Select into-----1 fila
Cursores-------Muchas filas

Ejemplo

Borra los datos de un empleado cuyo nombre y apellido los escriba un usuario

```
begin
  delete from employees_temp where first_name like '&nombre' and last_name like ‘&apellido’;
  if sql%notfound then
     dbms_output.put_line('Empleado no encontrado');
  else
     dbms_output.put_line('Empleado borrado');
  end if;
end;
```

Ejercicio: Repite el ejemplo incrementando un 5% el salario de un empleado cuyo nombre y apellido escriba el usuario.

```
declare
nombre employees_temp.first_name%type :='&nombre';
apellido employees_temp.last_name%type :='&apellido';
begin
  update employees_temp
    set salary = salary*1.05 where first_name like nombre and last_name like apellido;
  if sql%notfound then
     dbms_output.put_line('Empleado '||nombre||’ ‘||apellido||' no encontrado');
  else
     dbms_output.put_line('El salario de '||nombre||' ‘||apellido||’ se ha modificado');
  end if;
end;
```

Checking SQL%ROWCOUNT After an UPDATE

```
BEGIN
UPDATE employees_temp SET salary = salary * 1.05 WHERE salary < 5000;
DBMS_OUTPUT.PUT_LINE('Updated ' || SQL%ROWCOUNT || ' salaries.');
END;
```

Realiza un ejemplo borrando todos aquellos departamentos que no tengan empleados. Muestra un mensaje indicando cuántos departamentos han sido borrados.

```
BEGIN
DELETE FROM DEPT_TEMP
WHERE DEPARTMENT_ID NOT IN (SELECT NVL(DEPARTMENT_ID,0)
                    FROM EMPLOYEES_TEMP);
 DBMS_OUTPUT.PUT_LINE('Se han borrado '||SQL%ROWCOUNT||' DEPARTAMENTOS');
END;
```

```
begin
  delete from employees_temp where first_name like '&nombre' and last_name like ‘&apellido’;
  if sql%notfound then
     dbms_output.put_line('Empleado no encontrado');
  else
     dbms_output.put_line('Empleado borrado');
```

```
  end if;
end;
```

Selecting At Most One Row: SELECT INTO Statement

If you expect a query to only return one row, you can write a regular SQL SELECT statement with an additional INTO clause specifying the PL/SQL variable to hold the result.

If you just want to check whether a condition exists in your data, you might be able to code the query with the COUNT(*) operator, which always returns a number and never raises the NO_DATA_FOUND exception.

Ejemplos

- **Muestra en pantalla el salario de un empleado cuyo nombre y apellido escriba el usuario. Asegúrate de que no se eleva una excepción.**

```
DECLARE

NOMBRE EMPLOYEES_TEMP.FIRST_NAME%TYPE:='&NOMBRE';
APELLIDO EMPLOYEES_TEMP.LAST_NAME%TYPE:='&APELLIDO';
SALARIO EMPLOYEES_TEMP.SALARY%TYPE;
CUENTA NUMBER(1);

BEGIN

 SELECT COUNT(*) INTO CUENTA
 FROM EMPLOYEES_TEMP
 WHERE FIRST_NAME=NOMBRE AND LAST_NAME=APELLIDO;
 IF CUENTA = 1 THEN
   SELECT SALARY INTO SALARIO
   FROM EMPLOYEES_TEMP
   WHERE FIRST_NAME=NOMBRE AND LAST_NAME=APELLIDO;
   DBMS_OUTPUT.PUT_LINE('El salario de '||NOMBRE||' '||APELLIDO||' es '||SALARIO);
 ELSE
   DBMS_OUTPUT.PUT_LINE('El empleado '||NOMBRE||' '||APELLIDO||' no existe');
 END IF;

END;
```

- **Muestra en pantalla el salario y la comisión de un empleado cuyo nombre y apellido escriba el usuario. Asegúrate de que no se eleva una excepción.**

  (Introducir %type)´

```
DECLARE

NOMBRE EMPLOYEES_TEMP.FIRST_NAME%TYPE:='&NOMBRE';
APELLIDO EMPLOYEES_TEMP.LAST_NAME%TYPE:='&APELLIDO';
SALARIO EMPLOYEES_TEMP.SALARY%TYPE;
COMISION EMPLOYEES_TEMP.COMMISSION_PCT%TYPE;
```

```
CUENTA NUMBER(1);

BEGIN

 SELECT COUNT(*) INTO CUENTA
 FROM EMPLOYEES_TEMP
 WHERE FIRST_NAME=NOMBRE AND LAST_NAME=APELLIDO;
 IF CUENTA = 1 THEN
   SELECT SALARY, COMMISSION_PCT INTO SALARIO,COMISION
   FROM EMPLOYEES_TEMP
   WHERE FIRST_NAME=NOMBRE AND LAST_NAME=APELLIDO;
   IF COMISION IS NULL THEN
     DBMS_OUTPUT.PUT_LINE('El salario de '||NOMBRE||' '||APELLIDO||' es '||SALARIO||' y no tiene
comisión.');
   ELSE
     DBMS_OUTPUT.PUT_LINE('El salario de '||NOMBRE||' '||APELLIDO||' es '||SALARIO||' y su comisión
es '||COMISION);
   END IF;
 ELSE
   DBMS_OUTPUT.PUT_LINE('El empleado '||NOMBRE||' '||APELLIDO||' no existe o está duplicado.');
 END IF;

END;
```

- **Muestra todos los datos de un empleado cuyo código de empleado lo escribe el usuario.**

  (Introducir %rowtype)

```
DECLARE
 CODIGO EMPLOYEES_TEMP.EMPLOYEE_ID%TYPE:=&CODIGO;
 CUENTA NUMBER(1);
 FILA EMPLOYEES_TEMP%ROWTYPE;

BEGIN
 SELECT COUNT(*) INTO CUENTA
 FROM EMPLOYEES_TEMP
 WHERE EMPLOYEE_ID=CODIGO;

 IF CUENTA=1 THEN

   SELECT * INTO FILA
   FROM EMPLOYEES_TEMP
   WHERE EMPLOYEE_ID = CODIGO;
   DBMS_OUTPUT.PUT_LINE(FILA.FIRST_NAME||' '||FILA.LAST_NAME||' '||FILA.SALARY||'
'||FILA.EMAIL||' '||FILA.JOB_ID);

 ELSE

   DBMS_OUTPUT.PUT_LINE('El empleado no existe o está duplicado.');

 END IF;

END;
```

- **Dado un empleado, si su salario está por debajo del salario medio de los empleados con el mismo empleo, debe incrementarse un 5%. Emite un mensaje que indique la operación llevada a cabo (empleado no encontrado, salario igual o superior a la media, salario incrementado)**

```
DECLARE
 NOMBRE EMPLOYEES_TEMP.FIRST_NAME%TYPE:='&NOMBRE';
 APELLIDO EMPLOYEES_TEMP.LAST_NAME%TYPE:='&APELLIDO';
 CUENTA NUMBER(1);
 SALARIO EMPLOYEES_TEMP.SALARY%TYPE;
 SALARIOMEDIO NUMBER(8,2);
BEGIN
 SELECT COUNT(*) INTO CUENTA
 FROM EMPLOYEES_TEMP
 WHERE FIRST_NAME=NOMBRE AND LAST_NAME=APELLIDO;

 IF CUENTA=1 THEN
  SELECT SALARY INTO SALARIO
  FROM EMPLOYEES_TEMP
  WHERE FIRST_NAME=NOMBRE AND LAST_NAME=APELLIDO;

  SELECT AVG(SALARY) INTO SALARIOMEDIO
  FROM EMPLOYEES_TEMP
  WHERE JOB_ID=(SELECT JOB_ID
          FROM EMPLOYEES
          WHERE FIRST_NAME=NOMBRE AND LAST_NAME=APELLIDO);

  IF SALARIO<SALARIOMEDIO THEN
   UPDATE EMPLOYEES_TEMP SET SALARY=SALARY*1.05 WHERE FIRST_NAME=NOMBRE AND
LAST_NAME=APELLIDO;
   DBMS_OUTPUT.PUT_LINE('El salario modificado de '||NOMBRE||' '||APELLIDO||' es '||SALARIO*1.05);

  ELSE
   DBMS_OUTPUT.PUT_LINE('El salario '||SALARIO||' de '||NOMBRE||' '||APELLIDO||' no ha sido
modificado. Es igual o superior a la media.');
  END IF;
 ELSE
  DBMS_OUTPUT.PUT_LINE('El empleado no existe o está duplicado.');
 END IF;
END;
```

- **Dada una región que escriba el usuario, emite 3 mensajes que indiquen el número de poblaciones (locations), departamentos y empleados de dicha región.**

```
DECLARE

REGION REGIONS.REGION_NAME%TYPE:='&REGION';
CUENTA NUMBER(1);
LOCALIDADES NUMBER(3);
DEPARTAMENTOS NUMBER(3);
EMPLEADOS NUMBER(3);

BEGIN
```

```
 SELECT COUNT(*) INTO CUENTA
 FROM REGIONS
 WHERE REGION_NAME=REGION;

 IF CUENTA=0 THEN
 DBMS_OUTPUT.PUT_LINE('La región escrita no existe.');
 ELSE

 SELECT COUNT(LOCATION_ID) INTO LOCALIDADES
 FROM LOCATIONS
 WHERE COUNTRY_ID IN(SELECT COUNTRY_ID
            FROM COUNTRIES
            WHERE REGION_ID IN(SELECT REGION_ID
                    FROM REGIONS
                    WHERE REGION_NAME=REGION));

 DBMS_OUTPUT.PUT_LINE('La región dada tiene '||LOCALIDADES||' localidades.');

 SELECT COUNT(DEPARTMENT_ID) INTO DEPARTAMENTOS
 FROM DEPT_TEMP
 WHERE DEPARTMENT_ID IN(SELECT DEPARTMENT_ID
            FROM DEPARTMENTS
            WHERE LOCATION_ID IN(SELECT LOCATION_ID
                    FROM LOCATIONS
                    WHERE COUNTRY_ID IN(SELECT COUNTRY_ID
                            FROM COUNTRIES
                            WHERE REGION_ID IN(SELECT REGION_ID
                                    FROM REGIONS
                                    WHERE REGION_NAME=REGION))));

 DBMS_OUTPUT.PUT_LINE('La región dada tiene '||DEPARTAMENTOS||' departamentos.');

 SELECT COUNT(EMPLOYEE_ID) INTO EMPLEADOS
 FROM EMPLOYEES_TEMP
 WHERE DEPARTMENT_ID IN(  SELECT DEPARTMENT_ID
              FROM DEPT_TEMP
              WHERE DEPARTMENT_ID IN(SELECT DEPARTMENT_ID
                      FROM DEPARTMENTS
                      WHERE LOCATION_ID IN(SELECT LOCATION_ID
                              FROM LOCATIONS
                             WHERE COUNTRY_ID IN(SELECT COUNTRY_ID
                                     FROM COUNTRIES
                                     WHERE REGION_ID IN(SELECT REGION_ID
                                             FROM REGIONS
                                             WHERE REGION_NAME=REGION)))));
 DBMS_OUTPUT.PUT_LINE('La región dada tiene '||EMPLEADOS||' empleados.');
 END IF;
END;
```

- **Dado el nombre de un departamento, y el salario medio de sus empleados, actualizar incrementando un 5% el de todos aquellos cuyo salario esté por debajo de esa media.**

```
DECLARE

 NOMBREDEPT DEPT_TEMP.DEPARTMENT_NAME%TYPE:='&NOMBREDEPT';
```

```
 SALARIOMEDIO EMPLOYEES_TEMP.SALARY%TYPE;

BEGIN

 SELECT AVG(SALARY) INTO SALARIOMEDIO
 FROM EMPLOYEES_TEMP JOIN DEPT_TEMP USING (DEPARTMENT_ID)
 WHERE DEPARTMENT_NAME=NOMBREDEPT;

 UPDATE EMPLOYEES_TEMP
 SET SALARY=SALARY*1.05
 WHERE DEPARTMENT_ID = (SELECT DEPARTMENT_ID
                FROM DEPARTMENTS
                WHERE DEPARTMENT_NAME=NOMBREDEPT)
 AND SALARY < SALARIOMEDIO;
 DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT||' Empleados actualizados.');

END;
```

Querying Data with PL/SQL: Implicit Cursor FOR Loop

```
BEGIN
FOR item IN
( SELECT last_name, job_id FROM employees WHERE job_id LIKE '%CLERK%'
AND manager_id > 120 )
LOOP
DBMS_OUTPUT.PUT_LINE('Name = ' || item.last_name || ', Job = ' ||
item.job_id);
END LOOP;
END;
```

- **Dado un título de libro, si tiene 1 autor mostrar su nombre, en caso de no tenerlo o tener más de 2 mostrarlo también.**

```
DECLARE
 TITULO TITLE.TITLE%TYPE:='&TITULO';
 CUENTA NUMBER(2);
 CUENTAAUTORES NUMBER(2);
 NOMBRE AUTHOR.AU_FNAME%TYPE;
 APELLIDO AUTHOR.AU_LNAME%TYPE;

BEGIN

 SELECT COUNT(*) INTO CUENTA
 FROM TITLE
 WHERE TITLE=TITULO;

 SELECT COUNT(AU_FNAME) INTO CUENTAAUTORES
 FROM AUTHOR JOIN TITLEAUTHOR USING (AU_ID) JOIN TITLE USING (TITLE_ID)
 WHERE TITLE=TITULO;

 IF CUENTA=1 THEN
```

```
   IF CUENTAAUTORES=0 THEN
    DBMS_OUTPUT.PUT_LINE('El libro '||TITULO||' no tiene autor.');
    IF CUENTAAUTORES=1 THEN

    SELECT AU_FNAME, AU_LNAME INTO NOMBRE, APELLIDO
    FROM AUTHOR JOIN TITLEAUTHOR USING (AU_ID)
            JOIN TITLE USING (TITLE_ID)
    WHERE TITLE=TITULO;
    DBMS_OUTPUT.PUT_LINE('El autor del libro '||TITULO||' es '||NOMBRE||' '||APELLIDO);
    ELSE
    DBMS_OUTPUT.PUT_LINE('El libro '||TITULO||' tiene más de 1 autor.');
    END IF;
   END IF;
  ELSE
   DBMS_OUTPUT.PUT_LINE('El titulo del libro '||TITULO||' no existe o está mal escrito.');
  END IF;

END;
```

- **Escribe un bloque que permita insertar un JOB nuevo dados su descripción, nivel mínimo y nivel máximo.**

```
DECLARE
  DESCRIPCION JOB.JOB_DESC%TYPE:='&DESCRIPCION';
  MINIMO JOB.MIN_LVL%TYPE:=&MINIMO;
  MAXIMO JOB.MAX_LVL%TYPE:=&MAXIMO;
  NEXT_ID JOB.JOB_ID%TYPE;
BEGIN
  SELECT MAX(JOB_ID) INTO NEXT_ID
  FROM JOB;

  INSERT INTO JOB
  (JOB_ID, JOB_DESC, MIN_LVL, MAX_LVL)
  VALUES (NEXT_ID+1,DESCRIPCION, MINIMO, MAXIMO);

  DBMS_OUTPUT.PUT_LINE('Se ha añadido un nuevo JOB.');
END;
```

CONVERSIÓN DE BLOQUE A PROCEDIMIENTO:

```
CREATE PROCEDURE INSERTAR_JOB (DESCRIPCION JOB.JOB_DESC%TYPE,MINIMO JOB.MIN_LVL%TYPE,MAXIMO
JOB.MAX_LVL%TYPE ) IS

  NEXT_ID JOB.JOB_ID%TYPE;
BEGIN
  SELECT MAX(JOB_ID) INTO NEXT_ID
  FROM JOB;

  INSERT INTO JOB
  (JOB_ID, JOB_DESC, MIN_LVL, MAX_LVL)
  VALUES (NEXT_ID+1,DESCRIPCION, MINIMO, MAXIMO);

  DBMS_OUTPUT.PUT_LINE('Se ha añadido un nuevo JOB.');
END;
```

EJECUCIÓN DEL PROCEDIMIENTO:

```
DECLARE
DESCRIPCION JOB.JOB_DESC%TYPE:='&DESCRIPCION';
NIVELMIN JOB.MIN_LVL%TYPE:=&NIVEL_MINIMO;
NIVELMAX JOB.MAX_LVL%TYPE:=&NIVEL_MAXIMO;
BEGIN
 INSERTAR_JOB(DESCRIPCION, NIVELMIN,NIVELMAX);
END;
```

- Crear un procedimiento incremento de ventas (YTD_SALE en la tabla TITLE) que dado un título de libro y un valor de ventas, incrementen las ventas de ese libro.

```
CREATE OR REPLACE PROCEDURE INCREMENTO_VENTAS (LIBRO TITLE.TITLE%TYPE,
                        VENTAS TITLE.YTD_SALE%TYPE) IS
 CUENTA NUMBER(1);

BEGIN

 SELECT COUNT(*) INTO CONTADOR
 FROM TITLE
 WHERE TITLE=TITULO;

 IF CUENTA=1 THEN
  UPDATE TITLE
  SET YTD_SALE=YTD_SALE+VENTAS
  WHERE TITLE=LIBRO;

  DBMS_OUTPUT.PUT_LINE(TITULO||' Actualizado.');

 ELSE
  DBMS_OUTPUT.PUT_LINE('El titulo de libro no existe.');
 END IF;
END;
```

PARA PROBAR ESTE PROCEDIMIENTO:

```
BEGIN
 INCREMENTO_VENTAS('&LIBRO', &VENTAS);
END;
```

O BIEN(NO SQL):

```
EXECUTE INCREMENTO_VENTAS('&LIBRO', &VENTAS);
```

- **Escribe un procedimiento de inserción en la tabla CITY que reciba como parametros: Nombre de ciudad y nombre de pais.**

- Emite un listado con los empleos de los trabajadores del departamento ‘Shipping’

Querying Data with PL/SQL: Explicit Cursor FOR Loops

```
DECLARE
CURSOR c1 IS SELECT last_name, job_id FROM employees
```

```
WHERE job_id LIKE '%CLERK%' AND manager_id > 120;
BEGIN
FOR item IN c1
LOOP
DBMS_OUTPUT.PUT_LINE('Name = ' || item.last_name || ', Job = ' ||
item.job_id);
END LOOP;
END;
```

- Emite un listado con el nombre y apellido de los empleados cuyo salario está por debajo del salario medio de los empleados con el mismo empleo

Defining Aliases for Expression Values in a Cursor FOR Loop

```
BEGIN
FOR item IN
( SELECT first_name || ' ' || last_name AS full_name,
salary * 10 AS dream_salary FROM employees WHERE ROWNUM <= 5 )
LOOP
DBMS_OUTPUT.PUT_LINE(item.full_name || ' dreams of making ' ||
item.dream_salary);
END LOOP;
END;
```

Ejercicios adicionales con base de datos Sakila

Writing Maintainable PL/SQL Queries

```
DECLARE
CURSOR c1 (job VARCHAR2, max_wage NUMBER) IS
SELECT * FROM employees WHERE job_id = job AND salary > max_wage;
BEGIN
FOR person IN c1('CLERK', 3000)
LOOP
-- process data record
DBMS_OUTPUT.PUT_LINE('Name = ' || person.last_name || ', salary = ' ||
person.salary || ', Job Id = ' || person.job_id );
END LOOP;
END;
```

```
DECLARE
emp_job employees.job_id%TYPE := 'ST_CLERK';
emp_salary employees.salary%TYPE := 3000;
my_record employees%ROWTYPE;
CURSOR c1 (job VARCHAR2, max_wage NUMBER) IS
SELECT * FROM employees WHERE job_id = job and salary > max_wage;
BEGIN
-- Any of the following statements opens the cursor:
-- OPEN c1('ST_CLERK', 3000); OPEN c1('ST_CLERK', emp_salary);
-- OPEN c1(emp_job, 3000); OPEN c1(emp_job, emp_salary);
OPEN c1(emp_job, emp_salary);
LOOP
FETCH c1 INTO my_record;
EXIT WHEN c1%NOTFOUND;
-- process data record
DBMS_OUTPUT.PUT_LINE('Name = ' || my_record.last_name || ', salary = ' ||
my_record.salary || ', Job Id = ' || my_record.job_id );
END LOOP;
END;
```