MAC0439 – Laboratório de Bancos de Dados

Aula 19 – Parte A
Programação SQL Real
SQL Embutida

04 de novembro de 2015

Profa. Kelly Rosa Braghetto

(Adaptação dos slides do prof. Jeffrey Ullman, da *Stanford University*)

SQL em Programas Reais

- Neste curso, até o momento, nós vimos somente como a SQL é usada em uma interface de consulta genérica – um ambiente onde sentamos em frente ao computador e submetemos consultas ao banco de dados.
- A realidade, quase sempre, é outra: programas convencionais interagindo com o BD.

Opções

1. Código em uma linguagem especializada, armazenado no próprio BD (e.g., PSM, PL/PgSQL).
2. Comandos SQL embutidos em uma *linguagem hospedeira* (e.g., C).
3. Ferramentas (APIs) de conexão, que possibilitam que linguagens convencionais acessem um BD (e.g., CLI, JDBC, PHP/DB).

SQL Embutida

- Ideia principal: Um pré-processador transforma comandos SQL em chamadas a procedimentos, com a sintaxe apropriada de um código da linguagem hospedeira.
- Todo comando SQL embutido começa com EXEC SQL; assim, eles são facilmente encontrados pelo pré-processador.

Variáveis compartilhadas

- Para conectar o código em SQL ao restante do programa, as partes envolvidas precisam compartilhar algumas variáveis.
- Declarações de variáveis compartilhadas aparecem delimitadas:
-

EXEC SQL BEGIN DECLARE SECTION;

<declarações na linguagem hospedeira>

EXEC SQL END DECLARE SECTION;

Sempre necessários

O Uso de Variáveis Compartilhadas

- Em SQL embutida, o uso de variáveis compartilhadas deve vir precedido de um ':'
  - Elas podem ser usadas como constantes fornecidas pelo programa da linguagem hospedeira.
  - Elas podem receber valores provenientes de comandos SQL e passá-los ao programa da linguagem hospedeira.
- Na linguagem hospedeira, variáveis compartilhadas se comportam da mesma forma que qualquer outra variável.

Exemplo: Buscando Preços

- Usaremos C com SQL embutida, para “rascunhar” as partes mais importantes de uma função que obtém um refri e uma lanchonete, e então consulta o preço desse refri na lanchonete dada.
- Suponha que o BD possua a tradicional relação

  Vendas(nome_lanch, nome_refri, preço) .

Exemplo: C + SQL

```
EXEC SQL BEGIN DECLARE SECTION;
   char aLanch[21], oRefri[21];
   float oPreco;
EXEC SQL END DECLARE SECTION;
   /* antes do comando a seguir, obtenha
       valores para aLanch e oRefri  */
EXEC SQL SELECT preço INTO :oPreco
   FROM Vendas
   WHERE nome_lanch = :aLanch AND
              nome_refri = :oRefri;
   /* faça alguma coisa com o preço */
```

Uma string de até 20 caracteres precisa de um vetor de tamanho 20 + 1 (para o '\0')

SELECT-INTO, como em Stored Procedures

Consultas embutidas

- Consultas embutidas possuem as mesmas limitações que PSM (*stored procedures*) no que se refere a consultas:
  - SELECT-INTO para uma consulta que garantidamente produz uma única tupla como resposta.
  - Caso constrário, é preciso usar um cursor.
    - Há pequenas diferenças sintáticas entre os cursores da PSM e os da SQL embutida, mas as ideias principais são as mesmas.

Comandos para Cursores

- Declare um cursor *c* com:

EXEC SQL DECLARE *c* CURSOR FOR <consulta>;

- Abra e fecha o cursor c com:

EXEC SQL OPEN c;

EXEC SQL CLOSE c;

- Recuperar dados de *c* por:

EXEC SQL FETCH c INTO <variável(is)>

Exemplo: Imprima o Menu do Sujinhos

- Vamos escrever um código em C + SQL para a impressão do menu do Sujinhos – a lista de pares refri-lanchonente que nós encontramos em Vendas(nome_lanch, nome_refri, preço) com nome_lanch = 'Sujinhos'.
- Um cursor visitará cada tupla de vendas que tenha nome_lanch = Sujinhos.

Exemplo: Declarações

```
EXEC SQL BEGIN DECLARE SECTION;
   char oRefri[21]; float oPreco;
EXEC SQL END DECLARE SECTION;
EXEC SQL DECLARE c CURSOR FOR
   SELECT nome_refri, preço FROM Vendas
   WHERE nome_lanch = 'Sujinhos';
```

A declaração do cursor fica fora da seção de declaração de variáveis compartilhadas

Exemplo: Parte executável

```
EXEC SQL OPEN c;
while(1) {
   EXEC SQL FETCH c
   INTO :oRefri, :oPreço;
   if (NO_MORE_TUPLES) break;
   /* formatar e imprimir oRefri e
      oPreço */
}
EXEC SQL CLOSE c;
```

O estilo de C para quebra de laços

A Macro NO_MORE_TUPLES

- A macro NO_MORE_TUPLES é verdadeira se e somente se o FETCH falhar na tarefa de encontrar uma tupla.
- Pode ser definida como:

  #define NO_MORE_TUPLES !(strcmp(SQLSTATE,”02000”))

Scroll Cursors

- Com cursores definidos com a cláusula *scroll, podemos navegar pelas tuplas de outras formas*

EXEC SQL DECLARE c ***SCROLL CURSOR*** *FOR* SELECT * FROM Vendas;

- Tipos de FETCH:
  - FETCH FIRST, FETCH LAST
  - FETCH NEXT, FETCH PRIOR
  - FETCH RELATIVE *i*, FETCH RELATIVE -*i*
  - FETCH ABSOLUTE *i*, FETCH ABSOLUTE -*i*,

Proteção Contra Atualizações Concorrentes

- Exemplo:

EXEC SQL DECLARE c **INSENSITIVE CURSOR**

FOR SELECT * FROM Vendas;

- A cláusula INSENSITIVE CURSOR garante que mudanças na relação Vendas não afetarão o conjunto de tuplas “recuperadas” com o FETCH
- Cursores desse tipo costumam ser muito “custosos” em termos de tempo

Necessidade de SQL Dinâmico

- Muitas aplicações usam consultas específicas e comandos de modificação para interagir com os BDs.
  - O SGBD compila comandos EXEC SQL ... em chamadas a procedimentos específicas e produz um programa (como qualquer outro) na linguagem hospedeira que usa uma biblioteca.
- Mas e o **psql** (por exemplo), que não sabe o que precisa fazer até que seja executado de fato?

SQL Dinâmico

- Preparando uma consulta:

EXEC SQL PREPARE <nome-consulta>
FROM <texto-da-consulta>;

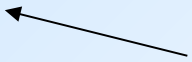
- Executando uma consulta:

EXEC SQL EXECUTE <nome-consulta>;

- “Prepare” = otimize a consulta.
- Prepare uma vez, execute muitas vezes.

Exemplo: Uma Interface Genérica

```
EXEC SQL BEGIN DECLARE SECTION;
   char mConsulta[TAM_MAX];
EXEC SQL END DECLARE SECTION;
while(1) {
   /* mostre o prompt SQL> */
   /* leia a consulta do usuário no vetor
      mConsulta */
   EXEC SQL PREPARE q FROM :mConsulta;
   EXEC SQL EXECUTE q;
}
```



q é uma variável SQL representando a forma otimizada do comando armazenado em :consulta

Execução Imediata

- Se vamos executar a consulta apenas uma vez, podemos combinar os passos PREPARE e EXECUTE em um só.
- Use:

EXEC SQL EXECUTE IMMEDIATE <texto-da-consulta>;

Exemplo: Interface Genérica Novamente

```
EXEC SQL BEGIN DECLARE SECTION;
  char mConsulta[TAM_MAX];
EXEC SQL END DECLARE SECTION;
while(1) {
  /* mostre o promp SQL> */
  /* leia a consulta do usuário no
     vetor mConsulta */
  EXEC SQL EXECUTE IMMEDIATE :mConsulta;
}
```

SQL Embutida no PostgreSQL

- ECPG – *Embedded SQL in C package for Postgres*

  http://www.postgresql.org/docs/9.4/static/ecpg.html
- Instruções para a compilação dos programas em C+SQL+ECPG:

  http://www.postgresql.org/docs/9.4/static/ecpg-process.html
- Veja um exemplo de programa que acessa um BD em PostgreSQL no Paca, na página da disciplina.