

REPUBLICA DE GUATEMALA

INTECAP

Alumno: Hugo Yoel Alexander Morales Batz.

No. De Carné: 2024-005101.

Carrera: Base de Datos Oracle.

LENGUAJE PL/SQL.

Guatemala, 19 de agosto de 2024.

INDICE

Contenido

| | |
|------------------------|-----------|
| INDICE | 2 |
| INTRODUCCION..... | 3 |
| CONCLUSION..... | 10 |
| BIBLIOGRAFIA..... | 11 |

INTRODUCCION

En el ámbito del desarrollo en PL/SQL, el manejo de colecciones y registros es fundamental para gestionar y manipular conjuntos de datos en memoria de manera eficiente. En esta práctica, exploraremos el uso de colecciones y registros a través de un bloque PL/SQL que realiza una serie de operaciones sobre datos de empleados en una base de datos.

Practica de SELECT INTO

Practica 1

- Crear un bloque PL/SQL que devuelva al salario máximo del departamento 100 y lo deje en una variable denominada salario_maximo y la visualice.

```
SET SERVEROUTPUT ON;

DECLARE
    salario_maximo NUMBER;
BEGIN
    SELECT MAX(salary)
    INTO salario_maximo
    FROM EMPLOYEES
    WHERE DEPARTMENT_ID = 100;

    DBMS_OUTPUT.PUT_LINE('El salario máximo del departamento 100 es: ' || salario_maximo);
END;
```

Salida de Script x | Tarea terminada en 0.262 segundos

PL/SQL: ORA-00942: la tabla o vista no existe
ORA-06550: línea 5, columna 4:
PL/SQL: SQL Statement ignored
06550. 00000 - "line %s, column %s:\n%s"
*Cause: Usually a PL/SQL compilation error.
*Action:
Procedimiento PL/SQL terminado correctamente.
Procedimiento PL/SQL terminado correctamente.
Procedimiento PL/SQL terminado correctamente.
El salario máximo del departamento 100 es: 12008

Procedimiento PL/SQL terminado correctamente.
El salario máximo del departamento 100 es: 12008

Practica 2

- Visualizar el tipo de trabajo del empleado número 100.

```
SET SERVEROUTPUT ON;

DECLARE
    TIPO_TRABAJO VARCHAR2(50);
BEGIN
    SELECT JOB_ID
    INTO TIPO_TRABAJO
    FROM EMPLOYEES
    WHERE EMPLOYEE_ID = 100;

    DBMS_OUTPUT.PUT_LINE('El TIPO DE TRABAJO DEL EMPLEADO 100 es: ' || TIPO_TRABAJO);
END;
```

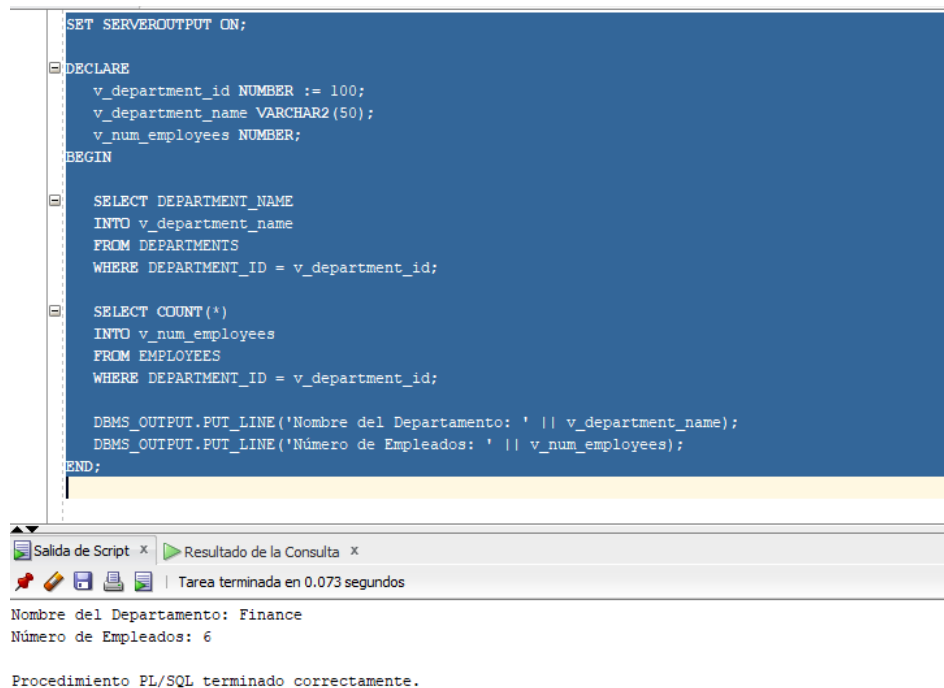
Salida de Script x | Resultado de la Consulta x | Tarea terminada en 0.043 segundos

El TIPO DE TRABAJO DEL EMPLEADO 100 es: AD_PRES

Procedimiento PL/SQL terminado correctamente.

Practica 3

- Crear una variable de tipo DEPARTMENT_ID y ponerla algún valor, por ejemplo 10.
 - Visualizar el nombre de ese departamento y el número de empleados que tiene,
- poniendo. Crear dos variables para albergar los valores.



```
SET SERVEROUTPUT ON;

DECLARE
    v_department_id NUMBER := 100;
    v_department_name VARCHAR2(50);
    v_num_employees NUMBER;
BEGIN
    SELECT DEPARTMENT_NAME
    INTO v_department_name
    FROM DEPARTMENTS
    WHERE DEPARTMENT_ID = v_department_id;

    SELECT COUNT(*)
    INTO v_num_employees
    FROM EMPLOYEES
    WHERE DEPARTMENT_ID = v_department_id;

    DBMS_OUTPUT.PUT_LINE('Nombre del Departamento: ' || v_department_name);
    DBMS_OUTPUT.PUT_LINE('Número de Empleados: ' || v_num_employees);
END;
```

Salida de Script x Resultado de la Consulta x

Tarea terminada en 0.073 segundos

Nombre del Departamento: Finance
Número de Empleados: 6

Procedimiento PL/SQL terminado correctamente.

Prácticas con INSERT, UPDATE y DELETE.

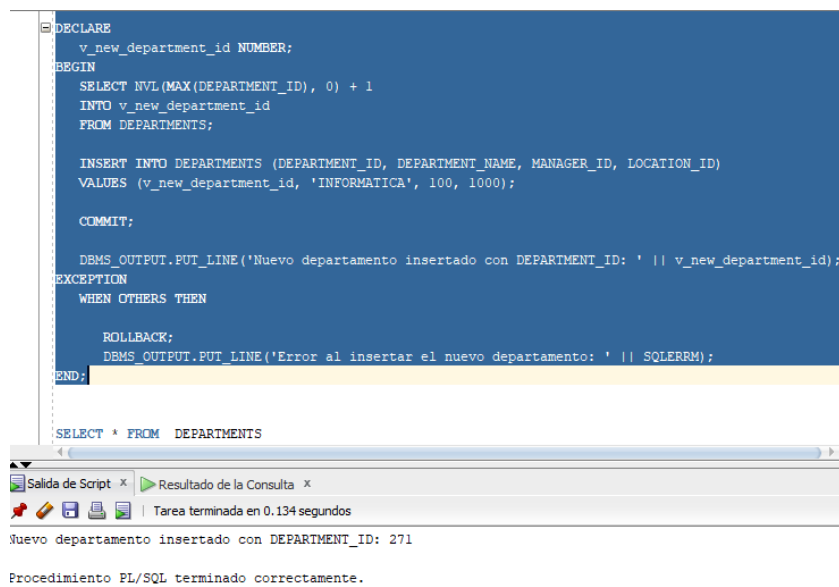
Práctica 1

- Crear un bloque que inserte un nuevo departamento en la tabla DEPARTMENTS. Para saber el DEPARTMENT_ID que debemos asignar al nuevo departamento primero
- debemos averiguar el valor mayor que hay en la tabla DEPARTMENTS y sumarle uno
- para la nueva clave.

- Location_id debe ser 1000
- Manager_id debe ser 100
- Department_name debe ser "INFORMATICA"
- NOTA: en PL/SQL debemos usar COMMIT y ROLLBACK de la misma forma que lo

hacemos en SQL. Por tanto, para validar definitivamente un cambio debemos usar

COMMIT.



```

DECLARE
    v_new_department_id NUMBER;
BEGIN
    SELECT NVL(MAX(DEPARTMENT_ID), 0) + 1
    INTO v_new_department_id
    FROM DEPARTMENTS;

    INSERT INTO DEPARTMENTS (DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, LOCATION_ID)
    VALUES (v_new_department_id, 'INFORMATICA', 100, 1000);

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Nuevo departamento insertado con DEPARTMENT_ID: ' || v_new_department_id);
EXCEPTION
    WHEN OTHERS THEN

        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error al insertar el nuevo departamento: ' || SQLERRM);
END;

SELECT * FROM DEPARTMENTS
  
```

Salida de Script x Resultado de la Consulta x

Tarea terminada en 0.134 segundos

Nuevo departamento insertado con DEPARTMENT_ID: 271

Procedimiento PL/SQL terminado correctamente.

Práctica 2

- Crear un bloque PL/SQL que modifique la LOCATION_ID del nuevo departamento a 1700. En este caso usemos el COMMIT dentro del bloque PL/SQL.

```
SET SERVEROUTPUT ON;
DECLARE
    v_department_id NUMBER;
BEGIN

    SELECT DEPARTMENT_ID
    INTO v_department_id
    FROM DEPARTMENTS
    WHERE DEPARTMENT_NAME = 'INFORMATICA';

    UPDATE DEPARTMENTS
    SET LOCATION_ID = 1700
    WHERE DEPARTMENT_ID = v_department_id;

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('El LOCATION_ID del departamento con DEPARTMENT_ID ' ||
    v_department_id || ' ha sido actualizado a 1700.');
```

EXCEPTION

```
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No se encontró un departamento con el nombre INFORMATICA.');
```

WHEN OTHERS THEN

```
        DBMS_OUTPUT.PUT_LINE('Error al actualizar el LOCATION_ID: ' || SQLERRM);
END;
```

Salida de Script x Resultado de la Consulta x

Tarea terminada en 0.08 segundos

El LOCATION_ID del departamento con DEPARTMENT_ID 271 ha sido actualizado a 1700.

Procedimiento PL/SQL terminado correctamente.

Prácticas de Colecciones y Records

- Creamos un TYPE RECORD que tenga las siguientes columnas
NAME_VARCHAR2(100),
SALEMPLOYEES.SALRY%TYPE,
COD_DEPTEMPLOYEES.DEPARTMENT_ID%TYPE;
- Creamos un TYPE TABLE basado en el RECORD anterior
- Mediante un bucle cargamos en la colección los empleados. El campo NAME debe contener FIRST_NAME y LAST_NAME concatenado.
- Para cargar las filas y siguiendo un ejemplo parecido que hemos visto en el vídeo usamos el EMPLOYEE_ID que va de 100 a 206
- A partir de este momento y ya con la colección cargada, hacemos las siguientes operaciones, usando métodos de la colección.
 - Visualizamos toda la colección
 - Visualizamos el primer empleado

- Visualizamos el último empleado
- Visualizamos el número de empleados
- Borramos los empleados que ganan menos de 7000 y visualizamos de nuevo la colección
- Volvemos a visualizar el número de empleados para ver cuantos se han borrado

```

SET SERVEROUTPUT ON;
DECLARE
    -- Definir el tipo RECORD
    TYPE emp_record_type IS RECORD (
        NAME VARCHAR2(100),
        SALARY NUMBER, -- Tipo de datos para el salario
        DEPARTMENT_ID NUMBER -- Tipo de datos para el DEPARTMENT_ID
    );

    -- Definir el tipo TABLE basado en el RECORD
    TYPE emp_table_type IS TABLE OF emp_record_type INDEX BY PLS_INTEGER;

    -- Declaración de la colección basada en el TYPE TABLE
    emp_table emp_table_type;

    -- Variable para almacenar cada registro temporalmente
    emp_record emp_record_type;

    -- Cursor para seleccionar empleados
    CURSOR emp_cursor IS
        SELECT FIRST_NAME || ' ' || LAST_NAME AS NAME,
               SALARY,
               DEPARTMENT_ID
        FROM EMPLOYEES
        WHERE EMPLOYEE_ID BETWEEN 100 AND 206;
BEGIN

```



```

-- Cargar la colección con los empleados
FOR emp IN emp_cursor LOOP
    emp_record.NAME := emp.NAME;
    emp_record.SALARY := emp.SALARY;
    emp_record.DEPARTMENT_ID := emp.DEPARTMENT_ID;

    -- Agregar el registro a la colección
    emp_table(emp_table.COUNT + 1) := emp_record;
END LOOP;

-- Visualizar toda la colección
DBMS_OUTPUT.PUT_LINE('Toda la colección:');
FOR i IN 1..emp_table.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(emp_table(i).NAME || ' - ' || emp_table(i).SALARY || ' - ' || emp_table(i).DEPARTMENT_ID);
END LOOP;

-- Visualizar el primer empleado
DBMS_OUTPUT.PUT_LINE('Primer empleado: ' || emp_table(1).NAME || ' - ' || emp_table(1).SALARY || ' - ' || emp_table(1).DEPARTMENT_ID);

-- Visualizar el último empleado
DBMS_OUTPUT.PUT_LINE('Último empleado: ' || emp_table(emp_table.COUNT).NAME || ' - ' || emp_table(emp_table.COUNT).SALARY || ' - ' || emp_table(emp_table.COUNT).DEPARTMENT_ID);

-- Visualizar el número de empleados
DBMS_OUTPUT.PUT_LINE('Número de empleados: ' || emp_table.COUNT);

-- Borrar empleados que ganan menos de 7000
FOR i IN REVERSE 1..emp_table.COUNT LOOP
    IF emp_table(i).SALARY < 7000 THEN
        emp_table.DELETE(i);
    END IF;
END LOOP;

```

```

-- Borrar empleados que ganan menos de 7000
FOR i IN REVERSE 1..emp_table.COUNT LOOP
    IF emp_table(i).SALARY < 7000 THEN
        emp_table.DELETE(i);
    END IF;
END LOOP;

-- Visualizar la colección después de borrar empleados
DBMS_OUTPUT.PUT_LINE('Colección después de borrar empleados que ganan menos de 7000:');
FOR i IN 1..emp_table.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(emp_table(i).NAME || ' - ' || emp_table(i).SALARY || ' - ' || emp_table(i).DEPARTMENT_ID);
END LOOP;

-- Visualizar el número de empleados después de borrar
DBMS_OUTPUT.PUT_LINE('Número de empleados después de borrar: ' || emp_table.COUNT);
END;

```

Salida de Script x

Tarea terminada en 0.166 segundos

```

Pat Fay - 6000 - 20
Susan Mavris - 6500 - 40
Hermann Baer - 10000 - 70
Shelley Higgins - 12008 - 110
William Gietz - 8300 - 110
Primer empleado: Steven King - 24000 - 90
Último empleado: William Gietz - 8300 - 110
Número de empleados: 107
Colección después de borrar empleados que ganan menos de 7000:
Steven King - 24000 - 90
Neena Kochhar - 17000 - 90
Lex De Haan - 17000 - 90
Alexander Hunold - 9000 - 60

```

CONCLUSION

Este ejercicio no solo refuerza el conocimiento sobre la definición y uso de tipos RECORD y TABLE en PL/SQL, sino que también proporciona una práctica integral en la manipulación de colecciones de datos en memoria, lo que es crucial para el desarrollo de aplicaciones PL/SQL más avanzadas y eficientes.

BIBLIOGRAFIA

Ayudas de instrucción dadas por el instructor.