

6

Escritura de Cursores Explícitos

ORACLE®

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Objetivos

Al finalizar este curso, debería estar capacitado para hacer lo siguiente:

- **Distinguir un cursor implícito de un cursor explícito**
- **Explicar cuándo y por qué se debe utilizar un cursor explícito**
- **Utilizar una variable de registro PL/SQL**
- **Escribir un bucle FOR de cursor**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Finalidad de la Lección

En esta lección aprenderá la diferencia entre cursores implícitos y explícitos. También aprenderá cuándo y por qué se debe utilizar un cursor explícito. Puede que deba utilizar una sentencia `SELECT` de múltiples filas en PL/SQL para procesar muchas filas. Para conseguirlo, hay que declarar y controlar los cursores explícitos.

Acerca de los Cursores

Cada sentencia SQL que ejecuta Oracle Server posee un cursor individual asociado a ella:

- **Cursores implícitos:** Declarados para todas las sentencias DML y `SELECT` de PL/SQL
- **Cursores explícitos:** El programador los declara y les asigna un nombre

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Cursores Implícitos y Explícitos

Oracle Server utiliza unas áreas de trabajo, que se denominan áreas SQL privadas, para ejecutar sentencias SQL y almacenar la información del procesamiento. Se pueden utilizar cursores PL/SQL para asignar un nombre a un área SQL privada y acceder a la información que tiene almacenada.

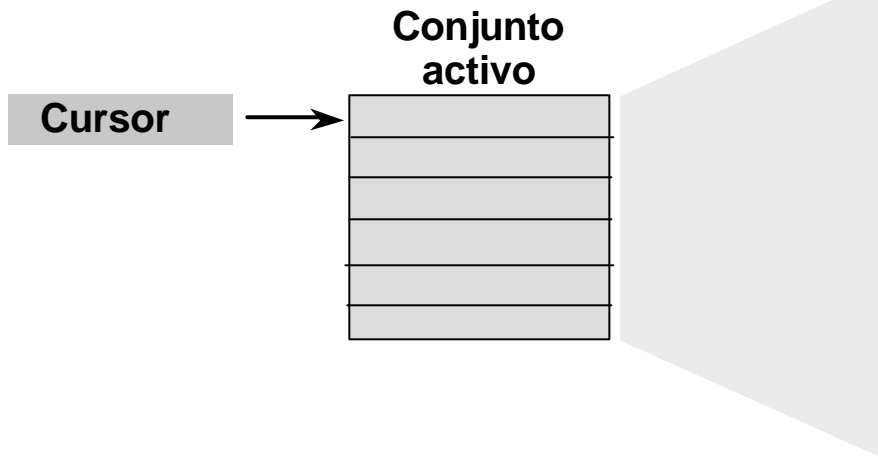
Tipo de Cursor	Descripción
Implícito	PL/SQL declara implícitamente los cursores implícitos para todas las sentencias DML y <code>SELECT</code> de PL/SQL, incluidas las consultas que sólo devuelven una fila.
Explícito	En las consultas que devuelven más de una fila, el programador declara y asigna un nombre a los cursores explícitos y se manipulan por medio de sentencias específicas en las acciones ejecutables del bloque.

Oracle Server abre implícitamente un cursor para procesar cada sentencia SQL que no está asociada a un cursor declarado explícitamente. PL/SQL le permite hacer referencia al cursor implícito más reciente como el cursor *SQL*.

Funciones de los Cursores Explícitos

Tabla

100	King	AD_PRES
101	Kochhar	AD_VP
102	De Haan	AD_VP
.	.	.
.	.	.
.	.	.
139	Seo	ST_CLERK
140	Patel	ST_CLERK
.	.	.



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Cursores Explícitos

Utilice cursores explícitos para procesar individualmente cada fila que se devuelve en una sentencia `SELECT` de múltiples filas.

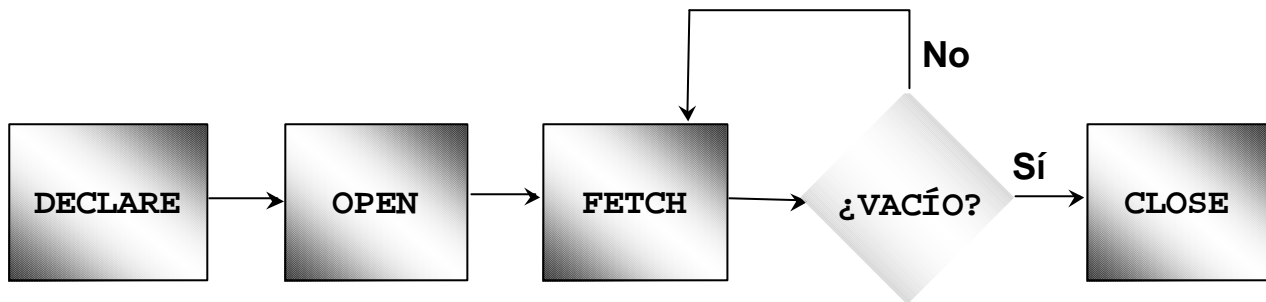
El conjunto de filas que devuelve una consulta de múltiples filas se denomina conjunto activo. Su tamaño equivale al número de filas que cumplen el criterio de búsqueda. El diagrama de la transparencia muestra cómo un cursor explícito “apunta” a la *fila actual* del conjunto activo. De esta manera, el programa puede procesar las filas de una en una.

Los programas PL/SQL abren un cursor, procesan las filas que devuelve una consulta y luego cierran el cursor. El cursor marca la posición actual en el conjunto activo.

Funciones de los cursores explícitos:

- Pueden realizar procesamientos más allá de la primera fila que devuelve la consulta, fila por fila
- Realizan un seguimiento de la fila que se está procesando en ese momento
- Permiten que el programador controle manualmente los cursores explícitos del bloque PL/SQL

Control de Cursores Explícitos



- Cree un área SQL con nombre
- Identifique el conjunto activo
- Cargue la fila actual en variables
- Compruebe si existen filas
- Vuelva a FETCH si se encuentran filas
- Libere el conjunto activo

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Cursores Explícitos (continuación)

Ahora que conoce el concepto de los cursores, revise los pasos necesarios para utilizarlos. En las siguientes páginas encontrará la sintaxis de cada paso.

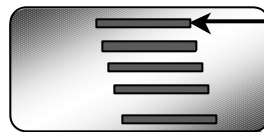
Control de Cursores Explícitos

1. Declare el cursor asignándole un nombre y definiendo la estructura de la consulta que se le va a realizar.
2. Abra el cursor. La sentencia OPEN ejecuta la consulta y enlaza cualquier variable a la que se haga referencia. Las filas que se identifican en la consulta se denominan conjunto activo y ya están disponibles para realizar recuperaciones.
3. Recupere datos desde el cursor. En el diagrama de flujo que se muestra en la transparencia, después de cada recuperación hay que probar el cursor para comprobar si hay alguna fila. Si no hay más filas que procesar, debe cerrar el cursor.
4. Cierre el cursor. La sentencia CLOSE libera el conjunto de filas activo. Ahora ya se puede volver a abrir el cursor para establecer un nuevo conjunto activo.

Control de Cursores Explícitos

1. Abra el cursor
2. Recupere una fila
3. Cierre el cursor

1. Abra el cursor.



Puntero del
cursor

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Cursores Explícitos (continuación)

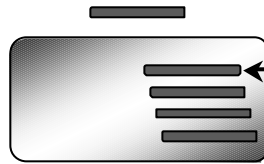
Para controlar un cursor, utilice las sentencias OPEN, FETCH y CLOSE.

La sentencia OPEN ejecuta la consulta asociada al cursor, identifica el conjunto de resultados y coloca el cursor delante de la primera fila.

Control de Cursores Explícitos

1. Abra el cursor
2. Recupere una fila
3. Cierre el cursor

2. Recupere una fila utilizando el cursor.



Puntero del
cursor

Continúe hasta que esté vacío.

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

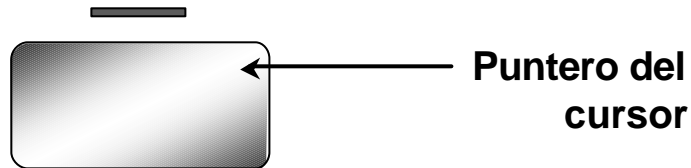
Cursores Explícitos (continuación)

La sentencia `FETCH` recupera la fila actual y avanza hasta la siguiente fila hasta que ya no quedan más filas o hasta que se cumple la condición especificada.

Control de Cursores Explícitos

1. Abra el cursor
2. Recupere una fila
3. Cierre el cursor

3. Cierre el cursor.



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Cursores Explícitos (continuación)

Cierre el cursor cuando se haya procesado la última fila. La sentencia CLOSE desactiva el cursor.

Declaración del Cursor

Sintaxis:

```
CURSOR nombre_cursor IS  
    sentencia_select;
```

- No incluya la cláusula INTO en la declaración del cursor.
- Si es necesario procesar las filas en una secuencia específica, utilice la cláusula ORDER BY en la consulta.

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Declaración del Cursor

Utilice la sentencia CURSOR para declarar un cursor explícito. Puede hacer referencia a variables en la consulta, pero debe declararlas antes que la sentencia CURSOR.

En la sintaxis:

<i>nombre_cursor</i>	es un identificador PL/SQL.
<i>sentencia_select</i>	es una sentencia SELECT sin una cláusula INTO.

Nota

- No incluya la cláusula INTO en la declaración del cursor ya que aparece después en la sentencia FETCH.
- El cursor puede ser cualquier sentencia SELECT ANSI válida, para incluir uniones, etc.

Declaración del Cursor

Ejemplo:

```
DECLARE
  CURSOR emp_cursor IS
    SELECT employee_id, last_name
    FROM   employees;

  CURSOR dept_cursor IS
    SELECT *
    FROM   departments
    WHERE  location_id = 170;
BEGIN
  ...
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Declaración del Cursor (continuación)

En el ejemplo de la transparencia, el cursor `emp_cursor` se declara para recuperar las columnas `EMPLOYEE_ID` y `LAST_NAME` de la tabla `EMPLOYEES`. Igualmente, se declara el cursor `DEPT_CURSOR` para recuperar todos los detalles del departamento `LOCATION_ID 170`.

```
DECLARE
  v_empno    employees.employee_id%TYPE;
  v_ename    employees.last_name%TYPE;
  CURSOR emp_cursor IS
    SELECT employee_id, last_name
    FROM   employees;
BEGIN
  ...
```

En esta lección, se explicará más adelante cómo recuperar los valores que ha obtenido el cursor en las variables declaradas en la sección `DECLARE`.

Apertura del Cursor

Sintaxis:

```
OPEN nombre_cursor;
```

- **Abra el cursor para ejecutar la consulta e identifique el conjunto activo.**
- **Si la consulta no devuelve ninguna fila, no se produce ninguna excepción.**
- **Utilice los atributos del cursor para probar el resultado después de realizar una recuperación.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Sentencia OPEN

La sentencia OPEN ejecuta la consulta asociada al cursor, identifica el conjunto de resultados y coloca el cursor delante de la primera fila.

En la sintaxis:

nombre_cursor es el nombre del cursor declarado previamente.

OPEN es una sentencia ejecutable que realiza las siguientes operaciones:

1. Asigna dinámicamente memoria para un área contextual que contendrá información crucial del procesamiento.
2. Analiza la sentencia SELECT.
3. Enlaza las variables de entrada: define el valor de las variables de entrada obteniendo sus direcciones en la memoria.
4. Identifica el conjunto activo: el conjunto de filas que cumple el criterio de búsqueda. Las filas del conjunto activo no se recuperan en variables cuando se ejecuta la sentencia OPEN. En su lugar, la sentencia FETCH recupera las filas.
5. Coloca el puntero justo delante de la primera fila del conjunto activo.

En el caso de los cursores que se han declarado utilizando la cláusula FOR UPDATE, la sentencia OPEN también bloquea esas filas. En una lección posterior explicaremos la cláusula FOR UPDATE.

Nota: Si la consulta no devuelve ninguna fila cuando el cursor está abierto, PL/SQL no produce ninguna excepción. Sin embargo, puede probar el estado del cursor después de cada recuperación utilizando el atributo de cursor SQL%ROWCOUNT.

Recuperación de Datos desde el Cursor

Sintaxis:

```
FETCH nombre_cursor INTO [variable1, variable2,  
...]  
/ nombre_registro];
```

- **Recupere los valores de fila actuales en variables.**
- **Incluya el mismo número de variables.**
- **Haga coincidir cada variable para que sus posiciones se correspondan con las de las columnas.**
- **Compruebe si el cursor contiene filas.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Sentencia **FETCH**

La sentencia **FETCH** recupera las filas del conjunto activo de una en una. Después de cada recuperación, el cursor avanza hasta la siguiente fila del conjunto activo.

En la sintaxis:

<i>nombre_cursor</i>	es el nombre del cursor declarado previamente.
<i>variable</i>	es una variable de salida para almacenar los resultados.
<i>nombre_registro</i>	es el nombre del registro en el cual se almacenan los datos recuperados. (La variable del registro se puede declarar utilizando el atributo %ROWTYPE.)

Instrucciones:

- Incluya tantas variables en la cláusula **INTO** de la sentencia **FETCH** como columnas en la sentencia **SELECT**, y asegúrese de que los tipos de datos son compatibles.
- Haga coincidir cada variable para que sus posiciones se correspondan con las de las columnas.
- Defina también un registro para el cursor y haga referencia al registro en la cláusula **FETCH INTO**.
- Compruebe si el cursor contiene filas. Si no se recupera ningún valor, no quedan más filas para procesar en el conjunto activo y no se registra ningún error.

Nota: La sentencia **FETCH** realiza las siguientes operaciones:

1. Lee los datos de la fila actual en las variables de salida PL/SQL.
2. Avanza el puntero hasta la siguiente fila del conjunto identificado.

Recuperación de Datos desde el Cursor

Ejemplo:

```
LOOP
  FETCH emp_cursor INTO v_empno,v_ename;
  EXIT WHEN ...;
  ...
  -- Procesa los datos recuperados
  ...
END LOOP;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Sentencia **FETCH** (continuación)

La sentencia **FETCH** se utiliza para recuperar los valores de la fila actual en variables de salida. Después de la recuperación, puede manipular los datos de las variables. Por cada valor de columna que devuelve la consulta asociada al cursor, debe haber una variable correspondiente en la lista **INTO**. Además, sus tipos de datos también deben ser compatibles.

Recupere los primeros 10 empleados, uno por uno.

```
SET SERVEROUTPUT ON
DECLARE
  v_empno  employees.employee_id%TYPE;
  v_ename  employees.last_name%TYPE;
  CURSOR   emp_cursor IS
    SELECT employee_id, last_name
    FROM   employees;
BEGIN
  OPEN emp_cursor;
  FOR i IN 1..10 LOOP
    FETCH emp_cursor INTO v_empno, v_ename;
    DBMS_OUTPUT.PUT_LINE (TO_CHAR(v_empno)
      || ' ' || v_ename);
  END LOOP;
END ;
```

Cierre del Cursor

Sintaxis:

```
CLOSE nombre_cursor;
```

- **Cierre el cursor después de terminar el procesamiento de las filas.**
- **Vuelva a abrir el cursor, si es necesario.**
- **Una vez cerrado, no intente recuperar datos desde el cursor.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Sentencia CLOSE

La sentencia CLOSE desactiva el cursor, y el conjunto activo queda indefinido. Cierre el cursor después de terminar el procesamiento de la sentencia SELECT. Este paso permite volver a abrir el cursor, si es necesario. Por lo tanto, puede establecer un conjunto activo varias veces.

En la sintaxis:

nombre_cursor es el nombre del cursor declarado previamente.

No intente recuperar datos desde un cursor después de haberlo cerrado, o se producirá la excepción INVALID_CURSOR.

Nota: La sentencia CLOSE libera el área de contexto.

Aunque se puede terminar el bloque PL/SQL sin cerrar los cursores, debería tener como costumbre cerrar cualquier cursor que haya declarado explícitamente para liberar recursos.

Existe un máximo de cursores abiertos por usuario, que se determina por medio del parámetro OPEN_CURSORS del archivo de parámetros de la base de datos. OPEN_CURSORS = 50 por defecto.

```
OPEN emp_cursor
FOR i IN 1..10 LOOP
    FETCH emp_cursor INTO v_empno, v_ename;
    ...
END LOOP;
CLOSE emp_cursor;
END;
```

Atributos de los Cursores Explícitos

Obtenga información acerca del estado del cursor.

Atributo	Tipo	Descripción
%ISOPEN	Booleano	Se evalúa como TRUE si el cursor está abierto
%NOTFOUND	Booleano	Se evalúa como TRUE si la recuperación más reciente no devuelve una fila
%FOUND	Booleano	Se evalúa como TRUE si la recuperación más reciente devuelve una fila; complementa a %NOTFOUND
%ROWCOUNT	Numérico	Se evalúa como el número total de filas recuperadas hasta ese momento

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Atributos de los Cursores Explícitos

Al igual que en los cursores implícitos, existen cuatro atributos para obtener información acerca del estado del cursor. Cuando se agregan al nombre de la variable de cursor, estos atributos devuelven información muy útil acerca de la ejecución de una sentencia de manipulación de datos.

Nota: No se puede hacer directamente referencia a los atributos de cursores en las sentencias SQL.

El Atributo %ISOPEN

- Recupere filas sólo cuando el cursor esté abierto.
- Utilice el atributo de cursor %ISOPEN antes de realizar una recuperación para comprobar si el cursor está abierto.

Ejemplo:

```
IF NOT emp_cursor%ISOPEN THEN
    OPEN emp_cursor;
END IF;
LOOP
    FETCH emp_cursor...
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

El Atributo %ISOPEN

- Sólo se pueden recuperar filas cuando el cursor está abierto. Utilice el atributo de cursor %ISOPEN para determinar si el cursor está abierto.
- Recupere filas en un bucle. Utilice atributos de cursor para determinar el momento de salida del bucle.
- Utilice el atributo de cursor %ROWCOUNT para hacer lo siguiente:
 - Recuperar un número de filas exacto
 - Recuperar las filas en un bucle FOR numérico
 - Recuperar las filas en un bucle simple y determinar el momento de salida del bucle.

Nota: %ISOPEN devuelve el estado del cursor: TRUE si está abierto y FALSE si no lo está.

Control de Recuperaciones Múltiples

- **Procese varias filas desde un cursor explícito utilizando un bucle.**
- **Recupere una fila en cada iteración.**
- **Utilice atributos del cursor explícito para comprobar el éxito de cada recuperación.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Control de Recuperaciones Múltiples desde Cursores Explícitos

Para procesar varias filas desde un cursor explícito, normalmente se define un bucle para realizar una recuperación en cada iteración. Se procesan de forma eventual todas las filas del conjunto activo y cualquier recuperación sin éxito hace que el atributo `%NOTFOUND` se defina como `TRUE`. Utilice atributos del cursor explícito para comprobar el éxito de cada recuperación antes de hacer más referencias al cursor. Si omite algún criterio de salida, se producirá un bucle infinito.

Para obtener más información, consulte el capítulo “Interaction with Oracle” de *PL/SQL User’s Guide and Reference*.

Los Atributos %NOTFOUND y %ROWCOUNT

- Utilice el atributo de cursor %ROWCOUNT para recuperar un número de filas exacto.
- Utilice el atributo de cursor %NOTFOUND para determinar el momento de salida del bucle.

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Los Atributos %NOTFOUND y %ROWCOUNT

%NOTFOUND

%NOTFOUND es el opuesto lógico de %FOUND. %NOTFOUND devuelve FALSE si en la última recuperación se obtuvo una fila, o TRUE si en la última recuperación no se obtuvo ninguna fila. En el siguiente ejemplo, se utiliza %NOTFOUND para salir de un bucle cuando FETCH no devuelve una fila:

```
LOOP
    FETCH c1 INTO my_ename, my_sal, my_hiredate;
    EXIT WHEN c1%NOTFOUND;
    ...
END LOOP;
```

Antes de la primera recuperación, %NOTFOUND se evalúa como NULL. Por lo tanto, si FETCH nunca se ejecuta con éxito, nunca se saldrá del bucle. Esto es debido a que la sentencia EXIT WHEN sólo se ejecuta si la condición WHEN es verdadera. Para estar seguro, utilice en su lugar la siguiente sentencia EXIT:

```
EXIT WHEN c1%NOTFOUND OR c1%NOTFOUND IS NULL;
```

Si el cursor no está abierto, haciéndole referencia con %NOTFOUND devuelve INVALID_CURSOR.

Los Atributos %NOTFOUND y %ROWCOUNT (continuación)

%ROWCOUNT

Cuando se abre su cursor o su variable de cursor, a %ROWCOUNT se le asigna el valor cero. Antes de la primera recuperación, %ROWCOUNT devuelve 0. Después, devuelve el número de filas que se han recuperado hasta ese momento. Este número aumenta si en la última recuperación se ha devuelto una fila. En el siguiente ejemplo, se utiliza %ROWCOUNT para realizar acciones siempre que se hayan recuperado más de diez filas:

```
LOOP
    FETCH c1 INTO my_ename, my_deptno;
    IF c1%ROWCOUNT > 10 THEN
        ...
    END IF;
    ...
END LOOP;
```

Si el cursor no está abierto, haciéndole referencia con %ROWCOUNT devuelve INVALID_CURSOR.

Ejemplo

```
DECLARE
    v_empno employees.employee_id%TYPE;
    v_ename employees.last_name%TYPE;
    CURSOR emp_cursor IS
        SELECT employee_id, last_name
        FROM employees;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor INTO v_empno, v_ename;
        EXIT WHEN emp_cursor%ROWCOUNT > 10 OR
                emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE (TO_CHAR(v_empno)
                               || ' ' || v_ename);
    END LOOP;
    CLOSE emp_cursor;
END ;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Ejemplo

En el ejemplo de la transparencia, se recuperan los diez primeros empleados uno por uno.

Nota: Antes de la primera recuperación, %NOTFOUND se evalúa como NULL. Por lo tanto, si FETCH nunca se ejecuta con éxito, nunca se saldrá del bucle. Esto es debido a que la sentencia EXIT WHEN sólo se ejecuta si la condición WHEN es verdadera. Para estar seguro, utilice la siguiente sentencia EXIT:

```
EXIT WHEN emp_cursor%NOTFOUND OR emp_cursor%NOTFOUND IS NULL;
```

Si utiliza %ROWCOUNT, compruebe que no haya ninguna fila utilizando el atributo %NOTFOUND, porque el contador de filas no aumenta si en la recuperación no se obtiene ninguna fila.

Cursores y Registros

Procese las filas del conjunto activo recuperando valores en RECORD de PL/SQL.

```
DECLARE
  CURSOR emp_cursor IS
    SELECT  employee_id, last_name
    FROM    employees;
  emp_record  emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO emp_record;
    ...
```

emp_record employee_id	last_name
---------------------------	-----------

100	King
-----	------

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Cursores y Registros

Ya ha comprobado que puede definir registros que tienen una estructura de columnas de una tabla. También puede definir un registro basándose en la lista de columnas seleccionada en un cursor explícito. Esto es conveniente para procesar las filas del conjunto activo, porque puede limitarse a realizar recuperaciones en el registro. Por lo tanto, los valores de la fila se cargan directamente en los campos correspondientes del registro.

Ejemplo

Utilice un cursor para recuperar los números y los nombres de los empleados y para rellenar una tabla de base de datos, TEMP_LIST, con esta información.

```
DECLARE
  CURSOR emp_cursor IS
    SELECT  employee_id, last_name
    FROM    employees;
  emp_record  emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO emp_record;
    EXIT WHEN emp_cursor%NOTFOUND;
    INSERT INTO temp_list (empid, empname)
    VALUES (emp_record.employee_id, emp_record.last_name);
  END LOOP;
  COMMIT;
  CLOSE emp_cursor;
END ;
/
```

Bucles FOR de Cursor

Sintaxis:

```
FOR nombre_registro IN nombre_cursor LOOP  
    sentencia1;  
    sentencia2;  
    . . .  
END LOOP;
```

- El bucle FOR de cursor es un método abreviado para procesar los cursores explícitos.
- Se producen aperturas, recuperaciones, salidas y cierres implícitos.
- El registro se declara implícitamente.

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Bucles FOR de Cursor

Los bucles FOR de cursor procesan las filas en un cursor explícito. Se trata de un método abreviado, porque el cursor se abre, se recuperan las filas una vez por cada iteración del bucle, se sale del bucle cuando se procesa la última fila y el cursor se cierra automáticamente. El propio bucle termina automáticamente al final de la iteración, donde se recupera la última fila.

En la sintaxis:

<i>nombre_registro</i>	es el nombre del registro declarado implícitamente.
<i>nombre_cursor</i>	es un identificador PL/SQL del cursor declarado previamente.

Instrucciones

- No declare el registro que controla el bucle, porque ya está declarado implícitamente.
- Pruebe los atributos del cursor durante el bucle, si es necesario.
- Proporcione los parámetros de un cursor, si es necesario, entre paréntesis y después del nombre del cursor en la sentencia FOR. En una lección posterior, encontrará más información acerca de los parámetros de cursores.
- No utilice un bucle FOR de cursor cuando haya que gestionar explícitamente las operaciones del cursor.

Nota: Se puede definir una consulta al principio del propio bucle. La expresión de la consulta se denomina subsentencia SELECT y el cursor es interno del bucle FOR. Dado que el cursor no está declarado con un nombre, no puede probar sus atributos.

Bucles FOR de Cursor

Imprima una lista de los empleados que trabajan en el departamento de ventas.

```
DECLARE
  CURSOR emp_cursor IS
    SELECT last_name, department_id
    FROM   employees;
BEGIN
  FOR emp_record IN emp_cursor LOOP
    -- se produce una apertura implícita y una
recuperación implícita
    IF emp_record.department_id = 80 THEN
      ...
    END LOOP; -- se produce un cierre implícito
END;
/
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Ejemplo

Recupere los empleados, uno por uno, e imprima una lista de aquellos que trabajen actualmente en el departamento de ventas (DEPARTMENT_ID = 80). El ejemplo de la transparencia continúa más abajo.

```
SET SERVEROUTPUT ON
DECLARE
  CURSOR emp_cursor IS
    SELECT last_name, department_id
    FROM   employees;
BEGIN
  FOR emp_record IN emp_cursor LOOP
    -- se produce una apertura implícita y una recuperación
implícita
    IF emp_record.department_id = 80 THEN
      DBMS_OUTPUT.PUT_LINE ('Employee ' || emp_record.last_name
                             || ' works in the Sales Dept. ');
    END IF;
  END LOOP; -- se produce un cierre implícito y una salida del bucle
implícita
END ;
/
```

Bucles FOR de Cursor con Subconsultas

No es necesario declarar el cursor.

Ejemplo:

```
BEGIN
  FOR emp_record IN (SELECT last_name, department_id
                     FROM   employees) LOOP
    -- se produce una apertura implícita y una
recuperación implícita
    IF emp_record.department_id = 80 THEN
      ...
    END LOOP; -- se produce un cierre implícito
END ;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Bucles FOR de Cursor con Subconsultas

Si utiliza una subconsulta en un bucle FOR, no es necesario que declare un cursor. Este ejemplo hace lo mismo que el de la página anterior. A continuación está el código completo:

```
SET SERVEROUTPUT ON
BEGIN
  FOR emp_record IN (SELECT last_name, department_id
                     FROM   employees) LOOP
    --se produce una apertura implícita y una recuperación
implícita
    IF emp_record.department_id = 80 THEN
      DBMS_OUTPUT.PUT_LINE ('Employee ' || emp_record.last_name
                           || ' works in the Sales Dept. ');
    END IF;
  END LOOP;  --se produce un cierre implícito
END ;
/
```


Ejemplo

Recupere los primeros cinco empleados con su historial de trabajo.

```
SET SERVEROUTPUT ON
DECLARE
    v_employee_id  employees.employee_id%TYPE;
    v_job_id       employees.job_id%TYPE;
    v_start_date   DATE;
    v_end_date     DATE;
    CURSOR emp_cursor IS
        SELECT          employee_id, job_id, start_date, end_date
        FROM    job_history
        ORDER BY    employee_id;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor
            INTO v_employee_id, v_job_id, v_start_date, v_end_date;
        DBMS_OUTPUT.PUT_LINE ('Employee #: ' || v_employee_id ||
            ' held the job of ' || v_job_id || ' FROM ' ||
            v_start_date || ' TO ' || v_end_date);
        EXIT WHEN emp_cursor%ROWCOUNT > 4 OR
            emp_cursor%NOTFOUND;
    END LOOP;
    CLOSE emp_cursor;
END ;
/
```

Resumen

En esta lección, ha aprendido a:

- **Distinguir los tipos de cursores**
 - **Cursores implícitos:** Se utilizan para todas las sentencias **DML** y las consultas de una única fila
 - **Cursores explícitos:** Se utilizan para las consultas sin filas, con una fila o con varias filas
- **Manipular cursores explícitos**
- **Evaluar el estado del cursor utilizando los atributos del cursor**
- **Utilizar bucles **FOR** de cursor**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Resumen

Oracle utiliza áreas de trabajo para ejecutar sentencias SQL y almacenar la información del procesamiento. Las construcciones PL/SQL que se denominan cursores permiten asignar un nombre a un área de trabajo y acceder a la información que se tiene almacenada. Existen dos tipos de cursores: implícitos y explícitos. PL/SQL declara implícitamente un cursor para todas las sentencias de manipulación de datos SQL, incluidas las consultas que sólo devuelven una fila. En el caso de las consultas que devuelven más de una fila, se puede declarar un cursor explícitamente para procesar las filas de manera individual.

Cada cursor explícito y cada variable de cursor posee cuatro atributos: `%FOUND`, `%ISOPEN`, `%NOTFOUND` y `%ROWCOUNT`. Cuando se agregan al cursor o a la variable de cursor, estos atributos devuelven información muy útil acerca de la ejecución de una sentencia de manipulación de datos. Los atributos de cursores se pueden utilizar en las sentencias procedurales, pero no en las sentencias SQL.

Visión General de la Práctica 6

Esta práctica cubre los siguientes temas:

- **Declaración y uso de cursores explícitos para consultar las filas de una tabla**
- **Uso de un bucle FOR de cursor**
- **Aplicación de atributos de cursores para probar el estado del cursor**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Visión General de la Práctica 6

Esta práctica le permitirá aplicar sus conocimientos acerca de los cursores a la hora de procesar una serie de filas desde una tabla y rellenar otra tabla con los resultados utilizando un bucle FOR de cursor.

Práctica 6

1. Ejecute el comando del archivo de comandos lab06_1.sql para crear una nueva tabla para almacenar los sueldos de los empleados.

```
CREATE TABLE      top_dogs
( salary           NUMBER( 8, 2 ) );
```

2. Cree un bloque PL/SQL que determine cuáles son los empleados que tienen los sueldos más altos.
 - a. Acepte un número n del usuario, donde n representa el número de los n que más ganan de la tabla EMPLOYEES. Por ejemplo, escriba 5 para ver a los cinco empleados que más ganan.
Nota: Utilice el comando DEFINE para proporcionar el valor de n . Transfiera el valor al bloque PL/SQL mediante una variable de sustitución de iSQL*Plus.
 - b. En un bucle, utilice el parámetro de sustitución iSQL*Plus que creó en el paso 1 y recopile los sueldos de las n primeras personas de la tabla EMPLOYEES. Los sueldos no deberían estar duplicados. Si dos empleados ganan el mismo sueldo, este sólo se debería recopilar una vez.
 - c. Almacene los sueldos en la tabla TOP_DOGS.
 - d. Pruebe diversos casos especiales como, por ejemplo, $n = 0$, o donde n es mayor que el número de empleados de la tabla EMPLOYEES. Vacíe la tabla TOP_DOGS después de cada prueba. El resultado que se muestra representa los cinco salarios más altos de la tabla EMPLOYEES.

SALARY	
	24000
	17000
	14000
	13500
	13000

3. Cree un bloque PL/SQL que haga lo siguiente:
 - a. Utilice el comando DEFINE para proporcionar el identificador del departamento. Transfiera el valor al bloque PL/SQL mediante una variable de sustitución iSQL*Plus.
 - b. En un bloque PL/SQL, recupere el apellido, el sueldo y el identificador del jefe (MANAGER ID) de los empleados que trabajan en ese departamento.
 - c. Si el sueldo del empleado es menor de 5000 y el identificador del jefe es 101 o 124, muestre el mensaje <<last_name>> Due for a raise. De lo contrario, muestre el mensaje <<last_name>> Not due for a raise.

Nota: Defina SET ECHO OFF para evitar que aparezca el código PL/SQL cada vez que ejecute el archivo de comandos.

Práctica 6 (continuación)

d. Pruebe el bloque PL/SQL para los siguientes casos:

Identificador de Departamento	Mensaje
10	Whalen Due for a raise
20	Hartstein Not Due for a raise Fay Not Due for a raise
50	Weiss Not Due for a raise Fripp Due for a raise Kaufling Due for a raise Vollman Due for a raise Mourgas Due for a raise
80	Russel Not Due for a raise Partners Not Due for a raise Errazuriz Not Due for a raise Cambrault Not Due for a raise



Conceptos Avanzados de Cursores Explícitos

ORACLE®

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Objetivos

Al finalizar este curso, debería estar capacitado para hacer lo siguiente:

- **Escribir un cursor que utilice parámetros**
- **Determinar cuándo se necesita una cláusula `FOR UPDATE` en un cursor**
- **Determinar cuándo se debe utilizar la cláusula `WHERE CURRENT OF`**
- **Escribir un cursor que utilice una subconsulta**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Finalidad de la Lección

En esta lección profundizará en el tema de la escritura de cursores explícitos y, específicamente, en la escritura de cursores que utilizan parámetros.

Cursores con Parámetros

Sintaxis:

```
CURSOR cursor_name
  [(parameter_name datatype, ...)]
IS
  select_statement;
```

- Transfiera valores de parámetros a un cursor cuando dicho cursor se abra y se ejecute la consulta.
- Abra un cursor explícito varias veces con un conjunto activo diferente cada vez.

```
OPEN cursor_name(parameter_value,.....) ;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Cursores con Parámetros

Para transferir parámetros al cursor, puede utilizar un bucle FOR de cursor. Eso significa que puede abrir y cerrar varias veces un cursor explícito en un bloque, para devolver un conjunto activo diferente en cada ocasión. Por cada ejecución, el cursor anterior se cierra y se vuelve a abrir con un nuevo conjunto de parámetros.

Cada parámetro de la declaración formal del cursor debe tener un parámetro real correspondiente en la sentencia OPEN. Los tipos de dato de los parámetros son los mismos que los de las variables escalares, pero no se les asignan tamaños. Los nombres de los parámetros se utilizan para las referencias en la expresión de consulta del cursor.

En la sintaxis:

<i>cursor_name</i>	es un identificador PL/SQL del cursor declarado previamente.
<i>parameter_name</i>	es el nombre de un parámetro.

parameter_name

<i>datatype</i>	es un tipo de dato escalar del parámetro.
<i>select_statement</i>	es una sentencia SELECT sin la cláusula INTO.

Cuando se abre el cursor, se transfieren valores a cada uno de los parámetros por posición o por nombre. Puede transferir valores desde variables del host o PL/SQL, así como desde literales.

Nota: La notación de los parámetros no es muy funcional; simplemente permite especificar los valores de entrada de una manera fácil y clara. Esto es especialmente útil cuando se hace referencia repetidas veces al mismo cursor.

Cursores con Parámetros

Transfiera el número de departamento y el puesto de trabajo a la cláusula WHERE, en la sentencia SELECT del cursor.

```
DECLARE
  CURSOR emp_cursor
    (p_deptno NUMBER, p_job VARCHAR2) IS
    SELECT employee_id, last_name
    FROM   employees
    WHERE  department_id = p_deptno
    AND    job_id = p_job;
BEGIN
  OPEN emp_cursor (80, 'SA_REP');
  . . .
  CLOSE emp_cursor;
  OPEN emp_cursor (60, 'IT_PROG');
  . . .
END;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Cursores con Parámetros

Los tipos de dato de los parámetros son los mismos que los de las variables escalares, pero no se les asignan tamaños. Los nombres de los parámetros se utilizan para las referencias en la consulta del cursor. En el siguiente ejemplo, se declara y se define un cursor con dos parámetros.

```
DECLARE
  CURSOR emp_cursor(p_deptno NUMBER, p_job VARCHAR2) IS
  SELECT ...
```

Las siguientes sentencias abren el cursor y devuelven conjuntos activos diferentes:

```
OPEN emp_cursor(60, v_emp_job);
OPEN emp_cursor(90, 'AD_VP');
```

Puede transferir parámetros al cursor que se ha utilizado en un bucle FOR de cursor:

```
DECLARE
  CURSOR emp_cursor(p_deptno NUMBER, p_job VARCHAR2) IS
  SELECT ...
BEGIN
  FOR emp_record IN emp_cursor(50, 'ST_CLERK') LOOP ...
```

La Cláusula FOR UPDATE

Sintaxis:

```
SELECT ...  
FROM      ...  
FOR UPDATE [OF column_reference][NOWAIT];
```

- **Utilice bloqueos explícitos para denegar el acceso durante el transcurso de una transacción.**
- **Bloquee las filas *antes* de realizar la actualización o la supresión.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

La Cláusula FOR UPDATE

Puede que desee bloquear las filas antes de actualizarlas o suprimirlas. Agregue la cláusula FOR UPDATE en la consulta del cursor para bloquear las filas afectadas cuando se abra el cursor. Dado que Oracle Server libera los bloqueos al final de la transacción, no debería realizar validaciones en las recuperaciones desde un cursor explícito si se utiliza FOR UPDATE.

En la sintaxis:

column_reference es una columna de la tabla en la cual se realiza la consulta. (También se puede utilizar una lista de columnas.)

NOWAIT devuelve un error de Oracle si las filas están bloqueadas por otra sesión

La cláusula FOR UPDATE es la última cláusula de una sentencia select y está incluso después de ORDER BY, si se ha utilizado esta cláusula. Cuando consulte múltiples tablas, puede utilizar la cláusula FOR UPDATE para restringir el bloqueo de las filas a unas tablas en particular. Las filas de una tabla sólo se bloquean si la cláusula FOR UPDATE se refiere a una columna de esa tabla. FOR UPDATE OF *col_name(s)* bloquea sólo las filas de las tablas que contienen *col_name(s)*.

La sentencia SELECT ... FOR UPDATE identifica las filas que se van a actualizar o a suprimir y luego bloquea cada fila en el conjunto de resultados. Esto resulta muy útil cuando se quiere basar una actualización en los valores existentes de una fila. En tal caso, debe asegurarse de que ningún otro usuario cambie la fila antes de la actualización.

La palabra clave opcional NOWAIT le indica a Oracle que no espere si otro usuario ha bloqueado las filas solicitadas. El control vuelve inmediatamente al programa, para que pueda hacer otro trabajo antes de volver a intentar realizar el bloqueo. Si omite la palabra clave NOWAIT, Oracle espera hasta que las filas estén disponibles.

La Cláusula FOR UPDATE

Recupere los empleados que trabajan en el departamento 80 y actualice sus sueldos.

```
DECLARE
  CURSOR emp_cursor IS
    SELECT employee_id, last_name
    FROM   employees,departments
    WHERE  employees.department_id =
           departments.department_id
    AND employees.department_id = 80
    FOR UPDATE OF salary NOWAIT;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

La Cláusula FOR UPDATE (continuación)

Nota: Si Oracle Server no puede obtener los bloqueos de las filas que necesita para SELECT FOR UPDATE, esperará indefinidamente. Puede utilizar la cláusula NOWAIT de la sentencia SELECT FOR UPDATE y probar el código de error que devuelve cuando no ha conseguido obtener los bloqueos en un bucle. Puede volver a intentarlo abriendo el cursor *n* veces antes de terminar el bloque PL/SQL. Si tiene una tabla de gran tamaño, puede mejorar el rendimiento utilizando la sentencia LOCK TABLE para bloquear todas las filas de la tabla. Sin embargo, al utilizar LOCK TABLE, no puede usar la cláusula WHERE CURRENT OF y debe emplear la notación WHERE *column* = *identifier*.

No es obligatorio que la cláusula FOR UPDATE OF haga referencia a una columna, pero es recomendable que lo haga para mejorar la legibilidad y facilitar el mantenimiento.

Nota: La cláusula WHERE CURRENT OF se explica más adelante, en esta misma lección.

La cláusula FOR UPDATE identifica las filas que se van a actualizar o a suprimir y luego bloquea cada fila del conjunto de resultados. Esto resulta muy útil cuando se quiere basar una actualización en los valores existentes de una fila. En tal caso, debe asegurarse de que ningún otro usuario cambie la fila antes de la actualización.

La Cláusula WHERE CURRENT OF

Sintaxis:

```
WHERE CURRENT OF cursor ;
```

- Utilice cursores para actualizar o para suprimir la fila actual.
- Utilice la cláusula FOR UPDATE en la consulta del cursor para bloquear primero las filas.
- Utilice la cláusula WHERE CURRENT OF para hacer referencia a la fila actual desde un cursor explícito.

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

La Cláusula WHERE CURRENT OF

Para hacer referencia a la fila actual desde un cursor explícito, utilice la cláusula WHERE CURRENT OF. De esta manera, puede aplicar actualizaciones y supresiones en la fila con la que está trabajando en ese momento sin que sea necesario hacer explícitamente referencia a ROWID. Debe incluir la cláusula FOR UPDATE en la consulta del cursor para que las filas se bloqueen en OPEN.

En la sintaxis:

cursor es el nombre de un cursor declarado. (El cursor se ha tenido que declarar con la cláusula FOR UPDATE.)

La Cláusula WHERE CURRENT OF

```
DECLARE
CURSOR sal_cursor IS
    SELECT e.department_id, employee_id, last_name, salary
    FROM   employees e, departments d
    WHERE  d.department_id = e.department_id
    and    d.department_id = 60
    FOR UPDATE OF salary NOWAIT;
BEGIN
    FOR emp_record IN sal_cursor
LOOP
    IF emp_record.salary < 5000 THEN
        UPDATE employees
        SET    salary = emp_record.salary * 1.10
        WHERE CURRENT OF sal_cursor;
    END IF;
END LOOP;
END;
/
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

La Cláusula WHERE CURRENT OF (continuación)

Ejemplo

El ejemplo de la transparencia realiza un bucle por cada empleado del departamento 60 y comprueba si su sueldo es menor de 5000. Si es así, sube el sueldo un 10%. La cláusula WHERE CURRENT OF de la sentencia UPDATE se refiere al registro recuperado en ese momento. Observe que se puede actualizar una tabla con la cláusula WHERE CURRENT OF, incluso si hay una unión en la declaración del cursor.

Además, se puede escribir una sentencia DELETE o UPDATE para que contenga la cláusula WHERE CURRENT OF *cursor_name* con el fin de hacer referencia a la última fila que ha procesado la sentencia FETCH. Puede actualizar filas desde un cursor basándose en criterios. Si utiliza esta cláusula, el cursor al que haga referencia debe existir y debe contener la cláusula FOR UPDATE en la consulta del cursor; de lo contrario, recibirá un error. Esta cláusula le permite aplicar actualizaciones y supresiones en la fila con la que está trabajando en ese momento sin que sea necesario hacer explícitamente referencia a la pseudo columna ROWID.

Cursores con Subconsultas

Ejemplo:

```
DECLARE
  CURSOR my_cursor IS
    SELECT t1.department_id, t1.department_name,
           t2.staff
    FROM   departments t1, (SELECT department_id,
                                   COUNT(*) AS STAFF
                            FROM employees
                            GROUP BY department_id) t2
    WHERE  t1.department_id = t2.department_id
    AND    t2.staff >= 3;
  ...
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Cursores con Subconsultas

Una subconsulta es una consulta (normalmente entre paréntesis) que aparece en el interior de otra sentencia de manipulación de datos SQL. Cuando se evalúa, la subconsulta proporciona un valor o un conjunto de valores a la consulta exterior. Las subconsultas se utilizan con frecuencia en la cláusula WHERE de una sentencia select. También se pueden utilizar en la cláusula FROM, lo que crea un origen de datos temporal para esa consulta.

En este ejemplo, la subconsulta crea un origen de datos que se compone de los números de los departamentos y el número de empleados de cada departamento (conocido con el alias STAFF). Hay un alias de tabla, t2, que hace referencia a este origen de datos temporal en la cláusula FROM. Cuando se abra este cursor, el conjunto activo contendrá el número del departamento, el nombre del departamento y el número total de empleados que trabaja en él, siempre y cuando haya tres o más empleados que trabajen en ese departamento.

Resumen

En esta lección, ha aprendido a:

- **Devolver diferentes conjuntos activos utilizando cursores con parámetros.**
- **Definir cursores con subconsultas y subconsultas correlacionadas.**
- **Manipular cursores explícitos con comandos, utilizando:**
 - **La cláusula FOR UPDATE**
 - **La cláusula WHERE CURRENT OF**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Resumen

Un cursor explícito puede aceptar parámetros. Se puede especificar un parámetro de cursor en una consulta, siempre que aparezca una constante. Una de las ventajas de utilizar parámetros es que le permite decidir el conjunto activo en tiempo de ejecución.

PL/SQL posee un método para modificar las filas que ha recuperado el cursor. Este método se compone de dos partes: La cláusula FOR UPDATE de la declaración del cursor y la cláusula WHERE CURRENT OF de una sentencia UPDATE o DELETE .

Visión General de la Práctica 7

Esta práctica cubre los siguientes temas:

- **Declaración y uso de cursores explícitos con parámetros**
- **Uso de un cursor FOR UPDATE**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Visión General de la Práctica 7

Esta práctica le permitirá aplicar sus conocimientos acerca de los cursores con parámetros a la hora de procesar una serie de filas desde múltiples tablas.

Práctica 7

1. En un bucle, utilice un cursor para recuperar el número del departamento y el nombre del departamento de la tabla DEPARTMENTS de aquellos departamentos cuyo identificador (DEPARTMENT_ID) sea menor de 100. Transfiera el número del departamento a otro cursor para recuperar de la tabla EMPLOYEES los apellidos, los puestos, las fechas de contratación y los sueldos de los empleados cuyo identificador (EMPLOYEE_ID) sea menor de 120 y que trabajen en ese departamento.

Department Number : 10 Department Name : Administration

Department Number : 20 Department Name : Marketing

Department Number : 30 Department Name : Purchasing

Raphaely PU_MAN 07-DEC-94 11000
Khoo PU_CLERK 18-MAY-95 3100
Baida PU_CLERK 24-DEC-97 2900
Tobias PU_CLERK 24-JUL-97 2800
Himuro PU_CLERK 15-NOV-98 2600
Colmenares PU_CLERK 10-AUG-99 2500

Department Number : 40 Department Name : Human Resources

Department Number : 50 Department Name : Shipping

Department Number : 60 Department Name : IT

Hunold IT_PROG 03-JAN-90 9000
Ernst IT_PROG 21-MAY-91 6000
Austin IT_PROG 25-JUN-97 5280
Pataballa IT_PROG 05-FEB-98 5280
Lorentz IT_PROG 07-FEB-99 4620

Department Number : 70 Department Name : Public Relations

Department Number : 80 Department Name : Sales

Department Number : 90 Department Name : Executive

King AD_PRES 17-JUN-87 24000
Kochhar AD_VP 21-SEP-89 17000
De Haan AD_VP 13-JAN-93 17000

PL/SQL procedure successfully completed.

Práctica 7 (continuación)

2. Modifique el código de `sol104_4.sql` para incorporar un cursor utilizando la funcionalidad `FOR UPDATE` y `WHERE CURRENT OF` en el procesamiento del cursor.

- a. Defina las variables del host.

```
DEFINE p_empno=104
```

```
DEFINE p_empno=174
```

```
DEFINE p_empno=176
```

- b. Ejecute el bloque PL/SQL modificado.

- c. Ejecute el siguiente comando para comprobar si el bloque PL/SQL ha funcionado correctamente:

```
SELECT employee_id,salary,stars  
FROM EMP  
WHERE employee_id IN (176,174,104);
```

EMPLOYEE_ID	SALARY	STARS
104	6000	*****
174	11000	*****
176	8600	*****

8

Manejo de Excepciones

ORACLE®

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Objetivos

Al finalizar esta lección, debería estar capacitado para hacer lo siguiente:

- **Definir excepciones PL/SQL**
- **Reconocer excepciones no tratadas**
- **Enumerar y utilizar diferentes tipos de manejadores de excepciones PL/SQL**
- **Interrumpir errores no anticipados**
- **Describir el efecto de la propagación de excepciones en bloques anidados**
- **Personalizar mensajes de excepciones PL/SQL**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Finalidad de la Lección

En esta lección aprenderá qué son las excepciones PL/SQL y cómo tratarlas utilizando manejadores de excepciones predefinidos, no predefinidos y definidos por el usuario.

Manejo de Excepciones con PL/SQL

- Una excepción es un identificador en PL/SQL que se emite durante la ejecución.
- ¿Cómo se inicia?
 - Se produce un error de Oracle.
 - Se inicia explícitamente.
- ¿Cómo se maneja?
 - Se interrumpe con un manejador.
 - Se propaga al entorno de llamada.

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

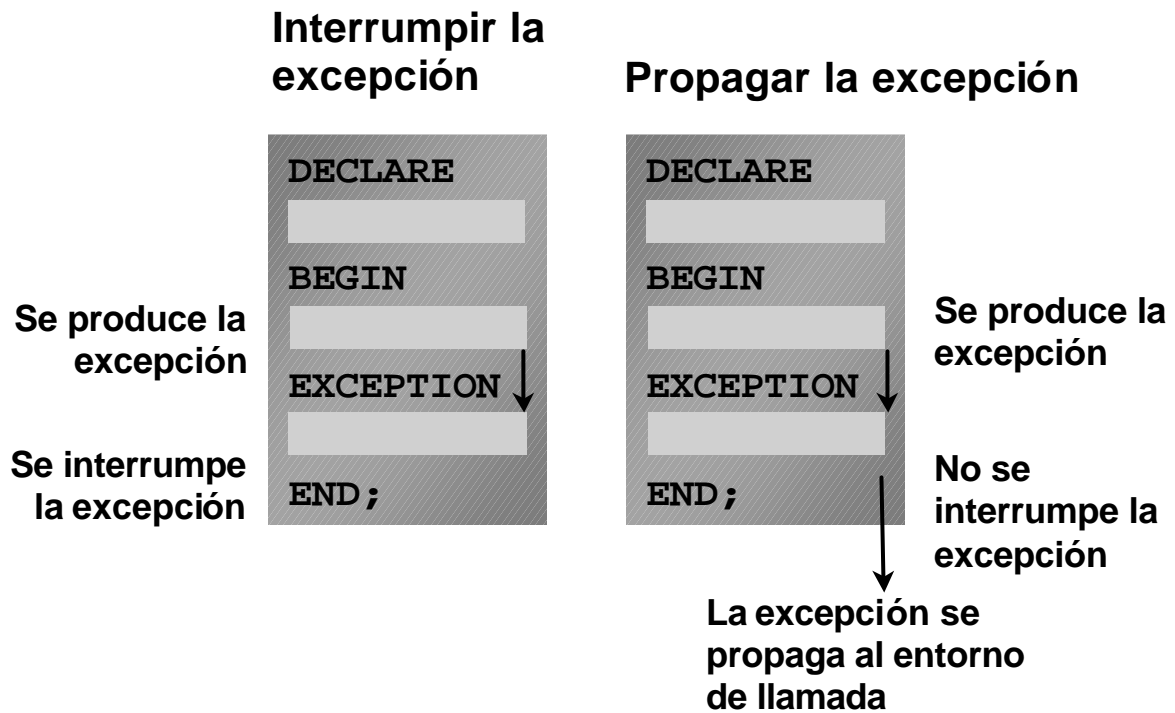
Visión General

Una excepción es un identificador en PL/SQL que se emite durante la ejecución de un bloque que termina su cuerpo principal de acciones. Los bloques terminan siempre cuando PL/SQL produce una excepción, pero se puede especificar un manejador de excepciones para realizar las acciones finales.

Dos Métodos para Producir una Excepción

- Cuando se produce un error de Oracle, la excepción asociada se produce automáticamente. Por ejemplo, si se produce el error ORA-01403 cuando no se recupera ninguna fila de la base de datos en una sentencia `SELECT`, PL/SQL emite la excepción `NO_DATA_FOUND`.
- Para emitir una excepción explícitamente, se produce la sentencia `RAISE` en el interior del bloque. La excepción que se produce puede ser definida por el usuario o predefinida.

Manejo de Excepciones



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Interrupción de una Excepción

Si la excepción se produce en la sección ejecutable del bloque, el procesamiento deriva al manejador de excepciones correspondiente en la sección de excepciones del bloque. Si PL/SQL consigue manejar la excepción, dicha excepción no se propaga al bloque o al entorno delimitador. El bloque PL/SQL finaliza con éxito.

Propagación de una Excepción

Si la excepción se produce en la sección ejecutable del bloque y no existe un manejador de excepciones correspondiente, el bloque PL/SQL termina con un error y la excepción se propaga al entorno de llamada.

Tipos de Excepciones

- Predefinidas de Oracle Server
 - No predefinidas de Oracle Server
- } Producidas implícitamente
- Definidas por el usuario
- Producidas explícitamente

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Tipos de Excepciones

Se pueden programar excepciones para evitar problemas en tiempo de ejecución. Existen tres tipos de excepciones.

Excepción	Descripción	Instrucciones de Manejo
Error predefinido de Oracle Server	Uno de los aproximadamente 20 errores que se producen con más frecuencia en el código PL/SQL	No hay que declararlas y hay que dejar que Oracle Server las emita implícitamente
Error no predefinido de Oracle Server	Cualquier otro error estándar de Oracle Server	Hay que declararlas en la sección declarativa y hay que dejar que Oracle Server las emita implícitamente
Error definido por el usuario	Una condición que el desarrollador considera que no es normal	Hay que declararlas en la sección declarativa y emitirlas explícitamente

Nota: Algunas herramientas de aplicaciones con PL/SQL de cliente como, por ejemplo, Oracle Developer Forms, poseen sus propias excepciones.

Interrupción de Excepciones

Sintaxis:

```
EXCEPTION
  WHEN exception1 [OR exception2 . . .] THEN
    statement1;
    statement2;
    . . .
  [WHEN exception3 [OR exception4 . . .] THEN
    statement1;
    statement2;
    . . .]
  [WHEN OTHERS THEN
    statement1;
    statement2;
    . . .]
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Interrupción de Excepciones

Puede interrumpir cualquier error utilizando la rutina correspondiente en el interior de la sección de manejo de excepciones del bloque PL/SQL. Cada manejador se compone de una cláusula WHEN, que especifica una excepción, seguida de una secuencia de sentencias que se van a ejecutar cuando se produzca esa excepción.

En la sintaxis:

excepción es el nombre estándar de una excepción predefinida o el nombre de una excepción definida por el usuario que se ha declarado en la sección

sentencia es una o varias sentencias PL/SQL o SQL.

OTHERS es una cláusula de manejo de excepciones opcional que interrumpe las excepciones no especificadas.

Manejador de Excepciones WHEN OTHERS

La sección de manejo de excepciones sólo interrumpe aquellas excepciones que se hayan especificado; no se interrumpe ninguna otra excepción, a menos que se utilice el manejador de excepciones OTHERS. Este manejador interrumpe cualquier excepción que todavía no se haya manejado. Por esta razón, OTHERS es el último manejador de excepciones que se define.

El manejador OTHERS interrumpe *todas* las excepciones que no se hayan interrumpido aún. Algunas herramientas de Oracle poseen sus propias excepciones predefinidas que se pueden emitir para producir eventos en la aplicación. El manejador OTHERS también interrumpe estas excepciones.

Instrucciones de la Interrupción de Excepciones

- La palabra clave **EXCEPTION** inicia la sección de manejo de excepciones.
- Se pueden utilizar varios manejadores de excepciones.
- Sólo se procesa un manejador antes de salir del bloque.
- **WHEN OTHERS** es la última cláusula.

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Instrucciones

- Comience la sección de manejo de excepciones del bloque con la palabra clave **EXCEPTION**.
- Defina varios manejadores de excepciones, cada uno con su propio conjunto de acciones, para el bloque.
- Cuando se produce una excepción, PL/SQL procesa *sólo un* manejador antes de salir del bloque.
- Coloque la cláusula **OTHERS** después de todas las demás cláusulas de manejo de excepciones.
- Sólo puede utilizar una sola cláusula **OTHERS**.
- Las excepciones no pueden aparecer en las sentencias de asignación ni en las sentencias **SQL**.

Interrupción de Errores Predefinidos de Oracle Server

- Haga referencia al nombre estándar en la rutina de manejo de excepciones.
- Ejemplos de excepciones predefinidas:
 - NO_DATA_FOUND
 - TOO_MANY_ROWS
 - INVALID_CURSOR
 - ZERO_DIVIDE
 - DUP_VAL_ON_INDEX

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Interrupción de Errores Predefinidos de Oracle Server

Para interrumpir un error predefinido de Oracle Server, haga referencia a su nombre estándar en el interior de la rutina de manejo de excepciones correspondiente.

Si desea ver una lista completa de las excepciones predefinidas, consulte el capítulo “Error Handling” de *PL/SQL User's Guide and Reference*.

Nota: PL/SQL declara las excepciones predefinidas en el paquete STANDARD.

Es conveniente manejar siempre las excepciones NO_DATA_FOUND y TOO_MANY_ROWS, que son las más habituales.

.

Excepciones Predefinidas

Nombre de Excepción	Número de Error de Oracle Server	Descripción
ACCESS_INTO_NULL	ORA-06530	Se han intentado asignar valores a los atributos de un objeto que no está inicializado
CASE_NOT_FOUND	ORA-06592	No se ha seleccionado ninguna de las opciones en las cláusulas WHEN de una sentencia CASE y no hay ninguna cláusula ELSE.
COLLECTION_IS_NULL	ORA-06531	Se han intentado aplicar métodos de recopilación que no son EXISTS a una tabla o varray que no están inicializados.
CURSOR_ALREADY_OPEN	ORA-06511	Se ha intentado abrir un cursor que ya está abierto
DUP_VAL_ON_INDEX	ORA-00001	Se ha intentado insertar un valor duplicado
INVALID_CURSOR	ORA-01001	Se ha producido una operación no válida del cursor
INVALID_NUMBER	ORA-01722	Ha fallado la conversión de la cadena de caracteres a números
LOGIN_DENIED	ORA-01017	Se ha conectado a Oracle con un usuario o una contraseña que no son válidos
NO_DATA_FOUND	ORA-01403	Un SELECT de una sola fila no ha devuelto ningún dato
NOT_LOGGED_ON	ORA-01012	El programa PL/SQL ha realizado una llamada a la base de datos sin estar conectado a Oracle
PROGRAM_ERROR	ORA-06501	PL/SQL tiene un problema interno
ROWTYPE_MISMATCH	ORA-06504	La variable de cursor del host y la variable de cursor PL/SQL que participan en una asignación tienen tipos de retorno incompatibles

Excepciones Predefinidas (continuación)

Nombre de Excepción	Número de Error de Oracle Server	Descripción
STORAGE_ERROR	ORA-06500	PL/SQL se ha quedado sin memoria o la memoria está corrupta
SUBSCRIPT_BEYOND_COUNT	ORA-06533	Se ha hecho referencia a una tabla anidada o a un elemento varray utilizando un número de índice mayor que el número de elementos de la recopilación.
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	Se ha hecho referencia a una tabla anidada o a un elemento varray utilizando un número de índice que está fuera del rango válido (-1, por ejemplo).
SYS_INVALID_ROWID	ORA-01410	Ha fallado la conversión de una cadena de caracteres en un ROWID universal porque la cadena de caracteres no representa un ROWID válido.
TIMEOUT_ON_RESOURCE	ORA-00051	Se ha producido un timeout mientras Oracle esperaba un recurso.
TOO_MANY_ROWS	ORA-01422	Un SELECT de una sola fila ha devuelto más de una fila.
VALUE_ERROR	ORA-06502	Se ha producido un error aritmético, de conversión, de truncado o de restricción de tamaño.
ZERO_DIVIDE	ORA-01476	Se ha intentado dividir por cero

Excepciones Predefinidas

Sintaxis:

```
BEGIN
. . .
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    statement1;
    statement2;

  WHEN TOO_MANY_ROWS THEN
    statement1;
  WHEN OTHERS THEN
    statement1;
    statement2;
    statement3;
END;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Interrupción de Excepciones Predefinidas de Oracle Server

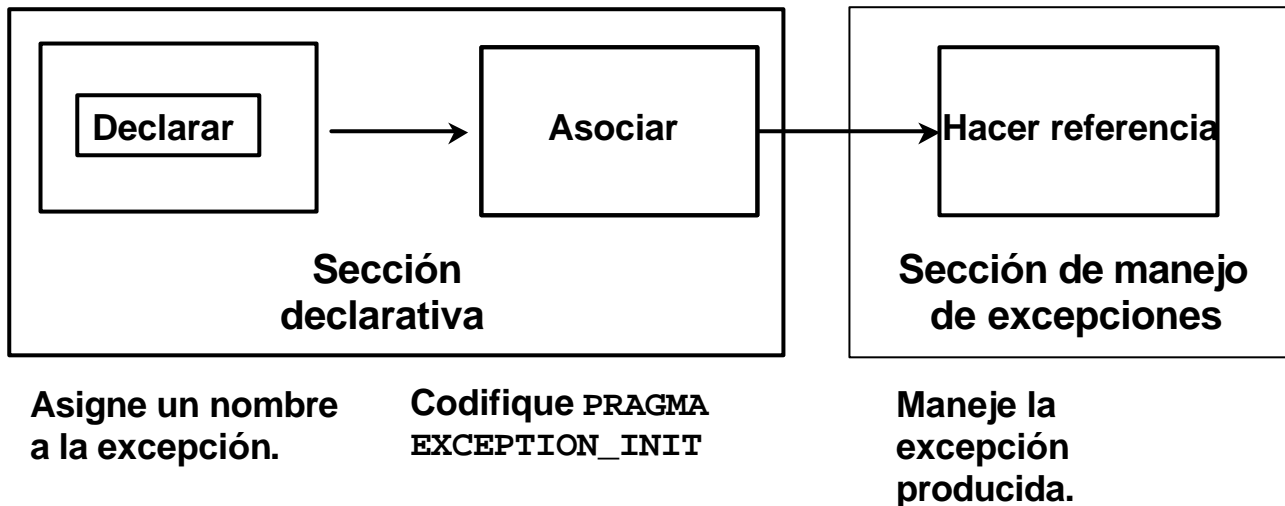
Cuando se produce una excepción, la ejecución normal del bloque o el subprograma PL/SQL se detiene y el control se transfiere a la parte de manejo de excepciones, que tiene el formato que se muestra en la transparencia.

Para atrapar las excepciones producidas, hay que escribir manejadores de excepciones. Cada manejador se compone de una cláusula WHEN, que especifica una excepción, seguida de una secuencia de sentencias que se van a ejecutar cuando se produzca esa excepción. Estas sentencias completan la ejecución del bloque o el subprograma; el control no vuelve al lugar en el que se produjo la excepción. Dicho de otro modo, no se puede continuar el procesamiento donde se quedó.

El manejador de excepciones opcional OTHERS, que, si existe, siempre es el último manejador del bloque o el subprograma, funciona como manejador de todas las excepciones a las que no se les ha asignado un nombre específicamente. Por eso, un bloque o un subprograma sólo puede tener un manejador OTHERS. Tal y como muestra el siguiente ejemplo, el uso del manejador OTHERS garantiza que ninguna excepción se quedará sin ser tratada:

```
EXCEPTION
  WHEN ... THEN
    -- gestiona el error
  WHEN ... THEN
    -- gestiona el error
  WHEN OTHERS THEN
    -- gestiona el resto de los errores
END;
```

Interrupción de Errores No Predefinidos de Oracle Server



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Interrupción de Errores No Predefinidos de Oracle Server

Para interrumpir un error no predefinido de Oracle Server, primero hay que declararlo, o bien hay que utilizar el manejador OTHERS. Las excepciones declaradas se producen implícitamente. En PL/SQL, `PRAGMA EXCEPTION_INIT` le indica al compilador que asocie el nombre de una excepción a un número de error de Oracle. Esto le permite hacer referencia a cualquier excepción interna por su nombre y escribir para ella un manejador específico.

Nota: `PRAGMA` (también denominada *pseudo instrucciones*) es la palabra clave que indica que la sentencia es una directiva del compilador, que no se procesa cuando se ejecuta el bloque PL/SQL. En su lugar, le indica al compilador de PL/SQL que interprete todas las incidencias del nombre de la excepción en el interior del bloque como el número de error de Oracle Server que tiene asociado

Errores No Predefinidos

Realice la interrupción para el número de error de Oracle Server –2292, que es una violación de la restricción de integridad.

```
DEFINE p_deptno = 10
DECLARE
    e_emps_remaining EXCEPTION;
    PRAGMA EXCEPTION_INIT
        (e_emps_remaining, -2292);
BEGIN
    DELETE FROM departments
    WHERE department_id = &p_deptno;
    COMMIT;
EXCEPTION
    WHEN e_emps_remaining THEN
        DBMS_OUTPUT.PUT_LINE ('Cannot remove dept ' ||
            TO_CHAR(&p_deptno) || '. Employees exist. ');
END;
```

1

2

3

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Interrupción de Excepciones No Predefinidas de Oracle Server

1. Declare el nombre de la excepción en la sección declarativa.

Sintaxis

exception EXCEPTION;

donde: *exception* es el nombre de la excepción.

2. Asocie la excepción declarada con el número de error estándar de Oracle Server utilizando la sentencia PRAGMA EXCEPTION_INIT.

Sintaxis

PRAGMA EXCEPTION_INIT(*exception*, *error_number*);

donde: *exception* es el nombre de la excepción declarada previamente.

error_number es el número de error estándar de Oracle Server.

3. Haga referencia a la excepción declarada en el interior de la rutina de manejo de excepciones correspondiente.

Ejemplo

Si en un departamento hay empleados, imprima un mensaje que informe al usuario de que el departamento no se puede eliminar.

Funciones para Interrumpir Excepciones

- **SQLCODE:** Devuelve el valor numérico del código de error
- **SQLERRM:** Devuelve el mensaje asociado al número de error

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Funciones de Interrupción de Errores

Cuando se produce una excepción, se puede identificar el código de error o el mensaje de error asociado utilizando dos funciones. Basándose en los valores del código o del mensaje, puede decidir qué acción va a realizar posteriormente en función del error.

SQLCODE devuelve el número de error de Oracle de las excepciones internas. Puede enviar un número de error a SQLERRM, para que le devuelva el mensaje asociado a ese número de error.

Función	Descripción
SQLCODE	Devuelve el valor numérico del código de error (puede asignarlo a una variable NUMBER).
SQLERRM	Devuelve datos de caracteres que contienen el mensaje asociado al número de error

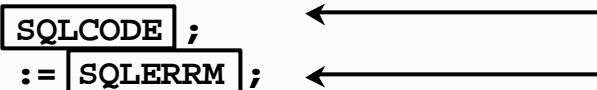
Ejemplo de Valores SQLCODE

Valor de SQLCODE	Descripción
0	No se ha encontrado ninguna excepción
1	Excepción definida por el usuario
+100	Excepción NO_DATA_FOUND
número negativo	Otro número de error de Oracle Server

Funciones para Interrumpir Excepciones

Ejemplo:

```
DECLARE
    v_error_code      NUMBER;
    v_error_message    VARCHAR2(255);
BEGIN
    ...
EXCEPTION
    ...
    WHEN OTHERS THEN
        ROLLBACK;
        v_error_code := SQLCODE;
        v_error_message := SQLERRM;
        INSERT INTO errors
            VALUES(v_error_code, v_error_message);
END;
```



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

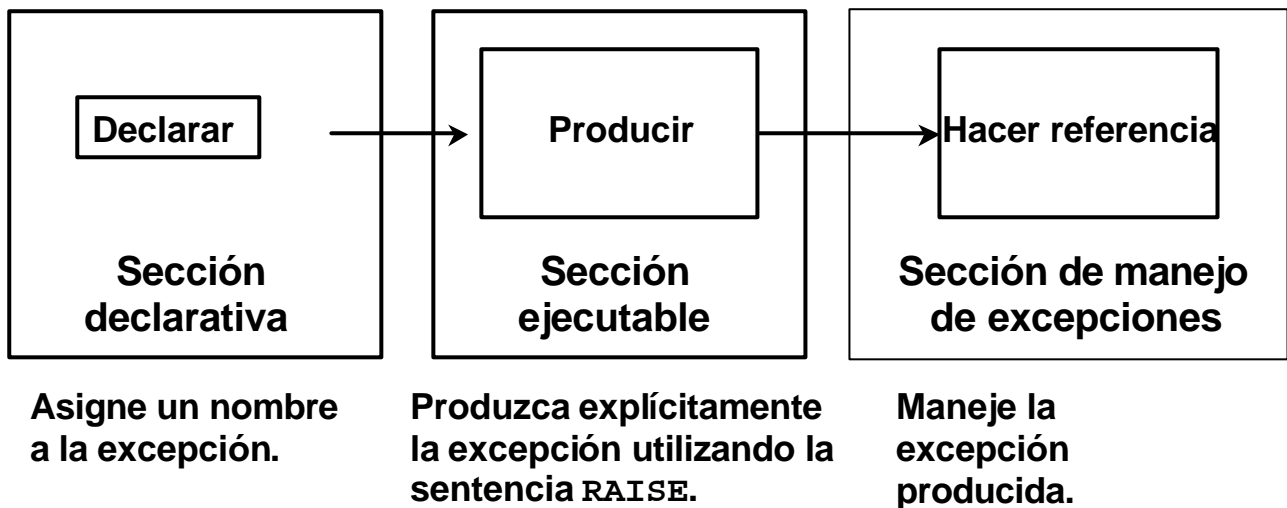
Funciones de Interrupción de Errores (continuación)

Cuando se interrumpe una excepción en el manejador de excepciones WHEN OTHERS, puede utilizar un conjunto de funciones genéricas para identificar esos errores. El ejemplo de la transparencia ilustra los valores SQLCODE y SQLERRM que se están asignando a variables y muestra cómo esas variables se utilizan luego en una sentencia SQL.

No puede utilizar SQLCODE ni SQLERRM directamente en una sentencia SQL. En su lugar, debe asignar sus valores a variables locales y luego utilizar las variables de la sentencia SQL, tal y como se muestra en el siguiente ejemplo:

```
DECLARE
    err_num NUMBER;
    err_msg VARCHAR2(100);
BEGIN
    ...
EXCEPTION
    ...
    WHEN OTHERS THEN
        err_num := SQLCODE;
        err_msg := SUBSTR(SQLERRM, 1, 100);
        INSERT INTO errors VALUES (err_num, err_msg);
END;
```

Interrupción de Excepciones Definidas por el Usuario



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Interrupción de Excepciones Definidas por el Usuario

PL/SQL le permite definir sus propias excepciones. Las excepciones definidas por el usuario deben ser:

- Declaradas en la sección declarativa de un bloque PL/SQL.
- Producidas explícitamente con sentencias `RAISE`.

Excepciones Definidas por el Usuario

Ejemplo:

```
DEFINE p_department_desc = 'Information Technology '  
DEFINE P_department_number = 300
```

```
DECLARE  
    e_invalid_department EXCEPTION;  
BEGIN  
    UPDATE      departments  
    SET          department_name = '&p_department_desc'  
    WHERE        department_id = &p_department_number;  
    IF SQL%NOTFOUND THEN  
        RAISE e_invalid_department;  
    END IF;  
    COMMIT;  
EXCEPTION  
    WHEN e_invalid_department THEN  
        DBMS_OUTPUT.PUT_LINE('No such department id.');
```

1

2

3

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Interrupción de Excepciones Definidas por el Usuario (continuación)

Para interrumpir una excepción definida por el usuario, declárela y luego emítala explícitamente.

1. Declare el nombre de la excepción definida por el usuario en la sección declarativa.

Sintaxis:

```
exception          EXCEPTION;
```

donde: *exception* es el nombre de la excepción.

2. Utilice la sentencia RAISE para producir la excepción explícitamente en la sección ejecutable.

Sintaxis:

```
RAISE exception;
```

donde: *exception* es el nombre de la excepción declarada previamente.

3. Haga referencia a la excepción declarada en la rutina de manejo de excepciones correspondiente.

Ejemplo

Este bloque actualiza la descripción de un departamento. El usuario proporciona el número del departamento y el nuevo nombre. Si el usuario introduce un número de departamento que no existe, no se actualizará ninguna fila de la tabla DEPARTMENTS. Produzca una excepción e imprima un mensaje que informe al usuario de que ha introducido un número de departamento que no es válido.

Nota: Utilice la sentencia RAISE propia en un manejador de excepciones para volver a producir la misma excepción en el entorno de llamada.

Entornos de Llamada

iSQL*Plus	Muestra en pantalla el número y el mensaje de error
Oracle Procedure Builder	Muestra en pantalla el número y el mensaje de error
Oracle Developer Forms	Accede al número y al mensaje de error en un disparador por medio de las funciones empaquetadas <code>ERROR_CODE</code> y <code>ERROR_TEXT</code>
Aplicación de precompilador	Accede al número de la excepción mediante la estructura de datos de <code>SQLCA</code>
Un bloque PL/SQL delimitador	Interrumpe una excepción en una rutina de manejo de excepciones del bloque delimitador

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Propagación de Excepciones

En lugar de atrapar una excepción en el interior del bloque PL/SQL, propague la excepción para que el entorno de llamada pueda manejarla. Cada entorno de llamada tiene su propia forma de mostrar y de acceder a los errores.

Propagación de Excepciones

Los subbloques pueden manejar una excepción o transferirla al bloque delimitador.

```
DECLARE
    . . .
    e_no_rows      exception;
    e_integrity     exception;
    PRAGMA EXCEPTION_INIT (e_integrity, -2292);
BEGIN
    FOR c_record IN emp_cursor LOOP
        BEGIN
            SELECT ...
            UPDATE ...
            IF SQL%NOTFOUND THEN
                RAISE e_no_rows;
            END IF;
        END;
    END LOOP;
EXCEPTION
    WHEN e_integrity THEN ...
    WHEN e_no_rows THEN ...
END;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Propagación de una Excepción en un Subbloque

Cuando un subbloque maneja una excepción, finaliza de la manera habitual y el control continúa en el bloque delimitador inmediatamente después de la sentencia END del subbloque.

Sin embargo, si PL/SQL produce una excepción y el bloque actual no tiene un manejador para esa excepción, la excepción se propaga por los sucesivos bloques delimitadores hasta que encuentra un manejador.

Cuando la excepción se propaga a un bloque delimitador, las acciones ejecutables restantes de ese bloque se ignoran.

Una de las ventajas de este comportamiento es que puede encerrar sentencias que requieren su propio manejador de errores de su propio bloque, mientras que se deja el manejo de excepciones más general para el bloque delimitador.

Observe en el ejemplo que las excepciones, `e_no_rows` y `e_integrity`, se declaran en el bloque exterior. En el bloque interior, cuando se produce la excepción `e_no_rows`, PL/SQL la busca en el subbloque. Dado que la excepción no está declarada en el subbloque, se propaga al bloque exterior, donde PL/SQL encuentra la declaración.

El Procedimiento

RAISE_APPLICATION_ERROR

Sintaxis:

```
raise_application_error (error_number,  
                        message[, {TRUE | FALSE}]);
```

- Puede utilizar este procedimiento para producir mensajes de error definidos por el usuario desde subprogramas almacenados.
- Puede informar de los errores a su aplicación y evitar que se devuelvan excepciones no tratadas.

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

El Procedimiento RAISE_APPLICATION_ERROR

Utilice el procedimiento RAISE_APPLICATION_ERROR para comunicar una excepción predefinida de manera interactiva devolviendo un código de error y un mensaje de error no estándar. Con RAISE_APPLICATION_ERROR, puede informar de los errores a su aplicación y evitar que se devuelvan excepciones no tratadas.

En la sintaxis:

<i>error_number</i>	es un número especificado por el usuario para la excepción, entre -20000 y -20999.
<i>message</i>	es el mensaje especificado por el usuario para la excepción. Se trata de una cadena de caracteres de una longitud de hasta 2.048 bytes.
TRUE FALSE	es un parámetro booleano opcional. (Si es TRUE, el error se coloca en la pila de errores anteriores. Si es FALSE, el valor por defecto, el error sustituye a todos los errores anteriores.)

El Procedimiento

RAISE_APPLICATION_ERROR

- **Se utiliza en dos lugares diferentes:**
 - **En la sección ejecutable**
 - **En la sección de excepciones**
- **Devuelve al usuario de manera consistente condiciones de errores junto con otros errores de Oracle Server**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

El Procedimiento RAISE_APPLICATION_ERROR (continuación)

RAISE_APPLICATION_ERROR se puede utilizar tanto en la sección ejecutable como en la sección de excepciones (o en ambas) de un programa PL/SQL. El error que se devuelve es consistente con la manera en que Oracle Server produce un error predefinido, no predefinido o definido por el usuario. El número y el mensaje del error se muestran al usuario.

RAISE_APPLICATION_ERROR

Sección ejecutable:

```
BEGIN
...
DELETE FROM employees
WHERE manager_id = v_mgr;
IF SQL%NOTFOUND THEN
RAISE_APPLICATION_ERROR(-20202,
    'This is not a valid manager');
END IF;
...
```

Sección de excepciones:

```
...
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR (-20201,
            'Manager is not a valid employee. ');
END;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Ejemplo

La transparencia demuestra que el procedimiento RAISE_APPLICATION_ERROR se puede utilizar tanto en la sección ejecutable como en la sección de excepciones de un programa PL/SQL. Aquí tiene otro ejemplo de un procedimiento RAISE_APPLICATION_ERROR que se puede utilizar tanto en la sección ejecutable como en la sección de excepciones de un programa PL/SQL:

```
DECLARE
    e_name EXCEPTION;
    PRAGMA EXCEPTION_INIT (e_name, -20999);
BEGIN
...
DELETE FROM employees
WHERE last_name = 'Higgins';
IF SQL%NOTFOUND THEN
RAISE_APPLICATION_ERROR(-20999, 'This is not a valid last name');
END IF;
EXCEPTION
    WHEN e_name THEN
        -- gestiona el error
    ...
END;
/
```

Resumen

En esta lección, ha aprendido:

- **Tipos de excepciones:**
 - **Error predefinido de Oracle Server**
 - **Error no predefinido de Oracle Server**
 - **Error definido por el usuario**
- **Interrupción de excepciones**
- **Manejo de excepciones:**
 - **Interrumpir la excepción en el bloque PL/SQL.**
 - **Propagar la excepción.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Resumen

En PL/SQL, las condiciones de error o de advertencia se denominan excepciones. Las excepciones predefinidas son condiciones de error que están definidas por Oracle Server. Las excepciones no predefinidas son cualquier otro error no estándar de Oracle Server. Las excepciones definidas por el usuario son excepciones específicas de la aplicación. Entre algunos ejemplos de excepciones predefinidas se incluyen la división por cero (`ZERO_DIVIDE`) y la falta de memoria (`STORAGE_ERROR`). A las excepciones que no tienen nombres definidos se les pueden asignar nombres utilizando la sentencia `PRAGMA EXCEPTION_INIT`.

Puede definir sus propias excepciones en la parte declarativa de cualquier bloque, cualquier subprograma o cualquier paquete PL/SQL. Por ejemplo, podría definir una excepción que se llamara `INSUFFICIENT_FUNDS` para indicar las cuentas bancarias que tienen descubiertos. Hay que asignar nombres a las excepciones definidas por el usuario.

Cuando se produce un error, se produce una excepción. Es decir, la ejecución normal se detiene y el control se transfiere a la parte de manejo de excepciones de su bloque o su subprograma PL/SQL. El sistema de tiempo de ejecución produce las excepciones internas implícitamente (automáticamente). Las excepciones definidas por el usuario se deben producir explícitamente por medio de sentencias `RAISE`, que también pueden producir excepciones predefinidas.

Para manejar las excepciones producidas, es necesario escribir unas rutinas independientes denominadas manejadores de excepciones. Una vez ejecutado un manejador de excepciones, el bloque actual deja de ejecutarse y el bloque delimitador continúa en la siguiente sentencia. Si no hay ningún bloque delimitador, el control vuelve al entorno del host.

Visión General de la Práctica 8

Esta práctica cubre los siguientes temas:

- **Manejo de excepciones con nombre**
- **Creación y llamada de excepciones definidas por el usuario**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Visión General de la Práctica 8

En esta práctica, tendrá que crear manejadores de excepciones para situaciones específicas.

Práctica 8

1. Escriba un bloque PL/SQL que seleccione el nombre del empleado que tiene un sueldo determinado.
 - a. Utilice el comando `DEFINE` para proporcionar el sueldo.
 - b. Transfiera el valor al bloque PL/SQL mediante una variable de sustitución `iSQL*Plus`. Si el sueldo que ha introducido devuelve más de una fila, maneje la excepción con el manejador de excepciones adecuado e incluya en la tabla `MESSAGES` el mensaje “More than one employee with a salary of <sueldo>.”
 - c. Si el sueldo que ha introducido no devuelve ninguna fila, maneje la excepción con el manejador de excepciones adecuado e incluya en la tabla `MESSAGES` el mensaje “No employee with a salary of <sueldo>.”
 - d. Si el sueldo que ha introducido sólo devuelve una fila, incluya en la tabla `MESSAGES` el nombre del empleado y su sueldo.
 - e. Maneje cualquier otra excepción con el manejador de excepciones adecuado e incluya en la tabla `MESSAGES` el mensaje “Some other error occurred.”
 - f. Pruebe el bloque en una amplia variedad de casos de prueba. Muestre las filas de la tabla `MESSAGES` para verificar si el bloque PL/SQL se ha ejecutado con éxito. A continuación se muestra un resultado de ejemplo.

RESULTS
More than one employee with a salary of 6000
No employee with a salary of 5000
More than one employee with a salary of 7000
No employee with a salary of 2000

2. Modifique el código del archivo `p3q3.sql` para agregar un manejador de excepciones.
 - a. Utilice el comando `DEFINE` para proporcionar el identificador del departamento y la ubicación de dicho departamento. Transfiera los valores al bloque PL/SQL mediante variables de sustitución `iSQL*Plus`.
 - b. Escriba un manejador de excepciones de error con el fin de que envíe un mensaje al usuario que le informe de que el departamento especificado no existe. Utilice una variable ligada para enviar el mensaje al usuario.
 - c. Ejecute el bloque PL/SQL introduciendo un departamento que no existe.

G_MESSAGE
Department 200 is an invalid department

Práctica 8 (continuación)

3. Escriba un bloque PL/SQL que imprima el número de empleados que gana \$100 más o menos que el valor del sueldo que se ha especificado para una variable de sustitución *iSQL*Plus*. Utilice el comando `DEFINE` para proporcionar el sueldo. Transfiera el valor al bloque PL/SQL mediante una variable de sustitución *iSQL*Plus*.
- Si no hay ningún empleado que tenga un sueldo en ese rango, imprima un mensaje para el usuario que le indique la situación. Utilice una excepción para este caso.
 - Si hay uno o varios empleados en ese rango, el mensaje debería indicar cuántos empleados se encuentran en ese rango de sueldo.
 - Maneje cualquier otra excepción con el manejador de excepciones adecuado. El mensaje debería indicar que se ha producido otro tipo de error.

```
DEFINE p_sal = 7000
```

```
DEFINE p_sal = 2500
```

```
DEFINE p_sal = 6500
```

G_MESSAGE

There is/are 4 employee(s) with a salary between 6900 and 7100

G_MESSAGE

There is/are 12 employee(s) with a salary between 2400 and 2600

G_MESSAGE

There is/are 3 employee(s) with a salary between 6400 and 6600