Stored Routines

o MySQL permite que o usuário guarde código dentro de um banco de dados. O objetivo deste procedimento pode ser resumido olhando para os aspectos de

centralização Muitos usuários, muitas aplicações, muitas linguagens podem usar todos o mesmo código escrito, testado e mantido em um local único.

segurança pode-se restringir a manipulação do banco de dados ao uso de rotinas armazenadas. Ao agir assim restringe-se a execução de códigos mal-escritos ou mal-intencionados.

velocidade havendo cuidados na sua geração, esta ferramenta pode efetivamente aumentar o desempenho (até pela diminuição de tráfego na rede)

suporte a transações havendo este cuidado, todas as transações podem ser concluídas (ou desfeitas) em bloco.

As stored procedures e as funções são guardadas na tabela PROC dentro do banco de dados MYSQL que é criado durante a instalação do MySQL.

A diferença entre procedure e função não é muita: ambas podem receber parâmetros, mas apenas a segunda devolve um resultado. Conceitualmente suas vocações são bem diferentes: enquanto o procedimento é usado para blocos mais robustos de códigos, as funções são usadas como mini blocos de código, aptas a serem usadas – por exemplo – no cálculo de uma função aritmética qualquer.

Dentro de funções e procedimentos podem aparecer comandos compostos, que são reconhecidos pelas cláusulas BEGIN e END. Note que estes blocos podem ser aninhados (um dentro do outro). Dentro do bloco podem ser usados IF, CASE, ITERTAE, LOOP, LEAVE LOOP, REPEAT e WHILE. Outro comando que só pode ser emitido dentro de um BEGIN...END é DECLARE, para a criação de variáveis. Os nomes das variáveis não são sensíveis à caixa. ITERATE só pode ser emitido dentro de LOOP, REPEAT e WHILE e significa "comece o loop novamente". LEAVE, idem, e deixa o laço mais interno de onde foi emitido. Se ele for do último nível, significa fim de programa.

O LOOP implementa um loop infinito, do qual se sai por LEAVE ou por RETURN (fim de função ou procedimento).

O REPEAT segue o padrão do pascal, saindo quando a condição for verdadeira.

Variáveis de usuário tem o nome iniciando por arroba. Elas sobrevivem à sessão, mas não podem ser passadas a outro usuário/programa. Se você usar uma varíavel sem o arroba, ela só sobrevive no comando onde é emitida.

Na criação de um bloco armazenado surge uma pequena dificuldade operacional relacionada a delimitar comandos. Como o caractere ; encerra comandos, ao digitar uma procedure formada por comandos (que precisam ser encerrados por ;, o MySQL acha que a digitação encerrou.

Para evitar o conflito, antes de nada, deve-se alterar o delimitador

```
DELIMITER **
```

A partir deste ponto, os comandos MySQL devem terminar por **. Então as linhas das stored procedures podem ser terminadas por ; sem que haja nenhum problema. Lembrando que após o seu armazernamento o delimitador deve ser restaurado a ;.

Para criar um procedimento armazenado o comando é

```
CREATE PROCEDURE <nome>([parâmetros])
[opções]
BEGIN
   <código SQL>
END**
```

A maneira ideal de proceder é escrever este código em um arquivo txt (para poder alterar e regravar – em caso de erro), sem ter que reescrever todo o bloco de código a cada vez. Criado e salvo o arquivo de script, digamos com o nome de PROC1.SQL sua carga no BD é feita pelo comando

```
SOURCE <caminho completo do arquivo de script>
```

Havendo algum erro, o arquivo txt é alterado, e pode ser carregado de novo (desde que se elimine o objeto de mesmo nome ja carregado, através do comando DROP PROCEDURE <nome>).

Criado o procedimento armazenado, para executá-lo o comando é

```
CALL <nome()>
```

O comando USE não pode ser usado dentro de um procedimento armazenado. O MySQL associa um banco de dados aos procedimentos. Para deixar isso explícito, o comando CALL pode incluir o banco esperado, fazendo-se

```
CALL bd.<nome()>
```

Veja um exemplo completo

```
DELIMITER **   // no arquivo lixomy.sql
DROP PROCEDURE IF EXISTS lixomy**
CREATE PROCEDURE lixomy()
BEGIN
select * from linh;
END**
DELIMITER ;
...
SOURCE c:/tcc/lixomy.sql;
...
CALL d76.lixomy();
```

Criando transações Você pode usar uma *stored procedure* para criar uma transação. A idéia é criar uma transação atômica (ou é realizada em bloco, ou nada é realizado). Para isso são necessários alguns comandos

`START TRANSACTION`
marca o início de uma transação.

`SAVEPOINT <nome>`
pontos seguros, aos quais pode-se retornar sem precisar voltar ao início. O <nome> identifica o ponto.

`ROLLBACK`
Desfaz a transação inteira.

`ROLLBACK TO SAVEPOINT<nome>`
Desfaz a transação até o ponto <nome> identificado. Inclusive, desfaz outros SAVEPOINT porventura emitidos.

`COMMIT`
Identifica o final da transação, garantindo que todas as alterações pedidas sejam efetivamente implementadas no BD.

Este conceito de transação só está garantido para os métodos InnoDB e BDB (Berkeley).

Manutenção Para examinar o conteúdo de uma *stored procedure* o comando é SHOW CREATE PROCEDURE <nome>. Para alterar, usando a interface de linha de comando, a coisa não é muito ergonômica: é necessário recriar um arquivo de texto (.SQL), depois eliminar a procedure (comando DROP PROCEDURE <nome> e finalmente criá-la de novo. (CREATE PROCEDURE <nome>).

FUNÇÔES As funções também são armazenadas no arquivo PROC do MySQL. Sua operacionalização é bastante similar aos das procedures, como se verá:

```
CREATE FUNCTION <nome>(<parâmetros>)
RETURNS <tipo de retorno>
DETERMINISTIC[<opções>]
BEGIN
   <código SQL>
END**

SHOW CREATE FUNCTION <nome>;

DROP FUNCTION

SELECT função(parametros); // para executar
```

Pode-se (deve-se) criar um arquivo texto de nome xx.SQL e depois fazer SOURCE xx.SQL. Veja-se a seguir um exemplo:

```
DELIMITER $$
DROP FUNCTION IF EXISTS exemplo$$
CREATE FUNCTION exemplo (a int) RETURNS int(11)
DETERMINISTIC
BEGIN
DECLARE r int;
SET r = a * 221;
RETURN r;
END$$
DELIMITER ;
```

Utilizando o banco de dados D76 (aquele das linhas de transporte intermunicipal) escreva uma função que receba 2 códigos de município (3 decimais) e devolva a distância pitagórica entre eles: (Note como são apanhados campos do BD...)

```
DELIMITER $$
DROP FUNCTION IF EXISTS dmun$$
CREATE FUNCTION dmun(a int, b int)
   RETURNS double
   DETERMINISTIC
BEGIN
DECLARE t double;
declare t1 double;
declare t2 double;
DECLARE ax decimal(5,0);
DECLARE ay decimal(5,0);
DECLARE bx decimal(5,0);
DECLARE bb decimal(5,0);
select municorx into ax from muni
    where MUNICODM = a;
select municory into ay from muni
    where MUNICODM = a;
select municorx into bx from muni
    where MUNICODM = b;
select municory into bb from muni
    where MUNICODM = b;
set t1 = (ax-bx)*(ax-bx);
set t2 = (ay-bb)*(ay-bb);
set t = sqrt(t1+t2);
RETURN t;
END$$
DELIMITER ;
```

☞ Para você fazer

Usando o banco de dados D76, aquele da estatística de transporte intermunicipal, escreva uma função que

Receba 2 códigos de linha no arquivo LINH e devolva a distância percorrida pela linha mais longa.

e entregue-a impressa em anexo ou transcrita no verso desta folha. Na listagem incluir também a execução da função e o resultado que ela apresentou.