BANCO DE DADOS II

Cursores

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Msc. Eliezio Soares
eliezio.soares@ifrn.edu.br | https://docente.ifrn.edu.br/elieziosoares

CURSOR

Cursores são utilizados para processar uma tupla de cada vez dentro do programa.

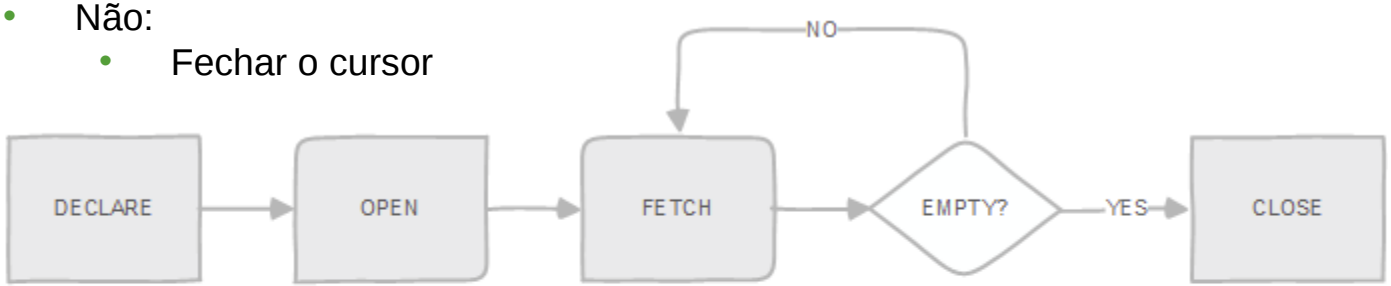
Um cursor pode ser representado através da metáfora de um ponteiro que aponta para uma única tupla do resultado de uma consulta que retornou diversas tuplas.

Um Cursor permite encapsular uma query e processar uma tupla por vez. Usa-se cursor para dividir um grande result set em partes e processá-las individualmente, evitando overflow de memória.

CURSOR

O diagrama abaixo ilustra o uso de um cursor em PostgreSQL:

1. Declarar um cursor;
2. Abrir o cursor;
3. Fetch (buscar) linhas do result set;
4. Checar se há linhas restantes para fetch;
   - Sim:
     - Passo 3;
   - Não:
     - Fechar o cursor



http://www.postgresqltutorial.com/plpgsql-cursor/

DECLARAÇÃO DE CURSORES

Há dois tipos de cursores:

- Unbound
- Bound

Unbound:
Todo acesso se dá por meio de uma variável de cursor.

Bound:
A declaração é ligada diretamente uma query.

DECLARAÇÃO DE CURSORES UNBOUND

Para ter acesso ao cursor é necessário declarar uma variável.

O PostgreSQL provê um tipo especial - REFCURSOR.

Nesse caso a variável pode ser utilizada com qualquer query.

```
DECLARE
        meu_cursor REFCURSOR;
```

DECLARAÇÃO DE CURSORES BOUND

Para ter acesso ao cursor é necessário declarar uma variável, mas nesse caso a variável está ligada (bound) a uma query especificada.

```
DECLARE
    cur_filme  CURSOR FOR SELECT * FROM filme;
```

DECLARAÇÃO DE CURSORES

```
DECLARE
    curs1 refcursor;
    curs2 CURSOR FOR SELECT * FROM tenk1;
    curs3 CURSOR (key integer) FOR SELECT * FROM tenk1 WHERE unique1 = key;
```

https://www.postgresql.org/docs/9.6/static/plpgsql-cursors.html

1. curs1:
   - Pode ser utilizada com qualquer query.
2. curs2:
   - Está ligada ao result set equivalente a todas as tuplas da relação tenk1.
3. curs3:
   - key será substituido por um inteiro quando o cursor for aberto.
   - Está ligada ao result set equivalente às tuplas da relação tenk1 que tenham o valor de unique1 igual ao valor de key.

DECLARAÇÃO DE CURSORES

```
name [ [ NO ] SCROLL ] CURSOR [ ( arguments ) ] FOR query;
```

https://www.postgresql.org/docs/9.6/static/plpgsql-cursors.html

- **name**: nome da variável
- **[NO] SCROLL:** Define se o cursor é ou não capaz de executar fetches para trás.
  - Default: Scroll
- **arguments:** Se utilizado deve apresentar uma lista de parâmetros contendo nome e datatype. Os itens devem ser separados por virgula.

ABRINDO UM UNBOUND CURSOR

```
OPEN unbound_cursorvar [ [ NO ] SCROLL ] FOR query;
```

https://www.postgresql.org/docs/9.6/static/plpgsql-cursors.html

A variável cursor é aberta e dada uma query específica para executar.

O cursor não pode ter sido aberto previamente.

Deve ser utilizado um cursor **unbound.**

A query deve ser um SELECT, ou algo que retorne linhas.

```
OPEN curs1 FOR SELECT * FROM foo WHERE key = mykey;
```

https://www.postgresql.org/docs/9.6/static/plpgsql-cursors.html

ABRINDO UM BOUND CURSOR

```
OPEN bound_cursorvar [ ( [ argument_name := ] argument_value [, ...] ) ];
```

https://www.postgresql.org/docs/9.6/static/plpgsql-cursors.html

O cursor não pode ter sido aberto previamente.

Uma lista dos valores dos argumentos devem ser declarados se o cursor foi declarado com argumentos.

Os argumentos podem ser passados usando uma notação posicional ou nominal.

```
OPEN curs2;
OPEN curs3(42);
OPEN curs3(key := 42);
```

https://www.postgresql.org/docs/9.6/static/plpgsql-cursors.html

USANDO CURSOR - FETCH

```
FETCH [ direction { FROM | IN } ] cursor INTO target;
```

https://www.postgresql.org/docs/9.6/static/plpgsql-cursors.html

O comando **FETCH** recupera a próxima linha do cursor e atribui a uma variável **target**

- Pode ser uma linha ou uma lista de variáveis - semelhante ao select into.
- Caso não haja mais linhas, será atribuído NULL à variável.

Por padrão um cursor recupera a próxima linha. As direções válidas são:

- NEXT
- LAST
- PRIOR
- FIRST
- ALL
- ABSOLUTE count
- RELATIVE count
- FORWARD (SCROLL)
- BACKWARD (SCROLL)

USANDO CURSOR - FETCH

```
FETCH curs1 INTO rowvar;
FETCH curs2 INTO foo, bar, baz;
FETCH LAST FROM curs3 INTO x, y;
FETCH RELATIVE -2 FROM curs4 INTO x;
```

https://www.postgresql.org/docs/9.6/static/plpgsql-cursors.html

...

USANDO CURSOR - MOVE

O comando MOVE reposiciona um cursor semelhantemente ao FETCH, no entanto sem recuperar qualquer dado.

```
MOVE [ direction { FROM | IN } ] cursor;
```

```
MOVE curs1;
MOVE LAST FROM curs3;
MOVE RELATIVE -2 FROM curs4;
MOVE FORWARD 2 FROM curs4;
```

https://www.postgresql.org/docs/9.6/static/plpgsql-cursors.html

USANDO CURSOR
UPDATE / DELETE WHERE CURRENT OF

Uma linha obtida ao percorrer o cursor pode ser atualizada ou removida utilizando os comandos **UPDATE** e **DELETE**, bem como utilizar a expressão **CURRENT OF** para identificar a linha atual.

```
UPDATE table SET ... WHERE CURRENT OF cursor;
DELETE FROM table WHERE CURRENT OF cursor;
```

```
UPDATE foo SET dataval = myval WHERE CURRENT OF curs1;
```

https://www.postgresql.org/docs/9.6/static/plpgsql-cursors.html

USANDO CURSOR - CLOSE

O comando **CLOSE** fecha um cursor. É utilizado para liberar os recursos antes do fim da transação, ou liberar a variável do cursor para ser aberto novamente.

```
CLOSE curs1;
```

https://www.postgresql.org/docs/9.6/static/plpgsql-cursors.html

USANDO CURSOR - EXEMPLO

```
CREATE OR REPLACE FUNCTION refAlunos()
RETURNS void AS $$
DECLARE
        exemplo_cursor_alunos CURSOR FOR SELECT * from ALUNOS;
        aluno alunos%ROWTYPE;
BEGIN
   OPEN exemplo_cursor_alunos;

   FETCH FIRST FROM exemplo_cursor_alunos into aluno;
   RAISE NOTICE 'Nome: %', aluno.nom_alu;

   FETCH exemplo_cursor_alunos into aluno;
   RAISE NOTICE 'Nome: %', aluno.nom_alu;

   FETCH LAST FROM exemplo_cursor_alunos into aluno;
   RAISE NOTICE 'Nome: %', aluno.nom_alu;

   CLOSE exemplo_cursor_alunos;
END;
$$LANGUAGE plpgsql;
```

```
BEGIN;
        SELECT refAlunos();
COMMIT;
```

USANDO CURSOR - EXEMPLO

Utilizando o **FOR** para iterar um cursor:

- O cursor é automaticamente aberto e fechado após o fim do LOOP;
- A variável de iteração é automaticamente definida como tipo record;

```
CREATE OR REPLACE FUNCTION refAlunos2()
RETURNS void AS $$
DECLARE
        exemplo_cursor_alunos CURSOR FOR SELECT * from ALUNOS;
BEGIN
   FOR aluno IN exemplo_cursor_alunos LOOP
        RAISE NOTICE 'Nome: %', aluno.nom_alu;
   END LOOP;
END;
$$LANGUAGE plpgsql;
```

```
BEGIN;
        SELECT refAlunos2();
COMMIT;
```

DÚVIDAS?

ATIVIDADE

1. Implemente uma função que utilize cursor para percorrer a relação historicos_escolares e imprima todos as matriculas com nota abaixo de 6.

2. Implemente uma função que utilize cursor para percorrer a relação cursos e caso não tenha coordenador definido, atualize o registro vinculando-o ao professor mais antigo que dê aula nesse curso.

   * Caso não haja cursos sem coordenador, altere um dos cursos removendo o cod_coord para testar o cursor.

REFERÊNCIAS BIBLIOGRÁFICAS

PostgreSQL 9.6.5 Documentation. Disponível em: <https://www.postgresql.org/docs/9.6/static/plpgsql-cursors.html#PLPGSQL-OPEN-BOUND-CURSOR>. Acesso em 15 Nov. 2017.

PL/pgSQL Cursor. Disponível em: <http://www.postgresqltutorial.com/plpgsql-cursor/
>. Acesso em 15 Nov. 2017.