

Sumário

- 1 Introdução ao Processamento de Consultas
- 2 Otimização de Consultas
- 3 Plano de Execução de Consultas
- 4 Introdução a Transações
- 5 Recuperação de Falhas
- 6 Controle de Concorrência
- 7 Fundamentos de BDs Distribuídos
- 8 SQL Embutida**

Motivação

- Aplicações precisam acessar o BD
- Linguagens BD X Linguagens Programação
 - paradigmas diferentes (*impedance mismatch*)
 - declarativo X {procedural, OO, ...}
 - tradutores independentes para cada linguagem
- Desenvolvimento de aplicações no SGBD
 - uso de uma linguagem de quarta geração
 - embutimento de SQL no código da aplicação

SQL Embutida

- O que é?
 - conjunto de instruções proposto para a SQL padrão que são incorporados no código de uma aplicação (Linguagem Hospedeira - LH)
- Objetivos
 - permitir a comunicação entre aplicação e BD
 - suprir certas deficiências da SQL
 - nem todo dialeto SQL é capaz de expressar adequadamente qualquer tipo de consulta
 - exemplos: consultas recursivas ou por similaridade
 - ela não é capaz de realizar ações não-declarativas
 - apresentar dados em interfaces gráficas ou relatórios, implementar algoritmos complexos, ...

SQL Embutida - Questões

- Questões a considerar
 1. tradução de duas linguagens diferentes
 2. intercâmbio de dados aplicação-BD
- 1. Tradução de linguagens diferentes
 - passo 1: pré-compilação de comandos SQL
 - substituídos por declarações na LH
 - invocação de procedimentos que realizam acesso (otimizado) ao BD
 - passo 2: compilação do código da aplicação
 - compilação da LH

SQL Embutida - Questões

2. Intercâmbio de Dados Aplicação-BD

- variáveis da LH utilizadas em comandos SQL devem estar identificadas no código da aplicação
 - reconhecidas pelo pré-compilador SQL
 - indicam onde atribuir os resultados de consultas nos procedimentos de acesso (**parâmetros de saída**)
 - indicam de onde obter dados para serem enviados ao BD nos procedimentos de acesso (**parâmetros de entrada**)
 - analisa-se a compatibilidade dos tipos de dados

Tipos de Instruções Embutidas

- Declarativas
 - variáveis da LH utilizadas em comandos SQL
 - inclusão de variáveis especiais do BD
- Executáveis
 - comandos SQL
 - instruções de definição e manipulação de **cursores**
 - instruções dinâmicas
- Instruções da SQL embutida são identificadas pela cláusula **EXEC SQL**
 - cláusula definida na SQL padrão

Instruções Declarativas

- Variáveis da LH utilizadas em comandos SQL são indicadas em uma **seção de declaração**

```
...  
EXEC SQL BEGIN DECLARE SECTION  
    int idade, codM;  
    char nome[30], esp[20];  
    ...  
EXEC SQL END DECLARE SECTION  
...
```

Instruções Executáveis

- Indicação de um comando SQL após a cláusula `EXEC SQL`
- Exemplo 1
 - comando SQL **sem** parâmetros

```
...  
EXEC SQL delete from Médicos  
           where idade > 70;  
...
```


Instruções Executáveis

- Exemplo 2
 - comando SQL **com** parâmetro de entrada

```
...  
EXEC SQL delete from Médicos  
         where CRM = :codM;  
...
```

Instruções Executáveis

- Exemplo 3
 - comando SQL **com** parâmetros de entrada e saída

```
...  
EXEC SQL  select nome, idade  
          into :nome, :idade  
          from Médicos  
          where CRM = :codM;  
...
```

Variável de Status – SQLCODE

- Indica o status de execução do comando SQL
 - definida na SQL padrão
- Campo de um registro especial chamado **SQLCA**
 - o registro deve ser definido no código da LH
- Exemplos de status
 - 0**: execução OK
 - 100**: não há mais tuplas a serem buscadas
 - < 0**: erro de execução
 - ...

Variável de Status

- Exemplo

```
...  
EXEC SQL INCLUDE SQLCA;  
...  
EXEC SQL delete from Médicos  
         where CRM = :codM;  
  
if (SQLCA.SQLCODE < 0) {  
    printf("Erro na exclusão!\n");  
    trataErroDeleteMedicos(); }  
...
```

Cursors

- Consultas unitárias
 - retornam uma única tupla do BD
- Consultas em nível de conjunto
 - retornam uma ou mais tuplas do BD
 - necessitam de **cursors**
- Cursor
 - mecanismo da SQL embutida que permite o acesso a cada tupla de um conjunto de dados buscado do BD
 - noção de “ponteiro” lógico

Cursores - Instruções

- Instrução `declare`
 - define um cursor
 - indica a estrutura do resultado de consulta que ele irá apontar
 - exemplo

```
...  
EXEC SQL DECLARE ptr CURSOR FOR  
        select nome, CRM  
        from Médicos  
        where especialidade = :esp;  
...
```

Cursors - Instruções

- Instrução `open`
 - ativa (ou “abre”) o cursor
 - executa a consulta associada ao cursor
 - gera uma tabela temporária
 - uma estrutura de resposta
 - posiciona o cursor na primeira tupla
 - exemplo

```
...  
EXEC SQL OPEN ptr;  
...
```

Cursores - Instruções

- Instrução `fetch-into`
 - transfere os dados apontados pelo cursor para variáveis da LH
 - avança o cursor para a próxima tupla
 - exemplo

```
...  
EXEC SQL FETCH ptr INTO :nome, :codM;  
...
```


Cursors - Instruções

- Instrução `close`
 - desativa (ou “fecha”) um cursor já ativo (ou “aberto”)
 - remove a tabela temporária
 - exemplo

```
...  
EXEC SQL CLOSE ptr;  
...
```

Cursores – Exemplo Completo

```
...
EXEC SQL BEGIN DECLARE SECTION
        int idade, codM;
        char nome[30], esp[20];
        ...
EXEC SQL END DECLARE SECTION
EXEC SQL INCLUDE SQLCA;
EXEC SQL DECLARE ptr CURSOR FOR
        select nome, CRM
        from Médicos
        where especialidade = :esp;

...
printf("\nInforme especialidade:"); gets(esp);
EXEC SQL OPEN ptr;
if(!SQLCA.SQLCODE)
    while (SQLCA.SQLCODE != 100)
        {EXEC SQL FETCH ptr INTO :nome, :codM;
        printf("\nMédico: ", nome, " CRM: ", codM);
        }
else printf("\nErro ou consulta vazia!");
EXEC SQL CLOSE ptr;
...

```

Cursors e Atualização do BD

- Cursors podem ser utilizados para atualização do BD
 - `update` e `delete` de tuplas apontadas pelo cursor
 - *por default*, todo cursor tem esta capacidade
 - exceto se um **FOR READ-ONLY** for declarado
- Recurso que permite consulta e atualização simultânea de dados
 - atualizações de dados apontados pelo cursor devem ser persistidos no BD
- Indicação de quais dados são passíveis de alteração pode ser definido no momento da declaração do cursor

Cursores e Alteração – Exemplo 1

```
...
EXEC SQL DECLARE ptr CURSOR FOR
        select nome, CRM, salário
        from Médicos
        where especialidade = :esp
        FOR UPDATE OF salário;
...

EXEC SQL FETCH ptr INTO :nome, :codM, :sal;
if (sal < salarioBase)
{
    EXEC SQL UPDATE Médicos
        SET salário = :salarioBase
        WHERE CURRENT OF ptr;
    printf("\nO salário de ", nome, " foi reajustado!");
}
...

```

Cursores e Exclusão – Exemplo 2

```
...
EXEC SQL DECLARE ptr CURSOR FOR
        select nome, CRM, salário
        from Médicos
        where especialidade = :esp;
...

EXEC SQL FETCH ptr INTO :nome, :codM, :sal;
if (sal < salarioBase)
{
    EXEC SQL DELETE FROM Médicos
        WHERE CURRENT OF ptr;
    printf("\nO médico ", nome, " foi excluído!");
}
...

```

Exercício usando SQL Embutida

- Considere a tabela Empregados abaixo

Empregados

<u>codEmp</u>	nome	salário	função	idade	<u>codGerente</u>
---------------	------	---------	--------	-------	-------------------

- [consulta recursiva](#): mostrar o nome e a função dos empregados que compõem a hierarquia de gerência de um empregado, dado o seu código;
- [RI](#): implementar um procedimento que verifica a existência de ciclos de gerência, dado um código de empregado cujo código do seu gerente foi recém-atualizado no BD. Ele deve desfazer essa transação de atualização de dados (`ROLLBACK TRANSACTION`), se um ciclo existir.

Instruções Dinâmicas

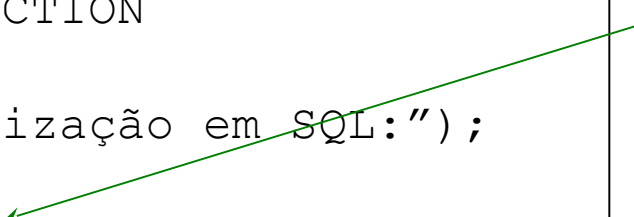
- Instruções estáticas
 - comandos SQL **pré-determinados** com ou sem parâmetros de entrada e/ou saída
- Instruções dinâmicas
 - comandos SQL total ou parcialmente definidos em tempo de execução
 - instruções
 - **PREPARE**
 - compila um comando SQL e gera código executável para ele
 - **EXECUTE**
 - executa o código gerado pela instrução PREPARE

Instruções Dinâmicas - Exemplo 1

- Execução de comandos de atualização (sem parâmetros)

```
...
EXEC SQL BEGIN DECLARE SECTION
    char stringAtSQL[30];
    ...
EXEC SQL END DECLARE SECTION
...
printf("\nInforme atualização em SQL:");
gets(stringAtSQL);
EXEC SQL PREPARE SQLExe from :stringAtSQL;
if(!SQLCA.SQLCODE) EXEC SQL EXECUTE SQLExe;
if(SQLCA.SQLCODE < 0)
    printf("\nErro de execução!");
...
```

nome simbólico para
um comando SQL
executável



Instruções Dinâmicas - Exemplo 2

- Execução de comandos de atualização (com parâmetros)

```
...
EXEC SQL BEGIN DECLARE SECTION
    int codigo;
    char stringSQL[40];
    ...
EXEC SQL END DECLARE SECTION
...
strcpy(stringSQL, "delete from médicos where CRM = :x");
EXEC SQL PREPARE SQLExe from :stringSQL;
if(!SQLCA.SQLCODE)
{ printf("\nInforme CRM de um médico para exclusão:");
  scanf("%d", codigo);
  EXEC SQL EXECUTE SQLExe USING :codigo;
}
if(SQLCA.SQLCODE < 0) printf("\nErro de execução!");
...
```

Instruções Dinâmicas - Exemplo 3

- Uso de cursores

```
...
EXEC SQL BEGIN DECLARE SECTION
    int codigo;
    char nome[40];
    ...
EXEC SQL END DECLARE SECTION
char condicao[100];
...
strcpy(stringSQL, "select CRM, nome from médicos where ");
printf("\nInforme uma condição para a consulta:");
gets(condicao);
strcat(stringSQL, condicao);
EXEC SQL PREPARE SQLExe from :stringSQL;
if(!SQLCA.SQLCODE) {
    EXEC SQL DECLARE ptr CURSOR FOR SQLExe;
    EXEC SQL OPEN ptr;
    while (SQLCA.SQLCODE < > 100) {
        EXEC SQL FETCH ptr INTO :codigo, :nome;
        ...}}...
```

Variações da SQL Padrão

Informix ESQL/C

```
...  
$ int resp1;  
$ char resp2[30], esp[20];  
...  
$ select CRM, nome  
  into :resp1, :resp2  
  from Médicos  
  where especialidade = :esp;  
...
```

DB2 SQLJ (Java)

```
...  
#sql { select CRM, nome  
  into :resp1, :resp2  
  from Médicos  
  where especialidade = :esp };  
...
```

SQL Embutida

- Trabalha-se com SQL e LH de forma independente e simples
- Instruções estáticas compiladas e otimizadas para acesso uma única vez
 - diferente de APIs para acesso a BDs
- Uso restrito ao ambiente do SGBD
 - SGBD deve conhecer a LH para adequar ao código da LH o código pré-compilado de acesso ao BD
 - APIs garantem maior independência de SGBD
 - uma aplicação pode acessar mais de um SGBD
 - com exceção de algumas diferenças em dialetos SQL