# Transações Seguras em Bancos de Dados (MySQL)

## Índice



## Entendendo os *storage engines* do MySQL 5

O MySQL 5 suporta vários tipos de *storage engines* (InnoBD, MyISAM, NDB, Merge Blackhole, Falcon, e por aí vai). Os mais comuns são MyISAM e InnoDB. E isso tem lá suas razões e implicações na hora de usar um ou outro recurso. O MyISAM não dá suporte a tabelas *transaction-safe*, enquanto o InnoDB provê essa funcionalidade. Então isto é um pré-requisito para continuarmos o artigo: usaremos InnoDB como *storage engine*.

Para maiores informações sobre storage engines e suas diferenças, veja o capítulo 14 do manual do MySQL 5.1.

## As ferramentas

Aqui vamos descrever as principais ferramentas de SQL que irão nos permitir operar as transações.

- **COMMIT**: isto é salvar a(s) operação(ões) realizadas até o momento;

- **SAVEPOINT**: cria um ponto de retorno;

- **ROLLBACK**: desfaz as operações até o início da transação;

- **ROLLBACK TO SAVEPOINT**: desfaz as operações até o ponto de retorno.

## Mais algumas coisas que você deve saber

Por padrão, o MySQL inicia suas sessões com o modo AUTOCOMMIT habilitado. Isso

significa que todas as operações são registradas (salvas) uma a uma, na ordem em que forem sendo executadas. Então precisamos usar:

(1) **SET AUTOCOMMIT=0;**

ou

(2) **START TRANSACTION;**

Para usar as ferrementas descritas acima. A diferença entre (1) e (2) é que usando (2) o AUTOCOMMIT volta a estar habilitado quando se termina a transação (um comando COMMIT ou um ROLLBACK).

Também há comandos que não podem ser desfeitos com ROLLBACK, isso será descrito mais adiante.

Quando um SAVEPOINT é criado usando um nome já atribuído a outro SAVEPOINT, o SAVEPOINT antigo é apagado o nome é atribuído ao novo SAVEPOINT.

## Com a mão na massa

Agora exemplos de uso das ferramentas de SQL para transações.

```
START TRANSACTION;
INSERT INTO tb1 id, nome, email VALUES (" , 'Nome X', 'xxx@zzz.com');
INSERT INTO tb2 id, email VALUES (" , 'xxx@zzz.com');
COMMIT;

SET AUTOCOMMIT = 0;
SAVEPOINT meu_ponto1;
UPDATE tb_dinheiro SET qtd = qtd-300 WHERE client_id = '10';
SAVEPOINT meu_ponto2;
UPDATE tb_dinheiro SET qtd = qtd+300 WHERE client_id = '15';
SELECT qtd FROM tb_dinheiro WHERE client_id = '10'; // qtd = 500
SELECT qtd FROM tb_dinheiro WHERE client_id = '15'; // qtd = 1100
ROLLBACK TO SAVEPOINT meu_ponto2;
SELECT qtd FROM tb_dinheiro WHERE client_id = '15'; // qtd = 800
ROLLBACK TO SAVEPOINT meu_ponto1;
SELECT qtd FROM tb_dinheiro WHERE client_id = '10'; // qtd = 800
COMMIT;
```

Linha a linha isso não parece ser muito útil, mas considere usar estas ferramentas dentro de funções, stored procedures, triggers e scheduled events, temas dos próximos artigos sobre bancos de dados que publicarei.

Uma outra possibilidade interessante é usar as transações em scripts de outras linguagens, como C, PHP, Java e Python, isso permitirá iterar sobre as queries e trabalhar com bastante comodidade, e é sobre isso que este artigo continuará falando.

## Mais ferramentas - Usando o PHP

O PHP tem uma biblioteca de funções que suporta diretamente as transações, a biblioteca mysqli. Vale lembrar que todas as regras, como usar o InnoDB como storage engine, e as restrições continuam valendo.

A idéia é trabalhar com *flags*. Vamos supor que você tenha um formulário para cadastro e que o processamento é feito nas seguintes etapas:

- Uma query INSERT na tabela clientes do seu banco de dados;

- O envio de um e-mail de confirmação para o cliente;

- Escrever uma linha num arquivo de log, mantida no site (contendo um time stamp, ip da máquina de onde o registro foi feito, e um hash md5 do e-mail);

Seria bom se algum destes passos falhar, que nada seja feito, principalmente no banco de dados. Imagine a inconveniência de não mandar um e-mail de confirmação, ou de não manter um log (manter um log, mesmo que seja num arquivo simples de XML, pode ser muito útil na hora de fazer uma auditoria, principalmente para detectar casos de *code-injection*).

Vamos ao script:

```
<?
    /* recebendo as variáveis do form */
    $nome = $_POST['nome'];
    $email = $_POST['email'];

    /* gerando as variáveis para o insert e para o log */
    $data = date("r");
    $ip = $_SERVER['REMOTE_ADDR'];
    $hash = md5($email);
```

```
/* instanciando a classe mysqli e fazendo o que tem que ser feito */
$my = new mysqli("localhost", "mysql_user", "mysql_password", "cadastro");
$my->autocommit(FALSE);
$sql = "INSERT INTO cadastro id, nome, email VALUES ('', '$nome' , '$email')";
$my->query($sql);

/*seta as variáveis de e-mail e faz o envio assim jogando numa flag */
...
$x = mail($email, $subject, $message);

/* usa uma função ou um método para inserir os dados no arquivo de log */
/* depois coloco no blog ou escrevo um artigo sobre como se faz isso */
$y = addToLog("log.xml", $data, $ip, $hash, $email); //retorna true em caso de sucesso ou false em caso de erro

/* Agora o que esperávamos */
if ($x == TRUE && $y == TRUE) {
        $my->commit();
        $my->close();
        echo "Sucesso !";
} else {
        $my->rollback();
        $my->close();
        echo "Nada feito!";
}
?>
```

Claro que este script está bem simplificado, mas elucida como usar as transações.

**IMPORTANTE:** Não há suporte a SAVEPOINT nesta classe.

## Nem tudo pode ser desfeito!

Há comandos SQL que não podem ser desfeitos com o ROLLBACK ou com o ROLLBACK TO SAVE POINT, isso porque causam um COMMIT implicitamente ao serem declarados. Portanto deve-se evitar esse comandos dentro de suas transações.

Tecnicamente falando são os comandos pertencentes ao grupo DDL (Data Definition Language) - Linguagem de Definição de Dados, ou seja comandos usados para definir tabelas novas, elementos associados e realizar modificações estruturais nestas entidades (tabelas, funções, triggers, stored procedures, scheduled events, etc...).

Na prática são os seguintes comandos (além dos similares e *alias* dos mesmos):

*ALTER FUNCTION,*

*ALTER PROCEDURE,*

*ALTER TABLE,*

*BEGIN,*

*CREATE DATABASE,*

*CREATE FUNCTION,*

*CREATE INDEX,*

*CREATE PROCEDURE,*

*CREATE TABLE,*

*DROP DATABASE,*

*DROP FUNCTION,*

*DROP INDEX,*

*DROP PROCEDURE,*

*DROP TABLE,*

*LOAD MASTER DATA,*

*LOCK TABLES,*

*LOAD DATA INFILE,*

*RENAME TABLE,*

*SET AUTOCOMMIT=1,*

*START TRANSACTION,*

*TRUNCATE TABLE,*

*UNLOCK TABLES.*

## Referências


- Manual do PHP 5, capítulo VI - LXXX.
- Manual do MySQL 5.1, capítulos 13.4, 14.2, 14.5.10
- Wikipédia: http://pt.wikipedia.org/wiki/SQL
- Para este artigo foi usado o servidor MySQL 5.0, Apache 2.0 e PHP 5.0

## Sobre este artigo

Este artigo é de autoria de Davis L. P. Peixoto.

Data: 20 de fevereiro de 2007