department_id = v_deptno;
END;
/
```