PUCP

1INF33 Base de Datos

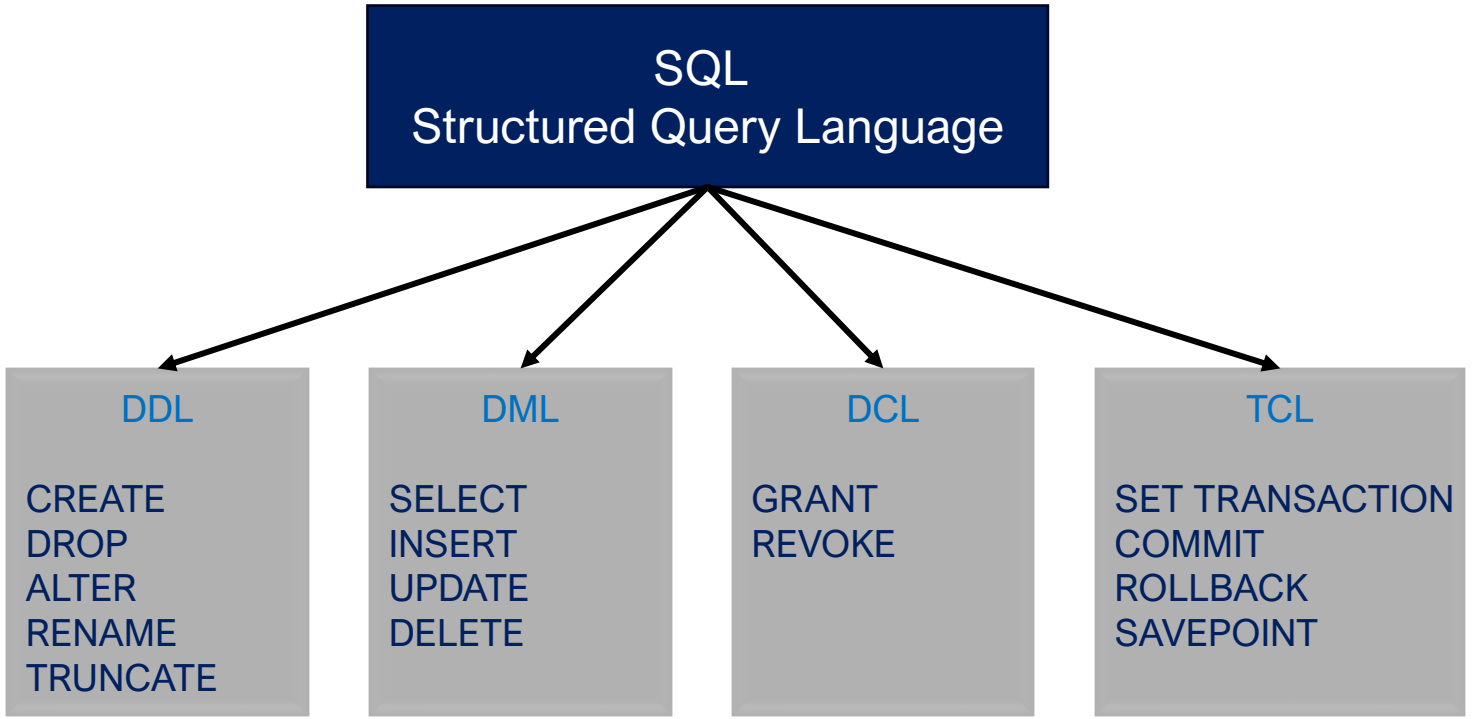
# Procedural Language / Structured Query Language

2024

Profesores del curso

# ÍNDICE

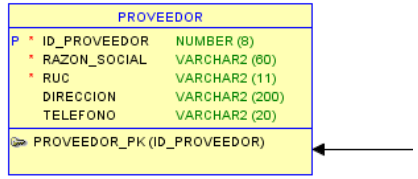
# Antecedentes



# Saberes previos

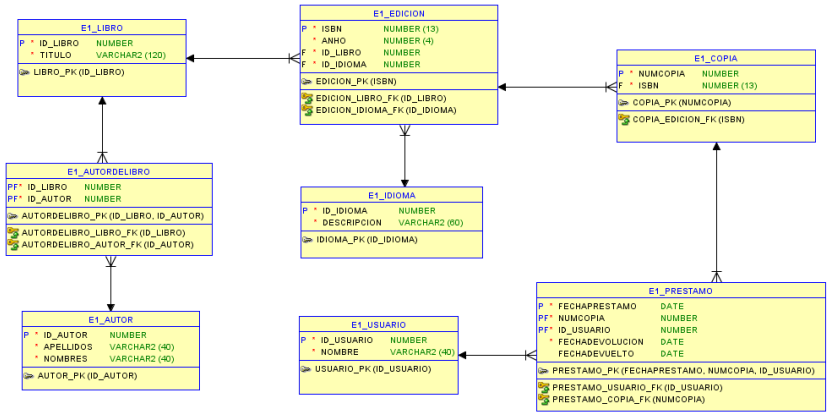
- Para consultas datos de una tabla

```
SELECT ID_PROVEEDOR, RAZÓN_SOCIAL, RUC
FROM PROVEEDOR;
```




- Para consultas recurrentes

```
CREATE VIEW V_LISTADO AS
SELECT AL.DESCRIPCION, M.DESCRIPCION_CORTA, COUNT(*)
FROM
X99_UNIDAD_MEDIDA M, X99_ARTICULO A,
X99_ALMACEN AL, X99_ARTICULOXALMACEN AXA
WHERE
M.ID_UNIDAD_MED = A. ID_UNIDAD_MED AND
A.ID_ARTICULO = AXA.ID_ARTICULO AND
AXA.ID_ALMACEN = AL.ID_ALMACEN AND
M.DESCRIPCION = 'LATA'
GROUP BY AL.DESCRIPCION, M.DESCRIPCION_CORTA;
```



# Introducción

# Definición

Temas que desarrollar:

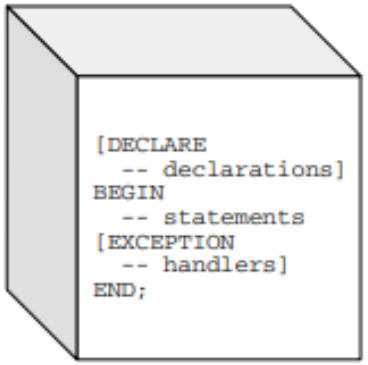
- Bloques (*Blocks*)
- Manejo de errores (*Error Handling*)
- Variables y constantes (*Variables and Constants*)
- Subprogramas (*Subprograms*)
- Paquetes (*Packages*)
- Disparadores (*Triggers*)
- Entrada y Salida (*Input and Output*)
- Abstracción de datos (*Data Abstraction*)
- Sentencias de control (*Control Statements*)
- Compilación condicional (*Conditional Compilation*)
- Procesamiento de un conjunto de resultados de consulta (*Processing a Query Result Set One Row at a Time*)

# BLOQUES (BLOCKS)

PL/SQL

# Bloque de Programa

- Las unidades básicas (procedimientos, funciones y bloques anónimos) que componen un programa PL/SQL son bloques lógicos que pueden anidarse unos dentro de otros.
- Un bloque agrupa declaraciones y sentencias relacionadas.
- Un bloque PL/SQL tiene tres partes básicas: una parte declarativa(DECLARE), una parte ejecutable (BEGIN .. END), y una parte de manejo de excepciones (EXCEPTION) que maneja las condiciones de error. Sólo se requiere la parte ejecutable.



# Bloque de Programa

Un bloque PL/SQL es definido por las palabras clave DECLARE, BEGIN, EXCEPTION y END que dividen el bloque en tres secciones:

1. Declarativa

   Sentencias que declaran variables, constantes y otros elementos de código, que después pueden ser usados dentro del bloque.

2. Ejecutable:

   Sentencias que se ejecutan cuando se ejecuta el bloque.

3. Manejo de excepciones

   Una sección especialmente estructurada para manejar cualquier excepción que se produzca durante la ejecución de la sección ejecutable.

# Bloque de Programa

Los bloques generalmente se clasifican de la siguiente manera:

- Bloque anónimo PL/SQL, es un bloque PL/SQL que aparece en su aplicación y no se nombra ni almacena en la base de datos. En muchas aplicaciones, los bloques PL/SQL pueden aparecer donde quiera que aparezcan sentencias SQL.
- Bloque PL/SQL (*named block*), un subprograma PL/SQL es un bloque PL/SQL que se almacena en la base de datos y se puede llamar por su nombre desde una aplicación. Cuando crea un subprograma, la base de datos analiza el subprograma y almacena su representación analizada en la base de datos.

  Puede declarar un subprograma como un procedimiento o una función.

# Bloque de Programa

```
DECLARE
/* Parte declarativa */
BEGIN
/* Parte de ejecución */
EXCEPTION
/* parte de excepciones */
WHEN OTHERS THEN
        dbms_output.put_line('Se ha producido un error' );
END;
```

# Bloque de Programa

```
SET SERVEROUTPUT ON;
DECLARE
    num_1 number;
    num_2 number;
BEGIN
    num_1:= 10;
    num_2:= 20;
    dbms_output.put_line(num_1); -- procedimiento para mostrar
    dbms_output.put_line(num_2); -- mensaje en la pantalla
END;
```

Es necesario activar SERVEROUTPUT (SET SERVEROUTPUT ON) para ver las salidas desde bloques, procedimientos o funciones almacenados.

# Bloque de Programa

## Delimitadores

- Operadores aritméticos y relacionales : +, - , *, >, <, >=, <= , …
- Delimitadores de comentarios : -- , /* */
- Delimitadores de cadenas de caracteres : ‘ ‘

## Identificadores

- Nombran objetos PL/SQL tales como: constantes, variables, excepciones, cursores, subprogramas y paquetes
- Hasta 30 caracteres
- No sensitivo a mayúsculas o minúsculas

## Literales

- Dato explícito, sea numérico, caracteres, cadena o *boolean* (TRUE o FALSE)

# Bloque de Programa

## Tipos de datos

Los tipos de datos más comunes son:

- NUMBER (numérico)
- CHAR (carácter)
- VARCHAR2 (carácter de longitud variable)
- BOOLEAN (lógico)
- DATE (fecha)
- Atributos de tipo, son:
  - %TYPE
  - %ROWTYPE

# VARIABLES Y CONSTANTES

PL/SQL

# Variables y Constantes

PL/SQL permite declarar variables y constantes para luego ser utilizadas en comandos SQL o en sentencias y expresiones PL/SQL.

- Declaración de Variables

```
DECLARE
    costo  NUMBER(5,2);
    activo BOOLEAN;
```

- Declaración de Constantes

```
DECLARE
    valor_maximo   CONSTANT INTEGER := 2;
    limite_credito CONSTANT REAL := 5000;
```

# Variables y Constantes

PL/SQL permite declarar variables y constantes para luego ser utilizadas en comandos SQL o en sentencias y expresiones PL/SQL.

- Asignación de valores a variables

  Utilizando el operador de asignación (:=)

```
costo  := precio_unitario * cantidad;
activo := FALSE;
```

  Utilizando un comando de selección SQL (SELECT)

```
SELECT monto*1.19 INTO valor_total
FROM factura
WHERE nfactura = nro_factura;
```

# ENTRADA Y SALIDA

PL/SQL

# Entrada y Salida

La mayoría de las entradas y salidas (E/S) de PL/SQL se realizan con declaraciones SQL que almacenan datos en tablas de bases de datos o consultan esas tablas. Todas las demás E/S de PL/SQL se realizan con paquetes PL/SQL que proporciona Oracle Database.

El package DBMS_OUTPUT, permite que los bloques, subprogramas, paquetes y disparadores (*triggers*) de PL/SQL muestren la salida. Especialmente útil para mostrar información de depuración de PL/SQL.

```
dbms_output.put_line(aa); -- procedimiento para mostrar mensaje
```

# SUBPROGRAMAS

PL/SQL

# Subprogramas

Los subprogramas son bloques PL/SQL con un nombre que los identifica (*named blocks*). Pueden llevar parámetros y ser invocados.

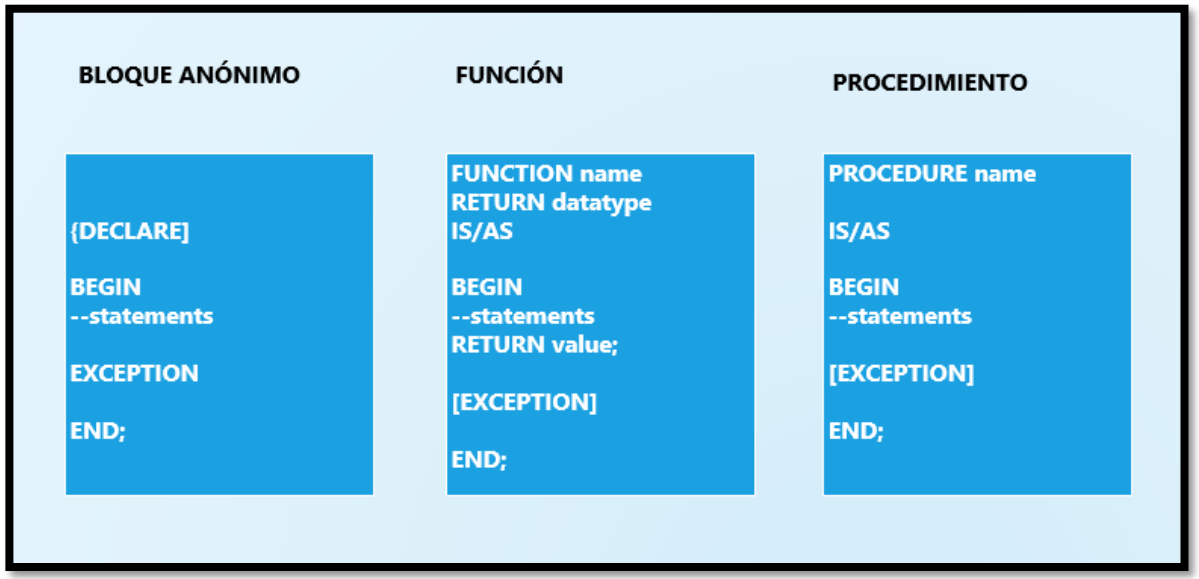
Existen dos tipos de subprogramas:

- Function (función)
- Procedure (procedimiento)

# Subprogramas

En resumen, tenemos:

- Bloque anónimo
- Bloque no anónimo, llamado subprograma que puede ser:
  - Function (función)
  - Procedure (procedimiento)



# Subprogramas

## Ventajas

1. Mayor alcance

   Permite utilizar el lenguaje PL/SQL para extender la funcionalidad de la base de datos.

2. Modularidad

   Permite separa un programa en módulos manejables y bien definidos.

3. Reutilización y mejor mantenimiento

   Un subprograma puede ser utilizado por cualquier número de aplicaciones. Por lo tanto, sólo el subprograma es afectado si las definiciones cambian.

4. Abstracción y Encapsulamiento

   Al usar un subprograma uno sólo requiere conocer qué es lo que hace y no cómo lo hace.

# Subprogramas

## Function

Una función es un subprograma que devuelve un valor. Tiene la siguiente sintaxis:

```
[CREATE  [OR REPLACE ] ]
FUNCTION function_name [ ( parameter [ , parameter ]... ) ]
RETURN datatype
  {IS | AS}
[ PRAGMA AUTONOMOUS_TRANSACTION; ]
[ local declarations ]
BEGIN
   executable statements
[ EXCEPTION
   exception handlers ]
END [ name ];
```

# Subprogramas

## Function

Una función es un subprograma que calcula un valor. Las funciones y procedimientos se estructuran de la misma manera, con la diferencia que las funciones llevan una sentencia de retorno (RETURN).

```
CREATE OR REPLACE FUNCTION sal_ok (salary REAL, title VARCHAR2)
      RETURN BOOLEAN IS
      min_sal    REAL;
      max_sal    REAL;
BEGIN
      SELECT losal, hisal INTO min_sal, max_sal
      FROM sals
      WHERE job = title;

      RETURN (salary >= min_sal AND salary <= max_sal);
END sal_ok;
```

# Subprogramas

## Function

Para invocar la función:

```
IF sal_ok(new_sal, new_title) THEN
        ……
END IF;
…
```

Sin embargo, una restricción sobre el uso de funciones: estas no pueden figurar directamente en sentencias SQL.

```
INSERT INTO bonus
VALUES (bonus_id, emp_id, sal_ok(new_sal, new_title)); -- esto es ilegal
…
```

# Subprogramas

## Function

- **Uso del RETURN**

La sentencia RETURN culmina la ejecución de la función y devuelve el control al origen de la llamada.

Es posible tener muchas sentencias RETURN dentro de una función, sin embargo, esto no es una buena práctica.

En las funciones debe haber por lo menos una sentencia RETURN, la que debe contener una expresión de manera obligatoria.

# Subprogramas

## Function

- Creando una función

```
CREATE OR REPLACE FUNCTION multiplica (n1 NUMBER, n2 NUMBER)
RETURN NUMBER
AS
BEGIN
    RETURN n1*n2;
END;
```

- Eliminando la función de la base de datos

```
DROP FUNCTION multiplica;
```

# Subprogramas

## Function

- Creando una función

```
CREATE OR REPLACE FUNCTION Obt_Sueldo (p_empno NUMBER)
RETURN NUMBER
AS
    nSal NUMBER;
BEGIN
    SELECT salary INTO nSal
    FROM employees
    WHERE empno = p_empno;
    RETURN nSal;
END;
```

# Subprogramas

## Function

- Invocar una función desde SQL

```
SQL> SELECT multiplica(10,3) FROM dual;
MULTIPLICA(10,3)
----------------
              30
```

- Invocar una función desde SQL*Plus

```
SQL> VARIABLE val NUMBER
SQL> EXEC :val := multiplica(10,3);
PL/SQL procedure successfully completed.
SQL> PRINT :val
     VAL
---------
      30
```

# Subprogramas

## Function

- Invocar una función desde PL/SQL

```
DECLARE
    nVal NUMBER;
BEGIN
    nVal:= multiplica(10,4);
    dbms_output.put_line('Valor es' || nVal);
END;
```

# Conclusiones

En esta sesión, debe haber comprendido lo siguiente:

- ¿Qué es PL/SQL?. Ventajas
- Sobre el uso en PL/SQL de bloques, variables y constantes, entrada y salida, y subprogramas (funciones)

# Referencias


- AR. Elmasri y S.B. Navathe. (2007). Fundamentos de Sistema de Base de Datos, 5ta edición

# ¡Gracias!

PUCP