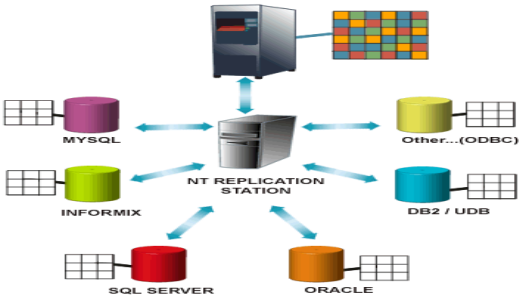
UNIVERSIDAD TECNOLOGICA DE PANAMA

FACULTAD DE INGENIERIA DE SISTEMAS

LICENCIATURA EN DESARROLLO SOFTWARE

BASE DE DATOS II
ORACLE PROGRAMACION PL/SQL

CURSORES –PL/SQL ORACLE



1

CONTENIDO

Capítulo IV. Procedimientos

- *Fundamentos de Lenguaje PL/SQL*
- *Cursores*
- *Procedimientos*

2

4.2. CURSORES

Que es un cursor?

Para poder procesar una orden SQL, Oracle asigna un área de memoria que se conoce como *área de contexto*. Esta área contiene la información necesaria para el procesamiento, incluyendo el numero de filas procesadas por la orden, un puntero a la versión analizada de la orden y en el caso de las consultas, el *conjunto activo*, que es el conjunto de filas resultado de la consulta.

Un *cursor* es un puntero al área de contexto. Mediante un cursor, un programa PL/SQL puede controlar el área de contexto y lo que en ella suceda a medida que se procesa la orden.

3

4.2. CURSORES

```
DECLARE
 /* Variable de salida para almacenar los resultados de la Consulta */
    v_StudentID              students.id%TYPE;
    v_FirstName              students.first_name%TYPE;
    v_ LastName              students.last_name%TYPE;

 /* Valores de acoplamiento utilizado  en la consulta */
    v_ Major                 students.major%TYPE  := 'Computer Science';

 /* Declaración del Curso  */
    CURSOR    c_Students  IS
              SELECT  is, first_name, last_name
              FROM students
              WHERE  major = v_Major;
BEGIN
 /* Identificar las filas en el conjunto activo y preparar el procesamiento ulterior de los datos */
    OPEN  c_Students ;
    LOOP
       /* Recuperar cada fila del conjunto active  y almacenarlos en las variables      PL/SQL */
       FETCH c_Students INTO  v_StudentID, v_FirstName, v_ LastName;

       /* SI no hay más filas que recuperar, salir del  bucle */
       EXIT WHEN  c_Students%NOTFOUND;
    END LOOP;
    /* Liberar los recursos usados para la consulta */
    CLOSE c_Students;
END;
```

El bloque mostrado ilustra el concepto de cursor explicito, donde se asigna explícitamente el nombre del cursor a una orden SELECT, mediante la orden CURSOR.. IS.

El resto de las ordenes SQL se utiliza un cursor implícito. *Estos cursores implícitos, por su parte son procesados automáticamente por PL/SQL.*

4

4.2. CURSORES

Procesamiento de Cursores Explicito

Los cuatro pasos PL/SQL necesario para el procesamiento de un cursor explicito son:

- *Declaración de Cursor*
- *Apertura del cursor para una consulta.*
- *Recogida de los resultados en variables PL/SQL*
- *Cierre del Cursor*

Declaración del cursor:

La declaración de un cursor define su nombre y asocia el cursor con una orden SELECT. La sintaxis es:

CURSOR *nombre_cursor* **IS** *order_SELECT;*

Donde *nombre_cursor* es el nombre del cursor y *orden_SELECT* es la consulta que el cursor procesará. Los nombres de cursores tienen las mismas reglas de ámbito y visibilidad a los identificadores PL/SQL

5

4.2. CURSORES

Procesamiento de Cursores Explicito

Declaración del cursor:

La declaración de un cursor define su nombre y asocia el cursor con una orden SELECT. La sintaxis es:

```
/* Declaración correcta de un curso */
DECLARE
  v_Departement          classes.department%TYPE;
  v_ Course              classes.course%TYPE;
   CURSOR  c_Classes   IS
     SELECT * from  classes
      WHERE  department  =  v_Department
       AND     course  =  v_Course;

/* Declaración incorrecta de un curso */
DECLARE
  CURSOR  c_Classes   IS
     SELECT * from  classes
      WHERE  department  =  v_Department
       AND     course  =  v_Course;
  v_Departement          classes.department%TYPE;
  v_ Course              classes.course%TYPE;
```

Una declaración de cursor puede hacer referencia a variables PL/SQL en la clausula WHERE.

Estas variables se consideran variables acopladas, deben ser visibles en el punto donde se declara el cursor, como la forma presentada.

Si embargo una declaración con la planteada en el segundo ejemplo seria ilegal.

Es recomendable que las variables de referencias en una declaración de cursor sean declaradas antes de la referencia y los cursores al final.

6

4.2. CURSORES

Procesamiento de Cursores Explicito

Apertura del cursor:

La sintaxis para abrir un cursor es:

OPEN *nombre_cursor;*

Donde *nombre_cursor* representa el nombre del cursor previamente declarado.

Cuando se abre el curso suelen ocurrir tres cosas:

- **Se examinan los valores de las variables acopladas**
- **Se determina el conjunto activo, basándose en los valores de dichas variables**
- **Se hace apuntar el puntero del conjunto activo a la primera fila.**

Las variables acopladas se examinan al momento de abrir el cursor y solo en el ese momento. Cualquier cambio que se de a estas variables en el proceso no tendrá efecto alguno sobre la consulta.

7

4.2. CURSORES

Procesamiento de Cursores Explicito

Apertura del cursor:

Las variables acopladas se examinan al momento de abrir el cursor y solo en el ese momento. Cualquier cambio que se de a estas variables en el proceso, no tendrá efecto alguno sobre la consulta.

```
DECLARE
  v_RoomID          classes.room_id%TYPE;
  v_Building        rooms.buliding%TYPE;
  v_Department      classes.department%TYPE;
  v_Course          classes.course%TYPE;
  CURSOR c_Building IS
     SELECT building
       FROM rooms, classes
       WHERE rooms.room_id  = classes.room_id
        AND  department  = v_Department
        AND  course  =  v_Course;
BEGIN
   -- Asignar las variables de Acoplamiento antes de abril el cursor
   v_Department  := 'HIS';
   v_Course  _=  101;
   -- Abril el Cursor
   OPEN c_Building;
   -- Reasignar las variables de acoplamiento – No tienen efecto alguno, ya que el cursor esta abierto
   v_Department :=  'XXX';
   v_Course  := -1;
END;
```

Aunque las variables de acoplamiento cambien después de la orden **OPEN,** el conjunto activo de la consulta no cambia. A este hecho se le conoce con el nombre de ***consistencia de lectura*** y se diseño así para asegurar la integridad de los datos.

Es legal abrir un cursor que ya esta abierto. PL/SQL ejecutara implícitamente una orden **CLOSE** antes de re-abril el cursor con la segundo orden **OPEN**. También puede haber mas de un cursor abierto al mismo tiempo

8

4.2. CURSORES

Procesamiento de Cursores Explicito

Extracción de los datos del cursor:

La clausula INTO de la consulta es parte de la orden FETCH. Dicha orden tiene dos forma posibles,

FETCH *nombre_cursor* **INTO** *lista_variables;*

Y

FETCH *nombre_cursor* **INTO** *registro_PL/SQL;*

Donde *nombre_cursor* representa el nombre del cursor previamente declarado y abierto, lista_variables es una lista de variables PL/SQL previamente declaradas y separadas por comas. registro_PL/SQL es un registro PL/SQL previamente declarado.

En ambos casos la variable o variables de la clausula INTO debe ser compatible en cuanto tipo con la lista de selección de la consulta.

Dada la declaración precedente del cursor c_Building, la siguiente orden FETCH sería valida:

FECTH c_Building **INTO** v_Building;

4.2. CURSORES

Procesamiento de Cursores Explicito

Extracción de los datos del cursor:

En el siguiente bloque se proporcionan ejemplos de órdenes FETCH legales e ilegales:

```
DECLARE
   v_Department          classes.department%TYPE;
   v_Course              classes.course%TYPE;
   CURSOR  c_AllClasses IS
   SELECT  *
   FROM  classes;
   v_ClassesRecord    c_AllClasses%ROWTYPE;
BEGIN
    OPEN  c_AllClasses;
   /* Esta es una orden FECH correcta, que almacena la primera fila en el registro PL/SQL con una estructura iguala a la lista la selección de la consulta */
    FETCH  c_AllClasses  INTO  v_ClassesRecord;
   /* Esta orden FETCH es incorrecta, ya que la lista de la selección de la consulta devuelve 7 columnas de la tabla classes, y solo estamos almacenando en dos 2 variables: Estos dará un error de asignación de valores E-PLS-394 */
    FETCH  c_AllClasses  INTO  v_Department, v_Course;
END;
```

Después de cada **FETCH** , se incrementa el puntero del conjunto activo, para que apunte a la siguiente fila.

De esta forma, cada **FETCH** devolverá filas sucesivas del conjunto activo, hasta que devuelva el conjunto completo.

El atributo **%NOTFOUND** se utiliza para determinar cuando se ha terminado de extraer todo el conjunto activo. Es uno de los atributos de los cursores

4.2. CURSORES

Procesamiento de Cursores Explicito

Cierre de un cursor:

Cuando se ha terminado de extraer el conjunto activo, debe cerrarse el cursor. Esta acción informa a PL/SQL de que el programa ha terminado de usar el cursor y que se pueden liberar los recursos con él asociados. Estos recursos incluyen el área de almacenamiento empleada para contener el conjunto activo, así como cualquier espacio temporal usado en la determinación de dicho conjunto.

CLOSE *nombre_cursor ;*

Donde *nombre_cursor* representa el nombre del cursor previamente declarado y abierto.

Una vez e cierra el cursor, es ilegal realizar extracciones de el. Si se intentara hacerlo, se produce el error oracle: **ORA-1001** or **ORA-1002,** cursor invalido, Fetch fuera de secuencia.

Dada la declaración precedente del cursor c_Building, la siguiente orden CLOSE sería valida:

CLOSE c_Building;

11

4.2. CURSORES

Procesamiento de Cursores Explicito

Atributos de los cursores:

Existen cuatro atributos que pueden ser aplicados a los cursores. Estos atributos se añaden, en un bloque PL/SQL al nombre del curso, de forma similar a %TYPE y %ROWTYPE, sin embargo la diferencia es que los atributos del cursor no devuelven un tipo, sino un valor que puede emplearse como parte de una expresión. Estos atributos son:

- **%FOUND: es un atributo booleano. Devuelve TRUE si la ultima orden FETCH devolvió una fila y FALSE en caso contrario. Si el cursor no esta abierto y tratamos de comprobar el valor de %FOUND mandara un error ORA-1001.**
- **%NOTFOUND: se comporta de forma opuesta a %FOUND; si la extracción anterior devuelve una fila , entonces %NOTFOUND tiene el valor FALSE. %NOTFOUND devuelve el valor TRUE solo si la extracción anterior no devuelve una fila. Este atributo se utiliza a menudo como condición de salida para un bucle de extracción. Si el cursor no esta abierto y tratamos de comprobar el valor de %NOTFOUND mandara un error ORA-1001**

12

4.2. CURSORES

Procesamiento de Cursores Explicito

Atributos de los cursores:

Existen cuatro atributos que pueden ser aplicados a los cursores. Estos atributos se añaden, en un bloque PL/SQL al nombre del curso, de forma similar a %TYPE y %ROWTYPE, sin embargo la diferencia es que los atributos del cursor no devuelven un tipo, sino un valor que puede emplearse como parte de una expresión. Estos atributos son: continuación….

- **%ISOPEN: este atributo booleano se utiliza para determinar si el cursor asociado esta o no abierto. Si la esta, %ISOPEN devuelve TRUE; si no devuelve FALSE.**
- **%ROWCOUNT: este atributo numérico devuelve el numero de filas extraídas por el cursor hasta el momento. Si el cursor no esta abierto y tratamos de comprobar el valor de %ROWCOUNT mandara un error ORA-1001**

13

4.2. CURSORES

Procesamiento de Cursores Explicito

Cursores Parametrizados:

Existen otras formas de emplear las variables de acoplamiento en un cursor. Existe un tipo de cursor, el cursor parametrizado, admite argumentos de la misma forma que los procedimientos.

Veamos la siguiente declaración de cursor:

```
DECLARE
    v_Department        classes.department%TYPE;
    v_Course            classes.course%TYPE;
    CURSOR   c_Classes  IS
      SELECT  *  FROM  classes
      WHERE   department  = v_Department
       AND    course    = v_Course;
```

```
DECLARE
CURSOR   c_Classes  (v_Department   classes.department%TYPE,
                     v_Course   classes.course%TYPE)  IS
      SELECT  *  FROM  classes
      WHERE   department  = v_Department
       AND    course    = v_Course;
```

- **c_Classes** tienen dos variables de acoplamiento, **v_Department y v_Course**.
- El cursor c_Clases podría ser modificado para obtener un cursor parametrizado.
- En el caso de los cursores parametrizados, se utiliza la orden **OPEN** para pasar os valores reales al cursor. Y se podría abrir con:

```
OPEN   c_Classes   ('HIS' ,  101);
```


14

4.2. CURSORES

Procesamiento de Cursores Implícito

Los cursores implícitos sirven para procesar ordenes SELECT que devuelven una fila. Todas las ordenes SQL se ejecutan dentro del área de contexto por lo tanto, tienen un cursor que apunta a dicha área. Este cursor se conoce con el nombre de *cursor SQL*.

A diferencia de los cursores explícitos, el programa no abre ni cierre el cursor SQL, sino que PL/SQL lo abre de forma implícita, procesa la orden SQL en el contenida y cierra el cursor después.

Los cursores implícitos sirven para procesar la ordenes INSERT, UPDATE, DELETE, y las ordenes SELECT .. INTO de una sola fila. A este tipo de cursores se le pueden aplicar los atributos del cursor explícito.

El siguiente bloque ejecutara una orden INSERT si la orden UPDATE no encuentra ninguna fila coincidente:

```
BEGIN
   UPDATE rooms
     SET number_seats = 100
     WHERE room_id = 99980;
  -- Si la anterior orden UPDATE no se aplica a ninguna fila, inserta una nueva fila en la tabla rooms.
   IF SQL%NOTFOUND THEN
      INSERT INTO rooms ( room_id, number_seats)
      VALUES (99980, 100);
   END IF;
END;
```


4.2. CURSORES

Procesamiento de Cursores Implícito

*El siguiente bloque ejecutara una orden INSERT si la orden UPDATE no encuentra ninguna fila coincidente y podríamos hacer el mismo proceso anterior si usamos el atributo **SQL%ROWCOUNT***

```
BEGIN
   UPDATE rooms
     SET number_seats = 100
     WHERE room_id = 99980;
  -- Si la anterior orden UPDATE no se aplica a ninguna fila, inserta una nueva fila en la tabla rooms.
   IF SQL%ROWCOUNT = 0 THEN
     INSERT INTO rooms ( room_id, number_seats)
       VALUES (99980, 100);
     END IF;
END;
```

Aunque se puede emplear SQL%NOTFOUND con las ordenes SELECT .. INTO no resulta útil, por la orden SELECT .. INTO produce el error ORA-1403 : no data found **y cuando esto ocurre la orden se pasa de forma inmediata al manejo de excepciones.**

4.2. CURSORES

Procesamiento de Cursores Implícito

Aunque se puede emplear SQL%NOTFOUND con las ordenes SELECT .. INTO no resulta útil, por que la orden SELECT .. INTO produce el error ORA-1403 : no data found **y cuando esto ocurre la orden se pasa de forma inmediata al manejo de excepciones.**

Veamos el siguiente ejemplo:

```
DECLARE
  -- Registro para almacenar la información acerca de una clase.
  v_RoomData    students%ROWTYPE;
BEGIN
   -- Extraer la información sobre la clase  ID -1
   SELECT  *  INTO  v_RoomData
   FROM  students
   WHERE  id  = -1;
   /* La siguiente orden no se ejecutará nunca, ya que el control pasa inmediatamente al gestor de  excepciones */
    IF  SQL%NOTFOUND THEN
        INSERT INTO temp_table ( char_col) VALUES ( 'Not Found');
   END IF;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
         INSERT INTO temp_table ( num_col, char_col) VALUES ( 5, 'Not Found,  Excpetion Handler');
END;
```


4.2. CURSORES

Bucles de Extracción Mediante Cursor

La operación mas común que se puede realizar con un cursor consiste en extraer todos las filas de un conjunto activo. Para esto usamos un *bucle de extracción*, que no es mas que un bucle que procesa una a una las filas del conjunto activo. Existen diferentes tipos de bucles de extracción mediante cursores.

Bucle Simple

Se usa la sintaxis de bucles simples conocidos (LOOP .. END LOOP) para el procesamiento, controlándose el numero de veces que se ejecuta el bucle mediante atributos explícitos del cursor.

4.2. CURSORES

Bucle Simple: Ejemplo1 de extracción usando este medio

```
DECLARE
  /* Declaración de variables para almacenar información acerca de los estudiantes que cursan la especialidad de Historia */
  v_StudentID      students.id%TYPE;
  v_FirstName      students.first_name%TYPE;
  v_LastName       students.las_name%TYPE;
-- Cursor para recuperar la informacion sobre los estudiantes de Historia
  CURSOR c_HistoryStudents IS
    SELECT id, first_name, las_name
    FROM students
    WHERE major = 'History';
BEGIN
     -- Abre el cursor e inicializa el conjunto activo
  OPEN c_HistoryStudents;
  LOOP
     -- Recupera la información del siguiente estudiante
    FETCH c_HistoryStudents INTO v_StudentID,   v_FirstName,   v_LastName  ;
    -- Salida del bucle cuando no hay más filas por recuperar
EXIT WHEN  c_HistoryStudents%NOTFOUND;
/* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla registered_students.
Registra también el nombre y el apellido en la tabla temp_table */
      INSERT INTO registered_students ( students_id, deparment, course)
           VALUES ( v_StudentID, 'HIS', 301);

      INSERT INTO temp_table ( num_col, char_col)
           VALUES ( v_studentID,  v_FirstName || ' '|| v_LastName);
  END LOOP;
   -- Libera los recursos utilizados por el curso
  CLOSE c_HistoryStudents;
-- Confirmamos el trabajo
---COMMIT;
END;
```

Para este bloque la orden **EXIT WHEN,** esta inmediatamente después de la orden **FETCH**. Después de extraer la ultima fila, c_HistoryStudents%NOTFOUND toma el valor de **TRUE** y sale del bucle.

Se estructuro de esa forma la orden **EXIT WHEN** antes del procesamiento de los datos, con el objeto de asegurar que no se procesen filas duplicadas

19

4.2. CURSORES

Bucle Simple: Ejemplo2 de extracción usando este medio

```
DECLARE
  /* Declaración de variables para almacenar información acerca de los estudiantes que cursan la especialidad de Historia */
  v_StudentID      students.id%TYPE;
  v_FirstName      students.first_name%TYPE;
  v_LastName       students.last_name%TYPE;
-- Cursor para recuperar la información sobre los estudiantes de Historia
  CURSOR c_HistoryStudents IS
    SELECT id, first_name, last_name
    FROM students
    WHERE major = 'History';
BEGIN
     -- Abre el cursor e inicializa el conjunto activo
  OPEN c_HistoryStudents;
  LOOP
     -- Recupera la información del siguiente estudiante
    FETCH c_HistoryStudents INTO v_StudentID,   v_FirstName,   v_LastName  ;
    /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla registered_students.
Registra también el nombre y el apellido en la tabla temp_table */
      INSERT INTO registered_students ( students_id, department, course)
           VALUES ( v_StudentID, 'HIS', 301);

      INSERT INTO temp_table ( numcol, char_col)
           VALUES ( v_studentID,  v_FirstName || ' '|| v_LastName);
    -- Salida del bucle cuando no hay más filas por recuperar
    EXIT WHEN  c_HistoryStudents%NOTFOUND;
  END LOOP;
   -- Libera los recursos utilizados por el curso
  CLOSE c_HistoryStudents;
-- Confirmamos el trabajo
COMMIT;
END;
```

La ultima orden **FETCH** no modificara v_StudentID, v_FirstName, ni v_LastName, puesto que ya no hay fila en el conjunto activo.

Las variables de salida mantendrán los valores correspondientes a la ultima fila extraída. Pero como la comprobación se realiza después el procesamiento de estos valores duplicados se insertan en las tablas registered_students y temp_table, lo que no deberá ser correcto.

20

4.2. CURSORES

Bucle WHILE

También se puede construir un bucle de extracción mediante un cursor usando la sintaxis WHILE .. LOOP

```
DECLARE
   -- Cursor para recuperar la información sobre los estudiantes de Historia
   CURSOR c_HistoryStudents IS
     SELECT id, first_name, last_name
     FROM students
     WHERE major = 'History';
   -- Declaración el registro para almacenar información extraída
   v_StudentData    c_HistoryStudents%ROWTYPE;
BEGIN
     -- Abre el cursor e inicializa el conjunto activo
   OPEN c_HistoryStudents;
     -- Recupera la información del siguiente estudiante
     FETCH c_HistoryStudents INTO v_StudentData;
     -- El bucle continua mientras haya mas filas que extraer
     WHILE c_HistoryStudents%FOUND LOOP
          /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla
registered_students. Registra también el nombre y el apellido en la tabla temp_table */
               INSERT INTO registered_students ( students_id, department, course)
                    VALUES ( v_StudentData.ID, 'HIS', 301);

               INSERT INTO temp_table ( numcol, char_col)
                    VALUES ( v_StudentData.ID, v_StudentData.first_name || ' '|| v_StudentData.last_name);
               -- Recuperar la fila siguiente. La condición %FOUND se comprobara antes de que el bucle continúe
               FETCH c_HistoryStudents INTO v_StudentData;
     END LOOP;
   -- Libera los recursos utilizados por el curso
   CLOSE c_HistoryStudents;
-- Confirmamos el trabajo
COMMIT;
END;
```

Este ejemplo de extracción se comporta de la misma forma que el ejemplo **LOOP .. END LOOP** de programas anteriores. En este caso del **FETCH** aparece dos veces en el bloque, una antes del bucle y la otra después del procesamiento incluido en este.

Esto es necesario para que la condición del bucle (c_HistoryStudents%FOUND) sea evaluada en cada iteración del bucle.

4.2. CURSORES

Bucle de cursor FOR

Los dos tipos de extracción descritos requieren un procesamiento explicito del cursor, mediante ordenes OPEN, FETCH y CLOSE. PL/SQL proporciona un tipo de bucle mas simple, que realiza de modo implícito el procesamiento del cursor. EL bucle del cursor FOR

```
DECLARE
   -- Cursor para recuperar la información sobre los estudiantes de Historia
   CURSOR c_HistoryStudents IS
     SELECT id, first_name, last_name
     FROM students
     WHERE major = 'History';
BEGIN
     /* Inicio del bucle. Aquí se ejecuta una orden OPEN
     implícita sobre c_HistoryStudents */
     FOR v_StudentData IN c_HistoryStudents LOOP
     -- Aquí se ejecuta una orden FETCH implícita
          /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla
registered_students. Registra también el nombre y el apellido en la tabla temp_table */
               INSERT INTO registered_students ( students_id, department, course)
                    VALUES ( v_StudentData.ID, 'HIS', 301);

               INSERT INTO temp_table ( numcol, char_col)
                    VALUES ( v_StudentData.ID, v_StudentData.first_name || ' '|| v_StudentData.last_name);
     -- Antes de continuar con el bucle, aquí se hace una comprobacion implícita de c_HistoryStudents %NOTFOUND.
     END LOOP;
-- Ahora el bucle ha terminado, se hace cierre implícito del cursor c_HistoryStudents
-- Confirmamos el trabajo
COMMIT;
END;
```

En primer lugar, el registro v_StudentData no se declara en la sección declarativa del bloque, el compilador PL/SQL la declara de forma implícita.

En segundo lugar, el bucle abre, extrae los datos y cierra c_HistoryStudents%FOUND de manera implícita.

Los bucles de cursor FOR tienen la ventaja de proporcionar la funcionalidad de un bucle de extracción mediante cursor de una forma simple y limpia, con una sintaxis mínima.