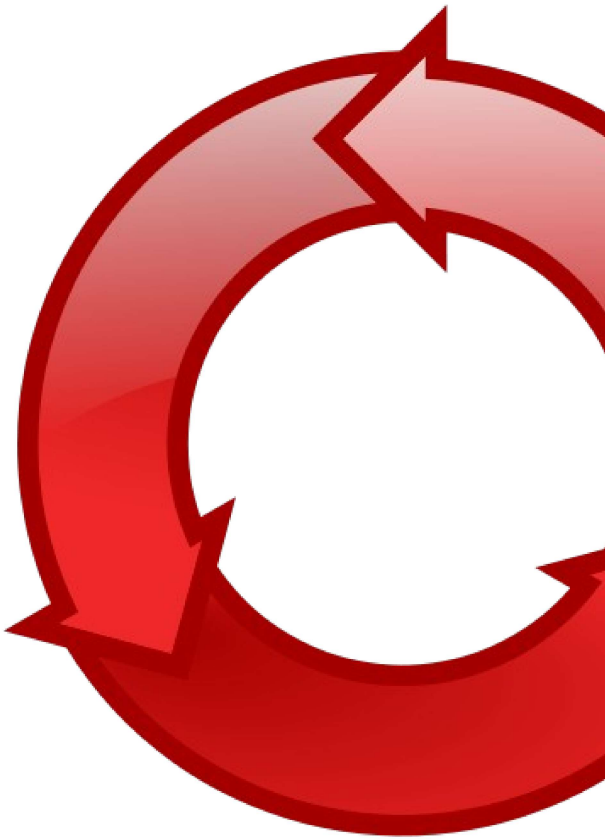
Banco de Dados II

Parte 9 – Curso

Professor Eduardo

Introdução

- Quando se executa um comando SELECT no banco de dados o resultado é um conjunto de dados que atendem aos critérios de seleção, mas que são recuperados em bloco. Todas as linhas do resultado são obtidas de uma vez só.
  - Qualquer tratamento de formatação condicional ou decisão lógica que dependa do conteúdo de cada linha individual deve ser tratada no próprio comando SELECT.
  - Isso pode tornar-se uma operação difícil de implementar, a depender da complexidade da lógica envolvida.



Definição de Cursor

- O uso de cursores pode simplificar a organização da estratégia de manipulação dos dados recuperados.

- Um cursor é um mecanismo de programação SQL que permite armazenar em área temporária o resultado de um SELECT e fornece mecanismos para que estes dados sejam acessados sequencialmente, linha a linha, em um loop controlado pela lógica do desenvolvedor.

Exemplo de Cursor (MySQL)

```
CREATE TABLE VEICULOS(
        VEICULO varchar(32) NOT NULL,
        QUILOMETRAGEM int NOT NULL);

INSERT INTO VEICULOS (VEICULO, QUILOMETRAGEM)
        VALUES ('Carro 1',  '5230');
INSERT INTO VEICULOS (VEICULO, QUILOMETRAGEM)
        VALUES ('Carro 2',  '15800');
INSERT INTO VEICULOS (VEICULO, QUILOMETRAGEM)
        VALUES ('Carro 3',  '2307');
INSERT INTO VEICULOS (VEICULO, QUILOMETRAGEM)
        VALUES ('Carro 4',  '26412');
INSERT INTO VEICULOS (VEICULO, QUILOMETRAGEM)
        VALUES ('Carro 5',  '9730');
```

Primeiramente
criar e povoar u
de veículos qu
fonte de dados
exemplo.

Exemplo de Cursor (MySQL)

```
DROP PROCEDURE IF EXISTS SomaKmVeiculos;

DELIMITER $$
CREATE PROCEDURE SomaKmVeiculos (OUT resultado
  INT)
BEGIN
    -- Definição de variáveis
    -- utilizadas na Procedure
    DECLARE existem_mais_linhas INT DEFAULT 0;
    DECLARE Km_do_veiculo INT DEFAULT 0;
    DECLARE Km_total INT DEFAULT 0;
```

Agora vamos c
STORED PRO
que vai efetu
simples soma
quilometragem
veículos ca
usando um curs

Neste primeir
vamos iniciar a
criando algumas
que serão usada

Exemplo de Cursor (MySQL)

(Continuação da stored procedure SomaKmVeiculos)

```
-- Definição do cursor
DECLARE meuCursor CURSOR FOR
        SELECT QUILOMETRAGEM FROM VEICULOS;

-- Definição da variável de
-- controle de looping do cursor
DECLARE CONTINUE HANDLER FOR
        NOT FOUND SET existem_mais_linhas=1;

-- Abertura do cursor
OPEN meuCursor;
```

No segundo t
procedure, vam
nosso cursor, es
uma variável de
de loop para
mesmo e in
carregando os d

Exemplo de Cursor (MySQL)

(Continuação da stored procedure SomaKmVeiculos)

```
-- Looping de execução do cursor
meuLoop: LOOP
    -- Carregar os dados da próxima linha do cursor
    FETCH meuCursor INTO Km_do_veiculo;
    -- Testar se existe mais registros para tratar
    IF existe_mais_linhas = 1
       THEN LEAVE meuLoop;
    END IF;
    -- Soma a quilometragem do registro atual
    -- com o total acumulado
    SET km_total = km_total + km_do_veiculo;
    -- Retorna para a primeira linha do loop
    END LOOP meuLoop;
```

No terceiro
procedure,
executar a
dos dados
dentro de u
acrescentan
quilometrage
cada veículo
variável de s

Exemplo de Cursor (MySQL)

```
(Continuação da stored procedure SomaKmVeiculos)
    -- Setando a variável com o resultado final
    SET resultado = km_total;
    -- Fechamento do cursor
    CLOSE meuCursor;
END
$$
DELIMITER ;

(Executando a stored procedure SomaKmVeiculos)
SET @variavel_temporaria = 0;
CALL SomaKmVeiculos(@variavel_temporaria);
SELECT @variavel_temporaria;
```

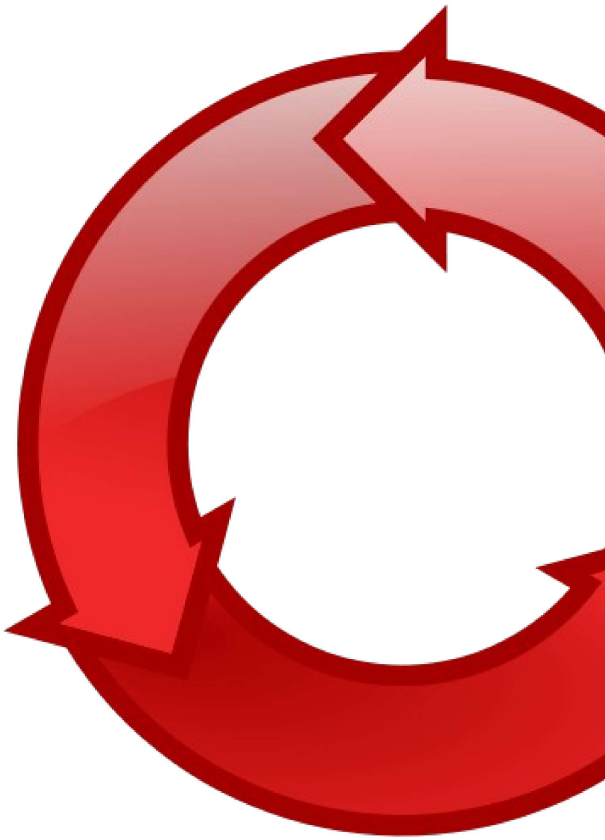
No quarto
trecho da proc
fora do loop
vamos colo
resultado fi
variável de
fechar o cu
encerrar a
procedure.

Resumindo...

1. Definir o cursor e o comando SELECT que obtém os dados.
2. Definir uma variável para identificar o fim dos dados do curso
3. Abrir o cursor (executar o SELECT para obter os dados).
4. Iniciar um loop.
    4.1. Ler uma linha de dados (e colocar em variáveis).
    4.2. Testar se o cursor chegou ao fim dos dados
        (se chegou, sair do loop, do contrário continuar no loop).
    4.3. Tratar a linha de dados lida com a lógica necessária.
    4.4. Retornar ao início do loop.
5. Fechar o cursor.

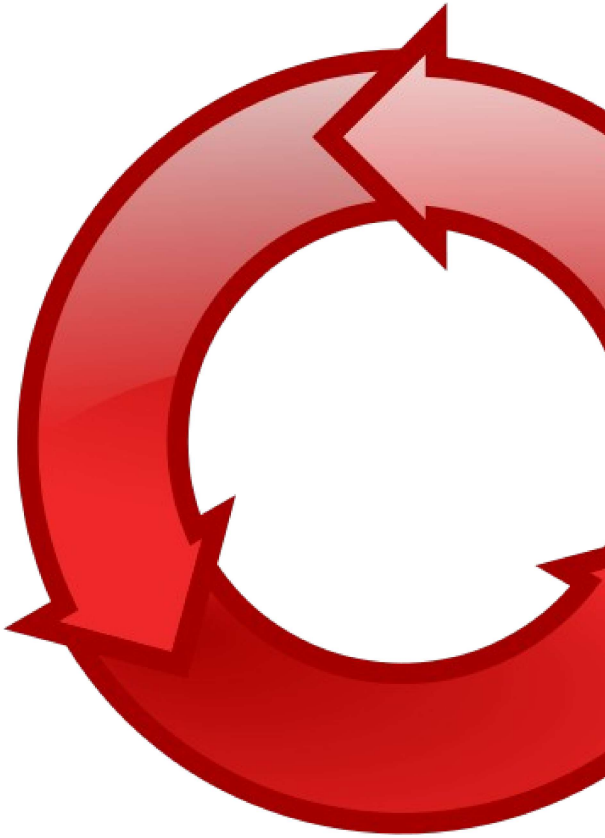
Observações Importantes

- Cursores, apesar de muitas vezes facilitarem a estruturação da lógica de programação, são gargalos de performance e de consumo de recursos temporários (memória, principalmente). Por isso é preciso ter cuidado e evitar seu uso indiscriminado.
- Cada fabricante tem suas variações de sintaxe e alguns comportamentos exclusivos. É sempre bom ler a documentação (manuais oficiais do SGBD) para estar a par das particularidades envolvidas.



Observações Importantes

- Particularidades do uso de cursores no MySQL:
  - **Read-only:** não é possível atualizar os dados dentro do cursor. Ele serve apenas para leitura.
  - **Non-scrollable:** só é possível varrer os dados contidos no cursor na ordem em que foram gerados pelo comando SELECT. Não é possível fazer um FETCH no sentido contrário ou saltar para uma linha específica. É sempre um fluxo top-down.



Referências Bibliográficas

- http://www.linhadecodigo.com.br/artigo/876/utilizando-cursores.aspx

- https://wagnercaetano.wordpress.com/2012/10/24/cursor-no-mysql/

- ELSMARI, Ramez & NAVATHE, Shamkant B. Sistema de Banco de Dados. Rio de Janeiro: Editora Addison-Wesley. 4ª. edição, 2005, 744p