SEGURANÇA E CONTROLE DE ACESSO - TRANSACTIONS

ANTONIO DAVID VINISKI
antonio.david@pucpr.br
PUCPR

SEGURANÇA E CONTROLE DE ACESSO

SISTEMA DE PRIVILÉGIOS MYSQL

- A função principal do sistema de privilégios do MySQL é autenticar um usuário que se conecta a partir de um determinado host e associar esse usuário a privilégios em um banco de dados como **SELECT**, **INSERT**, **UPDATE** e **DELETE**.
- Quando você se conecta a um servidor MySQL, sua identidade é determinada pelo host do qual você se conecta e pelo nome de usuário especificado.
  - Quando você emite solicitações após a conexão, o sistema concede privilégios de acordo com sua identidade e o que você quer fazer.

MECANISMOS DE CONTROLE DE ACESSO

- Os mecanismos de controle de acesso do MYSQL permitem:
  - Limitar o acesso de um usuário a apenas um banco, tabela ou coluna.
  - Controlar o acesso de acordo com o host a partir de onde está sendo feita a conexão com o servidor.
  - Conceder privilégios diferentes para cada host de onde o usuário possa estabelecer a conexão.
- Assim, é possível que determinados comandos possam ser executados somente quando o usuário estiver em um host específico, por exemplo o mesmo host do servidor MySQL (localhost).

COMO FUNCIONA?

- O controle de acesso MySQL envolve duas etapas:
  - Estágio 1: O servidor verifica se ele deve permitir que você se conecte.
  - Estágio 2: Supondo que você possa se conectar, o servidor verifica cada declaração que você emite para ver se você tem privilégios suficientes para executá-la.

SCHEMA mysql – GRANT TABLES

- As informações sobre os usuários do servidor MYSQL e seus privilégios são armazenados em 4 tabelas principais que estão localizadas no banco de dados **mysql**.
  - A tabela **user** armazena as informações de todos os usuários do banco e os privilégios globais deste usuário.
  - A tabela **db** armazena os privilégios dos usuários específicos de um banco de dados.
  - Finalmente, as tabelas **tables_priv** e **columns_priv** armazenam os privilégios associados a tabelas e colunas, respectivamente.
- Além dessas 4 tabelas, o banco **mysql** possui as tabelas **global_grants**, **procs_priv**, **proxies_priv**, **default_roles**, **role_edges** e **password_history** possui

GRANT TABLES - user

- Armazena informações de todos os usuários e seus privilégios.
- Para conexões permitidas, quaisquer privilégios concedidos na tabela **user** indicam os privilégios globais (superuser) do usuário.
- Esses privilégios se aplicam a todos os bancos de dados do servidor.

GRANT TABLES - db

- As colunas de escopo da tabela **db** determinam quais usuários podem acessar quais bancos de dados a partir de quais hosts.
- As colunas de privilégios determinam quais operações são permitidas.
- Um privilégio concedido no nível de banco de dados se aplica ao banco de dados e a todas as suas tabelas.

GRANT TABLES - tables_priv e columns_priv

- As tabelas **tables_priv** e **columns_priv** são semelhantes à tabela **db**, mas são mais refinadas: elas se aplicam nos níveis de tabela e coluna, em vez de no nível do banco de dados.
- Um privilégio concedido no nível da tabela se aplica à tabela e a todas as suas colunas.
- Um privilégio concedido no nível da coluna se aplica apenas a uma coluna específica.

NÍVEIS DE PRIVILÉGIOS

- Os privilégios concedidos a uma conta MySQL determinam quais operações a conta pode realizar.
- Os privilégios do MySQL diferem nos contextos em que se aplicam e em diferentes níveis de operação:
  - Privilégios administrativos.
  - Privilégios de banco de dados.
  - Privilégios para objetos de banco de dados como tabelas, índices, visualizações e rotinas armazenadas.

PRIVILÉGIOS ADMINISTRATIVOS

- Permitem que os usuários gerenciem a operação do servidor MySQL.
- Esses privilégios são globais porque não são específicos de um banco de dados.
- Privilégios globais se aplicam a todos os bancos de dados em um determinado servidor.
    - Esses privilégios são armazenados na tabela **mysql.user**

PRIVILÉGIOS DE BANCO DE DADOS

- Privilégios de banco de dados se aplicam a um banco de dados e a todos os objetos dentro dele.
- Esses privilégios podem ser concedidos para bancos de dados específicos ou globalmente para que se apliquem a todos os bancos de dados.
- Sintaxe:
  - **GRANT ALL ON** db_name.* e **REVOKE ALL ON** db_name.* concedem e revogam apenas privilégios de banco de dados.

PRIVILÉGIOS PARA OBJETOS

- Privilégios para objetos de banco de dados como tabelas, índices, visualizações e rotinas armazenadas podem ser concedidos para:
  - objetos específicos dentro de um banco de dados;
  - para todos os objetos de um determinado tipo dentro de um banco de dados (por exemplo, todas as tabelas em um banco de dados);
  - globalmente para todos objetos de um determinado tipo em todos os bancos de dados.

PRIVILÉGIOS PARA OBJETOS - tabela

- Nível de tabela
  - Os privilégios de tabela aplicam-se a todas as colunas em uma determinada tabela.
  - Esses privilégios estão armazenados na tabela **mysql.tables_priv**.
  - **GRANT ALL ON** db_name.tbl_name e **REVOKE ALL ON** db_name.tbl_name concedem e revogam apenas privilégios de tabela.

PRIVILÉGIOS PARA OBJETOS - coluna

- Nível da coluna
  - Os privilégios da coluna aplicam-se a colunas únicas em uma determinada tabela.
  - Esses privilégios estão armazenados na mesa **mysql.columns_priv**.
  - Ao usar **REVOKE**, você deve especificar as mesmas colunas para as quais foram concedidas acesso anteriormente.

PRIVILÉGIOS PARA OBJETOS - rotina

- Nível da Rotina
  - Os privilégios **CREATE ROUTINE**, **ALTER ROUTINE**, **EXECUTE** e **GRANT** (funções e procedures) aplicam-se a rotinas armazenadas.
  - Eles podem ser concedidos nos níveis global e de banco de dados.
  - Além disso, com exceção da **CREATE ROUTINE**, esses privilégios podem ser concedidos no nível de rotina para rotinas individuais e são armazenados na tabela **mysql.procs_priv**.

Quando as alterações de privilégios fazem efeito?

- Quando o servidor recarrega as tabelas de **GRANT**, os privilégios para as conexões existentes do cliente são afetados da seguinte forma:
  - As alterações de privilégio de tabela e coluna fazem efeito com a próxima solicitação do cliente.
  - Alterações de privilégio de banco de dados fazem efeito na próxima instrução **USE** db_name.
  - Alterações nos privilégios globais e senhas fazem efeito na próxima vez que o cliente se conectar.

Quando as alterações de privilégios fazem efeito?

- Se você modificar as tabelas de **GRANT** usando **GRANT**, **REVOKE** ou **SET PASSWORD**, o servidor perceberá essas alterações e recarregar as tabelas de **GRANT** na memória novamente imediatamente.
- Se você modificar as tabelas de **GRANT** diretamente usando declarações como **INSERT**, **UPDATE** ou **DELETE**, suas alterações não terão efeito na verificação de privilégios até que você reinicie o servidor ou diga para recarregar as tabelas.
- Para recarregar as tabelas de **GRANT** manualmente, emita uma instrução **FLUSH PRIVILEGES** ou execute um comando de recarga de **mysqladmin flush-privileges** ou **mysqladmin reload**.

Adicionando/Removendo novas contas de usuário ao MySQL

- Você pode criar/remover contas MySQL usando a instrução **CREATE/DROP USER**.

```
CREATE USER 'estudantes'@'localhost' IDENTIFIED BY 'curitiba123';
DROP USER 'estudantes'@'localhost';
```

Limitando recursos da conta

- No MySQL, você pode limitar os seguintes recursos do servidor para contas individuais:
  - O número de consultas que uma conta pode emitir por hora (**MAX_QUERIES_PER_HOUR**).
  - O número de atualizações que uma conta pode emitir por hora (**MAX_UPDATES_PER_HOUR**).
  - O número de vezes que uma conta pode se conectar ao servidor por hora (**MAX_CONNECTIONS_PER_HOUR**).
  - O número de conexões simultâneas ao servidor em uma base por conta (**MAX_USER_CONNECTIONS**).

Limitando recursos - exemplo

- Durante a criação do usuário:

```
CREATE USER 'estudantes'@'localhost' IDENTIFIED BY 'curitiba123'
    WITH MAX_QUERIES_PER_HOUR 100
    MAX_UPDATES_PER_HOUR 100
    MAX_CONNECTIONS_PER_HOUR 4
    MAX_USER_CONNECTIONS 2;
```

Para remover um limite, defina seu valor como zero

- Alterando o usuário:

```
ALTER USER 'estudantes'@'localhost'
    WITH MAX_QUERIES_PER_HOUR 100
    MAX_UPDATES_PER_HOUR 100
    MAX_CONNECTIONS_PER_HOUR 4
    MAX_USER_CONNECTIONS 2;
```

Para redefinir as contagens atuais para zero para todas as contas, emita uma instrução **FLUSH USER_RESOURCES**.

Criando/Removendo papéis no MySQL

- Os papéis (roles) são utilizados para facilitar a atribuição dos privilégios aos usuários.
- Para cada tipo de usuários específico do banco de dados podemos criar um papel, por exemplo, administrador, cliente, funcionário, gerente e garçom.
- Você pode criar/remover papéis no MySQL usando as instruções **CREATE/DROP ROLE**.

```
CREATE ROLE IF NOT EXISTS 'administrator', 'client', 'employee', 'manager', 'waiter';

DROP ROLE IF EXISTS 'estudantes';
```

SINTAXE

- Atribuindo permissões para usuário:

  GRANT priv [(colunas)] [, priv [(colunas)]] ...

  ON {*.* | db.* | db.tabela}

  TO user1 [, user2 ...]

  [**WITH GRANT OPTION**]

- Atribuindo permissões para role:

  GRANT priv [(colunas)] [, priv [(colunas)]] ...

  ON {*.* | db.* | db.tabela}

  TO role1 [, role2 ...]

  [**WITH GRANT OPTION**]

- Atribuindo uma role para um usuário:

  GRANT role1, role2 **TO** user1, user2;

Os [ ] indicam que o comando é opcional. O primeiro item a ser informado é(são) o(s) privilégio(s) a ser(em) concedido(s) ao(s) usuário(s).

Privilégio	Descrição
ALL [PRIVILEGES]	Todos os privilégios exceto **GRANT OPTION**
ALTER	Permite executar **ALTER TABLE**
CREATE	Permite executar **CREATE TABLE**
CREATE TEMPORARY TABLES	Permite executar **CREATE TEMPORARY TABLE**
DELETE	Permite executar **DELETE**
DROP	Permite executar **DROP TABLE**
EXECUTE	Permite executar stored procedures (MySQL 5.0)
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
RELOAD	Permite executar **FLUSH**
REPLICATION CLIENT	Permite ao usuário obter a localização do Master ou Slave
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
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

Especificação do nível de permissão

- Uma vez informados os privilégios do usuário, você deverá indicar o nível ao qual o privilégio se aplica, sendo possível especificar três níveis:

.	Privilégio global
db.*	Qualquer tabela do banco db
db.tb	Apenas a tabela tb do banco de dados db. Para especificar apenas algumas colunas de uma determinada tabela, estas deverão ser listadas ao lado do privilégio (priv (colunas))

Especificação do(s) usuário(s)

- Depois do nível você deverá indicar o usuário, ou a lista de usuários, para os quais os privilégios se aplicam.
- No MySQL o usuário é constituído de um nome mais o host de onde ele poderá acessar o servidor (user@host).
- Caso você não informe o host para o usuário, o MySQL assumirá "%", isto é, todos os hosts.
- A senha do usuário é opcional, mas é recomendado sempre informá-la no momento de criação do usuário, por questões de segurança.

EXEMPLO I

- No exemplo a seguir é criado um usuário com o nome **clientes** que pode se conectar somente do host onde o servidor está em execução (localhost), o usuário só poderá fazer **SELECT** nas colunas *nome* e *valor* da tabela **produto**, que se encontra no banco de dados **restaurante**. A senha do usuário é *12345*.

```
CREATE USER 'clientes'@'localhost' IDENTIFIED BY 'curitiba123'

GRANT SELECT (nome, valor) ON restaurante.produto TO 'clientes'@'localhost';
```

EXEMPLO II

- Você pode especificar um conjunto de hosts utilizando o caracter "%", neste caso é possível dar acesso a um usuário dentro de uma faixa de IPs ou DNS.
- No exemplo a seguir, o usuário remoto poderá executar **UPDATE** e **INSERT** em qualquer tabela do banco de dados restaurante, sendo possível a conexão ao servidor a partir de qualquer máquina no domínio restaurante.com.br:

```
CREATE USER 'remoto'@'%.restaurante.com.br' IDENTIFIED BY 'curitiba123'
GRANT UPDATE, INSERT ON restaurante.* TO 'remoto'@"%.restaurante.com.br";

CREATE USER 'remoto'@'200.236.13.%' IDENTIFIED BY 'curitiba123'
GRANT UPDATE, INSERT ON restaurante.* TO 'remoto'@"200.236.13.%";
```

EXEMPLO III

- Usuários anônimos também podem ser criados informando um nome com o caractere espaço (" ").
- Um usuário anônimo com os mesmos privilégios do usuário remoto seria criado da seguinte forma:

```
CREATE USER ' '@'localhost';

GRANT UPDATE, INSERT ON restaurante.* TO ' '@'localhost';
```

CLÁUSULAS ADICIONAIS

- Finalmente, a opção **GRANT OPTION** é utilizada para que o usuário possa conceder os seus privilégios para outros usuários do banco.

```
GRANT SELECT (nome, valor) ON restaurante.produto
TO 'clientes'@'localhost' WITH GRANT OPTION;
```

REMOVENDO PRIVILÉGIOS

- Para remover um privilégio do usuário utilize o comando **REVOKE**

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

Lembre-se de que a parte **ON** do **REVOKE** deverá coincidir com a parte **ON** do **GRANT** que você deseja remover, caso isto não se verifique o comando **REVOKE** não terá efeito algum

REMOVENDO USUÁRIO

- O comando **REVOKE** remove apenas os privilégios do usuário, mas o usuário continuará existindo.
- A remoção do usuário deverá ser feita com um **DELETE** explícito na tabela de usuários do MySQL, após terem sido Removidos todos os seus privilégios com o comando **REVOKE**

```
DELETE FROM mysql.user WHERE user= "clientes" AND host="localhost";
```

Observe que para remover o usuário você deverá ter privilégio para executar **DELETE** na tabela user do mysql

- Feito isto, você terá que executar um comando **FLUSH PRIVILEGES** para que o MySQL possa atualizar os privilégios que estão em memória

TRANSACTIONS

O QUE SÃO TRANSAÇÕES?

- Uma transação no MySQL é um grupo sequencial de instruções, consultas ou operações como **SELECT**, **INSERT**, **UPDATE** ou **DELETE** para executar como uma única unidade de trabalho que pode ser confirmada ou revertida.
- Se a transação fizer várias modificações no banco de dados, duas coisas acontecem:
  - Todas as modificações são bem-sucedidas quando a transação é confirmada.
  - Ou todas as modificações são desfeitas quando a transação é revertida.

Como funciona?

- A transação MySQL permite que você execute um conjunto de operações do MySQL para garantir que o banco de dados nunca contenha o resultado parcial de operações.
  - Em um conjunto de operações, se uma delas falhar, ocorre a reversão para restaurar o banco de dados ao seu estado original.
  - Se nenhum erro ocorrer, todo o conjunto de instruções será confirmado no banco de dados.

Como funciona?

- Em outras palavras, uma transação não pode ser bem-sucedida sem concluir cada operação disponível no conjunto.
  - Isso significa que se alguma instrução falhar, a operação de transação não poderá produzir resultados.
- Uma transação no MySQL começa com a primeira instrução SQL executável e termina quando encontra um commit ou rollback explicitamente ou implicitamente.
- Ele usa explicitamente a instrução **COMMIT** ou **ROLLBACK** e implicitamente quando uma instrução **DDL** é usada.

Propriedades das Transações

- A transação contém quatro propriedades principais, conhecidas como propriedades ACID.
  - Atomicidade.
  - Consistência.
  - Isolamento.
  - Durabilidade.

ACID - ATOMICIDADE

- **Atomicidade**: Esta propriedade garante que todas as instruções ou operações dentro da unidade de transação devem ser executadas com sucesso. Caso contrário, se alguma operação falhar, toda a transação será abortada e será revertida para seu estado anterior.
- Inclui recursos:
  - declaração **COMMIT**.
  - instrução **ROLLBACK**.
  - Configuração de confirmação automática.
  - Dados operacionais das tabelas INFORMATION_SCHEMA.

ACID - CONSISTÊNCIA

- **Consistência**: Esta propriedade garante que o banco de dados mude de estado somente quando uma transação for confirmada com sucesso. Também é responsável por proteger os dados de falhas.
- Inclui recursos:
  - InnoDB doublewrite buffer.
  - InnoDB crash recovery.

ACID - ISOLAMENTO

- **Isolamento**: Esta propriedade garante que cada operação na unidade de transação opere de forma independente. Também garante que as declarações sejam transparentes entre si.
- Inclui recursos:
  - instrução **SET ISOLATION LEVEL**.
  - Configuração de confirmação automática.
  - Os detalhes de baixo nível do bloqueio do InnoDB.

ACID - DURABILIDADE

- **Durabilidade**: Esta propriedade garante que o resultado das transações confirmadas persista permanentemente mesmo se o sistema travar ou falhar.
- Inclui recursos:
  - Buffer de gravação em um dispositivo de armazenamento.
  - Cache com bateria em um dispositivo de armazenamento.
  - Opção de configuração **innodb_file_per_table**.
  - Opção de configuração **innodb_flush_log_at_trx_commit**.
  - Opção de configuração **sync_binlog**.

Instruções de transação do MySQL

- O MySQL nos fornece a seguinte declaração importante para controlar transações:
  - Para iniciar uma transação, você usa a instrução **START TRANSACTION**. O **BEGIN** ou **BEGIN WORK** são os *aliases* da **START TRANSACTION**.
  - Para confirmar a transação atual e tornar suas alterações permanentes, use a instrução **COMMIT**.
  - Para reverter a transação atual e cancelar suas alterações, use a instrução **ROLLBACK**.
  - Para desabilitar ou habilitar o modo de autocommit para a transação atual, use a instrução **SET** autocommit.

EXEMPLO COMMIT

```
SET autocommit = FALSE;

START TRANSACTION; #BEGIN; #BEGIN WORK;

INSERT INTO restaurante.pessoa VALUES (NULL, "Jessica Rocha", "12341234544", "F", "1986-09-23");

SET @pessoa_id := (SELECT LAST_INSERT_ID());

INSERT INTO restaurante.cliente VALUES (@pessoa_id, NOW());

INSERT INTO restaurante.comanda VALUES (NULL, NOW(), 0.0, 0, @pessoa_id, NULL);

SET @comanda_id := (SELECT LAST_INSERT_ID());

COMMIT;
```

INSTRUÇÃO SAVEPOINT

- Um **SAVEPOINT** é um ponto de reversão lógico dentro de uma transação.
- Quando você define um ponto de salvamento, sempre que ocorrer um erro após um ponto de salvamento, você pode desfazer os eventos que você fez até o ponto de salvamento usando o **ROLLBACK**.
- MySQL InnoDB fornece suporte para as instruções **SAVEPOINT**, **ROLLBACK TO SAVEPOINT**, **RELEASE SAVEPOINT**.
- A instrução **SAVEPOINT** é usada para definir um ponto de salvamento para a transação com o nome especificado.
  - Se já existir um ponto de salvamento com o nome fornecido, o antigo será excluído.;

EXEMPLO SAVEPOINT

```
SET autocommit = FALSE;
START TRANSACTION;
SELECT * FROM garcom g JOIN funcionario f ON g.id = f.id JOIN pessoa p ON p.id = f.id;
UPDATE garcom SET valor_hora = valor_hora + 1;
SAVEPOINT insert_garcom;
INSERT INTO restaurante.pessoa VALUES (NULL, "Mariana Fernandes", "34341234544", "F", "1991-04-05");
SET @pessoa_id := (SELECT LAST_INSERT_ID());
INSERT INTO restaurante.funcionario VALUES (@pessoa_id, "mariana@restaurante.com","41999090900",
NOW());
 INSERT INTO restaurante.garcom VALUES (@pessoa_id, 90.00);
SELECT * FROM garcom g JOIN funcionario f ON g.id = f.id JOIN pessoa p ON p.id = f.id;
ROLLBACK TO SAVEPOINT insert_garcom;
COMMIT;
SELECT * FROM garcom g JOIN funcionario f ON g.id = f.id JOIN pessoa p ON p.id = f.id;
```