UA4B. PL/SQL

4.15. Cursores

En este apartado vamos a ver un tipo de dato, que aunque se puede asemejar a otros que ya conozcas, su uso es exclusivo en la programación de las bases de datos y que es el cursor .
Un cursor no es más que una estructura que almacena el conjunto de filas devuelto por una consulta a la base de datos.
Oracle usa áreas de trabajo para ejecutar sentencias SQL y almacenar la información procesada. Hay 2 clases de cursores: implícitos y explícitos. PL/SQL declara implícitamente un cursor para todas las sentencias SQL de manipulación de datos, incluyendo consultas que devuelven una sola fila. Para las consultas que devuelven más de una fila, se debe declarar explícitamente un cursor para procesar las filas individualmente.

En este primer apartado vamos a hablar de los cursores implícitos y de los atributos de un cursor (estos atributos tienen sentido con los cursores explícitos, pero los introducimos aquí para ir abriendo boca), para luego pasar a ver los cursores explícitos y terminaremos hablando de los cursores variables.

Cursores implícitos

Oracle abre implícitamente un cursor para procesar cada sentencia SQL que no esté asociada con un cursor declarado explícitamente.

Con un cursor implícito no podemos usar las sentencias OPEN, FETCH y CLOSE para controlar el cursor. Pero sí podemos usar los atributos del cursor para obtener

información sobre las sentencias SQL más recientemente ejecutadas.

Atributos de un cursor

Cada cursor tiene 4 atributos que podemos usar para obtener información sobre la ejecución del mismo o sobre los datos. Estos atributos pueden ser usados en PL/SQL, pero no en SQL. Aunque estos atributos se refieren en general a cursores explícitos y tienen que ver con las operaciones que hayamos realizado con el cursor, es deseable comentarlas aquí y en el siguiente apartado tomarán pleno sentido.

- **%FOUND**: Después de que el cursor esté abierto y antes del primer FETCH, %FOUND devuelve NULL. Después del primer FETCH, %FOUND devolverá TRUE si el último FETCH ha devuelto una fila y FALSE en caso contrario. Para cursores implícitos %FOUND devuelve TRUE si un INSERT, UPDATE o DELETE afectan a una o más de una fila, o un SELECT ... INTO ... devuelve una o más filas. En otro caso %FOUND devuelve FALSE.
- **%NOTFOUND**: Es lógicamente lo contrario a %FOUND.
- **%ISOPEN**: Evalúa a TRUE si el cursor está abierto y FALSE en caso contrario. Para cursores implícitos, como Oracle los cierra automáticamente, %ISOPEN evalúa siempre a FALSE.
- **%ROWCOUNT**: Para un cursor abierto y antes del primer FETCH, %ROWCOUNT evalúa a 0. Después de cada FETCH, %ROWCOUNT es incrementado y evalúa al número de filas que hemos procesado. Para cursores implícitos %ROWCOUNT evalúa al número de filas afectadas por un INSERT, UPDATE o DELETE o el número de filas devueltas por un SELECT ... INTO ...

Cursores explícitos

Cuando una consulta devuelve múltiples filas, podemos declarar explícitamente un cursor para procesar las filas devueltas. Cuando declaramos un cursor, lo que hacemos es darle un nombre y asociarle una consulta usando la siguiente sintaxis:

```
CURSOR nombre_cursor [(parametro [, parametro] ...)] [RETURN tipo_dev
```

Donde tipo_devuelto debe representar un registro o una fila de una tabla de la base de datos, y parámetro sigue la siguiente sintaxis:

```
parametro := nombre_parametro [IN] tipo_dato [{:= | DEFAULT} expresic
```

Ejemplos:

```
CURSOR cAgentes IS SELECT * FROM agentes;
CURSOR cFamilias RETURN familias%ROWTYPE IS SELECT * FROM familias WH
```

Además, como hemos visto en la declaración, un cursor puede tomar parámetros, los cuales pueden aparecer en la consulta asociada como si fuesen constantes. Los parámetros serán de entrada, un cursor no puede devolver valores en los parámetros actuales. A un parámetro de un cursor no podemos imponerle la restricción NOT NULL.

```
CURSOR c1 (cat INTEGER DEFAULT 0) IS SELECT * FROM agentes WHERE cate
```

Cuando abrimos un cursor, lo que se hace es ejecutar la consulta asociada e identificar el conjunto resultado, que serán todas las filas que emparejen con el criterio de búsqueda de la consulta. Para abrir un cursor usamos la sintaxis:

```
OPEN nombre_cursor [(parametro [, parametro] ...)];
```

Ejemplos:

```
OPEN cAgentes;
OPEN c1(1);
OPEN c1;
```

La sentencia FETCH devuelve una fila del conjunto resultado. Después de cada FETCH, el cursor avanza a la próxima fila en el conjunto resultado.

```
FETCH cFamilias INTO mi_id, mi_nom, mi_fam, mi_ofi;
```

Para cada valor de columna devuelto por la consulta asociada al cursor, debe haber una variable que se corresponda en la lista de variables después del INTO.

Para procesar un cursor entero deberemos hacerlo por medio de un bucle.

```
BEGIN
...
OPEN cFamilias;
LOOP
FETCH cFamilias INTO mi_id, mi_nom, mi_fam, mi_ofi;
EXIT WHEN cFamilias%NOTFOUND;
...
END LOOP;
...
END;
```

Una vez procesado el cursor, deberemos cerrarlo, con lo que desabilitamos el cursor y el conjunto resultado queda indefinido.

```
CLOSE cFamilias;
```

Una vez cerrado el cursor podemos reabrirlo, pero cualquier otra operación que hagamos con el cursor cerrado lanzará la excepción **INVALID_CURSOR**.

También podemos simplificar la operación de procesamiento de un cursor, por medio de los bucles para cursores, los cuales declaran implícitamente una variable índice definida como **%ROWTYPE** para el cursor, abren el cursor, se van trayendo los valores de cada fila del cursor, almacenándolas en la variable índice, y finalmente cierran el cursor.

```
BEGIN
...
FOR cFamilias_rec IN cFamilias LOOP
--Procesamos las filas accediendo a
--cFamilias_rec.identificador, cFamilias_rec.nombre,
--cFamilias_rec.familia, ...
END LOOP;
...
END;
```