Universidad de Playa Ancha

Ingeniería en Informática

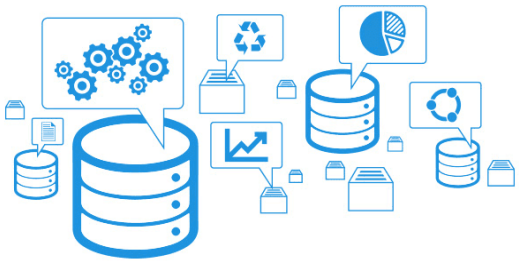
Sistemas de Bases de Datos

PL/SQL 04

Procedimientos almacenados

Docente Tatiana Ilabaca W.

Segundo semestre de 2023

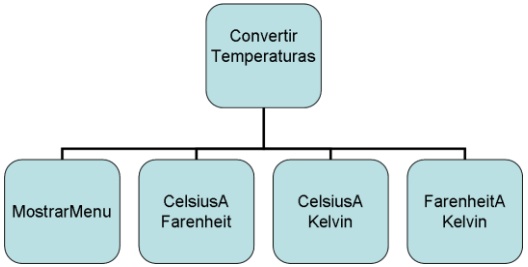


Objetivo

Comprender la importancia de la modularización a través de subprogramas

Crear e invocar Procedimientos Almacenados

Identificar y definir aspectos concernientes al uso de parámetros



Modularización

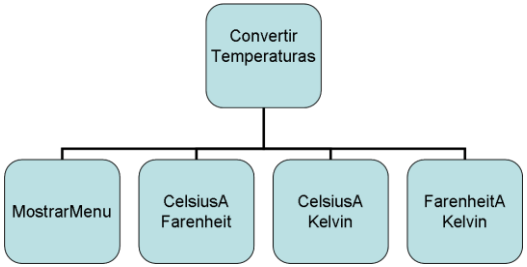
- ¿Qué es modularizar?

  Dividir o descomponer un problema en partes funcionalmente independientes que encapsulen operaciones y datos.

  Su objetivo es obtener:

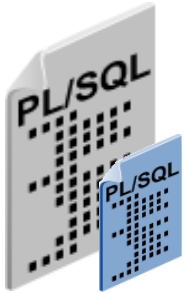
  ✓ Alta cohesión: grado de identificación de un módulo con una función concreta.

  ✓ Bajo acoplamiento: grado de interacción de los módulos que constituyen un programa.



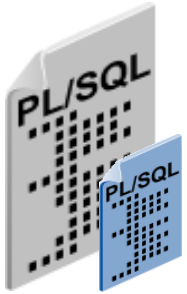
Ventajas de la modularización

- Facilita el proceso de diseño, implementación y operación
- Facilita la comprensión (cada módulo puede ser estudiado separadamente)
- Reduce el tamaño del código
- Facilita la corrección de errores y el mantenimiento (localización rápida)
- La modificación de un módulo no afecta a los demás
- Reutilización de código



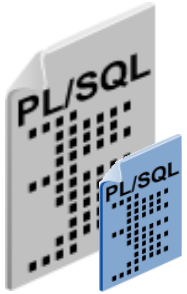
Modularización con bloques PL/SQL

- PL/SQL es un lenguaje estructurado en bloques
- Los bloques ayudan a modularizar el código usando:
  - Bloques anónimos
  - Procedimientos y funciones
  - Packages
  - Triggers
- Beneficios:
  - Fácil mantenimiento
  - Mayor seguridad e integridad de los datos
  - Mayor rendimiento
  - Mayor claridad del código



Subprograma PL/SQL

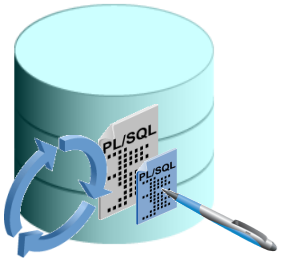
- Un subprograma PL/SQL es un bloque PL/SQL con nombre que se puede llamar con un conjunto de parámetros
- Se compone de una especificación y de un cuerpo
- Puede ser un procedimiento o una función
- Normalmente, se utiliza un procedimiento para realizar una acción y una función para calcular y devolver un valor



Subprograma PL/SQL

Diferencias entre bloques anónimos y subprogramas

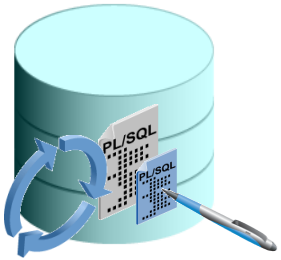
Bloques anónimos	Subprogramas
Bloque PL/SQL sin nombre	Bloques PL/SQL con nombre
Compilado cada vez	Compilado solo una vez
No está almacenado en la base de datos	Almacenado en la base de datos
No puede ser invocado por otras aplicaciones	Tiene nombre, por lo tanto, puede ser invocado por otras aplicaciones
No retorna valores	Las funciones pueden retornar valores
No utilizan parámetros	Utilizan parámetros



Procedures

PROCEDIMIENTOS

- Son un tipo de subprograma PL/SQL que ejecuta una acción
- Son compilados y almacenados en la base de datos como un objeto del esquema
- Promueve la reutilización y la mantenibilidad
- A diferencia de las funciones, los procedimientos no retornan valores, sólo pueden hacerlo a través de parámetros.



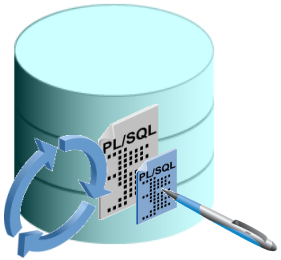
Procedures

PROCEDIMIENTOS
Creación

- Usar la sentencia SQL CREATE para crear un procedimiento
- Usar la opción OR REPLACE para sobreescribir un procedimiento existente

```
CREATE [OR REPLACE] PROCEDURE procedure_name
 [(parameter1 [mode] datatype1,
   parameter2 [mode] datatype2, ...)]
IS|AS
  [local_variable_declarations; ...]
BEGIN
      -- actions;
END [procedure_name];
```

PL/SQL block

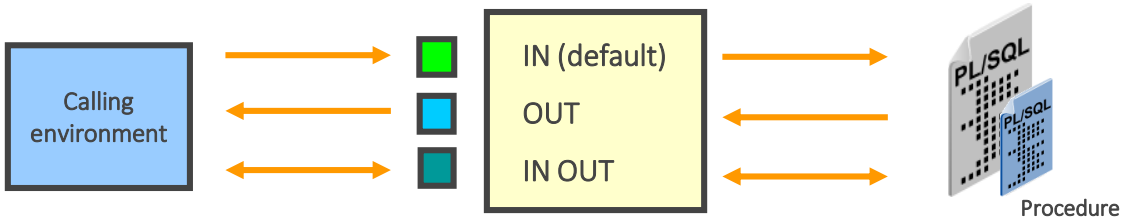


Procedures

PROCEDIMIENTOS
Parámetros

- Se declaran después del nombre del procedimiento, en el encabezado
- Se utilizan como variables locales pero dependen de su modo de paso de parámetros:
  - IN (predeterminado): proporciona valores para que el procedimiento procese
  - OUT devuelve un valor al llamador
  - IN OUT: proporciona un valor de entrada que puede devolverse (salida) con un valor modificado

PROCEDIMIENTOS
Parámetros



IN	OUT	IN OUT
Modo por defecto	Debe ser especificado	Debe ser especificado
Valor es pasado al subprograma	Valor retornado al ambiente de llamada	Valor pasado al subprograma; retornado al ambiente de llamada
Parámetro formal actua como una constante	Variable no inicializada	Variable no inicializada
Parámetro formal puede ser un literal, una expresión, una constante, o una variable inicializada.	Debe ser una variable	Debe ser una variable
Puede ser asignado un valor por defecto	No puede ser asignado un valor por defecto	No puede ser asignado un valor por defecto

PROCEDIMIENTOS
Modo IN

- Ejemplo: Crear una apuesta

```
CREATE OR REPLACE PROCEDURE CrearApuesta(codTJ IN SORTEO.codigoTJ%TYPE, nroST IN SORTEO.numeroST%TYPE,
                                         codAG IN AGENCIA.codigoAG%TYPE, tipAP IN APUESTA.tipoAP%TYPE )
AS
codAP   APUESTA.codigoAP%TYPE;

BEGIN
SELECT NVL(MAX(codigoAP),0) INTO codAP FROM APUESTA;

INSERT INTO APUESTA VALUES(codAP + 1, TO_DATE(SYSDATE), codTJ, nroST, NULL, 0, NULL, NULL, codAG, 0, tipAP);
END;
```

```
EXECUTE CrearApuesta(100, 2659, 23575, 1);
```

PROCEDIMIENTOS
Modo IN

- Ejemplo: Crear una apuesta (invocación 2)

PROCEDIMIENTOS
Modo OUT

- Ejemplo: Consultar el monto del premio de una apuesta

```
CREATE OR REPLACE PROCEDURE ConsultarPremio(codAP IN APUESTA.codigoAP%TYPE,
                                            premio OUT APUESTA.montoPremioAP%TYPE)
AS
BEGIN
SELECT montoPremioAP INTO premio FROM APUESTA WHERE codigoAP=codAP;
END;
```

PROCEDIMIENTOS
Modo OUT

- Ejemplo: Consultar el monto del premio de una apuesta (invocación)

```
SET SERVEROUTPUT ON
DECLARE
codAP   APUESTA.codigoAP%TYPE;
existe  NUMBER(1,0);
premio  APUESTA.montoPremioAP%TYPE;

APUESTA_NO_EXISTE EXCEPTION;
BEGIN
codAP := &CODIGO_APUESTA;

SELECT COUNT(codigoAP) INTO existe FROM APUESTA WHERE codigoAP = codAP;

IF(existe = 0)THEN
    RAISE APUESTA_NO_EXISTE;
END IF;

ConsultarPremio(codAP,premio);

DBMS_OUTPUT.PUT_LINE('Monto del premio de la apuesta '||codAP||': '||premio);

EXCEPTION
WHEN APUESTA_NO_EXISTE THEN DBMS_OUTPUT.PUT_LINE('La apuesta ingresada no existe');
END;
```

PROCEDIMIENTOS
Modo IN OUT

- Ejemplo: Dar formato xx.xxx.xxx-x a un rut.

```
CREATE OR REPLACE PROCEDURE formatoRut(rut IN OUT VARCHAR2)
AS
longitud    NUMBER(1,0);
BEGIN
longitud:=LENGTH(rut);
DBMS_OUTPUT.PUT_LINE('Longitud: '||longitud);
IF(longitud = 9)THEN
    rut:=SUBSTR(rut,1,2)||'.'||SUBSTR(rut,3,3)||'.'||SUBSTR(rut,6,3)||'-'||SUBSTR(rut,9,1);
ELSIF(longitud = 8)THEN
    rut:=SUBSTR(rut,1,1)||'.'||SUBSTR(rut,2,3)||'.'||SUBSTR(rut,5,3)||'-'||SUBSTR(rut,8,1);
ELSE
    rut:=SUBSTR(rut,1,3)||'.'||SUBSTR(rut,4,3)||'-'||SUBSTR(rut,7,1);
END IF;
END;
```

```
DECLARE
rut    VARCHAR2(12);
BEGIN
rut:='&RUT';
formatoRut(rut);
DBMS_OUTPUT.PUT_LINE('Rut: '||rut);
END;
```

PROCEDIMIENTOS
Sin parámetros

- Ejemplo: Procesar las apuestas de los sorteos realizados