
FIAP

Sistemas de Informação

Prof. Me. Alexandre Barcelos
profalexandre.barcelos@fiap.com.br

Registros

PL/SQL

O Atributo %ROWTYPE

- **Declarar uma variável segundo um conjunto de colunas de uma visão ou tabela de banco de dados.**
- **Prefixar `%ROWTYPE` com a tabela de banco de dados.**
- **Campos no registro obtêm seus nomes e**

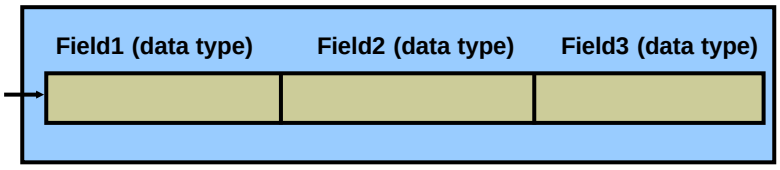
Sintaxe:

```
DECLARE
   identifier  reference%ROWTYPE;
```

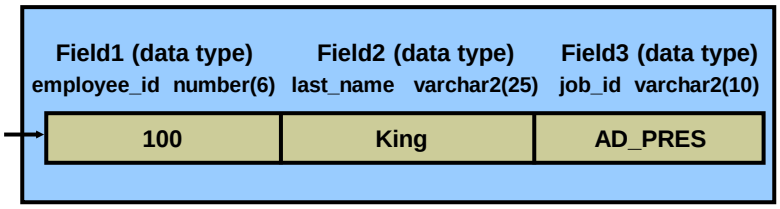
1-4

Declarando Registros com o Atributo %ROWTYPE
Para declarar um registro com base em um conjunto de colunas em uma view ou tabela de banco de dados, use o atributo %ROWTYPE. Os campos no registro obtêm seus nomes e tipos de dados a partir das colunas da tabela ou view. O registro também pode armazenar uma linha inteira de dados extraída
de um cursor ou variável de cursor.
No exemplo a seguir, um registro é declarado usando %ROWTYPE como um especificador de tipo de dados.
O registro, *emp_record,* terá uma estrutura consistindo nos campos a seguir, cada um deles representando uma coluna na tabela EMP.
Observação: Isso não é código, mas simplesmente a estrutura da variável composta.
O Atributo %ROWTYPE
• Declarar uma variável segundo um conjunto de colunas em uma view ou tabela de banco de dados.
• Prefixar %ROWTYPE com a tabela de banco de dados.
• Campos no registro obtêm seus nomes e tipos de dados a partir das colunas da tabela ou view.

Estrutura de Registro PL/SQL



Exemplo



1-5

Fazendo Referência e Inicializando Registros

Os campos em um registro são acessados por nome. Para fazer referência ou inicializar um campo individual, use notação em pontos e a seguinte sintaxe:

Por exemplo, você faz referência ao campo no registro *emp_record* .

É possível, em seguida, atribuir um valor ao campo de registro.

Em um bloco ou subprograma, os registros definidos pelo usuário são concretizados quando você inclui o bloco ou subprograma e deixam de existir quando você sai do bloco ou subprograma.

Atribuindo Valores aos Registros

Você pode atribuir uma lista de valores comuns a um registro usando a instrução SELECT ou FETCH. Certifique-se de que os nomes de colunas aparecem na mesma ordem dos campos no registro. Você também pode atribuir um registro a outro se eles tiverem o mesmo tipo de dados.

Um registro definido pelo usuário e um registro %ROWTYPE nunca têm o mesmo tipo de dados.

Características do Atributo %ROWTYPE

- **O número e tipos de dados das colunas de banco de dados subjacentes podem não ser conhecidas.**
- **O número e tipos de dados da coluna de banco de dados subjacente pode alterar no tempo de execução.**
- **O atributo é útil ao recuperar uma linha com a instrução SELECT *.**

Declarando Registros com o Atributo %ROWTYPE (continuação)
Sintaxe
onde: *identificador* é o nome escolhido para o registro como um todo *referência* é o nome da tabela, view, cursor ou variável de cursor em que o
registro deve ser baseado. (Você deve certificar-se de que essa referência é válida ao declarar o registro, isto é, a tabela ou view precisa existir.)
Pode-se, em seguida, atribuir um valor ao campo de registro.
Vantagens de Usar %ROWTYPE
• O número e tipos de dados das colunas de banco de dados subjacentes podem nãoser conhecidas.
• O número e tipos de dados da coluna de banco de dados subjacente pode alterar no tempo de execução.
• O atributo é útil ao recuperar uma linha com a instrução SELECT.

O Atributo %ROWTYPE

Exemplos

Declarar uma variável para armazenar a mesma informação sobre um departamento que está armazenada na tabela DEPT.

```
dept_record dept%ROWTYPE;
```

Declarar uma variável para armazenar a mesma informação sobre um funcionário que está armazenada na tabela EMP.

```
emp_record emp%ROWTYPE;
```

1-7

Exemplos

A primeira declaração no slide cria um registro com os mesmos nomes de campo e tipos de dados de campo da linha na tabela DEPT. Os campos são DEPTNO, DNAME e LOCATION.

A segunda declaração cria um registro com os mesmos nomes de campo e tipos de dados de campo de uma linha na tabela EMP. Os campos são EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM e DEPTNO.

O Atributo %ROWTYPE

Exemplos

Declarar uma variável para armazenar a mesma informação sobre um departamento que está armazenada na tabela DEPT.

```
dept_record dept%ROWTYPE;
```

Declarar uma variável para armazenar a mesma informação sobre um funcionário que está armazenada na tabela EMP.

```
emp_record emp%ROWTYPE;
```

1-8

No exemplo a seguir, um funcionário está se aposentando. As informações sobre esse funcionário são adicionadas a uma tabela que contém informações sobre funcionários aposentados. O usuário fornece o número do funcionário.

```
DECLARE
emp_rec emp%ROWTYPE;
BEGIN
SELECT * INTO emp_rec
FROM emp
WHERE empno = &employee_number;
INSERT INTO retired_emps(empno, ename, job, mgr,
hiredate,
leavedate, sal, comm, deptno)
VALUES (emp_rec.empno, emp_rec.ename, emp_rec.job,
emp_rec.mgr,
emp_rec.hiredate, SYSDATE, emp_rec.sal, emp_rec.comm,
emp_rec.deptno);
COMMIT;
END
```

O Atributo %ROWTYPE

```
...
DEFINE employee_number = 124
 DECLARE
  emp_rec   employees%ROWTYPE;
 BEGIN
  SELECT * INTO emp_rec FROM employees
  WHERE  employee_id = &employee_number;
  INSERT INTO retired_emps(empno, ename, job, mgr,
  hiredate, leavedate, sal, comm, deptno)
  VALUES (emp_rec.employee_id, emp_rec.last_name,
  emp_rec.job_id,emp_rec.manager_id,
  emp_rec.hire_date, SYSDATE, emp_rec.salary,
  emp_rec.commission_pct, emp_rec.department_id);
END;
/
```

Inserindo um registro com %ROWTYPE

```
...
DEFINE employee_number = 124
DECLARE
   emp_rec  retired_emps%ROWTYPE;
BEGIN
 SELECT employee_id, last_name, job_id, manager_id,
 hire_date, hire_date, salary, commission_pct,
 department_id INTO emp_rec FROM employees
 WHERE  employee_id = &employee_number;
 INSERT INTO retired_emps VALUES emp_rec;
END;
/
SELECT * FROM retired_emps;
```

Atualizando um linha em uma Tabela utilizando um Registro

```
SET SERVEROUTPUT ON
SET VERIFY OFF
DEFINE employee_number = 124
DECLARE
   emp_rec retired_emps%ROWTYPE;
BEGIN
 SELECT * INTO emp_rec FROM retired_emps;
 emp_rec.leavedate:=SYSDATE;
 UPDATE retired_emps SET ROW = emp_rec WHERE
  empno=&employee_number;
END;
/
SELECT * FROM retired_emps;
```

Cursores

1-12

Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

- **Distinguir entre um cursor implícito e um explícito**
- **Usar uma variável de registro PL/SQL**
- **Criar um loop FOR de cursor**

Sobre os Cursores

Cada instrução SQL executada pelo Oracle Server tem um cursor individual associado:

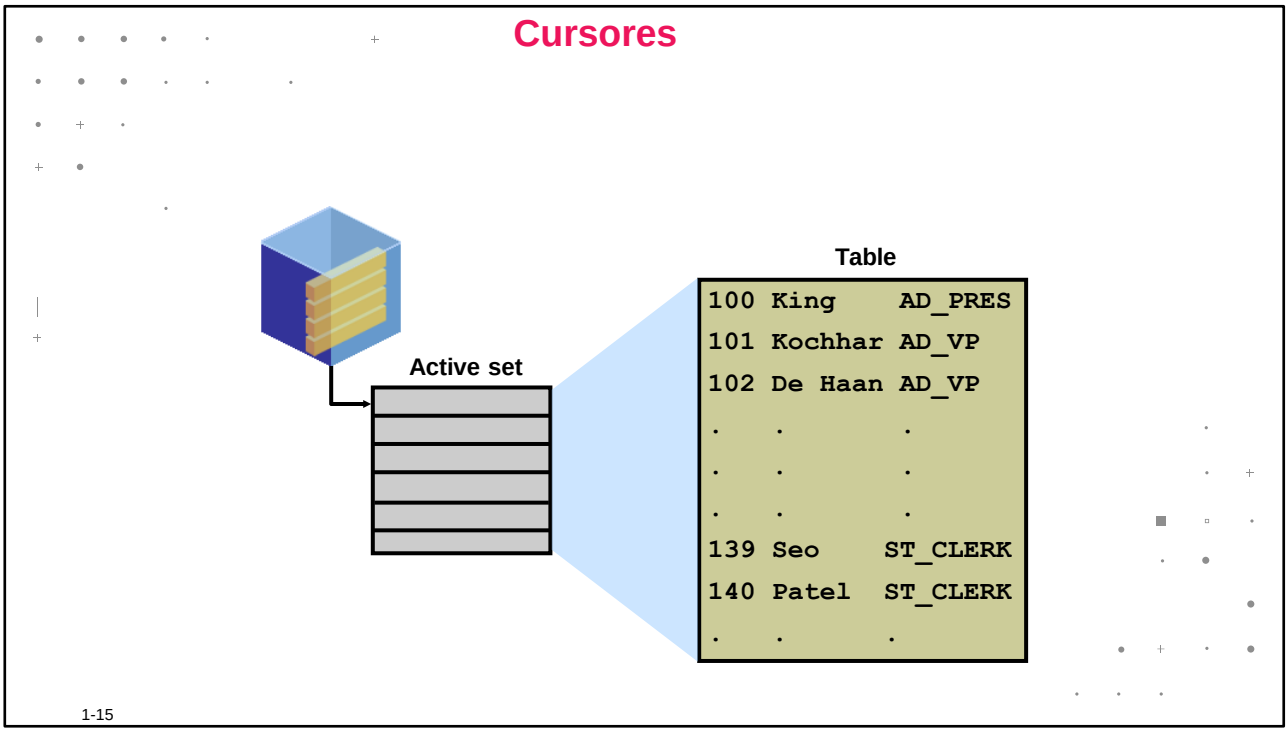
- **Cursores implícitos: Declarados para todas as instruções DML e PL/SQL SELECT**
- **Cursores explícitos: Declarados e nomeados pelo programador**

1-14

Objetivo da Lição

Listas as diferenças entre cursores implícitos e explícitos. Quando e porque usar um cursor explícito.

Você pode precisar usar uma instrução SELECT de várias linhas no PL/SQL para processar diversas linhas. Para isso, você declara e controla cursores explícitos, que são usados em loops, incluindo o loop FOR de cursor.



Cursores Explícitos

Use cursores explícitos para processar individualmente cada linha retornada por uma instrução SELECT de várias linhas.

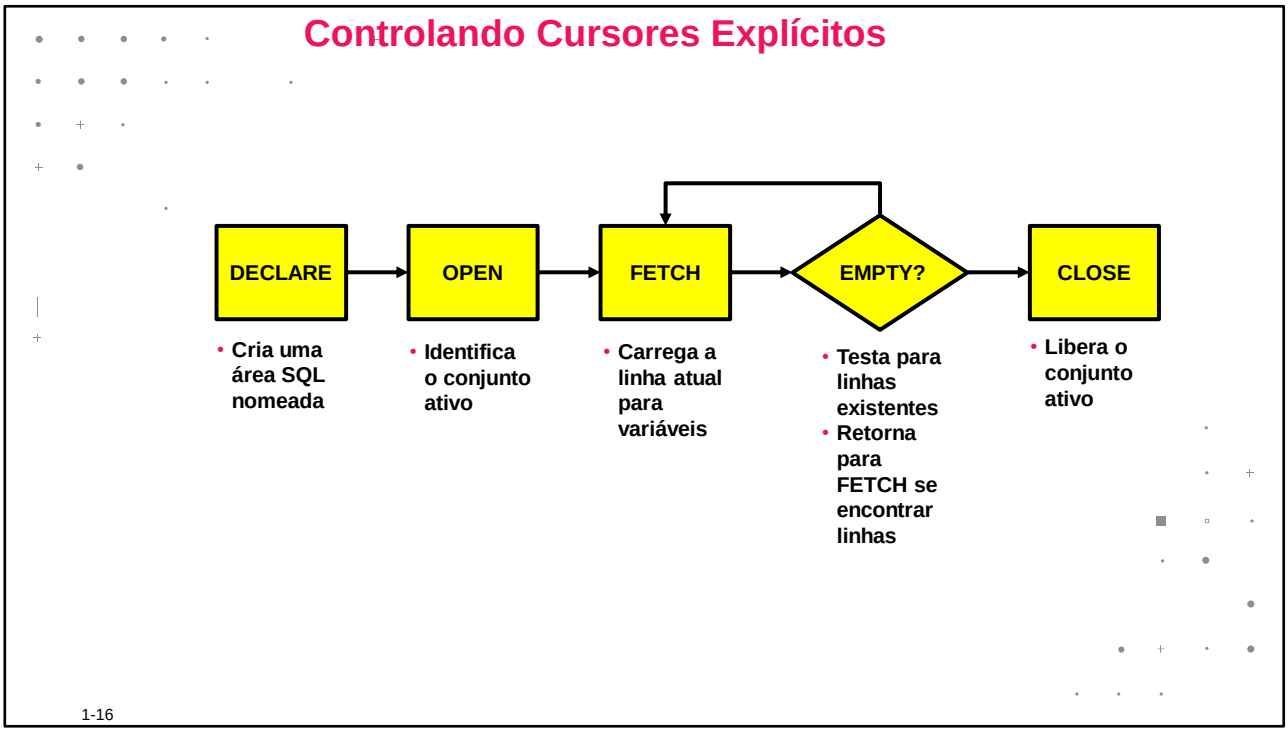
O conjunto de linhas retornado por uma consulta de várias linhas é chamado *conjunto ativo*. Seu tamanho é o número de linhas que atende aos critérios da pesquisa. O diagrama no slide mostra como um cursor explícito "aponta" para a *linha atual* do conjunto ativo. Isso permite que o programa processe as linhas uma de cada vez.

Um programa PL/SQL abre um cursor, processa linhas retornadas por uma consulta e, em seguida, fecha o cursor. O cursor marca a posição atual no conjunto ativo.

Funções do cursor explícito:

- Pode processar além da primeira linha retornada pela consulta, linha por linha
- Controla que linha está sendo processada no momento
- Permite que o programador controle as linhas manualmente no bloco PL/SQL

Observação: A extração de um cursor implícito é uma extração de array e a existência de uma segunda linha ainda criará a exceção TOO_MANY_ROWS. Além disso, você pode usar cursores explícitos para realizar diversas extrações e para executar novamente consultas analisadas na área de trabalho.



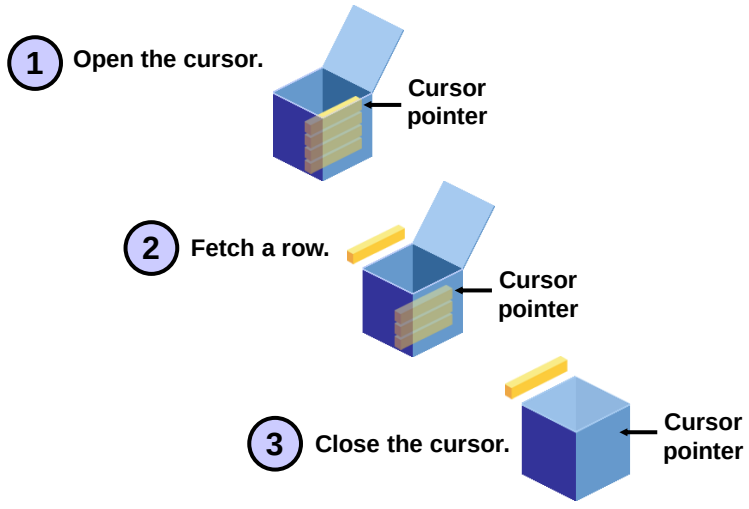
Cursores Explícitos (continuação)

Agora que você obteve uma compreensão conceitual dos cursores, verifique as etapas para usá-los. A sintaxe de cada etapa pode ser encontrada nas páginas a seguir.

Controlando Cursores Explícitos Usando Quatro Comandos

1. Declare o cursor nomeando-o e definindo a estrutura da consulta a ser executada dentro dele.
2. Abra o cursor. A instrução OPEN executa a consulta e vincula as variáveis que estiverem referenciadas. As linhas identificadas pela consulta são chamadas *conjunto ativo* e estão agora disponíveis para extração.
3. Extraia dados do cursor. No diagrama de fluxo mostrado no slide, após cada extração você testa o cursor para qualquer linha existente. Se não existirem mais linhas para serem processadas, você precisará fechar o cursor.
4. Feche o cursor. A instrução CLOSE libera o conjunto ativo de linhas. Agora é possível reabrir o cursor e estabelecer um novo conjunto ativo.

Controlando Cursores Explícitos



1-17

Cursores Explícitos (continuação)

Você usa as instruções OPEN, FETCH e CLOSE para controlar um cursor. A instrução OPEN executa a consulta associada ao cursor, identifica o conjunto ativo e posiciona o cursor (indicador) antes da primeira linha. A instrução FETCH recupera a linha atual e avança o cursor para a próxima linha. Após o processamento da última linha, a instrução CLOSE desativa o cursor.

Declarando um Curso

Sintaxe:

```
CURSOR cursor_name IS
     select_statement;
```

Examplo

```
DECLARE
  CURSOR emp_cursor IS
  SELECT employee_id, last_name FROM employees
  WHERE department_id =30;
```

```
DECLARE
  locid NUMBER:= 1700;
  CURSOR dept_cursor IS
  SELECT * FROM departments
  WHERE location_id = locid;
...
```

1-18

Declaração de Cursor Explícito

Use a instrução CURSOR para declarar um cursor explícito. Pode-se fazer referência a variáveis dentro da consulta, mas você deve declará-las antes da instrução CURSOR.

Na sintaxe:

cursor_name é um identificador do PL/SQL

select_statement é uma instrução SELECT sem uma cláusula INTO

Observação: Não inclua a cláusula INTO na declaração de cursor porque ela aparecerá posteriormente na instrução FETCH.

Declaração de Cursor Explícito (continuação)
Recupere os funcionários um a um.

Recupere os funcionários um a um.

```
DECLARE
  v_empno              emp.empno%TYPE;
  v_ename              emp.ename%TYPE;
  CURSOR emp_cursor IS
    SELECT empno, ename
    FROM   emp;
BEGIN
...
```

/e declará-las

Abrindo um Cursor

```
DECLARE
  CURSOR emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE department_id =30;
...
BEGIN
  OPEN emp_cursor;
```


Insrtrução OPEN

Abe o cursor para executar a consulta e identificar o conjunto ativo, que consiste em todas as linhas que atendem aos critérios de pesquisa da consulta. O cursor aponta agora para a primeira linha do conjunto ativo.

Na sintaxe:

cursor_name é o nome do cursor declarado anteriormente

OPEN é uma instrução executável que realiza as seguintes operações:

1. Aloca memória dinamicamente para uma área de contexto que finalmente conterá informações cruciais de processamento.
2. Analisa a instrução SELECT.
3. Vincula as variáveis de entrada — isto é, define o valor das variáveis de entrada obtendo seus endereços de memória.
4. Identifica o conjunto ativo — isto é, o conjunto de linhas que satisfaz os critérios de pesquisa. As linhas no conjunto ativo não são recuperadas para variáveis quando a instrução OPEN é executada. Em vez disso, a instrução FETCH recupera as linhas.
5. Posiciona o indicador imediatamente antes da primeira linha no conjunto ativo.

Observação: Se a consulta não retornar qualquer linha quando o cursor for aberto, o PL/SQL não criará uma exceção. Entretanto, você pode testar o status do cursor após a extração.

Extraindo Dados do Cursor

```
SET SERVEROUTPUT ON
DECLARE
  CURSOR emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE department_id =30;
  empno employees.employee_id%TYPE;
  lname employees.last_name%TYPE;
BEGIN
  OPEN emp_cursor;
  FETCH emp_cursor INTO empno, lname;
  DBMS_OUTPUT.PUT_LINE( empno ||' '||lname);
  ...
END;
/
```

1-21

nstrução FETCH

A instrução FETCH recupera as linhas no conjunto ativo uma de cada vez. Após cada extração, o cursor avança para a próxima linha no conjunto ativo.

Na sintaxe:

cursor_name é o nome do cursor declarado anteriormente

variável é uma variável de saída para armazenar os resultados

record_name é o nome do registro em que os dados recuperados são armazenados

(A variável de registro pode ser declarada usando o atributo %ROWTYPE.)

Diretrizes

• Inclua o mesmo número de variáveis na cláusula INTO da instrução FETCH do que as colunas na instrução SELECT e certifique-se de que os tipos de dados são compatíveis.

• Faça a correspondência de cada variável para coincidir com a posição das colunas.

• Como alternativa, defina um registro para o cursor e faça referência do registro na cláusula FETCH INTO.

• Teste para verificar se o cursor possui linhas. Se uma extração não obtiver valores, não existem linhas remanescentes para serem processadas no conjunto ativo e nenhum erro será registrado.

Observação: A instrução FETCH realiza as seguintes operações:

1. Avança o indicador para a próxima linha no conjunto ativo.
2. 2. Lê os dados da linha atual para as variáveis PL/SQL de saída.

Extraindo Dados do Cursor

```
SET SERVEROUTPUT ON
DECLARE
  CURSOR emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE  department_id =30;
  empno employees.employee_id%TYPE;
  lname employees.last_name%TYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO empno, lname;
    EXIT WHEN emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE( empno ||' '||lname);
  END LOOP;
  ...
END;
/
```

1-23

Instrução FETCH (continuação)

Use a instrução FETCH para recuperar os valores da linha ativa para variáveis de saída. Após a extração, você pode manipular as variáveis através de instruções futuras. Para cada valor de coluna retornado pela consulta associada ao cursor, deve existir uma variável correspondente na lista INTO.

Além disso, seus tipos de dados devem ser compatíveis.

Recupere os primeiros 10 funcionários um por um.

```
DECLARE
  v_empno   emp.empno%TYPE;
  v_ename   emp.ename%TYPE;
  CURSOR    emp_cursor IS
    SELECT empno, ename
    FROM   emp;
BEGIN
  OPEN emp_cursor;
  FOR i IN 1..10 LOOP
    FETCH emp_cursor INTO v_empno, v_ename;
    ...
  END LOOP;
END ;
```

Fechando o Cursor

```
...
  LOOP
    FETCH emp_cursor INTO empno, lname;
    EXIT WHEN emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE( empno ||' '||lname);
  END LOOP;
 CLOSE emp_cursor;
END;
/
```


Instrução CLOSE

A instrução CLOSE desativa o cursor e o conjunto ativo se torna indefinido. Feche o cursor após completar o processamento da instrução SELECT. Essa etapa permite que o cursor seja reaberto, se necessário. Assim, você pode estabelecer um conjunto ativo diversas vezes.

Na sintaxe:

cursor_name é o nome do cursor declarado anteriormente.

Não tente extrair dados de um cursor após ele ter sido fechado ou será criada a exceção

INVALID_CURSOR.

Observação: A instrução CLOSE libera a área de contexto.

Embora seja possível terminar o bloco PL/SQL sem fechar cursores, você deve criar o hábito de fechar qualquer cursor declarado explicitamente para liberar recursos.

Existe um limite máximo para o número de cursores abertos por usuário, que é determinado pelo parâmetro OPEN_CURSORS no campo de parâmetros do banco de dados. OPEN_CURSORS = 50 por default.

Cursores e Registros

Processe as linhas do active set ao buscar valores em um registro PL/SQL.

```
DECLARE
  CURSOR emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE  department_id =30;
   emp_record  emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO emp_record;
  ...
```

1-25

Cursores e Registros

Você já constatou que pode definir registros para usar a estrutura de colunas em uma tabela. Você também pode definir um registro com base na lista selecionada de colunas em um cursor explícito.

Isso é conveniente para processar as linhas do conjunto ativo, porque você pode simplesmente extrair para o registro. Assim, os valores das linhas são carregados diretamente para os campos correspondentes do registro.

Loops FOR de Cursor

Syntax:

```
FOR record_name IN cursor_name LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

- **O loop FOR de cursor é um atalho para processar cursores explícitos.**
- **Ocorrem abertura, extração e fechamento implícitos.**
- **O registro é declarado implicitamente.**

1-26

Loops FOR de Cursor

Um loop FOR de cusor processa linhas em um cursor explícito. Ele é um atalho porque o cursor é aberto, linhas são extraídas uma vez para cada iteração no loop e o cursor é fechado automaticamente após o processamento de todas as linhas. O loop em si é terminado automaticamente ao final da iteração quando a última linha for extraída.

Na sintaxe:

record_name é o nome do registro declarado implicitamente

cursor_name é um identificador PL/SQL para o cursor declarado anteriormente

Diretrizes

- Não declare o registro que controla o loop. Seu escopo é somente no loop.
- Teste os atributos do cursor durante o loop, se necessário.
- Forneça os parâmetros de um cursor, se necessário, entre parênteses após o nome do cursor na instrução FOR. Mais informações sobre parâmetros de cursor são abrangidas em uma lição subsequente.
- Não use um loop FOR de cursor quando as operações do cursor precisarem ser manipuladas manualmente.

Loops FOR de Cursor

Recuperar funcionários um a um até não restar nenhum.

Exemplo

```
SET SERVEROUTPUT ON
DECLARE
  CURSOR emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE department_id =30;
BEGIN
   FOR emp_record IN emp_cursor
    LOOP
     DBMS_OUTPUT.PUT_LINE( emp_record.employee_id
     ||' ' ||emp_record.last_name);
    END LOOP;
END;
/
```

Atributos do Cursor Explícito

Obter informações de status sobre um cursor.

Atributo	Tipo	Descrição
%ISOPEN	Booleano	Será avaliado para TRUE se o cursor estiver aberto
%NOTFOUND	Booleano	Será avaliado para TRUE se a extração mais recente não retornar uma linha
%FOUND	Booleano	Será avaliado para TRUE se a extração mais recente não retornar uma linha; complemento de %NOTFOUND
%ROWCOUNT	Número	Será avaliado para o número total de linhas retornadas até o momento

Atributos do Cursor Explícito

Da mesma forma que para cursores implícitos, existem quatro atributos para obter informações de status sobre um cursor. Quando anexado ao nome da variável do cursor, esses atributos retornam informações úteis sobre a execução de uma instrução de manipulação de dados.

O Atributo %ISOPEN

Extrair linhas somente quando o cursor estiver aberto.

• Usar o atributo de cursor %ISOPEN antes de executar uma extração para testar se o cursor está aberto.

Exemplo

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

1-29

Atributos do Cursor Explícito

• Você pode extrair linhas somente quando o cursor está aberto. Use o atributo de cursor %ISOPEN para determinar se o cursor está aberto, se necessário.

• Extraia linhas em um loop. Use atributos de cursor para determinar o momento para sair do loop.

• Use o atributo de cursor %ROWCOUNT para recuperar um número exato de linhas, extrair as linhas em um loop FOR numérico ou extrair as linhas em um loop simples e determinar o momento para sair do loop.

Observação: %ISOPEN retorna o status do cursor: TRUE se ele estiver aberto e FALSE se não estiver. Não é normalmente necessário examinar %ISOPEN.

%ROWCOUNT e %NOTFOUND: Exemplo

```
SET SERVEROUTPUT ON
DECLARE
  empno  employees.employee_id%TYPE;
  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 empno, ename;
   EXIT WHEN emp_cursor%ROWCOUNT > 10 OR
                      emp_cursor%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(TO_CHAR(empno)
                        ||' '|| ename);
  END LOOP;
  CLOSE emp_cursor;
END ;
/
```

1-30

Observação: Antes da primeira extração, %NOTFOUND é avaliado para NULL. Assim, se FETCH nunca executar com êxito, jamais ocorrerá saída do loop. Isso porque a instrução EXIT WHEN executará somente se sua condição WHEN for verdadeira. Como segurança, convém usar a seguinte
instrução EXIT:
Se usar %ROWCOUNT, adicione um teste para nenhuma linha no cursor usando o atributo %NOTFOUND, já que a contagem de linhas não será incrementada se a extração não recuperar qualquer linha.

Loops FOR do Cursor Usando Subconsultas

Não é necessário declarar o cursor.

Exemplo

```
SET SERVEROUTPUT ON
BEGIN
  FOR emp_record IN (SELECT employee_id, last_name
   FROM employees WHERE department_id =30)
  LOOP
   DBMS_OUTPUT.PUT_LINE( emp_record.employee_id ||'
   '||emp_record.last_name);
  END LOOP;
END;
/
```

1-31

Loops FOR de Cursor

Um loop FOR de cusor processa linhas em um cursor explícito. Ele é um atalho porque o cursor é aberto, linhas são extraídas uma vez para cada iteração no loop e o cursor é fechado automaticamente após o processamento de todas as linhas. O loop em si é terminado automaticamente ao final da iteração quando a última linha for extraída.

Na sintaxe:

record_name é o nome do registro declarado implicitamente

cursor_name é um identificador PL/SQL para o cursor declarado anteriormente

Diretrizes

- Não declare o registro que controla o loop. Seu escopo é somente no loop.
- Teste os atributos do cursor durante o loop, se necessário.
- Forneça os parâmetros de um cursor, se necessário, entre parênteses após o nome do cursor na instrução FOR. Mais informações sobre parâmetros de cursor são abrangidas em uma lição subsequente.
- Não use um loop FOR de cursor quando as operações do cursor precisarem ser manipuladas manualmente.

Observação: Você pode definir uma consulta no início do próprio loop. A expressão da consulta é chamada sub instrução SELECT e o cursor é interno para o loop FOR. Como o cursor não é declarado com um nome, não é possível testar seus atributos.

Cursors com Subqueries

Exemplo

```
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;
...
```


Loops FOR do Cursor Usando Subconsultas
Você não precisa declarar um cursor porque o PL/SQL permite substituição por uma subconsulta. Este exemplo realiza o mesmo do exemplo na página anterior. Ele é o código completo do slide acima.

Sumário

Tipos de cursor:
– Cursores implícitos: Usados em todas as instruções DML e consultas de linha única.
– Cursores explícitos: Usados para consultas de zero, uma ou mais linhas.
• É possível manipular cursores explícitos.
• É possível avaliar o status do cursor usando atributos de cursor.
• É possível usar loops FOR de cursor.

OBRIGADO

profalexandre.barcelos@fiap.com.br

https://www.linkedin.com/in/alexandrebarcelos

FIAP


FIAP