Cursores

Cursores permitem que as linhas de uma consulta SQL sejam recuperadas aos poucos para evitar problemas com o uso de memória em consultas que retornam muitas linhas.

O padrão do SQL define cursores apenas para SQL embutido em aplicações. O PostgreSQL permite a utilização de cursores de modo interativo.

DECLARE

O comando DECLARE cria um cursor.

```
DECLARE nome [ BINARY ] [ INSENSITIVE ] [ [ NO ] SCROLL ]
    CURSOR [ { WITH | WITHOUT } HOLD ] FOR comando
```

Onde

nome – nome do cursor criado

comando – comando SELECT ou VALUES que gera as linhas do cursor

Normalmente um cursor retorna os dados em formato texto, para que os dados sejam retornados em formato binário, deve ser especificada a cláusula BINARY.

A cláusula INSENSITIVE indica que o cursor não é afetado por alterações em tabelas subjacentes ao cursor. Como no PostgreSQL todos os cursores são INSENTITIVE, essa cláusula e mantida apenas por compatibilidade com o padrão SQL.

DECLARE

A cláusula **NO SCROLL** determina que as linhas do cursor não podem ser recuperadas de modo não sequencial. Mesmo sem o uso da cláusula **SCROLL**, normalmente os cursores podem ser utilizados para recuperar as linhas de modo não sequencial, mas dependendo da consulta SQL isso pode não ser confiável. Se necessário recuperar as linhas de modo não sequencial, a cláusula **SCROLL** deve ser especificada para garantir esse comportamento.

A cláusula **WITH HOLD** especifica que o cursor pode ser utilizado após a transação que o criou ter sido efetivada com sucesso. O padrão é **WITHOUT HOLD** que especifica que o cursor não pode ser utilizados após o final da transação. Se a transação que criou o cursor for interrompida, o cursor será removido.

Se o comando **DECLARE** for utilizado fora de um bloco de transação, é obrigatório o uso da cláusula **WITH HOLD** pois o cursor não teria validade após a execução do comando.

DECLARE

Exemplo:

```
DECLARE alunocursor SCROLL CURSOR WITH HOLD FOR SELECT *
FROM aluno;
```

FETCH

O comando **FETCH** permite recuperar as linhas de um cursor.

```
FETCH [ direção { FROM | IN } ] nome_do_cursor
```

onde **direção** pode ser omitida ou pode ser um entre:

```
NEXT
PRIOR
FIRST
LAST
ABSOLUTE contador
RELATIVE contador
contador
ALL
FORWARD
FORWARD contador
FORWARD ALL
BACKWARD
BACKWARD contador
BACKWARD ALL
```

FETCH

O cursor possui uma posição associada, que é utilizada pelo comando FETCH. A posição do cursor pode ser: antes da primeira linha, em uma determinada linha, ou após a última linha do resultado da consulta. Ao ser criado, o cursor fica posicionado antes da primeira linha. Após retornar algumas linhas, o cursor fica posicionado na linha retornada mais recentemente. Se o comando FETCH ultrapassar o final das linhas disponíveis, então o cursor ficará posicionado após a última linha, ou antes da primeira linha se estiver retornando linhas para trás. Os comandos FETCH ALL e FETCH BACKWARD ALL sempre deixam o cursor posicionado após a última linha ou antes da primeira linha, respectivamente.

As formas NEXT, PRIOR, FIRST, LAST, ABSOLUTE e RELATIVE retornam uma única linha após mover o cursor de forma apropriada. Se a linha não existir, será retornado um resultado vazio, e o cursor será deixado posicionado antes da primeira linha ou após a última linha, conforme for apropriado.

FETCH

As formas que utilizam FORWARD e BACKWARD retornam o número indicado de linhas movendo o cursor para frente ou para trás, deixando o cursor posicionado na última linha retornada; ou após/antes de todas as linhas se o contador exceder o número de linhas disponíveis

RELATIVE 0, FORWARD 0 e BACKWARD 0 requerem que seja retornada a linha corrente sem mover o cursor, ou seja, retorna novamente a linha retornada mais recentemente. Sempre é bem-sucedido, a menos que o cursor esteja posicionado antes da primeira linha ou após a última linha; nestes casos, não é retornada nenhuma linha.

FETCH

O cursor deve ser declarado com a opção SCROLL se houver intenção de utilizar qualquer variante do comando FETCH que não seja FETCH NEXT ou FETCH FORWARD com um contador positivo. Para consultas simples o PostgreSQL permite retornar linhas rolando para trás a partir de cursores não declarados com a opção SCROLL, mas não é possível confiar nesse comportamento sem a cláusula SCROLL. Se o cursor for declarado com a opção NO SCROLL, então não será permitido retornar linhas rolando para trás.

FETCH

Exemplo:

```
FETCH FROM alunocursor;
 matricula |      nome        |    rg    | curso | serie | turma
-----------+------------------+----------+-------+-------+-------
         1 | Ana Lucia        | 20143531 | 0001  | 1     | A

FETCH 4 FROM alunocursor;
 matricula |      nome        |    rg    | curso | serie | turma
-----------+------------------+----------+-------+-------+-------
         2 | Luis Claudio     | 22336362 | 0001  | 1     | A
         3 | Marcelo          | 25343256 | 0001  | 1     | A
         4 | Debora           | 20356328 | 0001  | 1     | B
         5 | Fernanda         | 26344325 | 0001  | 1     | B
```

MOVE

O comando **MOVE** posiciona o cursor.

```
MOVE [ direção { FROM | IN } ] nome_do_cursor
```

O comando **MOVE** tem a mesma sintaxe e funcionamento que o comando **FETCH** mas sem retornar nenhuma linha.

O padrão do SQL não define o comando **MOVE**.

MOVE

Exemplo:

```
MOVE BACKWARD 2 FROM alunocursor;

FETCH RELATIVE 0 FROM alunocursor;
 matricula |      nome        |    rg    | curso | serie | turma
-----------+------------------+----------+-------+-------+-------
         3 | Marcelo          | 25343256 | 0001  | 1     | A
```

CLOSE

O comando CLOSE fecha um cursor.

```
CLOSE nome
```

O comando CLOSE libera os recursos associados a um cursor aberto. Após o cursor ser fechado, não é permitida nenhuma operação posterior que o utilize.

Todo cursor aberto sem a cláusula HOLD é fechado implicitamente quando a transação termina por um COMMIT ou ROLLBACK. O cursor aberto com a cláusula HOLD é fechado implicitamente quando a transação em que foi criado é interrompida através de ROLLBACK. Se a transação que criou o cursor for efetivada com sucesso, o cursor aberto com a cláusula HOLD permanecerá aberto até ser executado um CLOSE explícito, ou o encerramento da sessão.

CLOSE

Exemplo:

```
CLOSE alunocursor;
```

Cursores Atualizáveis

Um cursor pode permitir a atualização da tabela utilizada no comando SELECT do cursor.

Para que o cursor seja atualizável, o comando SELECT deve ser um comando simples sobre uma tabela. Não deve haver junções ou agregações.

Um cursor só permite atualização dentro da mesma transação que criou o cursor.

Os comandos UPDATE e DELETE permitem o uso da cláusula WITH CURRENT OF nome_do_cursor, para alterar ou deletar o registro correspondente a posição corrente do cursor.

Cursores Atualizáveis

Exemplo:

```
BEGIN;
DECLARE alunocursor CURSOR WITH HOLD FOR SELECT * FROM
aluno;
FETCH 4 FROM alunocursor;
 matricula |      nome       |    rg    | curso | serie | turma
-----------+-----------------+----------+-------+-------+-------
         1 | Ana Lucia       | 20143531 | 0001  | 1     | A
         2 | Luis Claudio    | 22336362 | 0001  | 1     | A
         3 | Marcelo         | 25343256 | 0001  | 1     | A
         4 | Debora          | 20356328 | 0001  | 1     | B
UPDATE aluno SET nome='Paula' WHERE CURRENT OF
alunocursor;
SELECT * FROM aluno WHERE matricula='4';
 matricula |      nome       |    rg    | curso | serie | turma
-----------+-----------------+----------+-------+-------+-------
         4 | Paula           | 20356328 | 0001  | 1     | B
```

Cursores Atualizáveis

O cursor não é atualizado automaticamente com o novo valor da tabela.

```
FETCH RELATIVE 0 FROM alunocursor;
 matricula |      nome       |    rg    | curso | serie | turma
-----------+-----------------+----------+-------+-------+-------
         4 | Debora          | 20356328 | 0001  | 1     | B
COMMIT;
FETCH RELATIVE 0 FROM alunocursor;
 matricula |      nome       |    rg    | curso | serie | turma
-----------+-----------------+----------+-------+-------+-------
         4 | Debora          | 20356328 | 0001  | 1     | B
SELECT * FROM aluno WHERE matricula=4;
 matricula |      nome       |    rg    | curso | serie | turma
-----------+-----------------+----------+-------+-------+-------
         4 | Paula           | 20356328 | 0001  | 1     | B
```

Cursores Atualizáveis

Para evitar que outras transações alterem os registros obtidos pelo cursor, é recomendável a utilização da cláusula FOR UPDATE no comando SELECT do cursor.

Desta forma, os registros serão bloqueados na primeira vez que forem recuperados pelo cursor, impedindo que esses registros sejam alterados por outras transações.

Sem o bloqueio dos registros, as alterações feitas por outras transações podem impedir que as atualizações no cursor tenham efeito na tabela.

As cláusulas WITH HOLD e SCROLL não pode ser utilizada com as cláusulas FOR UPDATE ou FOR SHARE do comando SELECT.