



Cursos


► Listagem de disciplinas

Selecione uma disciplina

Aulas

- 01 Introdu  o a Banco de Dados
- 02 Modelo de Entidade e Relacionamento
- 03 Modelo Relacional
- 04 Transforma  es ER para MR
- 05 Transforma  es ER para MR e dicion rio de dados
- 06 Normaliza  o b sica
- 07 Normaliza  o avan ada
- 08 Introdu  o   Linguagem SQL e Sistemas Gerenciadores de Banco de Dados
- 09 Linguagem SQL - cria  o, inser  o e modifica  o de tabelas
- 10 Linguagem SQL - Consulta simples de tabelas
- 11 Linguagem SQL - Consulta avan ada de tabelas
- 12 Linguagem SQL - Altera  o da estrutura de tabelas e ambientes de m ltiplas tabelas
- 13 Linguagem SQL - Subconsultas
- 14 Linguagem SQL - VIS  ES
- 15 Linguagem SQL - STORED PROCEDURES
- 16 Linguagem SQL - Fun  es
- 17 Linguagem SQL - Seguran a
- 18 Engenharia Reversa
- 19 Utilizando SQL em Java
- 20 Utilizando conceitos avan ados de SQL em Java


◀ Voltar ◀ Imprimir ◀ Topo



Sistemas de Banco de Dados

Aula 17 – Linguagem SQL – Seguran a

Professores autores
Jos  Josemar de Oliveira J nior (josemar@ect.ufrn.br)
Luciana Ribeiro Veloso (luciana.veloso@globo.com)



Apresenta o

Nesta aula, estudaremos seguran a de sistema e seguran a de banco de dados. Em seguran a de sistemas, veremos como criar uma conta de **usu rio** no *MySQL* e como acessar o sistema *MySQL* a partir de uma determinada conta. Em seguida, estudaremos seguran a de banco de dados. Aprenderemos como controlar o que os usu rios podem fazer com os objetos (tabelas, vis es, fun  es e *stored procedures*) baseados nos privil gios atribuídos a cada usu rio. Finalizaremos a nossa aula estudando o comando que revoga permiss es dos usu rios.



Objetivo

Ao final desta aula, voc  ser  capaz de:

- criar contas no sistema *MySQL*;
- acessar o sistema *MySQL* a partir da linha de **comando** do Windows;
- conceder permiss es aos usu rios;
- revogar permiss es dos usu rios.

Segurança

Em ambientes com múltiplos usuários, é importante proteger o banco de dados de alterações indevidas nos dados ou nas estruturas das tabelas, as quais podem comprometer a integridade do banco de dados. Além disso, evita o acesso de determinados usuários a dados sigilosos, como, por exemplo, a folha de pagamento dos empregados de uma empresa. Com esse propósito, os SGBDs possuem um conjunto de regras e mecanismos de proteção de acesso ao banco de dados denominado segurança ou autorização.

A segurança em banco de dados pode ser classificada em duas categorias:

- **segurança de sistema:** relaciona-se com o controle de acesso ao banco de dados no nível de sistema, como, por exemplo, nome de usuário e senha;
- **segurança de banco de dados:** relaciona-se com o controle de uso dos objetos do banco de dados e as ações que esses usuários podem realizar sobre os objetos.

Segurança de sistema

Até o momento, trabalhamos com apenas um único usuário em nosso banco de dados, o usuário **root**, que por definição é o primeiro usuário do SGBD. Como nos demais sistemas, o usuário **root** possui o controle completo sobre o banco de dados, tendo inclusive a permissão de incluir novos usuários no banco de dados.

Mas, como permitir que mais usuários utilizem o banco de dados? Todos eles devem acessar o banco de dados através da conta **root**? Permitir o acesso a um sistema através de uma única conta com todas as permissões, como a conta **root**, é geralmente perigoso. Cada usuário deve possuir um **login**, ao qual está associada uma conta de acesso ao banco de dados, com determinadas permissões, conforme for o caso.

Então, como proceder para adicionar uma nova conta de acesso? A sintaxe para adicionar uma conta ao sistema **MySQL** é descrita no destaque abaixo.

```
mysql> CREATE USER nome_da_conta  
IDENTIFIED BY 'password';
```

O comando **CREATE USER** cria uma nova conta no **MySQL**. Para cada conta criada, esse comando insere uma nova linha na tabela **mysql.user** sem qualquer privilégio. A tabela **mysql.user** é mantida pelo SGBD e contém informações sobre todas as contas de acesso (**login**, senha e o que cada conta tem permissão de fazer em cada banco de dados).

No **MySQL**, o usuário é constituído de um nome mais o **host** de onde ele poderá acessar o **servidor** do banco de dados (**usuario@host**). Caso você não informe o **host** para o usuário, o **MySQL** assumirá "%", isto é, todos os **hosts**.

Vamos criar, inicialmente, um usuário com **login josemar** e senha (**password**) **111111** no SGBD **MySQL**? Para criarmos a conta **josemar**, utilizamos o seguinte comando.

```
mysql> CREATE USER josemar  
IDENTIFIED BY '111111';
```

A resposta do sistema SGBD, no caso **MySQL**, para o comando **CREATE USER**, é ilustrada na Figura 1.

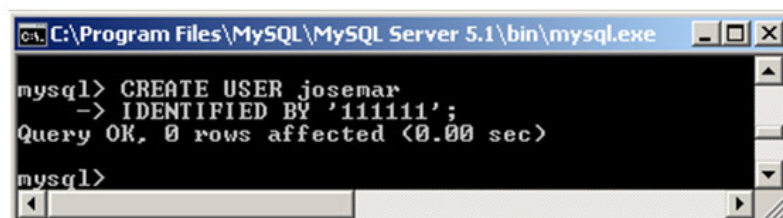
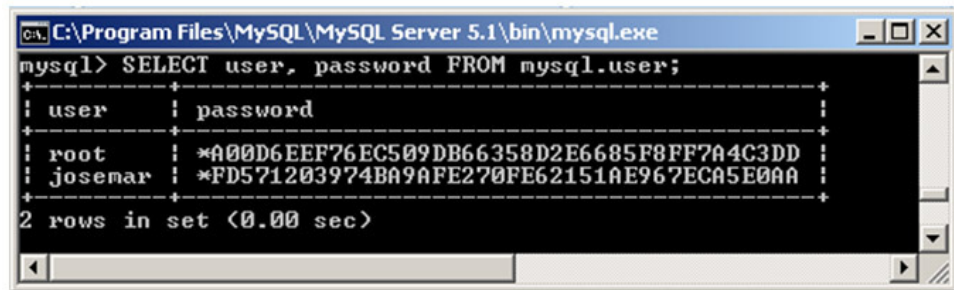


Figura 1 – Tela do **MySQL** após o comando **CREATE USER**
Fonte: **MySQL Server 5.1**

É importante esclarecer que a linguagem **SQL** não especifica como gerenciar os usuários. A criação dos usuários varia de sistema para sistema. Sendo necessário consultar a documentação para encontrar o comando correto para criar um usuário em cada sistema **SQL**.

Para visualizarmos os atributos **user** e **password** da tabela **mysql.user**, utilizamos o comando **SELECT** (Aula 10), conforme apresentado na Figura 2. Observe que as informações referentes ao campo **password** estão codificadas.



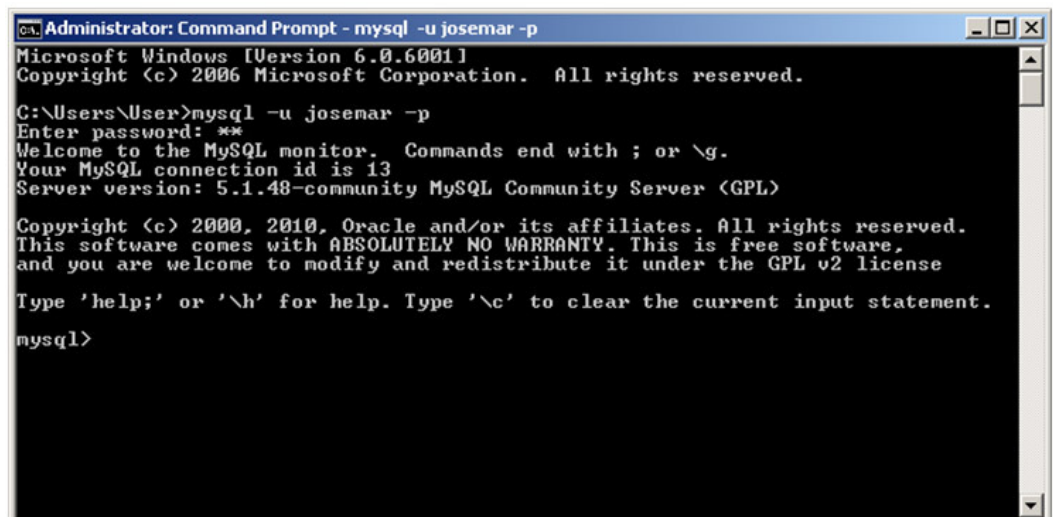
```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe
mysql> SELECT user, password FROM mysql.user;
+-----+-----+
| user      | password |
+-----+-----+
| root      | *A00D6EEF76EC509DB66358D2E6685F8FF7A4C3DD |
| josemar   | *FD571203974BA9AFE270FE62151AE967ECA5E0AA |
+-----+-----+
2 rows in set (0.00 sec)
```

Figura 2 - Tela do MySQL após o comando SELECT. Fonte: MySQL Server 5.1

Você deve estar se perguntando: "Como proceder para acessar o banco de dados com uma conta diferente da *root*?". Bom, para ter acesso ao MySQL a partir de uma conta qualquer e começar a interagir com seu banco de dados, você deve procurar o prompt de comando do *Windows*, acessível diretamente pelo menu Iniciar/Acessórios do seu ambiente *Windows*. Ao clicar no ícone, aparecerá a tela da linha de comando no qual se deve digitar o seguinte comando para entrar no SGBD MySQL.

> `mysql -u login_do_usuario -p`

Como resposta a esse comando, será solicitada a senha do usuário. Digite a senha e tecla *enter*. O sistema SGBD será acessado, aparecendo a tela de boas vindas do ambiente MySQL (Figura 3).



```
Administrator: Command Prompt - mysql -u josemar -p
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

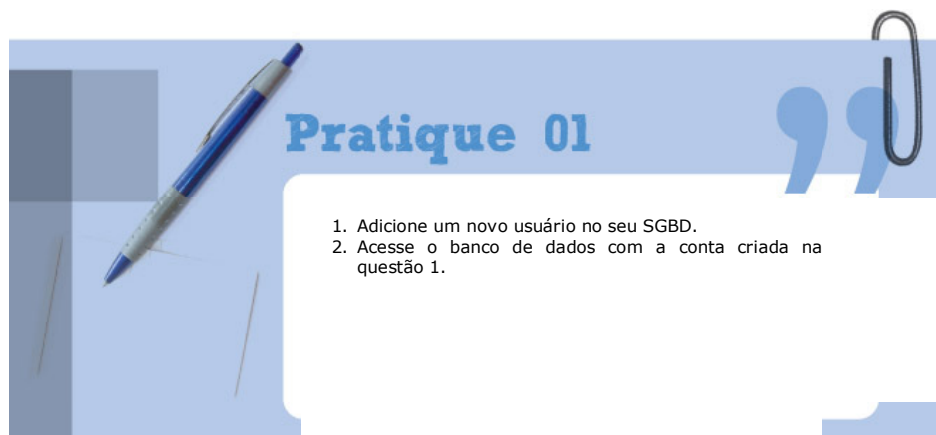
C:\Users\User>mysql -u josemar -p
Enter password: **
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 5.1.48-community MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Figura 3 - Tela inicial do MySQL. Fonte: MySQL Server 5.1



Pratique 01

1. Adicione um novo usuário no seu SGBD.
2. Acesse o banco de dados com a conta criada na questão 1.

Segurança de banco de dados

Para executar qualquer atividade em um banco de dados, o usuário deve ter as permissões adequadas. Diferentemente da conta *root*, os novos usuários que forem criados utilizando o comando *CREATE USER*, conforme foi descrito, não possuem permissão para executar nenhum comando SQL. Portanto, para cada novo usuário, é necessário especificar quais dados e comandos ele terá a permissão de acessar e utilizar, evitando assim o uso não autorizado, através da concessão de permissão.

Para conceder permissão no *MySQL*, utiliza-se o comando *GRANT*. Esse comando concede permissões específicas no objeto (tabela, visão, função e *stored procedures*) para um ou mais usuários ou grupos de usuário. Essas permissões são adicionadas às já concedidas, caso existam. A sintaxe resumida do comando *GRANT* é exibida no destaque a seguir.

mysql> GRANT lista_de_privilegios ON lista_do_objeto
TO lista-de-usuarios ;

No comando acima, o primeiro item a ser informado é a lista de privilégios a serem concedidos aos usuários. Os privilégios mais comuns são:

- *SELECT* - permite consultar qualquer coluna da tabela, visão ou sequência especificada;
- *INSERT* - permite incluir novas linhas na tabela especificada;
- *DELETE* - permite excluir linhas da tabela especificada;
- *UPDATE* - permite modificar os dados de qualquer coluna da tabela especificada;
- *ALTER* - permite modificar a estrutura da tabela especificada;
- *CREATE* - permite criar tabelas;
- *DROP* - permite excluir tabelas especificadas;
- *SHOW DATABASES* - permite exibir todos os bancos de dados;
- *ALL* - concede todos os privilégios descritos nessa lista de uma só vez.

Uma vez informados os privilégios do usuário, deverá ser indicada a lista de objetos ao qual o privilégio se aplica, sendo possível especificar três níveis:

- *.** - Privilégio global;
- *db.** - Qualquer tabela do banco de dados denominado de **db**;
- *db.tb* - Apenas a tabela **tb** do banco de dados **db**.

É importante destacar que para especificar apenas algumas colunas de uma determinada tabela, essas deverão ser listadas ao lado do privilégio (*priv (colunas)*).

Depois da lista de objetos, deverá ser indicada a lista de usuários, para os quais os privilégios se aplicam.

Vamos praticar o comando *GRANT*, concedendo sucessivamente diversas permissões ao usuário **josemar**, utilizando para tanto uma janela de linha de comando do sistema *MySQL* conectado como *root*. Para efeito de verificação das permissões concedidas, abrimos uma segunda janela de linha de comando do sistema *MySQL*, conectado como o usuário **josemar** (Figura 4).

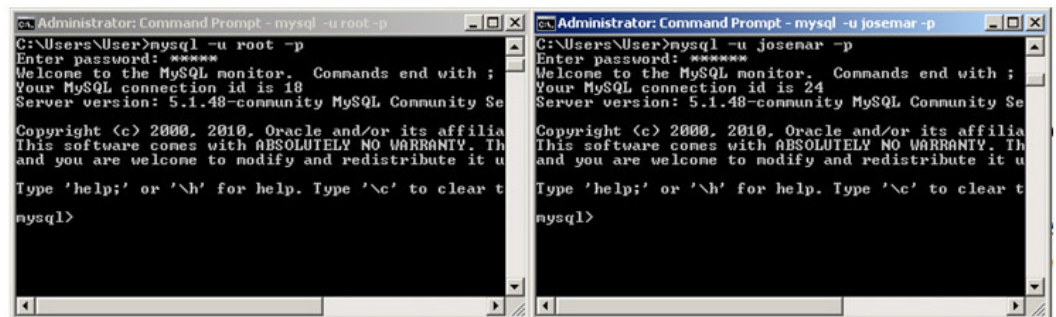


Figura 4 – Telas do ambiente *MySQL* para os usuários *root* e **josemar**
Fonte: *MySQL Server 5.1*

Inicialmente, usando a conta *root*, vamos conceder ao usuário **josemar** o direito de selecionar os dados da tabela **clientes** do nosso banco de dados **sistvendas** (Aula 13).

mysql> GRANT SELECT ON sistvendas.clientes
TO josemar;

A resposta do sistema *MySQL* ao comando é *QUERY OK*, que informa que o comando foi executado com sucesso, conforme ilustrado na Figura 5.

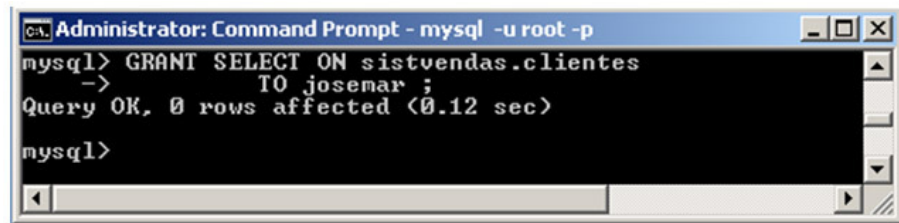


Figura 5 – Tela do MySQL após o comando GRANT SELECT
Fonte: MySQL Server 5.1

Nesse momento, é interessante verificar a permissão concedida ao usuário **josemar**. Para tanto, utilizando a janela do MySQL conectada com o usuário **josemar**, selecionamos todos os dados da tabela **clientes** através do comando SELECT, conforme é apresentado no quadro abaixo. Lembre-se de antes informar ao sistema que deseja trabalhar com o banco de dados **sistvendas** utilizando o comando USE (Aula 9).

```
mysql> SELECT *
FROM clientes;
```

A resposta do sistema ao comando anterior é ilustrada na Figura 6. Conforme pode ser visualizado na figura, o usuário **josemar** pode selecionar e visualizar os dados da tabela **clientes**. Entretanto, não é permitido a esse usuário visualizar as informações pertencentes a nenhuma outra tabela desse banco de dados (**produtos** e **compras**). Caso esse usuário tente selecionar os dados das tabelas **produtos** ou **compras**, terá como resposta do sistema a mensagem que o uso do comando SELECT nessas tabelas foi negado ao usuário **josemar**, conforme ilustrado na Figura 6.

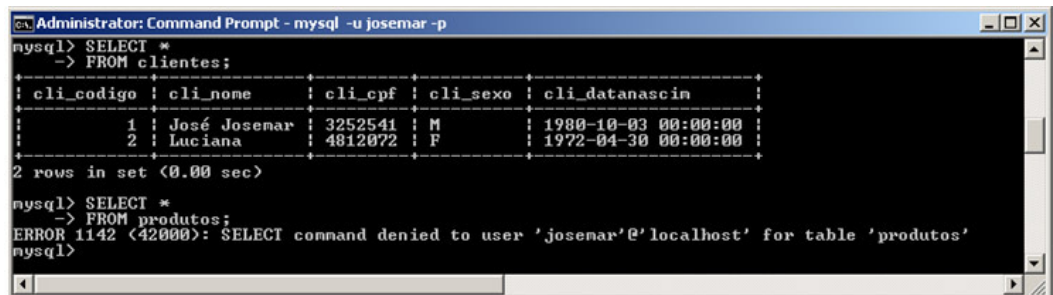


Figura 6 - Tela do MySQL após os comandos SELECT. Fonte: MySQL Server 5.1

Para que o usuário **josemar** tenha acesso aos dados de todas as tabelas pertencentes ao banco de dados **sistvendas**, devemos, conectado como **root**, executar o seguinte comando.

```
mysql> GRANT SELECT ON sistvendas.*
TO josemar;
```

Até o momento, o usuário **josemar** só tem permissão de visualizar os dados nas tabelas pertencentes ao banco de dados **sistvendas**, mas não tem permissão de incluir, atualizar ou excluir nenhum registro nessas tabelas. Para que esse usuário tenha permissão para realizar tais tarefas, é necessário que o usuário **root** lhe conceda as permissões através do seguinte comando.

```
mysql> GRANT INSERT, UPDATE, DELETE ON sistvendas.*
TO josemar;
```

É importante destacar que para especificar apenas algumas colunas de uma determinada tabela, essas deverão ser listadas ao lado do privilégio. Para maior esclarecimento, analise o seguinte exemplo.

```
mysql> GRANT SELECT (cli_nome) ON locadora.clientes
TO josemar;
```

Nesse exemplo, é concedido ao usuário **josemar** a permissão de visualizar apenas os nomes dos clientes do banco de dados **locadora**.

Vale salientar que não são apenas as tabelas que podem ser objeto de permissões, essas podem ser estendidas para

outros objetos do banco de dados como as visões e as *stored procedures* vistas nas aulas anteriores, conforme mostra o exemplo abaixo.

```
mysql> GRANT SELECT ON pagamentos.funcionario  
TO josemar;
```

Nesse exemplo, **funcionario** é uma visão pertencente ao banco de dados **pagamentos** (Aula 15).

O usuário **root** também pode conceder a qualquer usuário o direito de repassar para um terceiro o privilégio concedido a ele. Para isso, basta acrescentar a cláusula **WITH GRANT OPTION** no final de um comando **GRANT** qualquer. Veja o seguinte exemplo:

```
mysql> GRANT ALL ON cineonline.*  
TO josemar  
  
WITH GRANT OPTION;
```

Com esse comando, o usuário **josemar** não só possui todos os privilégios em todas as tabelas do banco de dados **cineonline**, como pode conceder, a outro usuário, qualquer um dos seus privilégios nas tabelas do banco de dados **cineonline**.

É importante esclarecer que mesmo que o usuário possua várias permissões, ele só poderá conceder, a outros usuários, aqueles privilégios que lhe forem atribuídos com a cláusula **WITH GRANT OPTION**.

Suponha que o usuário **josemar**, por alguma razão, não deve mais ter acesso ao banco de dados **locadora**. O que se deve fazer? Excluir esse usuário e novamente adicioná-lo, concedendo novamente todas as suas permissões com exceção daquelas referentes ao banco de dados **locadora**? Embora isso seja possível, não é a maneira mais prática.

De maneira semelhante ao que foi utilizado para conceder privilégios a usuários, existe um comando para remover ou excluir privilégios concedidos. O comando **REVOKE** permite ao administrador de sistemas (usuário **root**) revogar permissões concedidas.

A sintaxe do comando **REVOKE** é semelhante à sintaxe do comando **GRANT**, entretanto, ao invés de utilizar a palavra **GRANT**, utiliza-se a palavra **REVOKE**, e ao invés de **TO**, utiliza-se **FROM**, conforme ilustrado no quadro abaixo.

```
mysql> REVOKE lista_de_privilegios ON lista_do_objeto  
FROM lista-de-usuarios;
```

Para revogar toda e qualquer permissão que o usuário **josemar** tenha sobre as tabelas do banco de dados **locadora**, o usuário **root** deve executar o seguinte comando.

```
mysql> REVOKE ALL ON locadora.*  
FROM josemar;
```

Para revogar um privilégio com a cláusula **WITH GRANT OPTION** no qual o usuário pode conceder seus direitos a outros usuários, utiliza-se comando semelhante ao exemplo a seguir.

```
mysql> REVOKE GRANT OPTION ON cineonline.*  
FROM josemar;
```

Nesse exemplo, o usuário **josemar** ainda poderá utilizar todos os comandos nas tabelas do banco de dados **cineonline**, mas não poderá conceder a mais ninguém a permissão de utilizar os comandos **SQL**.

Suponha que o usuário **Josemar** passou a permissão de atualizar dados de **cineonline** (**UPDATE**) para outro usuário chamado **Pedro**. Quando a permissão de atualizar (**UPDATE**) de **Josemar** é revogada (comando abaixo), também é revogada a permissão de **Pedro** de atualizar tabelas.

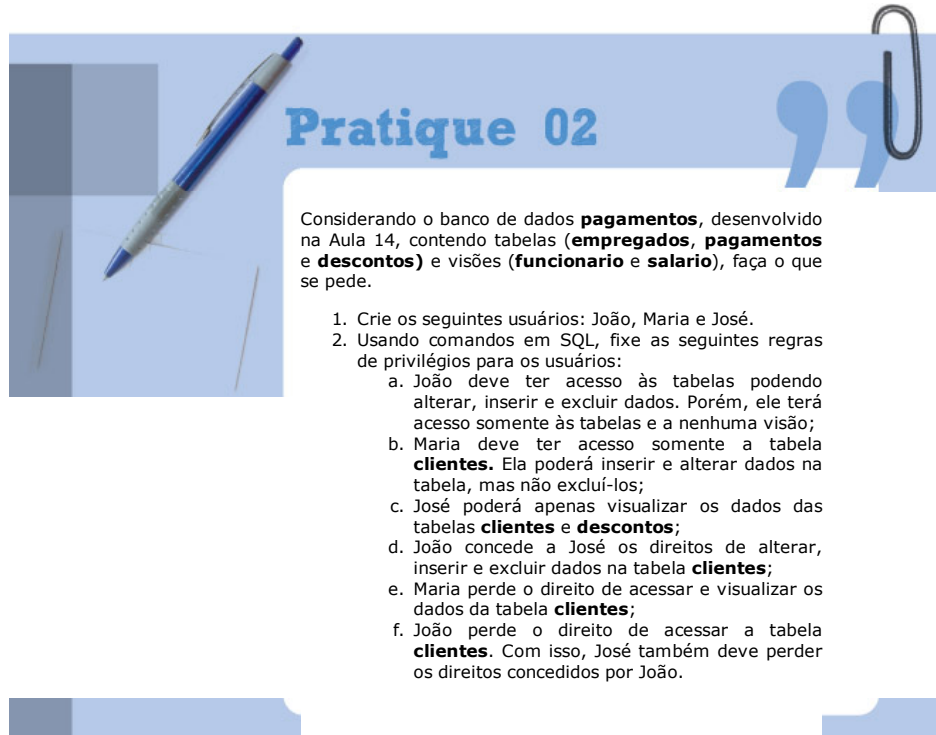
```
mysql> REVOKE UPDATE ON cineonline.*  
FROM josemar;
```

É importante destacar que é possível unir os comandos **CREATE USER** e **GRANT** em um só comando, criando um usuário e lhe concedendo as permissões devidas. Para tanto, deve-se utilizar o comando **GRANT** acrescido da cláusula **IDENTIFIED BY**, conforme o exemplo a seguir.

```
mysql> GRANT SELECT ON sistvendas.*  
TO Jose
```

IDENTIFIED BY '22222';

No comando acima, criamos um usuário chamado Jose no nosso sistema de banco de dados e lhe concedemos a permissão de visualizar todos os dados contidos no banco de dados **sistvendas**.



Pratique 02

Considerando o banco de dados **pagamentos**, desenvolvido na Aula 14, contendo tabelas (**empregados**, **pagamentos** e **descontos**) e visões (**funcionario** e **salario**), faça o que se pede.

1. Crie os seguintes usuários: João, Maria e José.
2. Usando comandos em SQL, fixe as seguintes regras de privilégios para os usuários:
 - a. João deve ter acesso às tabelas podendo alterar, inserir e excluir dados. Porém, ele terá acesso somente às tabelas e a nenhuma visão;
 - b. Maria deve ter acesso somente a tabela **clientes**. Ela poderá inserir e alterar dados na tabela, mas não excluí-los;
 - c. José poderá apenas visualizar os dados das tabelas **clientes** e **descontos**;
 - d. João concede a José os direitos de alterar, inserir e excluir dados na tabela **clientes**;
 - e. Maria perde o direito de acessar e visualizar os dados da tabela **clientes**;
 - f. João perde o direito de acessar a tabela **clientes**. Com isso, José também deve perder os direitos concedidos por João.

Encerramos por aqui nossa aula sobre segurança de sistemas e dados na linguagem SQL. Na próxima aula, aprenderemos como integrar uma aplicação desenvolvida em Java com o seu banco de dados **MySQL**. Faça a autoavaliação com atenção e veja se precisa parar e refletir mais um pouco sobre como modelar, criar e manipular dados utilizando a linguagem SQL. É uma boa prática escrever no seu caderno todos os comandos SQL (e respectivas funções) que você estudou para não esquecer. Bons estudos e boa sorte!

Resumo

Nesta aula, estudamos segurança de sistemas e de banco de dados. Em segurança de sistemas, vimos que o comando **CREATE USER** cria uma conta de usuário no **MySQL**. Aprendemos como ter acesso ao **MySQL** a partir de uma conta qualquer. Em segurança de banco de dados, estudamos o comando **GRANT**, que permite controlar exatamente o que os usuários podem fazer com os objetos (tabelas, visões, funções e *stored procedures*) baseados nos privilégios atribuídos a cada usuário. Estudamos como utilizar o comando **REVOKE** para revogar as permissões de um usuário.



Evolucao

Considere o banco de dados **CursoX** criado na autoavaliação da Aula 9 cuja estrutura de tabelas é mostrada abaixo:

TABELA alunos

ATRIBUTO	TIPO	DESCRIÇÃO
aluno_cod	Número inteiro	Código do aluno
aluno_nome	Alfanumérico	Nome do aluno

aluno_endereco Alfanumérico Endereço do aluno

aluno_cidade Alfanumérico Cidade do aluno

TABELA disciplina

ATRIBUTO	TIPO	DESCRIÇÃO
dis_cod	Número inteiro	Código da disciplina
dis_nome	Alfanumérico	Nome da disciplina
dis_carga	Número inteiro	Carga horária da disciplina
dis_professor	Alfanumérico	Professor da disciplina

TABELA professores

ATRIBUTO	TIPO	DESCRIÇÃO
prof_cod	Número inteiro	Código do professor
prof_nome	Alfanumérico	Nome do professor
prof_endereco	Alfanumérico	Endereço do professor
prof_cidade	Alfanumérico	Cidade do professor

1. Considere os comandos a seguir e as tabelas pertencentes ao banco de dados **CursoX**, para responder às próximas questões:

- CREATE USER prof IDENTIFIED BY = '111111';
- CREATE USER coord IDENTIFIED BY = '222222';
- CREATE USER maria IDENTIFIED BY = '333333';
- CREATE USER marcos IDENTIFIED BY = '444444';
- GRANT SELECT ON Cursox.alunos TO marcos;
- GRANT ALL ON Cursox.* TO coord WITH GRANT OPTION;
- GRANT SELECT, UPDATE (aluno_endereco, aluno_cidade) ON Cursox.alunos TO Maria;
- GRANT SELECT, UPDATE, INSERT ON Cursox.professores TO Maria;
- REVOKE SELECT ON Cursox.alunos TO marcos;
- REVOKE INSERT ON Cursox.professores TO maria;

Considerando a execução dos comandos acima, responda as questões propostas.

- Quais os nomes das pessoas que podem se conectar ao banco de dados **CursoX**? O que cada uma delas está autorizada a fazer nesse banco de dados? Explique.
- O que o usuário **maria** pode fazer?
- O usuário **coord** poderá conceder a outro usuário permissão para atualizar a tabela **professores**? Explique.
- O usuário **marcos** poderá cadastrar um novo professor? Explique.
- O usuário **maria** poderá cadastrar um novo aluno? Explique.

Referencias

BEIGHLEY, L. **Use a cabeça SQL**. Rio de Janeiro: Editora AltaBooks, 2008.

MySQL 5.1 Reference Manual. Disponível em: <<http://dev.mysql.com/doc/refman/5.1/en/>>. Acesso em: 24 set. 2010.

WIKIPÉDIA. **SQL**. Disponível em: <<http://pt.wikipedia.org/wiki/SQL>>. Acesso em: 24 set. 2010.

