Universidad
Inca Garcilaso de la Vega
Nuevos Tiempos. Nuevas Ideas
Facultad de Ingeniería de Sistemas, Cómputo y Telecomunicaciones

GESTIÓN DE BASE DE DATOS

CURSORES EN PL/SQL

Mg. Christian Almóguer Martínez.
almoguer@uigv.edu.pe

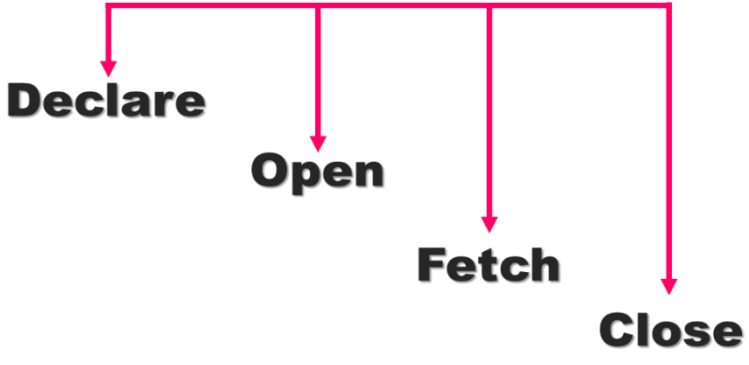
Universidad
Inca Garcilaso de la Vega
Nuevos Tiempos. Nuevas Ideas
Facultad de Ingeniería de Sistemas, Cómputo y Telecomunicaciones

Cursores en Pl/Sql

Asignatura: Gestión de Base de Datos
Semana 08
Docente: Mg. Christian Almóguer Martínez
Mail: almoguer@uigv.edu.pe

Agenda

- Antecedentes.
- Procedimientos para aplicar cursores.
- Cursores con parámetros.
- Ciclo FOR en cursores.



Antecedentes

- Dependiendo del numero de registros que retorna el SELECT, podemos dividir el problema en dos partes:
  - Si retorna un registro o
  - Si retorna más de un registro.

Retorna una fila

- El registro se podrá almacenar en tantas variables como campos consultados.
- Es decir, si escribimos un SELECT de tres campos, se retornara un registro y tres campos (matriz 1x3). Ejemplo:

```
Select c1, c2, c3 into v1, v2, v3
From  tabla(s)
Where  codigo=dato;
```

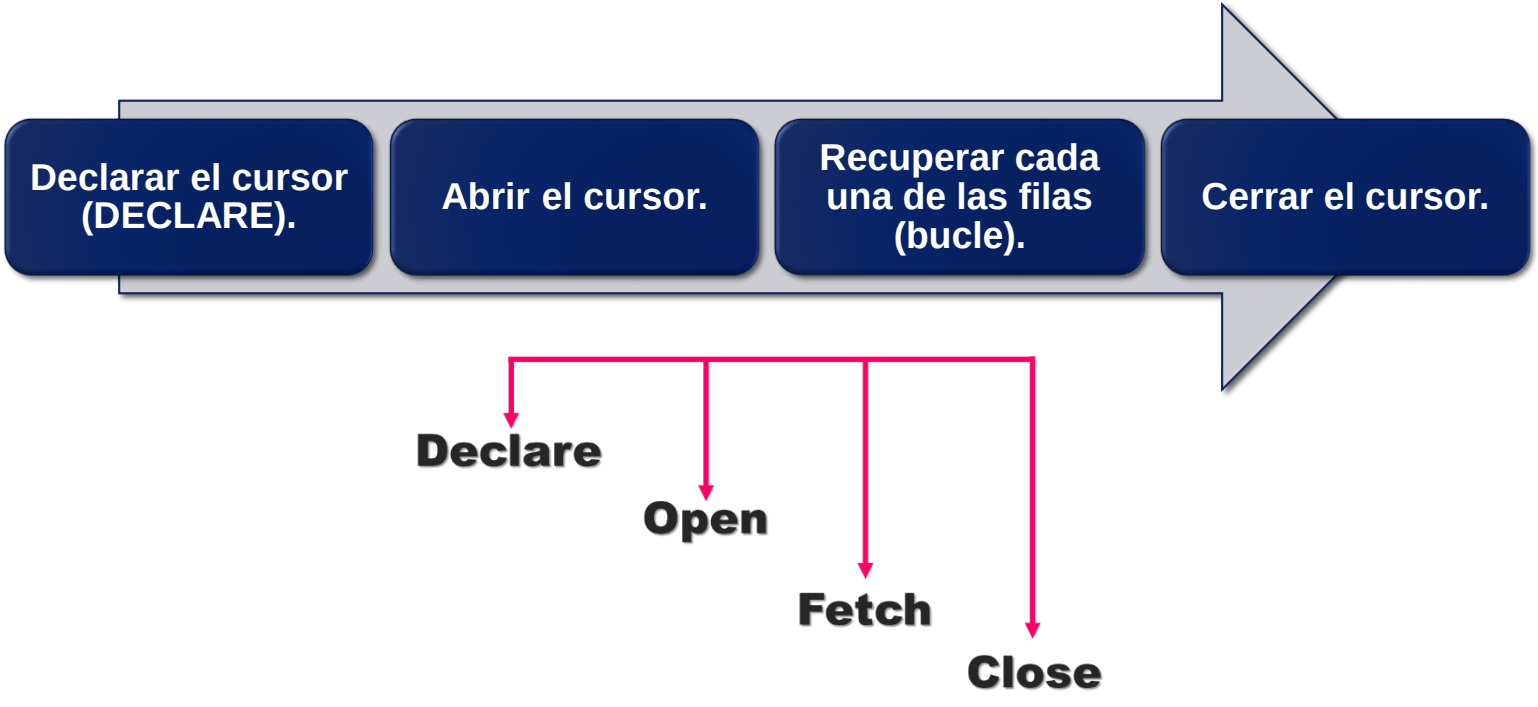
Retorna más de una fila

- En este caso no es posible almacenar directamente los valores en variables. Para ello emplearemos los cursores.

Son consultas que recuperan fila a fila y no todo el conjunto de registros a la vez.

Procedimiento para aplicar Cursores

- El procedimiento adecuado es:



1. Declarar el cursor

- En este paso se define el nombre que tendrá el cursor y qué consulta SELECT ejecutará.
- La sintaxis es:

```
Declare
    .....
    Cursor nombre_cursor  is
    select …….
```

- Una vez declarado ya podrá ser utilizado dentro del bloque de código.

2. Abrir el Cursor

- La apertura del cursor debe realizarse sólo una vez.

```
Begin
    Open nombre_cursor;
```

- Una vez que el cursor está abierto, se podrá empezar a pedir los resultado al servidor.

3. Recuperar cada Fila

- Este paso es equivalente a realizar una consulta SELECT de una sola fila.
- La sintaxis de recuperación es:

```
FETCH NOMBRE_CURSOR INTO VARIABLES;
```

- Podremos recuperar filas mientras la consulta SELECT tenga filas pendientes. Para saber cuando no hay más filas podemos emplear **los atributos**.

Atributos del Cursor

Nombre de Atributo	Descripción
Nombre_cursor**%FOUND**	**Retorna true si la ultima fila recuperada fue valida**
Nombre_cursor**%ISOPEN**	**Retorna true si el cursor esta abierto ó no.**
Nombre_cursor**%NOTFOUND**	**Retorna true si la última fila leída no tuvo éxito.**
Nombre_cursor**%ROWCOUNT**	**Retorna el numero de filas recuperadas.**

Recuperar cada Fila

- El siguiente bloque muestra la propuesta:

```
LOOP
    ……
    ……
    FETCH nombre_cursor into variables;
    ……
    Exit when Nombre_cursor%notfound;
    ……
    ……
END LOOP;
```

4. Cerrar el Cursor

- Una vez recuperado todas las filas del cursor, se debe cerrar para que se libere de la memoria del servidor los objetos temporales creados.
- En caso de no cerrar, la tabla temporal quedaría almacenado en el servidor con el nombre dado al cursor. Y esto produciría una excepción si volviéramos a ejecutar el código.

CURSOR_ALREADY_OPEN

- Para cerrar el cursor, usaremos la sintaxis:

```
.....
    CLOSE  nombre_cursor;
End;
```

Ejemplo

A través de un cursor en PL/SQL mostrar el nombre y salario de todos los empleados.

Cursor para más de una fila.

```
DECLARE
  nombre  employees.first_name%type;
  salario    employees.salary%type;
  CURSOR cur_empleado IS   --Declarar el cursor
     SELECT  first_name, salary
     FROM  employees;
BEGIN
  OPEN cur_empleado;      -- Abrir el cursor
  dbms_output.put_line('Empleado   Salario:  ');
  LOOP                              -- Recuperar cada fila
     FETCH cur_empleado INTO nombre, salario;
     EXIT WHEN cur_empleado%NOTFOUND;
     dbms_output.put_line(substr(nombre,1,15)||chr(9)||chr(9)||salario);
  END LOOP;
  CLOSE  cur_empleado;       -- Cerrar el cursor
END;
```

Ejemplo de Aplicación

Mostrar el nombre del empleado, nombre del departamento, puesto y salario de los empleados, para todos aquellos contratados en el segundo semestre de cualquier año.

Cursores con Parámetros

Ingresar el nombre del puesto y luego mostrar los nombres y salarios asociados al puesto.

Ejemplo: Cursor con parámetros

```
DECLARE
NOMBRE employees.first_name%TYPE;
SALARIO employees.salary%TYPE;
CURSOR cur_puesto (cargo employees.job_id%type)  IS
  SELECT  first_name, salary
  FROM employees
  where  job_id=upper(cargo);
BEGIN
   OPEN cur_puesto(:puesto);
  LOOP
      FETCH cur_puesto INTO  nombre, salario;
      EXIT WHEN cur_puesto%NOTFOUND;
      Dbms_output.put_line (nombre ||' - '|| salario);
   END LOOP;
   Dbms_output.put_line ('Registros mostrados : '|| cur_puesto%rowcount);
   CLOSE cur_puesto;
END;
```

EJEMPLO DE APLICACIÓN

Mostrar el nombre del empleado, nombre del departamento, fecha de contrato y localización del departamento; para todos aquellos cuyos salarios sean mayor a la cantidad solicitada (ingresar este valor por pantalla).

Ciclo FOR en Cursores

- Trabajando con el ciclo FOR podemos usar un procesamiento mucho más sencillo en relación a los ciclos del cursor.
- Esto permitirán abrir, recuperar los datos y cerrar el cursor de forma automática.

Estructura Normal del For

```
FOR <variable> IN <min>..<max>  LOOP
    ..........
    ..........
END LOOP;
```

Estructura del For con Cursor

```
FOR <variable> IN nombre_cursor  LOOP
 ..............
 ..............
END LOOP;
```

Ejemplo: Ciclo For en Cursores

```
Declare
  n number:=0;
Cursor cur_empleado IS
   SELECT * FROM employees;
Begin
Dbms_output.put_line('Nombre      Apellido        Salario ');
For  i IN cur_empleado LOOP
    Dbms_output.put_line(substr(i.first_name,1,6)||chr(9)||chr(9)||
    substr(i.last_name,1,7)||chr(9)||chr(9)||i.salary);
    n:=n+1;
End loop ;
  Dbms_output.put_line (‘Total de Registros: '|| n);
End;
```

Ejemplo:

Ingresar el código del departamentos y luego mostrar el nombre del empleado, fecha de contrato y salario. Finalmente mostrar la suma total de los salarios (Emplear estructura FOR).