

Feita para Desenvolvedores de Software e DBAs



Edição 43 :: Ano 4 : R\$11,90

DevMedia

Desenvolvimento

Acesse seu banco de dados através
do NetBeans

Coluna Desafio SQL

Controle de empréstimo de Cds

Banco de Dados

Veja como migrar seus dados com sucesso

Busca textual com Full-text search

Veja neste mini curso como realizar pesquisas textuais avançadas, ordenadas por relevância, utilizando o SQL Server 2005

Brinde!

Mini curso sobre Crystal Reports.

Compre esta edição e ganhe **10 vídeo-aulas** sobre Crystal Reports:

- Introdução ao Crystal Reports;
- Construindo relatórios máster/detail
- Trabalhando com fórmulas
- Relatórios agrupados
- Relatórios zebreados e formatações
- Agrupando relatórios por condição
- Passando parâmetros para relatórios
- Relatórios Top N



SQL Server

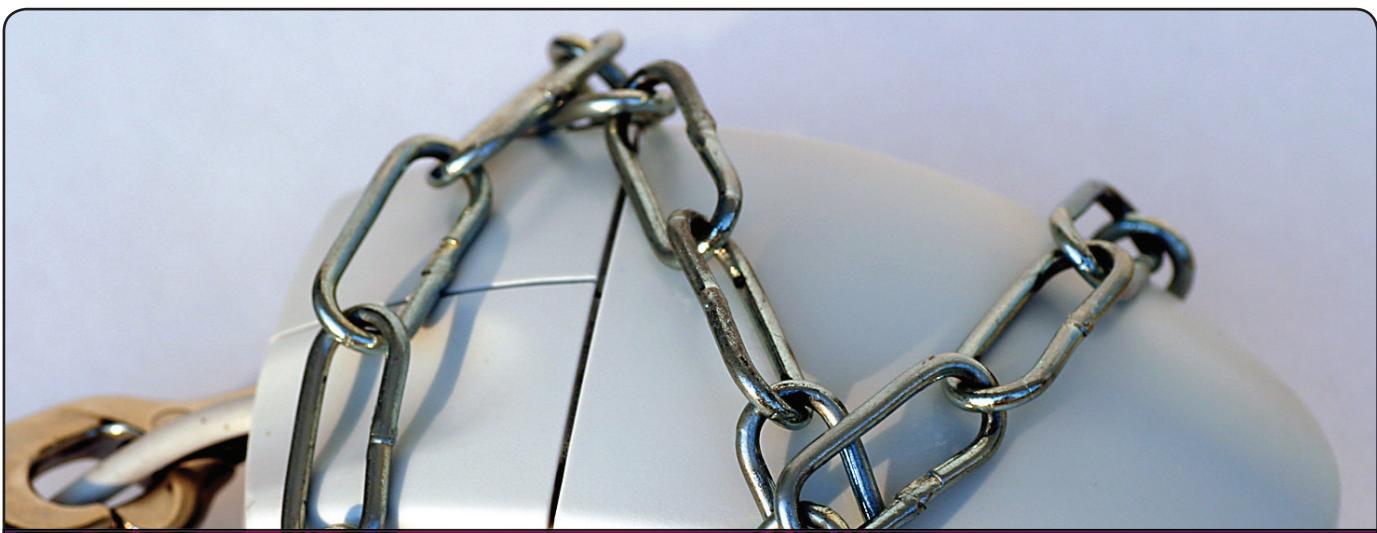
Implemente uma solução de auditoria
utilizando triggers

Modelagem de Dados

Elabore modelo de dados utilizando a
System Architect

Banco OO

Construa aplicações orientadas a objetos
acessando o Jade



Gerenciamento de usuários, privilégios e controle de acessos em MySQL

O MySQL apresenta uma estratégia para controle de acessos bastante flexível e robusta que permite controlar as ações de cada usuário em diversos níveis dentro do sistema. Portanto, é possível controlar quais os comandos cada usuário pode executar, bem como em qual nível de dados: em apenas um banco de dados, tabela ou coluna. Além disto, pode-se fazer um controle apurado levando-se em conta o host (máquina cliente) de onde a conexão com o servidor MySQL foi estabelecida. Desta forma, é factível que a partir do host local o usuário possua privilégios distintos dos que estão disponíveis quando o acesso é feito a partir de um host remoto.

Assim, torna-se viável uma definição de privilégios de forma a conceder apenas os acessos necessários a cada aplicação que utiliza o MySQL, evitando que sejam feitos acessos indevidos aos dados. Ao longo deste artigo serão apresentados todos os aspectos relacionados à manipulação de

usuários, bem como determinadas dicas para resolução de problemas comuns envolvendo o controle de privilégios. Inicialmente será abordada a forma de criação de um novo usuário, e a seguir os tipos de privilégios definidos no MySQL, bem como os comandos para a concessão e remoção dos mesmos. Finalmente, serão apresentados os comandos para visualização e exclusão de usuários, bem como os métodos para exibição de seus privilégios.

Em última análise, será dada uma visão geral de como são armazenados os dados dos usuários e como o MySQL realiza a autenticação e o controle de acesso dos seus clientes.

Vale ressaltar que este artigo tem como base o MySQL 5.0, portanto alguns comandos e comportamentos aqui apresentados podem não se aplicar às versões anteriores deste SGBD.



Éber Duarte

(eber@eacsoftware.com.br)

é bacharel em Ciência da Computação, pós-graduado em Engenharia Elétrica e MySQL Professional Certified. Trabalha há 5 anos na EAC Software (BH/MG) como Analista e desenvolvedor de sistemas, atuando especialmente no desenvolvimento de sistemas Web. Atualmente, também é consultor e instrutor do banco de dados MySQL.

Criação de usuários

No MySQL, um usuário é definido a partir de duas informações, seu nome e o *host* de onde ele poderá se conectar ao servidor MySQL. Neste caso, toda manipulação de usuários dentro do SGBD levará em consideração estas duas informações.

O comando para a criação de usuários é o CREATE USER, que é ilustrado na **Listagem 1**.

O comando apresentado na **Listagem 1** cria um usuário chamado sqlmagazine, que pode se conectar apenas do *host* local, isto é, o cliente deve estar instalado na mesma máquina em que se encontra o servidor MySQL. O comando ainda atribui uma senha, senhasecreta, para este usuário. Vale ressaltar que a senha é opcional, mas é recomendado sempre informar uma senha para os usuários, de forma a eliminar acessos não autorizados.

Um usuário que tenha sido criado com o comando CREATE USER não possui nenhum privilégio. Ele pode apenas se conectar e executar alguns comandos básicos do sistema, tais como exibir a data e hora do sistema.

O usuário ilustrado pelo exemplo anterior está limitado a se conectar apenas do *host* local, mas em situações práticas, é usual que as pessoas possam se conectar a partir de vários *hosts*. Por exemplo, em uma empresa é bastante conveniente que os funcionários tenham acesso ao MySQL a partir de qualquer *host* dentro da sua rede ou domínio, mas não fora deste universo. O MySQL possui recurso para a definição de usuários com acessos a diversos *hosts*, para isto deve-se utilizar o caractere '%' (percentual) para determinar uma faixa de *hosts*.

Supondo uma empresa onde todas as máquinas estão no domínio exemplo.com.br e que utilizam endereços IP inválidos no formato 192.168.1.xxx, a **Listagem 2** apresenta dois comandos para a criação de usuários que possam se conectar a partir de qualquer *host* dentro desta empresa.

Os usuários *remoto1* e *remoto2* são equivalentes, e podem acessar o MySQL a partir de qualquer *host* dentro do domínio desta empresa fictícia. No primeiro exemplo, foi utilizada uma faixa de DNS para controlar os *hosts* do qual ele pode

acessar o sistema, enquanto no segundo foi utilizada uma faixa de IPs. A escolha da abordagem fica a cargo do administrador do sistema, que deverá considerar os aspectos de segurança e facilidades de manutenção existentes em ambos os casos. O uso do nome do *host* (DNS) é menos seguro que o uso do endereço IP, uma vez que o cliente pode forjar o seu DNS para tentar invadir o sistema. Por outro lado, a utilização de endereços IP pode ser trabalhosa para o administrador uma vez que havendo mudanças nos endereços da rede, os usuários do MySQL deverão ser reconfigurados para terem acesso a partir destes novos endereços.

Vale ressaltar que o '%' pode ser utilizado sozinho na definição de um *host*, e neste caso, o usuário poderá acessar o sistema a partir de qualquer local que tenha acesso à porta TCP/IP utilizada pelo MySQL, lembrando que configurações de *firewall* podem inibir o acesso remoto, mesmo tendo o acesso com '%'. Não se recomenda criar usuários que possam acessar o sistema de qualquer *host*, pois isto pode introduzir vulnerabilidades de segurança ao sistema desnecessariamente.

Caso exista a necessidade de acesso remoto, recomenda-se criar dois usuários com o mesmo nome, mas com privilégios diferentes. Ou seja, o usuário com acesso remoto teria privilégios restritos em relação ao seu correspondente que possui acesso apenas do *host* local. Este metadado elimina problemas de segurança sem restringir a criação de usuários com

acesso a partir de qualquer *host*.

Uma observação importante é que o MySQL fornece um usuário chamado *root* que possui todos os privilégios e acessa o servidor apenas do *host* local. Este é o administrador do sistema, e pode executar quaisquer comandos dentro do MySQL. Esse usuário é criado automaticamente durante a instalação do MySQL. Além disso, há um outro usuário anônimo também com acesso apenas do *host* local, mas que possui privilégios apenas no banco de dados *test*. Para criar um usuário anônimo, basta utilizar um nome de usuário vazio, conforme ilustra a **Listagem 3**. Este usuário anônimo é criado pelo processo de instalação do MySQL, e serve apenas para permitir o acesso de qualquer pessoa ao servidor.

Para estabelecer uma conexão com o usuário anônimo, basta informar um nome de usuário que não seja um usuário cadastrado no sistema, como mostra a **Listagem 4**. Neste caso, deve-se informar a senha do usuário anônimo, lembrando que a instalação padrão cria uma senha vazia para este usuário.

A conexão do exemplo anterior será feita considerando o anônimo, pois o usuário *teste* não existe como um usuário cadastrado dentro do sistema. É possível perceber que a senha utilizada no exemplo foi a senha do anônimo, por isto a conexão foi estabelecida com sucesso. Portanto, os privilégios utilizados para esta conexão serão os do anônimo, uma vez que a conexão foi estabelecida para este usuário.

Listagem 1. Criação do usuário sqlmagazine com acesso do host local

```
eber@linux:~/sqlMagazine> mysql -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.20-standard

mysql> CREATE USER sqlmagazine@localhost IDENTIFIED BY "senhasecreta";
Query OK, 0 rows affected (0.04 sec)
```

Listagem 2. Definição de dois usuários com acesso remoto ao servidor MySQL

```
mysql> CREATE USER remoto1@'%.exemplo.com.br' IDENTIFIED BY "senharemotol";
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE USER remoto2@'192.168.1.%' IDENTIFIED BY "senharemoto2";
Query OK, 0 rows affected (0.00 sec)
```

Listagem 3. Criando um usuário anônimo no MySQL.

```
mysql> CREATE USER ''@localhost IDENTIFIED BY "senhaanonimo";
Query OK, 0 rows affected (0.00 sec)
```

Listagem 4. Estabelecendo uma conexão com um usuário anônimo.

```
eber@linux:~/sqlMagazine> mysql -u teste -psenhaanonimo
```

Vale lembrar que o nome teste utilizado foi apenas uma possibilidade, qualquer outro nome pode ser empregado para estabelecer a conexão.

Visualização dos usuários e seus privilégios

Para visualizar todos os usuários cadastrados no MySQL será necessário listar o conteúdo da tabela *user*, que armazena os usuários do sistema. A **Listagem 5** ilustra esta situação.

A tabela fornece todos os usuários ca-

dastrados no sistema, inclusive aqueles criados nos exemplos anteriores.

Para visualizar os privilégios para um determinado usuário, deve-se utilizar o comando SHOW GRANTS, conforme ilustra a **Listagem 6**.

A listagem dos privilégios dos usuários recém criados fornece apenas o privilégio *USAGE*, que permite apenas a conexão do usuário ao servidor, como dito anteriormente. A seguir são descritos os mecanismos para atribuição de privilégios.

Listagem 5. Exibindo os usuários cadastrados no MySQL.

```
mysql> SELECT user, host FROM mysql.user;
+-----+-----+
| user | host   |
+-----+-----+
| remoto1 | % exemplo.com.br |
| remoto2 | 192.168.1.% |
| root   | localhost |
| sqlmagazine | localhost |
+-----+-----+
5 rows in set (0.00 sec)
```

Listagem 6. Exibição dos privilégios dos usuários *sqlmagazine*, *remoto2* e *root*.

```
mysql> SHOW GRANTS FOR sqlmagazine@localhost;
+-----+
| Grants for sqlmagazine@localhost |
+-----+
| GRANT USAGE ON *.* TO 'sqlmagazine'@'localhost' IDENTIFIED BY PASSWORD
'*0D8868F3E03112C0A389E531202CE9F005C2AOA' |
+-----+
1 row in set (0.00 sec)

mysql> SHOW GRANTS FOR remoto2@'192.168.1.%';
+-----+
| Grants for remoto2@192.168.1.% |
+-----+
| GRANT USAGE ON *.* TO 'remoto2'@'192.168.1.%' IDENTIFIED BY PASSWORD
'*DA7A1EE7B3EA839DB1DCA284545B9D73312BBB2B' |
+-----+
1 row in set (0.00 sec)

mysql> show grants for root@localhost;
+-----+
| Grants for root@localhost |
+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
+-----+
1 row in set (0.00 sec)
```

Listagem 7. Estrutura básica do comando GRANT.

```
GRANT priv [(colunas)] [, priv [(colunas)]]
ON (*.* | db.* | db.tabela)
TO usuario [IDENTIFIED BY 'senha'] [, usuario [IDENTIFIED BY 'senha']]
[WITH [GRANT OPTION |
MAX_QUERIES_PER_HOUR contador |
MAX_UPDATES_PER_HOUR contador |
MAX_CONNECTIONS_PER_HOUR contador]]
```

Listagem 8. Exemplos de concessão de privilégios com o comando GRANT.

```
Todos os privilégios no banco "teste"
mysql> GRANT ALL ON teste.* TO sqlmagazine@localhost;

INSERT e SELECT na tabela "pessoas" do banco "teste"
mysql> GRANT INSERT, SELECT ON teste.pessoas TO remoto1@'%exemplo.com.br';

Apenas SELECT na coluna "nome" da tabela "pessoas" do banco "teste"
mysql> GRANT SELECT(nome) ON teste.pessoas TO remoto2@'192.168.1.%';

Administrador do sistema, com privilégio de dar privilégios
mysql> GRANT ALL ON *.* TO admin@localhost WITH GRANT OPTION;

Permite se conectar 2 vezes por hora e executar apenas 50 consultas por hora
mysql> GRANT ALL ON *.* TO limitado@localhost WITH MAX_QUERIES_PER_HOUR
      50 MAX_CONNECTIONS_PER_HOUR 2;
```

Atribuição de privilégios com o comando GRANT

Nas versões anteriores do MySQL, o comando GRANT era utilizado para criar usuários e dar privilégios, uma vez que não havia o comando CREATE USER, introduzido apenas na versão 5.0. Desta forma, o comportamento do comando se dá da seguinte maneira: se for executado um GRANT para um usuário existente, os privilégios serão adicionados aos privilégios já existentes. Caso contrário, o usuário será criado e os privilégios serão concedidos a ele.

A estrutura básica do comando GRANT é ilustrada na **Listagem 7**.

O comando é composto de quatro partes, que permitem especificar os privilégios que serão dados ao usuário, o nível aos quais os mesmos se aplicam, quem receberá os privilégios e os limites de recursos do sistema impostos ao usuário. No exemplo da **Listagem 7**, tudo que se encontra entre "[]" são cláusulas opcionais do comando, podendo assim ser omitidas sem incorrer em erros.

A **Tabela 1** fornece todos os privilégios disponíveis no MySQL e uma descrição de suas funcionalidades.

A cláusula ON do comando GRANT define o nível ao qual o privilégio se aplica, sendo que podem existir quatro níveis de privilégios: global, por banco, por tabela e por coluna. Estes privilégios são concedidos respectivamente com **.**, *nome_do_banco.**, *nome_do_banco.nome_da_tabela* e *nome_da_tabela* acrescido do nome das colunas na frente do privilégio que serão dados a elas.

Uma vez definidos os privilégios e os níveis aos quais eles se aplicam, basta informar para quem serão dados estes acessos. Esta tarefa é realizada com a utilização da cláusula TO, onde serão listados os usuários no formato *usuário@host*. A **Listagem 8** fornece exemplos de utilização do comando GRANT.

Ao observar os exemplos anteriores, observa-se que o comando GRANT permite definir uma grande variedade de privilégios, tornando apurado o controle de acessos dos clientes ao servidor MySQL. É preciso ressaltar que há a possibilidade de criação de dois usuários com o mesmo nome, mas com privilé-

gios diferentes e que possam acessar o sistema a partir de *hosts* diferentes. Isto é possível devido ao fato de um usuário ser sempre identificado por um par de valores (nome do usuário e *host* de onde pode se conectar).

Remoção de privilégios

Em um sistema real, é comum que as atribuições de um usuário dentro do sistema, bem como os seus requisitos, variem ao longo do tempo. Por isto, um usuário previamente criado pode ter que receber novos acessos ou até mesmo ter a sua ação limitada dentro do sistema. Portanto, para remover privilégios de um usuário, será utilizado o comando REVOKE, cuja sintaxe básica é apresentada na **Listagem 9**.

A estrutura do comando é similar ao GRANT, ou seja, é preciso especificar os privilégios que se deseja retirar, o nível e o usuário do qual os privilégios serão removidos. Vale ressaltar que a cláusula ON do comando REVOKE deve coincidir com a cláusula ON do comando GRANT que foi executado. Caso contrário, nenhum privilégio será removido. Por exemplo, se o usuário tem um privilégio no nível banco.*, este deve ser removido utilizando na cláusula ON do comando REVOKE o mesmo nível usado durante a atribuição da permissão. Caso seja utilizado um nível diferente, por exemplo **, nenhum privilégio será removido, pois o usuário não apresenta privilégios neste nível, mas sim no nível de banco.

Vale ressaltar que o REVOKE ALL remove todos os privilégios, exceto o GRANT OPTION, que deverá ser removido explicitamente. A **Listagem 10** fornece exemplos de uso do comando REVOKE.

Um detalhe importante em relação ao comando REVOKE é que este remove apenas os privilégios do usuário, mas este ainda poderá se conectar ao MySQL. Assim, para remover usuários deve ser utilizado um outro comando que será apresentado na seção seguinte.

Remoção de usuários

Para remover um usuário do sistema, bem como todos os privilégios concedidos a ele, é necessário utilizar o comando DROP USER. A **Listagem 11** fornece uma visão geral de como utilizar este comando.

Privilégios	Descrição
ALL [PRIVILEGES]	Todos os privilégios exceto GRANT OPTION
ALTER	Permite executar ALTER TABLE
ALTER ROUTINE	Alteração e exclusão de rotinas
CREATE	Permite executar CREATE TABLE
CREATE ROUTINE	Criação de rotinas
CREATE TEMPORARY TABLES	Permite executar CREATE TEMPORARY TABLE
CREATE USER	Permite executar CREATE USER, DROP USER, RENAME USER, REVOKE ALL PRIVILEGES
CREATE VIEW	Permite criar VIEWS
DELETE	Permite executar DELETE
DROP	Permite executar DROP TABLE
EXECUTE	Permite executar stored procedures
FILE	Permite executar SELECT ... INTO OUTFILE e LOAD DATA INFILE
INDEX	Permite executar CREATE INDEX e DROP INDEX
INSERT	Permite executar INSERT
LOCK TABLES	Permite executar LOCK TABLES em tabelas que você tenha o privilégio SELECT
PROCESS	Permite executar SHOW FULL PROCESSLIST
REFERENCES	Ainda não está implementado, e não existe qualquer informação sobre sua função no manual do MySQL
RELOAD	Permite executar FLUSH
REPLICATION CLIENT	Permite ao usuário obter informação de onde o Master ou Slave estão
REPLICATION SLAVE	Necessário para a replicação slave (leitura dos eventos do log binário do master)
SELECT	Permite executar SELECT
SHOW DATABASES	Exibe todos os bancos de dados
SHOW VIEW	Permite executar SHOW CREATE VIEW
SHUTDOWN	Permite executar mysqladmin shutdown
SUPER	Permite executar CHANGE MASTER, KILL, PURGE MASTER LOGS e SET GLOBAL. Permite conectar-se ao servidor uma vez mesmo que o max_connections tenha sido atingido
UPDATE	Permite executar UPDATE
USAGE	Sinônimo para "no privileges"
GRANT OPTION	Permite ao usuário repassar os seus privilégios

Tabela 1. Listagem de todos os privilégios do MySQL e as suas funcionalidades.

Listagem 9. Estrutura básica do comando REVOKE.

```
REVOKE priv [(colunas)] [, priv [(colunas)]]  
ON (*.* | db.* | db.tabela)  
FROM usuario
```

Listagem 10. Exemplos de remoção de privilégios com o comando REVOKE.

```
Remove todos os privilégios no banco teste  
mysql> REVOKE ALL ON teste.* FROM sqlmagazine@localhost;
```

```
Remove apenas os privilégios INSERT e SELECT na tabela de pessoas do banco teste  
mysql> REVOKE INSERT, SELECT ON teste.pessoas FROM remot01@'%exemplo.com.br';
```

```
Remove todos os privilégios inclusive o GRANT OPTION  
mysql> REVOKE ALL, GRANT OPTION ON *.* FROM admin@localhost;
```

Listagem 11. Remoção de usuário com o comando DROP USER.

```
mysql> SHOW GRANTS FOR remot01@'%exemplo.com.br';  
+-----+  
| Grants for remot01@%exemplo.com.br |  
+-----+  
| GRANT USAGE ON *.* TO 'remot01'@'%exemplo.com.br'  
| GRANT SELECT, INSERT ON `teste`.`pessoas` TO 'remot01'@'%exemplo.com.br'  
+-----+  
2 rows in set (0.00 sec)  
mysql> DROP USER remot01@'%exemplo.com.br';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> SHOW GRANTS FOR remot01@'%exemplo.com.br';  
ERROR 1141 (42000): There is no such grant defined for user 'remot01' on host '%exemplo.com.br'
```

Como ilustra o exemplo da **Listagem 11**, o usuário *remoto1* foi removido do sistema com todos os seus privilégios, e não mais poderá acessar o servidor MySQL.

Armazenamento das informações dos usuários

Ao instalar o MySQL, é criado um banco de dados chamado *mysql* onde estão armazenadas as tabelas contendo os dados de todos os usuários do sistema, dentre outras informações. O mecanismo de privilégios faz uso de seis tabelas para armazenar os diversos níveis de acessos: *user*, *db*, *host*, *procs_priv*, *tables_priv* e *columns_priv*. A primeira tabela é a *user*, que contém todos os usuários do sistema, bem como os seus privilégios globais. A tabela *db* contém os privilégios do usuário no nível de banco. A tabela *host* não é mais utilizada pelos comandos GRANT e REVOKE, e era utilizada nas versões anteriores para realizar o controle de privilégios por *host*. As tabelas *procs_priv*, *tables_priv* e *columns_priv* armazenam respectivamente os privilégios relacionados a *Stored Procedures*, acessos por tabela e, finalmente, as restrições por colunas.

Portanto, o acesso ao banco de dados *mysql* deve ser restrito. Caso contrário, o usuário pode burlar o controle de acesso através da manipulação direta dos dados armazenados neste banco.

Autenticação e controle de acessos

Ao iniciar a execução do MySQL, este carrega para a memória as informações das tabelas de privilégios, facilitando

assim o acesso aos dados e reduzindo o tempo para realizar a autenticação e o controle de acessos ao sistema. Os dados são ordenados em memória colocando os *hosts* mais específicos (sem o símbolo %) na frente dos *hosts* menos específicos (com o símbolo %), e havendo dois usuários com *hosts* iguais, aqueles mais específicos vêm à frente dos anônimos. A autenticação do usuário consiste em pesquisar nesta lista ordenada se há um registro que coincida o nome de usuário, *host* e senha informados pelo cliente.

Uma vez conectado, para cada comando executado, o sistema verificará as tabelas *user*, *db* ou *procs_priv*, *tables_priv* e *columns_priv* para determinar se o usuário possui permissão para executar o comando em questão. Vale ressaltar que a busca é feita na ordem em que as tabelas foram apresentadas, isto é, o MySQL pesquisará primeiro no nível global, e apenas se não encontrar o acesso seguirá para o próximo nível e assim sucessivamente. Caso não encontre o registro em nenhum dos níveis, emitirá a mensagem de acesso negado para o comando. Isto significa dizer que um usuário contendo privilégios no nível global e no nível de banco, os últimos nunca serão testados, pois ao

encontrar o privilégio no nível global o MySQL terminará a busca.

Para ilustrar este cenário será considerado o usuário *teste@localhost* contendo os privilégios exibidos na Listagem 12.

Portanto, o usuário pode executar SELECT no nível do banco teste (teste.*), e há uma restrição para que ele acesse apenas a coluna nome da tabela de pessoas. Neste caso, mesmo havendo esta restrição por coluna, este usuário terá acesso a toda a tabela de pessoas, pois como há esta liberação no nível global, o nível específico não será avaliado, conforme descrito anteriormente.

Conclusão

O MySQL apresenta um mecanismo de controle de acessos bastante adequado para a implementação de controles de acessos sofisticados e consistentes. Apesar disto, a manipulação destes privilégios pode ser trabalhosa caso exista no sistema um perfil variado de usuários, pois não há um conceito de papéis (roles) neste SGBD.

Para facilitar a administração dos privilégios, a ferramenta MySQL Administrator, que pode ser obtida no endereço www.mysql.com/downloads, pode ser utilizada. ●

Listagem 12. Usuário com privilégio global e por coluna

```
mysql> SHOW GRANTS FOR teste@localhost;
+-----+
| Grants for remot01@%exemplo.com.br |
+-----+
| GRANT SELECT ON teste.* TO 'teste'@'localhost'; |
| GRANT SELECT(nome) ON `teste`.`pessoas` TO 'teste'@'localhost'; |
+-----+
2 rows in set (0.00 sec)
```



Assinaturas Online
www.javamagazine.com.br