

SQL

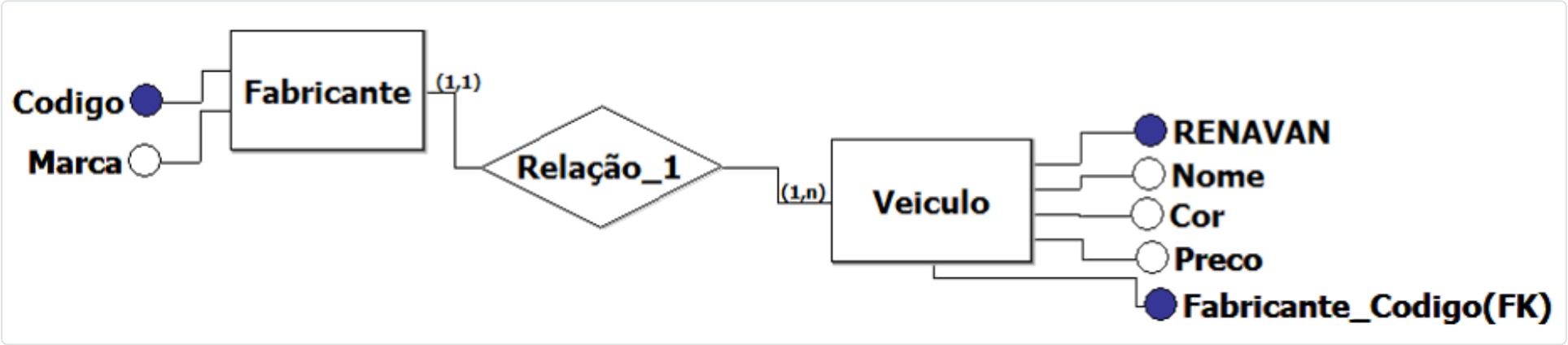
Visões e índices

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!

Devido à complexidade dos bancos de dados, e a quantidade de dados inseridos nas tabelas, as consultas realizadas podem sofrer um comprometimento no tempo de processamento. Portanto, nesta webaula vamos entender as VIEWS (visões), os INDEX (índices), e as buscas textuais com o FULLTEXT, que são as técnicas presentes na linguagem de programação de banco de dados SQL.

Contextualização

Para contextualizarmos as aplicações dos conceitos, vamos tomar o exemplo de banco de dados representado no diagrama de entidade relacionamento (DER) a seguir:



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

Script em SQL para a implementação do banco de dados:

```
1 CREATE DATABASE Car;
2 USE Car;
3
4 CREATE TABLE Fabricante (
5 Codigo INT(3) PRIMARY KEY AUTO_INCREMENT,
6 Marca CHAR(20) NOT NULL
7 );
8
9 CREATE TABLE Veiculo (
10 RENAVAN INT(8) PRIMARY KEY,
11 Nome VARCHAR(30) NOT NULL,
12 Cor VARCHAR (20) NOT NULL,
13 Preco DECIMAL(10,2) NOT NULL,
14 fabricante_Codigo INT(3) NOT NULL,
15 FOREIGN KEY (fabricante_Codigo) REFERENCES Fabricante (Codigo)
16 );
```

Depois que o banco de dados foi desenvolvido no sistema gerenciador de banco de dados (SGBD) MySQL, os registros foram inseridos conforme ilustrado na imagem a seguir.

```
mysql> select * from Fabricante;
```

Codigo	Marca
1	Volk
2	Fait
3	Cherroles
4	Fordys
5	Maudi
6	Junday

6 rows in set (0.00 sec)

```
mysql> select * from Veiculo;
```

RENAVAN	Nome	Cor	Preco	Codigo_fabricante
1234567	Cersas	azul	15000.00	3
1444558	Já	verde	49000.00	4
2582582	Montanha	lilas	62000.00	3
2589967	Hideas	prata	44000.00	2
4445566	AAR5	azul	80000.00	5
10102020	Cheveiro	preto	22000.00	1
11111111	EspacialPex	amarelo	39000.00	1
11122255	10S	preto	33000.00	3
12312312	Cersas	rosa	18000.00	3
12345678	AAR3	prata	44000.00	5
14714714	Jatus	prata	45000.00	1
22222222	Seniel	preto	18000.00	2
30303030	Estradus	preto	27000.00	2
33333333	Pins	preto	40000.00	3
36544477	Linearr	prata	35000.00	2
44444444	Pins	prata	38000.00	3
45645645	Hideas	branco	42000.00	2
55220044	Festinn	branco	25000.00	4
65465465	AAR3	verde	54000.00	5
66666666	Já	preto	19000.00	4
74174174	10S	azul	23000.00	3
77889966	Montanha	preto	32000.00	3
78889994	Jatus	prata	55000.00	1
78978998	Golos	dourado	82000.00	1
85285285	Linearr	amarelo	55000.00	2
87654321	Golos	azul	32000.00	1
95195195	Golos	preto	18000.00	1
96396396	Festinn	marrom	25000.00	4
98798798	AAR5	blindado	40000.00	5

29 rows in set (0.00 sec)

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

VIEW

O recurso SQL para gerar visões é uma alternativa para visualizar os dados de uma ou mais tabelas de um BD. Uma visão, pode ser considerada uma “tabela virtual”, ou ainda, uma consulta pré-armazenada por meio de scripts. Geralmente a técnica de VIEW encapsula uma seleção de dados (SELECT), na qual esses dados da tabela virtual são armazenados no cache do SGBD.

A VIEW é uma das técnicas que podem auxiliar na diminuição na carga de processamento. Ao utilizar uma VIEW para efetuar seleção de dados, as consultas tornam-se mais rápidas, e exige menos carga de processamento. Isso ocorre, porque uma VIEW não necessita fazer o retrabalho ao executar um SELECT, pois a seleção já está pré-armazenada.

A seguir, veja as sintaxes para criação, visualização, utilização e exclusão de uma VIEW:

Criação

Sintaxe utilizada para se desenvolver uma VIEW:

```
CREATE VIEW [nome_da_VIEW] AS
SELECT [coluna]
FROM [tabela]
WHERE [condições];
```

Exemplo: Partindo do cenário desenvolvido, vamos desenvolver uma VIEW que selecione a marca do fabricante, o nome, a cor e o preço do veículo, quando os valores desses veículos forem menor do que R\$ 50.000,00.

Sintaxe:

```
CREATE VIEW v_selecta AS
SELECT veiculo.nome as “Veiculo”, fabricante.marca as “Marca”, veiculo.cor as Cpr, veiculo.preco as
“Valor”
FROM veiculo IINER JOIN fabricante
WHERE veiculo.fabricante_Codigo = fabricante.código AND veiculo.preco <= 50000;
```

Visualização

Como as VIEWS são consideradas “tabelas virtuais”, para visualizá-las, basta exibirmos as tabelas inseridas no BD.

Sintaxe:

```
SHOW TABLES
```

Resultado:



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

Utilização

Para utilizarmos uma VIEW para exibir uma consulta, deve ser utilizada a sintaxe a seguir:

Sintaxe:

```
SELECT * FROM [nome_da_VIEW; ]
```

No exemplo desenvolvido:

```
SELECT * FROM v_select1;
```

Resultado:

```
mysql> select * from v_select1;
```

Veiculo	Marca	Cor	Valor
Cheveiro	Volk	preto	22000.00
EspacialFex	Volk	amarelo	39000.00
Jatus	Volk	prata	45000.00
Golos	Volk	azul	32000.00
Golos	Volk	preto	18000.00
Hideas	Fait	prata	44000.00
Seniel	Fait	preto	18000.00
Estradus	Fait	preto	27000.00
Linearr	Fait	prata	35000.00
Hideas	Fait	branco	42000.00
Cersas	Chervroles	azul	15000.00
10S	Chervroles	preto	33000.00
Cersas	Chervroles	rosa	18000.00
Pins	Chervroles	preto	40000.00
Pins	Chervroles	prata	38000.00
10S	Chervroles	azul	23000.00
Montanha	Chervroles	preto	32000.00
Já	Fordys	verde	49000.00
Festinn	Fordys	branco	25000.00
Já	Fordys	preto	19000.00
Festinn	Fordys	marrom	25000.00
AAR3	Maudi	prata	44000.00
AAR5	Maudi	blindado	40000.00

```
23 rows in set (0.02 sec)
```

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

Exclusão



Para excluir uma VIEW a sintaxe utilizada deve ser:

Sintaxe:

```
DROP VIEW [nome_da_VIEW];
```

INDEX

O recurso SQL para aumentar a velocidade das consultas nos bancos de dados é a utilização de índices. O recurso de índice (INDEX no MySQL) não era admitido até a versão SQL:1999. Após isso, os engenheiros buscaram um recurso para diminuir a taxa de processamento nas buscas nas tabelas, e para imposição das restrições de integridade.

A utilização dos índices é opcional para a seleção de dados, pois os índices são considerados estruturas redundantes. O SGBD pode decidir quais índices devem ser criados, porém nem sempre essa escolha automatizada, pode trazer algum benefício no processamento.

A seguir, veja as sintaxes para declaração, verificação, utilização e exclusão de um INDEX:

Declaração



- Sintaxe para declarar um índice, no desenvolvimento da tabela:

```
CREATE TABLE [nomeDaTabela] (  
    Campo1 tipo(tamanho),  
    Campo2 tipo(tamanho),  
    INDEX(Campo1)
```

- Sintaxe para declarar um índice, em uma tabela existente no BD:

```
CREATE TABLE [nomeDoIndice] ON  
    [nomeDaTabela](Campo);
```

Exemplo: Vamos criar um índice na chave primária RENAVAN (Registro Nacional de Veículos Automotores) da tabela veículo, no nosso exemplo, utilizamos a sintaxe:

```
CREATE INDEX idx_Renavam ON veiculo(RENAVAN);
```

Verificação



Sintaxe para nos certificarmos que os índices foram criados:

```
SHOW INDEX FROM [nomeDaTabela];
```

No exemplo desenvolvido:

```
SHOW INDEX FROM veiculo;
```

Resultado:

```
mysql> show index from veiculo;
+-----+-----+-----+-----+-----+
| Table      | Non_unique | Key_name      | Seq_in_index | Column_name |
+-----+-----+-----+-----+-----+
| veiculo    | 0          | PRIMARY       | 1            | RENAUVAN    |
| veiculo    | 1          | fabricante_Codigo | 1            | fabricante_Codigo |
| veiculo    | 1          | idx_Renavam   | 1            | RENAUVAN    |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Utilização

Sintaxe:

```
SELECT [coluna] FROM [nomeDaTabela]
USE INDEX (nomeDoIndice)
WHERE [condições];
```

No exemplo desenvolvido:

```
SELECT nome AS "Veiculo", cor AS "Cor", Preço AS "Valor"
FROM veiculo
USE INDEX(idx_Renavam)
WHERE preco<= 50000;
```

Resultado:

```
+-----+-----+-----+
| Veiculo      | Cor      | Valor      |
+-----+-----+-----+
| Cersas       | azul     | 15000.00   |
| Já           | verde    | 49000.00   |
| Hideas       | prata    | 44000.00   |
| Cheveiro     | preto    | 22000.00   |
| EspacialFex  | amarelo  | 39000.00   |
| 10S          | preto    | 33000.00   |
| Cersas       | rosa     | 18000.00   |
| AAR3         | prata    | 44000.00   |
| Jatus        | prata    | 45000.00   |
| Seniel       | preto    | 18000.00   |
| Estradus     | preto    | 27000.00   |
| Pins         | preto    | 40000.00   |
| Linearrrr    | prata    | 35000.00   |
| Pins         | prata    | 38000.00   |
| Hideas       | branco   | 42000.00   |
| Festinnnn    | branco   | 25000.00   |
| Já           | preto    | 19000.00   |
| 10S          | azul     | 23000.00   |
| Montanha     | preto    | 32000.00   |
| Golos        | azul     | 32000.00   |
| Golos        | preto    | 18000.00   |
| Festinnnn    | marrom   | 25000.00   |
| AAR5         | blindado | 40000.00   |
+-----+-----+-----+
23 rows in set (0.04 sec)
```

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

Exclusão

Sintaxe:

```
DROP INDEX (nomeDoIndice);
```

FULLTEXT

Outro recurso que tem uma função muito parecida com o INDEX é o FULLTEXT. Esse recurso tem a capacidade de buscar um trecho dentro de várias *strings*, assim como a função “localizar” existente nos navegadores de internet, editores de texto, etc.

Sintaxe:

```
ALTER TABLE [nome_tabela] ADD FULLTEXT (nome_da_coluna);
```

Nesse comando ao especificar uma determinada coluna como FULLTEXT, a mesma passa a ter as strings no interior de um texto monitoradas.

Para utilizar esse recurso, deve-se utilizar a sintaxe descrita a seguir:

```
SELECT [coluna] FROM nome_da_tabela  
WHERE MATCH(coluna)  
AGAINST('palavra_desejada');
```

Saiba Mais

Nesta webaula, pudemos compreender de que forma os recursos SQL denominados VIEW e INDEX, podem auxiliar na missão em economizar o processamento das informações quando se necessita efetuar consultas em base de dados muito extensa.



Fonte: Shutterstock.

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:

```
1 CREATE TABLE Enfermeiro;
2 Core INT PRIMARY KEY,
3     Nome VARCHAR(50) NOT NULL
4 );
5 CREATE TABLE Paciente (
6     Num INT PRIMARY KEY,
7     Nome VACHAR(50) NOT NULL
8 );
9 CREATE TABLE Remedio (
10    Cod INT PRIMARY KEY,
11    Nome VARCHAR (50) NOT NULL
12 );
13 CREATE TABLE Medicacao (
14    Id INT PRIMARY KEY AUTO_INCREMENT,
15    Data DATE NOT NULL,
16    Hora TIME NOT NULL,
17    PacienteNum INT NOT NULL,
18    RemedioCod INT NOT NULL,
19    EnfermeiroCoren INT NOT NULL,
20    FOREIGN KEY (PacienteNum) REFERENCES Paciente(Num)
21    FOREIGN KEY (RemedioCod) REFERENCES Remedio(Cod)
22    FOREIGN KEY (EnfermeiroCoren) REFERENCES Enfermeiro(Coren)
23 );
```

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.

```
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”.

```
mysql> update Medicacao
-> set EnfermeiroCoren = 22222
-> where Id = 1;
Query OK, 1 row affected (0.01 sec)
Linhas que combinaram: 1 - Alteradas: 1 - Avisos: 0

mysql> select * from Medicacao where Id = 1;
+----+-----+-----+-----+-----+-----+
| Id | Data      | Hora      | PacienteNum | RemedioCod | EnfermeiroCoren |
+----+-----+-----+-----+-----+-----+
| 1  | 2018-07-01 | 05:00:00 | 1003        | 104        | 22222           |
+----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

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.

```
mysql> select * from Medicacao where Id = 1;
+----+-----+-----+-----+-----+-----+
| Id | Data      | Hora      | PacienteNum | RemedioCod | EnfermeiroCoren |
+----+-----+-----+-----+-----+-----+
| 1  | 2018-07-01 | 05:00:00 | 1003        | 104        | 11111           |
+----+-----+-----+-----+-----+-----+
```

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

```
mysql> RENAME TABLE Medicacao TO AdmMedicacao;
Query OK, 0 rows affected (0.12 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
+-----+
| Tables_in_controle |
+-----+
| admmedicacao       |
| enfermeiro         |
| paciente           |
| remedio             |
+-----+
4 rows in set (0.07 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
+-----+
| Tables_in_controle |
+-----+
| admmedicacao       |
| enfermeiro         |
| paciente           |
| remedio             |
+-----+
4 rows in set (0.00 sec)
```

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.

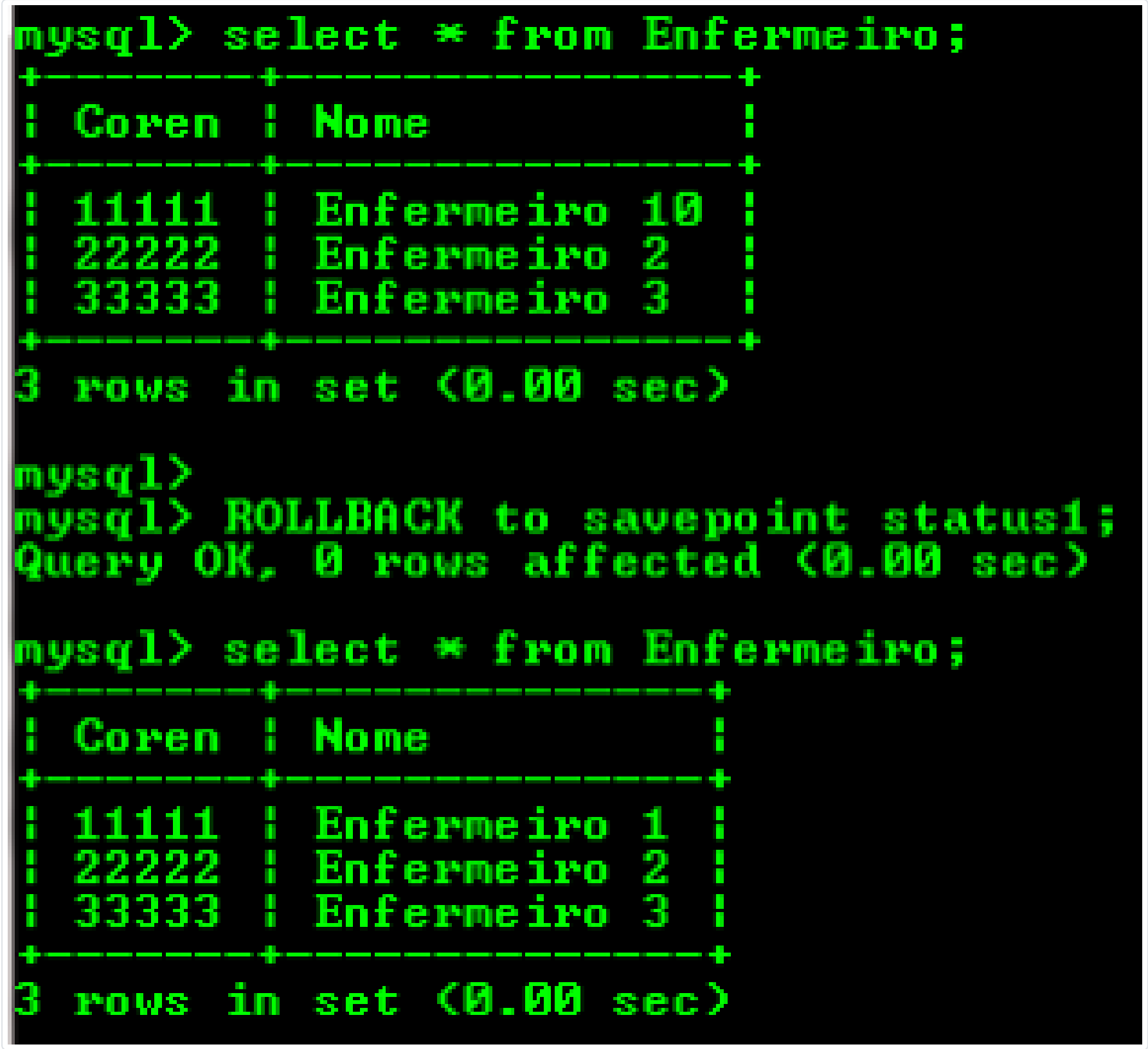
Exemplo: a tabela Enfermeiro possui três registros: “Enfermeiro 1”, “Enfermeiro 2” e “Enfermeiro 3”, com isso, vamos criar um ponto de salvag o, por meio do comando:

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

[Saiba Mais](#)

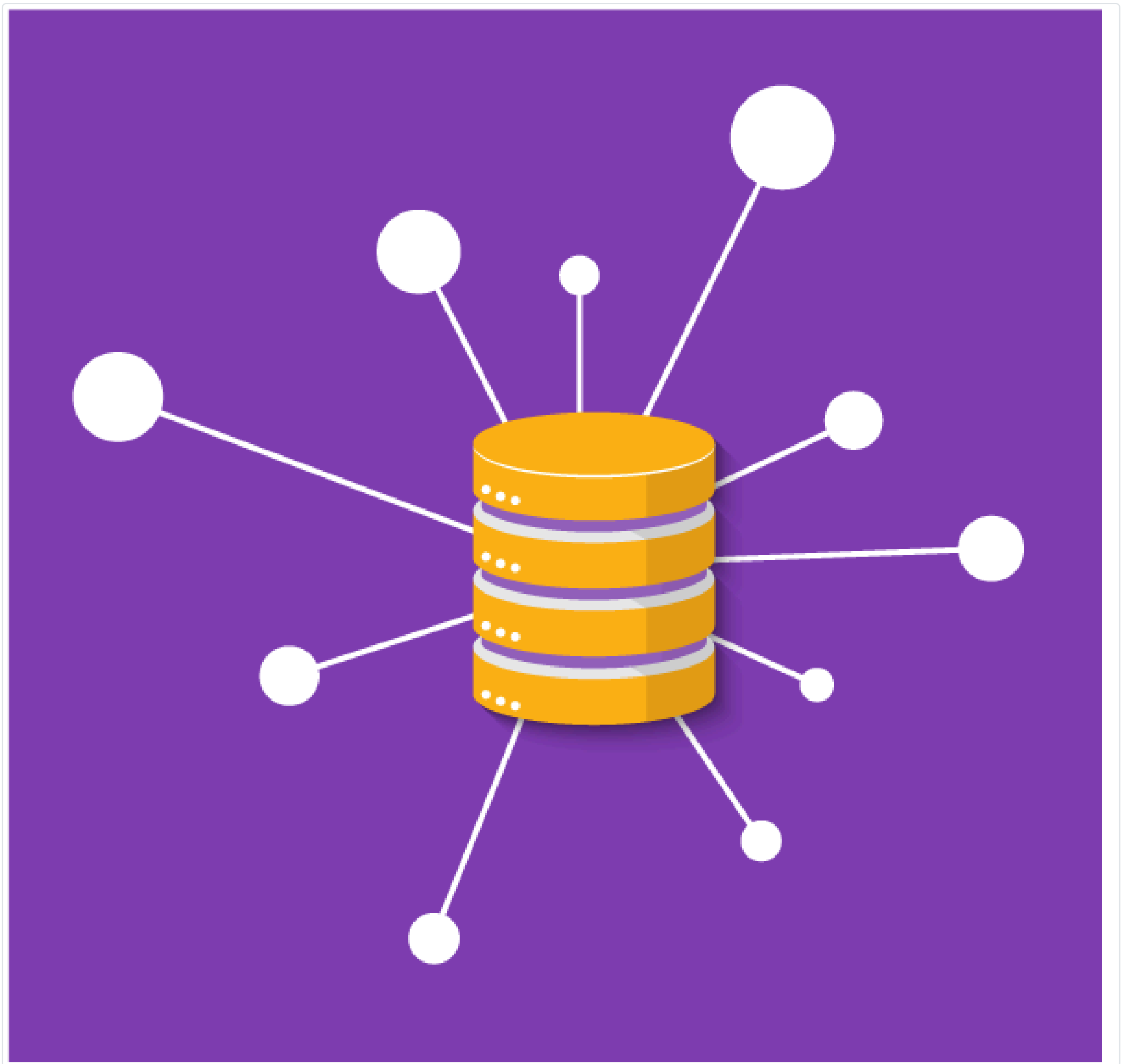
Posteriormente, o nome do “Enfermeiro 1” foi alterado para “Enfermeiro 10”. Para retornar o nome do enfermeiro, o ponto de salvag o deve ser retornado por meio do comando:

Resultado do exemplo ▾



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

Nesta webaula, foi poss vel compreender a import ncia de se manter a integridade dos bancos de dados, e com isso efetuar os desenvolvimentos de forma segura. Dessa forma,   poss vel criar pontos de restaura o no banco de dados (comando SAVE POINT) e, caso necess rio retornar a esse ponto, pode-se utilizar o comando ROLLBACK. Vimos, tamb m, que a confirma o das altera o das bases de dados tem papel importante para a perman ncia dos registros na tabela (comando COMMIT).



Fonte: Shutterstock.

SQL

Procedimentos e Funções

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!

A linguagem de programação de banco de dados SQL possui recursos específicos que podem automatizar algumas execuções, permitindo que o tempo de algumas consultas, o resultado dos cálculos e outros processos possíveis dentro de um banco de dados sejam efetuados em menor tempo e menor consumo de memória/processamento do servidor que o sistema de gerenciamento de banco de dados está instalado. Desta forma, nesta webaula vamos ver como desenvolver funções e procedimentos armazenados.

Banco de dados de exemplo

Para exemplificar os conceitos e aplicações, foi desenvolvido um banco de dados para guardar as informações. As tabelas desenvolvidas para esse fim foram: Aluno, Disciplina e Notas. A seguir veja os registros inseridos em cada uma delas.

```
mysql> select * from Aluno;
```

RA	Nome	Telefone
1234	Aluno_A	988776655
1235	Aluno_B	997975566
1236	Aluno_C	988225544
1237	Aluno_D	966887744
1238	Aluno_E	911223344
1239	Aluno_F	922334455

```
mysql> select * from Disciplina;
```

Id	Nome
1	Banco de dados
2	Programação estruturada
3	Redes de computadores
4	LFA

```
mysql> select * from Notas;
```

AlunoRA	DisciplinaId	NotaP1	NotaP2
1234	1	7.0	5.5
1235	1	7.0	5.5
1236	1	6.0	8.5
1237	1	5.0	3.5
1238	1	2.5	3.5
1239	1	9.0	5.5
1234	2	6.0	7.5
1235	2	6.5	8.5
1236	2	5.0	4.5
1237	2	8.0	7.0
1238	2	7.5	6.5
1239	2	6.0	5.5
1234	3	8.5	5.5
1235	3	3.5	7.5
1236	3	7.0	3.5
1237	3	2.0	7.0
1238	3	2.5	7.5
1239	3	4.0	9.5
1234	4	5.0	6.5
1235	4	7.5	7.5
1236	4	7.0	6.5
1237	4	6.0	7.0
1238	4	4.5	3.5
1239	4	2.0	2.5

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

Funções (FUNCTION)

É uma técnica que possibilita realizar cálculos aritméticos complexos utilizando os valores das colunas existentes em um banco de dados. Basicamente o intuito ao se utilizar uma FUNCTION, é retornar tabelas como resultado, conhecidas como funções de tabela.


```
CREATE FUNCTION nome_da_funcao (x tipo, y tipo)
RETURNS tipo
RETURN (função);
```

Utilizar uma função



```
SELECT nome_da_funcao (parâmetro x, parâmetro y)
FROM nome_da_tabela
WHERE nome_da_coluna (condição);
```

Excluir uma função



```
DROP FUNCTION nome_da_funcao;
```

Exemplo

Vamos desenvolver uma função no banco de dados de exemplo, para o cálculo da média final, onde: Média Final = (NotaP1 * 0,4) + (NotaP2 * 0,6).

Sintaxe:

```
CREATE FUNCTION fn_media(x DECIMAL(3,1), y DECIMAL(3,1))  
RETURNS DECIMAL(3,1)  
RETURN (x * 0.4) + (y * 0.6);
```

Para utilizarmos a **FUNCTION fn_media**, com notas entre 4,1 e 6,9 inclusive, deve ser utilizada a seguinte sintaxe SQL:

```
SELECT Aluno.Nome, Disciplina.Nome AS "Disciplina",
fn_media(NotaP1, NotaP2) AS "Média Final"
FROM Notas INNER JOIN Aluno
ON Notas.AlunoRA = Aluno.RA
INNER JOIN Disciplina
ON Notas.DisciplinaId = Disciplina.Id
WHERE fn_media(NotaP1, NotaP2) >= 4.0
AND fn_media(NotaP1, NotaP2) <= 6.9;
```

Resultado do exemplo ▾

Nome	Disiciplina	Média Final
Aluno_A	Banco de dados	6.1
Aluno_B	Banco de dados	6.1
Aluno_D	Banco de dados	4.1
Aluno_F	Banco de dados	6.9
Aluno_A	Programação estruturada	6.9
Aluno_C	Programação estruturada	4.7
Aluno_E	Programação estruturada	6.9
Aluno_F	Programação estruturada	5.7
Aluno_A	Redes de computadores	6.7
Aluno_B	Redes de computadores	5.9
Aluno_C	Redes de computadores	4.9
Aluno_D	Redes de computadores	5.0
Aluno_E	Redes de computadores	5.5
Aluno_A	LFA	5.9
Aluno_C	LFA	6.7
Aluno_D	LFA	6.6

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

Procedimento armazenado (PROCEDURE)

Os procedimentos armazenados foram definidos, para que fosse obtida capacidade procedural aos bancos de dados, porém não com a intenção de substituir técnicas que também estão disponíveis nas linguagens de programação como o Java, C++ ou C#.

Esse recurso deve permitir armazenar procedimentos como seleção de dados, exclusão de registros, alteração de dados, entre outras funções disponíveis na linguagem de programação de banco de dados SQL.

Criar um procedimento ▾

```
CREATE PROCEDURE nome_do_procedure (var_nome tipo)
Declarações.
```

Utilizar um procedimento ▾

```
CALL nome_do_procedure (var_nome);
```

Excluir um procedimento



```
DROP PROCEDURE nome_do_procedure;
```

Exemplo

Vamos desenvolver um procedimento armazenado para calcular a média geral de todos os alunos que estão de exame nas disciplinas:

1. Banco de Dados
2. Programação Estruturada
3. Redes de Computadores
4. LFA

Sintaxe:

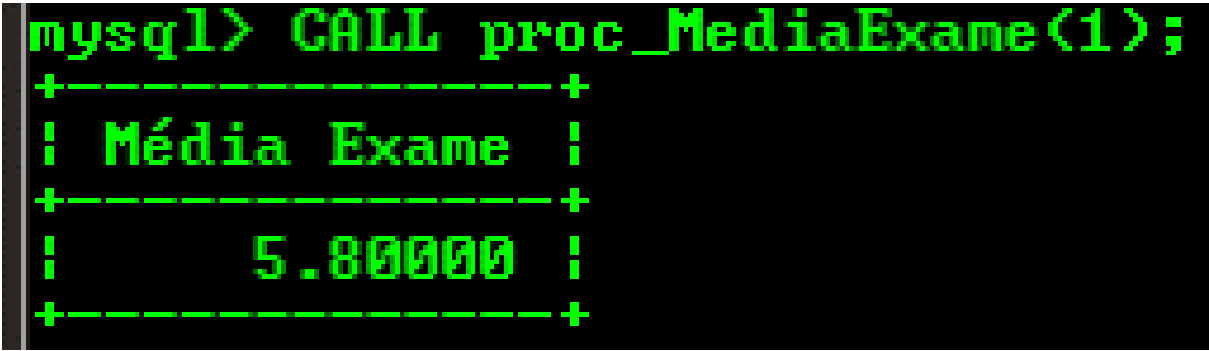
```
CREATE PROCEDURE proc_MediaExame(var_DisciplinaId int)
SELECT AVG (fn_media(NotaP1, NotaP2)) AS “Média Exame”
FROM Notas
WHERE DisciplinaId = var_DisciplinaId
      AND (fn_media(NotaP1, NotaP2) >= 4.0
      AND (fn_media(NotaP1, NotaP2) <= 6.9);
```

A seguir, veja como os procedimentos são utilizados.

Banco de Dados

Para utilizar um procedimento armazenado e calcular a média geral dos alunos em exame em Banco de Dados, utilizamos a sintaxe SQL:

CALL proc_MediaExame(1);



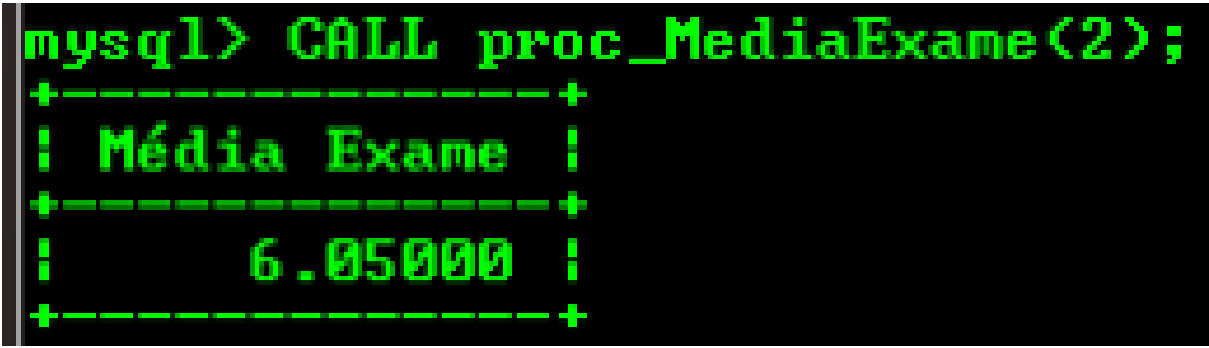
```
mysql> CALL proc_MediaExame(1);
+-----+
| Média Exame |
+-----+
|      5.800000 |
+-----+
```

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

Programação Estruturada

Para utilizar um procedimento armazenado e calcular a média geral dos alunos em exame em Programação Estruturada, utilizamos a sintaxe SQL:

CALL proc_MediaExame(2);



```
mysql> CALL proc_MediaExame(2);
+-----+
| Média Exame |
+-----+
|      6.050000 |
+-----+
```

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

Redes de Computadores

Para utilizar um procedimento armazenado e calcular a média geral dos alunos em exame em Rede de Computadores, utilizamos a sintaxe SQL:

CALL proc_MediaExame(3);

```
mysql> CALL proc_MediaExame(3);
+-----+
| Média Exame |
+-----+
|      5.60000 |
+-----+
```

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

LFA



Para utilizar um procedimento armazenado e calcular a média geral dos alunos em exame em LFA, utilizamos a sintaxe SQL:

```
CALL proc_MediaExame(4);
```

```
mysql> CALL proc_MediaExame(4);
+-----+
| Média Exame |
+-----+
|      6.40000 |
+-----+
```

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

Nesta webaula, vimos assuntos relacionados a automação de alguns processos dentro dos bancos. Foi possível compreender o papel das funções, onde foram efetuados cálculos aritméticos, por meio da sintaxe FUNCTIONS. Também vimos que os procedimentos armazenados permitem se deixar algumas execuções nas bases de dados pré-armazenadas, por meio da sintaxe PROCEDURE.

Para visualizar o vídeo, acesse seu material digital.