SQL

Controle transacional

Você sabia que seu material didático é interativo e multimídia? Isso significa que você pode interagir com o conteúdo de diversas formas, a qualquer hora e lugar. Na versão impressa, porém, alguns conteúdos interativos ficam desabilitados. Por essa razão, fique atento: sempre que possível, opte pela versão digital. Bons estudos!

As falhas nos bancos de dados, independente de sua natureza, podem ser extremamente sensível para a garantia da integridade dos dados. Com isso, nesta webaula vamos explorar os recursos presentes no SQL de controle transacional no banco de dados, são eles: SAVEPOINT, COMMIT e ROLLBACK.

Script do exemplo

BD e tabelas

Para que possamos demonstrar as técnicas, vamos tomar de exemplo o script de criação de BD e tabelas, a seguir:

```
CREATE TABLE Enfermeiro;
Core INT PRIMARY KEY,
   Nome VARCHAR(50) NOT NULL
);
CREATE TABLE Paciente (
Num INT PRIMARY KEY,
Nome VACHAR(50) NOT NULL
);
CREATE TABLE  Remedio (
Cod INT PRIMARY KEY,
    Nome VARCHAR (50) NOT NULL
);
CREATE TABLE Medicacao (
Id INT PRIMARY KEY AUTO_INCREMENT,
Data DATE NOT NULL,
    Hora TIME NOT NULL,
    PacienteNum INT NOT NULL,
    RemedioCod INT NOT NULL,
    EnfermeiroCoren INT NOT NULL,
    FOREIGN KEY (PacienteNum) REFERENCES Paciente(Num)
    FOREIGN KEY (RemedioCod) REFERENCES Remedio(Cod)
    FOREIGN KEY (EnfermeiroCoren) REFERENCES Enfermeiro(Coren)
);
```

Registros

Para que possamos compreender as operações realizadas de controle transacional foram inseridos os registros das tabelas. Veja a seguir, os scripts de inserção de dados nas tabelas.

Tabela Enfermeiro

```
INSERT Enfermeiro VALUES (11111, “Enfermeiro 1”)
(22222,”Enfermeiro 2”),
(33333, “Enfermeiro 3”);
```

Tabela Paciente

```
INSERT Paciente VALUES
(1000, “Paciente A”)
(1001,”Paciente B”),
(1002,”Paciente C”),
(1003,”Paciente D”),
(1004,”Paciente E”),
(1005,”Paciente F”),
(1006,”Paciente G”),
(1007,”Paciente H”),
(1008,”Paciente I”);
```

Tabela Remedio

```
INSERT Remedio VALUES (100, “Controle de pressao”)
(101,”Problemas no estomago”),
(102,”Soro”),
(103,”Calmante”),
(104,”Analgesico”),
(105,”Rins”);
```

Tabela Medicacao

```
INSERT Remedio VALUES
(0, current_date, “05:00:00”,1003, 104, 11111),
(0, current_date, “08:00:00”,1001, 100, 11111),
(0, current_date, “08:20:00”,1007, 102, 11111),
(0, current_date, “08:30:00”,1007, 105, 11111),
(0, current_date, “09:00:00”,1004, 104, 22222),
(0, current_date, “09:30:00”,1005, 105, 33333),
(0, current_date, “10:20:00”,1001, 103, 11111),
(0, current_date, “12:00:00”,1008, 102, 22222),
(0, current_date, “12:20:00”,1002, 105, 22222),
(0, current_date, “13:30:00”,1001, 100, 11111),
(0, current_date, “15:00:00”,1003, 104, 22222),
(0, current_date, “16:00:00”,1001, 103, 11111),
(0, current_date, “20:30:00”,1008, 100, 22222),
(0, current_date, “21:00:00”,1002, 105, 11111),
(0, current_date, “21:10:00”,1006, 102, 11111),
(0, current_date, “23:00:00”,1003, 104, 33333);
```

Nesse exemplo, vamos tomar as informações como: data, hora, paciente, remédio e o enfermeiro que foi responsável pela administração do medicamento, considerando a saída conforme a seguir:

Registro	Data	Hora	Paciente	Medicacao	Enfermeiro
1	2018-07-01	05:00:00	Paciente D	Analgesico	Enfermeiro 1
2	2018-07-01	08:00:00	Paciente B	Controle de pressao	Enfermeiro 1
3	2018-07-01	08:20:00	Paciente H	Soro	Enfermeiro 1
4	2018-07-01	08:30:00	Paciente H	Rins	Enfermeiro 1
5	2018-07-01	09:00:00	Paciente E	Analgesico	Enfermeiro 2
6	2018-07-01	09:30:00	Paciente F	Rins	Enfermeiro 3
7	2018-07-01	10:20:00	Paciente B	Calmante	Enfermeiro 1
8	2018-07-01	12:00:00	Paciente I	Soro	Enfermeiro 2
9	2018-07-01	12:20:00	Paciente C	Rins	Enfermeiro 2
10	2018-07-01	13:30:00	Paciente B	Controle de pressao	Enfermeiro 1
11	2018-07-01	15:00:00	Paciente D	Analgesico	Enfermeiro 2
12	2018-07-01	16:00:00	Paciente B	Calmante	Enfermeiro 1
13	2018-07-01	20:30:00	Paciente I	Controle de pressao	Enfermeiro 2
14	2018-07-01	21:00:00	Paciente C	Rins	Enfermeiro 1
15	2018-07-01	21:10:00	Paciente G	Soro	Enfermeiro 1
16	2018-07-01	23:00:00	Paciente D	Analgesico	Enfermeiro 3

Fonte: elaborada pelo autor, captura de tela do software MySQL.

Controle de transação

Uma transação pode ser considerada um conjunto de operações com uma única unidade lógica de trabalho, em que uma instrução pode acessar, alterar ou excluir vários dados em uma ou mais tabelas. O controle das transações ocorridas em um banco de dados deve garantir a integridade dos dados contidos nas tabelas.

Atomicidade

É a garantia que todas as operações serão corretamente refletidas em todo o banco de dados. Caso isso não seja possível, nenhuma das operações deve ser concluída, evitando que ocorra somente uma execução parcial de uma transação.

Consistência

Deve garantir que, se houverem duas transações executadas ao mesmo tempo, uma não interfira na outra.

Isolamento

Embora várias transações ocorram simultaneamente, na visão dos mecanismos envolvidos nos bancos de dados, tais execuções devem se comportar de forma isolada, de forma que, uma transação não fique “sabendo” o que está ocorrendo nas demais transações em execução.

Durabilidade

Após as transações serem finalizadas com sucesso, deve ser garantido que as alterações persistam nas tabelas do banco de dados.

COMMIT

Algumas vezes, por falha do SGBD, a garantia da durabilidade não é efetivada, comprometendo, assim, a integridade do BD. Quando uma transação se completa, é considerada **CONFIRMADA (committed)**, com isso, automaticamente é criado um novo estado, que deve garantir a persistência, mesmo em caso de falhas.

EXEMPLO

Imagine que você efetue um depósito em sua conta corrente e, posteriormente, o sistema confirma a entrada de um novo valor (COMMIT). Por uma falha de falta de energia elétrica, quando o sistema se restabelece, o valor depositado não está mais na sua conta. Nesse caso, a durabilidade não foi garantida e a integridade da transação não foi efetivada.

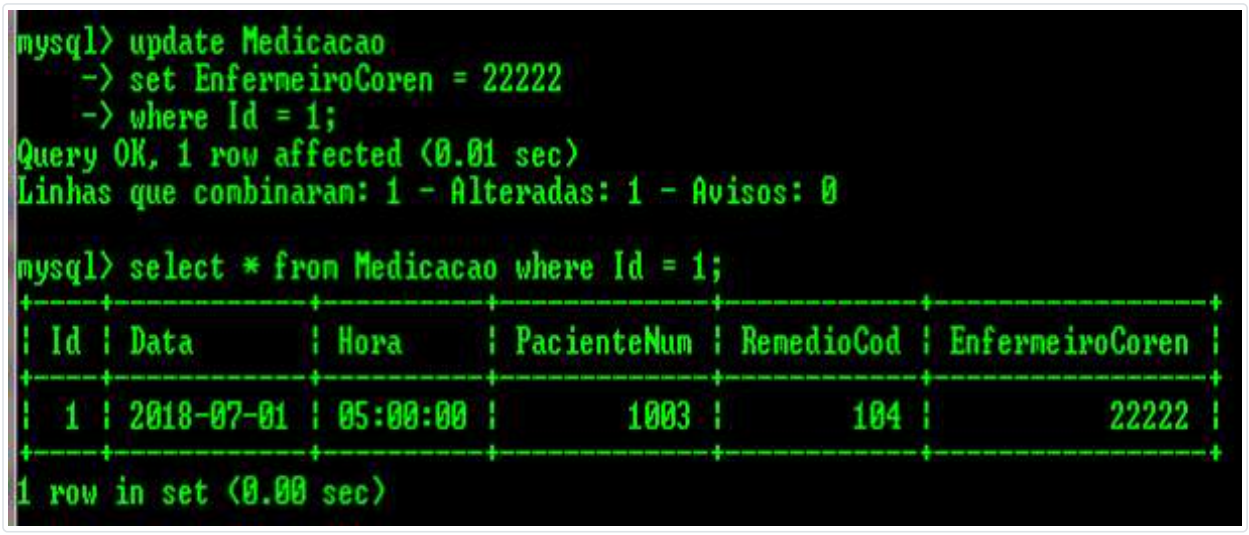
O MySQL vem como padrão de configuração, com o recurso COMMIT em modo automático, ou seja, AUTOCOMMIT. Nesse caso, para qualquer alteração feita no banco de dados, o SGBD armazena as atualizações no disco, sem que usuário necessite de um comando para fazer isso.

Porém, é possível que as confirmações sejam determinadas pelo administrador do banco de dados. Para isso o seguinte comando dever ser digitado no SGBD:

```
SET AUTOCOMMIT=0;
```

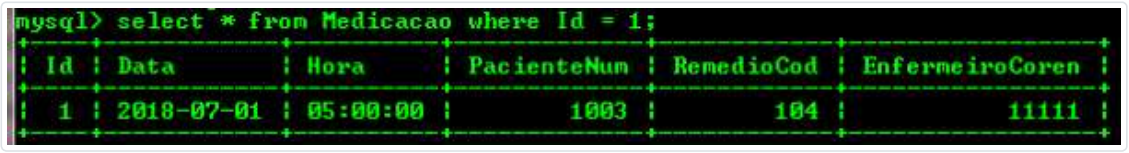
Desta forma, estamos concordando que o COMMIT seja feito manualmente, e não mais de forma automática como antes.

Exemplo: veja na imagem a seguir como efetuar a alteração do enfermeiro que fez o primeiro atendimento, de "Enfermeiro 1" para "Enfermeiro 2".



Fonte: elaborada pelo autor, captura de tela do software MySQL.

Porém não foi utilizado o COMMIT para registrar a alteração. Assim, ao fazer logout, e novamente se conectar ao banco para realizar a consulta, pode ser observado que a alteração do registro não foi efetuada.



Fonte: elaborada pelo autor, captura de tela do software MySQL.

Confirmar as alterações nas tabelas

Para que o UPDATE feito na tabela fique gravado (persistência), após a transação ser confirmada, é necessário utilizar o COMMIT. O comando é simples e direto:

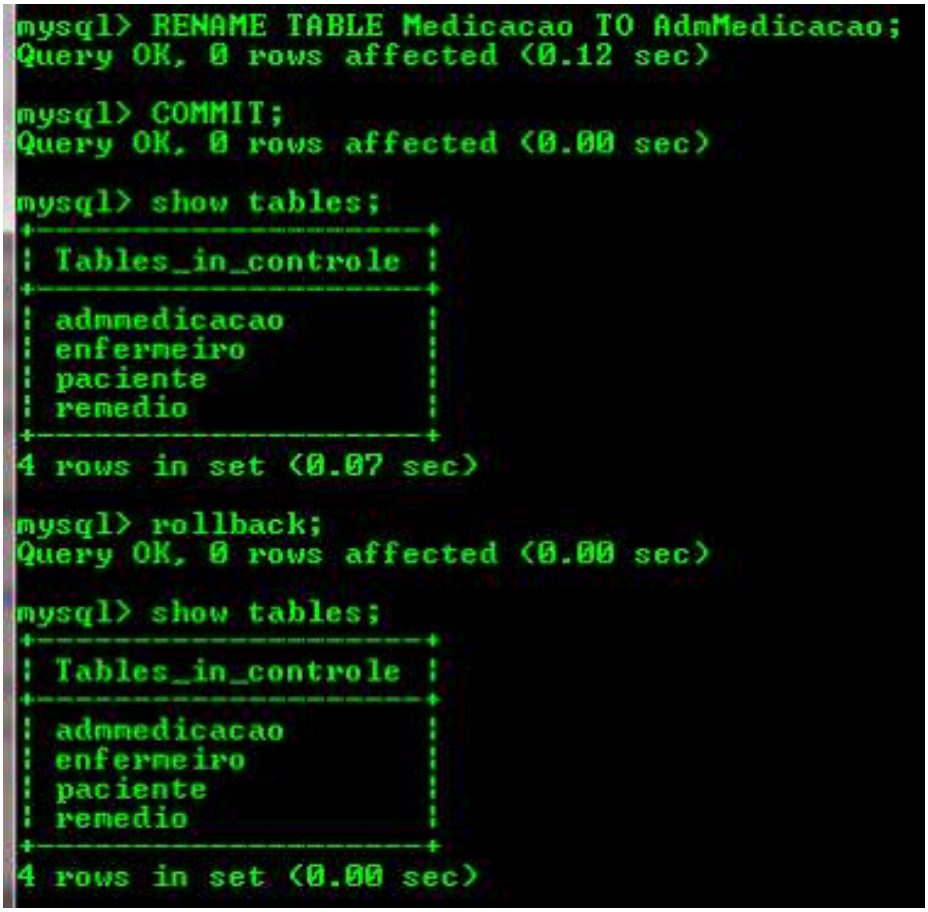
```
COMMIT;
```

Dessa forma, a integridade dos dados na tabela está garantida, fazendo com que a transação seja confirmada.

ROLLBACK

Para que uma transação feita no banco de dados possa ser revertida, é possível utilizar o recurso de **ROLLBACK para retornar ao estado anterior da transação**. Porém, as instruções linguagem de definição de dados (DDL, *data definition language*) de criação e exclusão de banco de dados, ou ainda, as alterações, exclusões e criação de tabelas não admitem o uso do ROLLBACK.

Exemplo



Fonte: elaborada pelo autor, captura de tela do software MySQL.

Criar um ponto de restauração

Para que possamos retornar a determinado ponto com o ROLLBACK, o SQL permite a criação de pontos de restauração, por meio da sintaxe:

```
SAVEPOINT [nomeDoPonto];
```

Utilizar um ponto de restauração

Para utilizar o ponto criado, deve ser usada a sintaxe:

```
ROLLBACK TO SAVEPOINT [nomeDoPonto];
```

Vale ressaltar que, para os controles transacionais SAVEPOINT e ROLLBACK funcionarem deve-se alterar o valor do AUTOCOMMIT para zero.

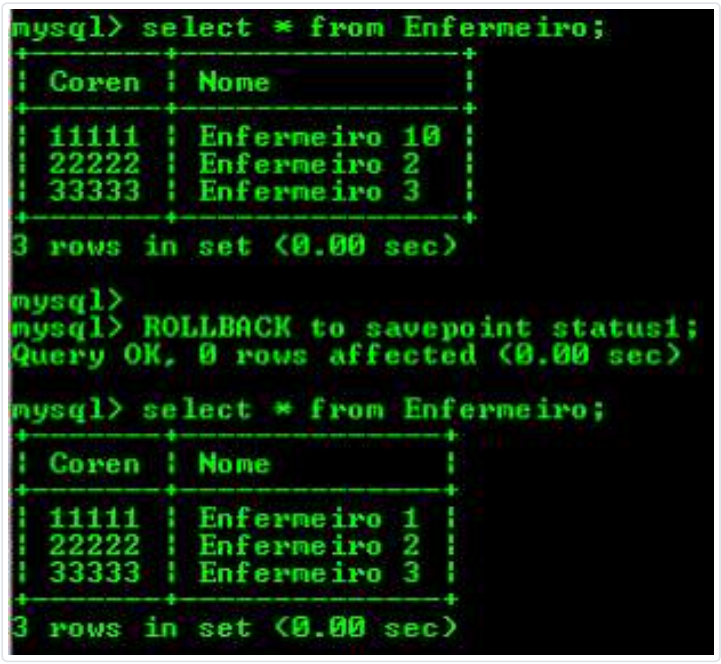
Exemplo: a tabela Enfermeiro possui três registros: “Enfermeiro 1”, “Enfermeiro 2” e “Enfermeiro 3”, com isso, vamos criar um ponto de salvação, por meio do comando:

```
SET AUTOCOMMIT=0;
SAVEPOINT status1;
```

Saiba Mais

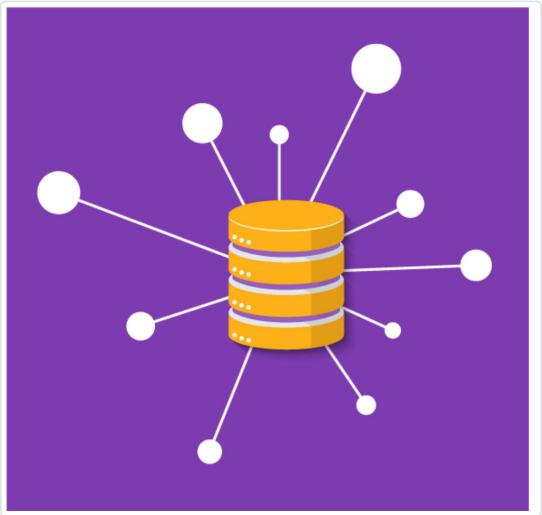
Posteriormente, o nome do “Enfermeiro 1” foi alterado para “Enfermeiro 10”. Para retornar o nome do enfermeiro, o ponto de salvação deve ser retornado por meio do comando:

Resultado do exemplo



Fonte: elaborada pelo autor, captura de tela do software MySQL.

Nesta webaula, foi possível compreender a importância de se manter a integridade dos bancos de dados, e com isso efetuar os desenvolvimentos de forma segura. Dessa forma, é possível criar pontos de restauração no banco de dados (comando SAVE POINT) e, caso necessário retornar a esse ponto, pode-se utilizar o comando ROLLBACK. Vimos, também, que a confirmação das alterações das bases de dados tem papel importante para a permanência dos registros na tabela (comando COMMIT).



Fonte: Shutterstock.