BASES DE DATOS II

PROCEDIMIENTOS ALMACENADOS Y ESTRUCTURAS DE CONTROL

MSc. Jimena Adriana Timaná Peña

Procedimiento Almacenado

- Es un bloque PL/SQL **con *nombre*** que puede tener parámetros o argumentos y que es invocado.
- En general se utiliza un procedimiento para realizar una acción específica.
- Un procedimiento tiene un encabezado, una sección de declaración, una sección ejecutable y una sección opcional de control de excepciones.

Procedimiento Almacenado

- **Sintaxis de un procedimiento almacenado**

```
CREATE [OR REPLACE] PROCEDURE procedure_name
 [(parameter1 [mode1] datatype1,
  parameter2 [mode2] datatype2,
   . . .)]
IS|AS
PL/SQL Block;
```

	Descripción
procedure_name	Nombre del procedimiento
parameter	Variables PL/SQL cuyo valor es pasado como parámetro
mode	Tipo del argumento: **IN** (por defecto): Pasa un valor constante desde el entorno del llamado al interior del procedimiento. **OUT:** pasa un valor desde el procedimiento al entorno del llamado. **IN OUT:** pasa un valor desde el entorno del llamado al interior del procedimiento y posiblemente un valor de regreso diferente desde el procedimiento al entorno utilizando el mismo parámetro.
datatype	Tipo de dato del argumento
PL/SQL block	Cuerpo del procedimiento que define la(s) acción(es) a realizar.

Procedimiento Almacenado

- **Sintaxis de un procedimiento almacenado**

```
CREATE [OR REPLACE] PROCEDURE procedure_name
 [(parameter1 [mode1] datatype1,
  parameter2 [mode2] datatype2,
   . . .)]
IS|AS
PL/SQL Block;
```

Tenga en cuenta además:

- El bloque PL/SQL empieza con BEGIN o con la declaración de las variables locales y finaliza con END o END *procedure_name.*
- No se pueden referenciar variables host o bind en un bloque PL/SQL de un procedimiento almacenado.
- No se debe especificar el tamaño o la precisión del tipo de dato. Solo el tipo de dato.

Procedimiento Almacenado

Para Consultar Procedimientos almacenados creados

```
SELECT *
FROM User_Procedures;
```

Para borrar un procedimiento almacenado

```
DROP PROCEDURE procedure_name
```

Procedimientos Almacenados

- **Ejemplo de un Procedimiento almacenado**

```
set serveroutput on

CREATE OR REPLACE PROCEDURE saludo
IS
    v_mensaje VARCHAR2(100) := 'Hola Mundo';
BEGIN
    DBMS_OUTPUT.PUT_LINE(v_mensaje);
END saludo;
```

Para ejecutarlo:

```
execute saludo
```

Procedimientos Almacenados

Es posible llamar el procedimiento desde un bloque anónimo:

```
BEGIN
  saludo;
END;


BEGIN
  saludo();
END;
```

Procedimientos Almacenados

- **Ejemplo uso de un Procedimiento almacenado**

```
create table EMPLEADO
 (
   EMPID                INTEGER                not null,
   EMPNOMBRE       VARCHAR2(50)          not null,
   EMPAPELLIDO      VARCHAR2(50)          not null,
   EMPSALARIO       NUMBER(14,0)           not null,
   constraint PK_EMPLEADO primary key (EMPID)
);

insert into empleado values(10, 'Juan' , 'Perez', 1000000);
insert into empleado values(20, 'Andres', 'Alvarez', 1100000);
insert into empleado values(30, 'Pedro' , 'Paz', 1000000);
insert into empleado values(40, 'Lucas', 'Lopez', 1500000);
insert into empleado values(50, 'Mateo' ,'Marin', 1600000);
insert into empleado values(60, 'Pablo', 'Paz', 1000000);
```

Procedimientos Almacenados

Ejemplo 1:

Cree un procedimiento almacenado que permita actualizar en un 10% el salario de todos los empleados.

Solución:

```
CREATE OR REPLACE PROCEDURE IncrementoSalario
IS

BEGIN
  UPDATE empleado
  SET emp_salario = emp_salario*1.10;
END IncrementoSalario;
```

Ahora ejecutamos el procedimiento almacenado:

```
EXECUTE IncrementoSalario
```

Procedimientos Almacenados

Ejercicio 1:

Cree un procedimiento almacenado que permita actualizar en un % específico el salario de un empleado específico. El id y el % serán pasados como parámetro al procedimiento almacenado.

Nota: *Para los ejercicio propuesto se trabajará con la tabla EMPLEADO creada anteriormente.*

Estructuras de Control

Dos tipos de estructuras de control en PL/SQL son las estructuras:

- Condicionales → sentencias IF
- Bucles (LOOP)

Estructuras de Control

Sintaxis Sentencia IF

```
IF condition THEN
   statements;
[ELSIF condition THEN
   statements;]
[ELSE
   statements;]
END IF;
```

Hay tres formas de sentencias IF:

IF-THEN-END IF *(Condicional simple)*
IF-THEN-ELSE-END IF *(Condicional Doble)*
IF-THEN-ELSIF-END IF

Estructuras de Control

- **Sentencia IF Simple:**

```
IF condition1 THEN
  statement1;

END IF;
```

```
. . .
IF v_ename       = 'MILLER' THEN
   v_job         := 'SALESMAN';
   v_deptno      := 35;
   v_new_comm    := sal * 0.20;
END IF;
. . .
```

Estructuras de Control

- **Sentencia IF Doble:**

```
IF condition1 THEN
  statement1;
ELSE
  statement2;
END IF;
```

```
...
IF v_shipdate - v_orderdate < 5 THEN
  v_ship_flag := 'Acceptable';
ELSE
  v_ship_flag := 'Unacceptable';
END IF;
...
```

Estructuras de Control

- **Sentencia IF Anidada**

```
IF condition1 THEN
  statement1;
ELSE
  IF condition2 THEN
    statement2;
  END IF;
END IF;
```

Recordar que cada Sentencia IF anidada debe terminar con el correspondiente **END IF;**

Estructuras de Control

- **Sentencia IF THEN ELSIF:**

```
IF condition1 THEN
   statement1;
ELSIF condition2 THEN
   statement2;
ELSIF condition3 THEN
   statement3;
END IF;
```

```
. . .
IF    v_start > 100 THEN
      v_start := 2 * v_start;
ELSIF v_start >= 50 THEN
      v_start := .5 * v_start;
ELSE
      v_start := .1 * v_start;
END IF;
. . .
```

- Se sugiere su utilización en vez de los IF anidados convencionales, ya que es más fácil de leer y además elimina la necesidad de usar siempre al final END IF;

Estructuras de Control

Construyendo Expresiones Lógicas

- Recuerde que los valores nulos se manejan con el operador **IS NULL**.
- Las expresiones concatenadas con valores nulos tranta los valores nulos como a una cadena vacía.
- Tenga en cuenta siempre las tablas lógicas de comparación y el comportamiento que toman con los operadores AND, OR, NOT a la hora de construir una expresión lógica

AND	*TRUE*	*FALSE*	*NULL*
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	*TRUE*	*FALSE*	*NULL*
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT	
TRUE	FALSE
FALSE	TRUE
NULL	NULL

Estructuras de Control

Sentencias LOOP

Permiten repetir una sentencia o una secuencia de sentencias varias veces

Hay tres tipos de bucles:

Bucle básico

Bucle For

Bucle While

Estructuras de Control

Bucle básico

```
LOOP
   statement1;
   . . .
   EXIT [WHEN condition];
END LOOP;
```

donde:	*condition*	es una expresión o variable booleana (TRUE, FALSE,o NULL);

Estructuras de Control

Ejemplo Bucle básico

```
LOOP
  statement1;
  . . .
  EXIT [WHEN condition];
END LOOP;
```

```
. . .
  v_ordid     item.ordid%TYPE := 101;
  v_counter   NUMBER(2) := 1;
BEGIN
. . .
  LOOP
    INSERT INTO item(ordid, itemid)
      VALUES(v_ordid, v_counter);
    v_counter := v_counter + 1;
    EXIT WHEN v_counter > 10;
  END LOOP;
. . .
```

Estructuras de Control

Bucle For

```
FOR counter in [REVERSE]
    lower_bound..upper_bound LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

- No declare el contador ya que éste se declara implícitamente como un entero.
- **REVERSE** hace que el contador disminuya con cada iteración desde el límite superior hasta el límite inferior (Tenga en cuenta que el límite inferior todavía está referenciado en primer lugar).

Estructuras de Control

Bucle For

```
FOR counter in [REVERSE]
    lower_bound..upper_bound LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

- El límite inferior y el límite superior del rango del bucle pueden ser literales, variables o expresiones, pero se deben evaluar en enteros.

```
DECLARE
  v_lower   NUMBER := 1;
  v_upper   NUMBER := 100;
BEGIN
  FOR i IN v_lower..v_upper LOOP
  ...
  END LOOP;
END;
```

```
DECLARE
  v_ordid     item.ordid%TYPE := 601;
BEGIN
  FOR i IN 1..10 LOOP
    INSERT INTO item(ordid, itemid)
    VALUES(v_ordid, i);
  END LOOP;
END;
```

Estructuras de Control

Bucle While

```
WHILE condition LOOP
   statement1;
   statement2;
   . . .
END LOOP;
```

- En el bucle WHILE, la condición es evaluada al principio de cada iteración
- El bucle WHILE se utiliza para repetir sentencias *mientras* haya una condición TRUE

Estructuras de Control

Cree la siguiente tabla e inserte los valores que se indican:

```
CREATE TABLE EmpleadoTemp
(
   id number(4) NOT NULL,
   cargo VARCHAR2(30) NOT NULL,
   nuevo_salario INTEGER  NOT NULL,
   constraint pk_EmpleadoTemp primary key(id)
);
INSERT INTO EmpleadoTemp(id,cargo,nuevo_salario) VALUES(111,'DOCTOR', 3000);
INSERT INTO EmpleadoTemp(id,cargo,nuevo_salario) VALUES(222,'ING', 3500);
```

Ejercicios con Estructuras de Control

Ejemplo 1:

- Cree un **bloque anónimo** donde inserte en la tabla EmpleadoTemp el cargo y el salario de un empleado de la Tabla Emp **si** el cargo es PRESIDENT(el mismo tipo de dato de las variables será igual al mismo tipo de dato de las columnas). Usted deberá buscar los datos del empleado, donde el identificador sea ingresado por usted a través de una variable de sustitución.

Parte de la Solución:

```
DECLARE
  v_job emp.job%TYPE;
  v_sal emp.sal%TYPE;
BEGIN
  SELECT job,sal
  INTO v_job,v_sal
 FROM emp
 WHERE empno=&v_empno;
 IF v_job='PRESIDENT' THEN
            INSERT INTO EmpleadoTemp VALUES(v_job,v_sal);
 END IF;
END;
```

Inserte Usted el identificador del empleado

Ejercicios con Estructuras de Control

Ejercicio 1:

Cree un **bloque anónimo** donde inserte en la tabla EmpleadoTemp el cargo y el salario de un empleado. Si el cargo es PRESIDENT, inserte el cargo y el salario multiplicado por 2, sino ingrese el cargo y el salario sin ninguna modificación. Usted deberá buscar la información del empleado, donde el identificador sea ingresado por usted, a través de una variable de sustitución.