Let's compare MariaDB and MySQL Roles

Vicențiu Ciorbaru
Software Developer Team Lead
@ MariaDB Foundation

MariaDB Day
2020 Feb 2 Brussels
whoami

- Vicențiu Ciorbaru
- MariaDB Foundation, Software Developer Team Lead
- MariaDB developer since 2013-...
- Implemented Roles, Window Functions and others
A brief history on roles

● In the open source database world:
  ○ PostGres added roles in 8.1 (2005)
  ○ MariaDB added roles in 10.0 (2013)
  ○ MySQL added roles in 8.0 (2016)

● Most features correspond to SQL Standard.
What are roles?

- Roles are almost like users, with some special powers.
- Roles can own (have access) to objects in the database.
  - Read / Write tables
  - Create / Drop databases
  - Create / Drop / Alter users
  - ...
- Roles can not be used to login. *

- Roles can inherit rights from other roles.
- Roles can be granted to users and subsequently activated.

*MariaDB Foundation
https://mariadb.org
Benefit of roles

- Roles simplify privilege granting / revoking.
  - Easier maintenance and auditing.

- Allow easy implementation of least-privilege principle.

- No practical performance impact. More details later.

- Can be integrated with legacy applications with the help of the default role.
Basic use case

- A simplified data warehouse.

```sql
CREATE DATABASE data_warehouse;
CREATE DATABASE staging;
CREATE TABLE data_warehouse.transactions (
  date datetime,
  type varchar(20),
  amount double,
  customer varchar(100));

CREATE USER rachel;
CREATE ROLE report;
CREATE USER import_robot;
CREATE ROLE import;
CREATE USER dave;
CREATE ROLE dev;
CREATE USER alex;
CREATE ROLE admin;
```
Basic use case

```
CREATE USER rachel;
CREATE ROLE report;
CREATE USER import_robot;
CREATE ROLE import;
CREATE USER dave;
CREATE ROLE dev;
CREATE USER alex;
CREATE ROLE admin;

GRANT SELECT ON data_warehouse.* TO report;
GRANT INSERT ON data_warehouse.* TO import;
GRANT ALL ON data_warehouse.* TO dev;
GRANT ALL ON staging.* TO dev;

GRANT ALL ON data_warehouse.* TO admin;
GRANT ALL ON staging.* TO admin;
GRANT CREATE USER ON *.* TO admin WITH GRANT OPTION;

GRANT report TO rachel;
GRANT import TO import_robot;
GRANT dev TO dave;
GRANT admin TO alex;
```
Basic use case

```
CREATE USER rachel;  CREATE ROLE report;
CREATE USER import_robot;  CREATE ROLE import;
CREATE USER dave;  CREATE ROLE dev;
CREATE USER alex;  CREATE ROLE admin;

GRANT SELECT ON data_warehouse.* TO report;
GRANT INSERT ON data_warehouse.* TO import;
GRANT ALL ON data_warehouse.* TO dev;
GRANT ALL ON staging.* TO dev;

GRANT ALL ON data_warehouse.* TO admin;
GRANT ALL ON staging.* TO admin;
GRANT CREATE USER ON *.* TO admin WITH GRANT OPTION;

GRANT report TO rachel;
GRANT import TO import_robot;
GRANT dev TO dave;
GRANT admin TO alex;
```
Basic use case

SELECT

INSERT

report

import

dev

admin

Rachel

import_robot

Dave

Alex

DWH

STAGING

MySQL
Basic use case

Can we do better?

SELECT

INSERT

DWH

STAGING

MySQL

report

import

dev

admin

Rachel

import_robot

Dave

Alex
Role hierarchies

- To eliminate duplicate grants, one can create a role hierarchy.
- Rights from the granted role propagate to the grantee role.
Role hierarchies

- To eliminate duplicate grants, one can create a role hierarchy.
- Rights from the granted role propagate to the grantee role.

```sql
CREATE USER rachel;
CREATE ROLE report;
CREATE USER import_robot;
CREATE ROLE import;
CREATE USER dave;
CREATE ROLE dev;
CREATE USER alex;
CREATE ROLE admin;

GRANT SELECT ON data_warehouse.* TO report;
GRANT INSERT ON data_warehouse.* TO import;
GRANT ALL ON data_warehouse.* TO dev;
GRANT ALL ON staging.* TO dev;

GRANT ALL ON data_warehouse.* TO admin;
GRANT ALL ON staging.* TO admin;
GRANT CREATE USER ON *.* TO admin WITH GRANT OPTION;
```
Role hierarchies

- To eliminate duplicate grants, one can create a role hierarchy.
- Rights from the granted role propagate to the grantee role.

CREATE USER rachel;
CREATE USER import_robot;
CREATE USER dave;
CREATE USER alex;
CREATE ROLE report;
CREATE ROLE import;
CREATE ROLE dev;
CREATE ROLE admin;

GRANT SELECT ON data_warehouse.* TO report;
GRANT INSERT ON data_warehouse.* TO import;
GRANT ALL ON data_warehouse.* TO dev;
GRANT ALL ON staging.* TO dev;

GRANT ALL ON data_warehouse.* TO admin;
GRANT ALL ON staging.* TO admin;
GRANT CREATE USER ON *.* TO admin WITH GRANT OPTION;
Role hierarchies

● To eliminate duplicate grants, one can create a role hierarchy.
● Rights from the granted role propagate to the grantee role.

CREATE USER rachel;
CREATE ROLE report;
CREATE USER import_robot;
CREATE ROLE import;
CREATE USER dave;
CREATE ROLE dev;
CREATE USER alex;
CREATE ROLE admin;

GRANT SELECT ON data_warehouse.* TO report;
GRANT INSERT ON data_warehouse.* TO import;
GRANT ALL ON data_warehouse.* TO dev;
GRANT ALL ON staging.* TO dev;

GRANT dev TO admin;
GRANT CREATE USER ON *.* TO admin WITH GRANT OPTION;
Basic use case

DWH

STAGING

MySQL

SELECT

INSERT

report

import

dev

admin

Rachel

import_robot

Dave

Alex
Basic use case

SELECT

INSERT

report

import

dev

admin

Rachel

import_robot

Dave

Alex
Basic use case

DWH

STAGING

MySQL

SELECT

INSERT

report

import

dev

admin

Rachel

import_robot

Dave

Alex
Role hierarchies

● To eliminate duplicate grants, one can create a role hierarchy.
● Rights from the granted role propagate to the grantee role.

● MariaDB disallows cycles.
● MySQL allows cycles.

● When a cycle is created, the whole hierarchy collapses. All roles have the same privileges (ALL from the hierarchy).
Role hierarchies

- Effective privileges for roles in a graph are cached.

- MariaDB and MySQL employ different caching strategies.
  - MariaDB
    - Computes the effective privileges when loading the role graph from disk.
    - Recomputes only necessary parts when grants are performed.
  - MySQL
    - Computes an "effective" authentication id (user + all active roles)
    - Computes effective rights for authentication id.
    - Stores effective rights in a cache for subsequent queries.
Role hierarchies

- Both databases optimise for the critical path.
- MariaDB slower GRANT operations (infrequent).
- Always fast access checks (frequent)
- MySQL no performance penalty on GRANT operations. (infrequent)
- First access check slow, then fast. (frequent)
Least privilege principle

- Dave has a tricky bug to debug. He needs a clone of some tables to investigate.
Least privilege principle

- Dave has a tricky bug to debug. He needs a clone of some tables to investigate.

```
SET ROLE dwh_development;
```

- Dave gets to work on the staging database.

```
CREATE TABLE staging.transactions LIKE transactions;
```
Least privilege principle

- Dave has a tricky bug to debug. He needs a clone of some tables to investigate.

```sql
SET ROLE dwh_development;
```

- Dave gets to work on the staging database.

```sql
CREATE TABLE staging.transactions LIKE transactions;
```

- Dave does some work, investigating experimenting with different transactions.
Least privilege principle

- Dave has a tricky bug to debug. He needs a clone of some tables to investigate.

```
SET ROLE dwh_development;
```

- Dave gets to work on the staging database.

```
CREATE TABLE staging.transactions LIKE transactions;
```

- Dave does some work, investigating experimenting with different transactions.
- After a long day he finishes and wants to clean up...
Least privilege principle

- Dave has a tricky bug to debug. He needs a clone of some tables to investigate.

```
SET ROLE dwh_development;
```

- Dave gets to work on the staging database.

```
CREATE TABLE staging.transactions LIKE transactions;
```

- Dave does some work, investigating experimenting with different transactions.

- After a long day he finishes and wants to clean up...

```
DROP TABLE transactions;
```
Dave has a tricky bug to debug. He needs a clone of some tables to investigate.

```
SET ROLE dwh_development;
```

Dave gets to work on the staging database.

```
CREATE TABLE staging.transactions LIKE transactions;
```

Dave does some work, investigating experimenting with different transactions.

After a long day he finishes and wants to clean up...

```
DROP TABLE transactions; -- oops, wrong current database;
```
Least privilege principle

- All this could have been prevented.
- Split roles into safe and dangerous ones.
- Only activate dangerous ones when you need them.
- This would not be possible without roles without switching users.
Roles related features

- MySQL allows enabling multiple roles at once.
- MySQL implements "mandatory roles".
  - Roles that are always active for a user.
  - Implicitly granted

- MySQL and MariaDB implement DEFAULT ROLE
  - Role is activated on login.
  - MySQL can activate multiple roles on login.
INFORMATION_SCHEMA for Roles

- **INFORMATION_SCHEMA.ENABLED_ROLES**
  - MySQL reports just the direct list of enabled roles.
  - MariaDB reports the enabled role, plus the effective inherited roles.

- **INFORMATION_SCHEMA.APPLICABLE_ROLES**
  - Same behaviour in MySQL and MariaDB.
  - MySQL extends the table with specific columns - is_mandatory

- **Newly introduced in MySQL 8.0.19 only**
  - INFORMATION_SCHEMA.ROLE_TABLE_GRANTS
  - INFORMATION_SCHEMA.ROLE_ROUTINE_GRANTS
  - INFORMATION_SCHEMA.ROLE_COLUMN_GRANTS
  - INFORMATION_SCHEMA.ADMINISTRABLE_ROLE_AUTHORIZATIONS
Overall comparison

● Both databases allow role-based access control.
  ○ From a functional perspective both implementations meet the core requirements.
● MariaDB never allows authentication via Roles. MySQL allows it.

● MySQL allows activating multiple roles at the same time. MariaDB can achieve the same result by creating an intermediate aggregate role.

● Since latest MySQL release, MySQL 8.0 has more complete INFORMATION_SCHEMA list.

● **If you stick to basic SQL Standard syntax, implementations are compatible.**
This talk would not have been possible if it not for the MariaDB Foundation's sponsors.
Thank you!

Contact details:

vicentiu@mariadb.org

About:

https://mariadb.org/vicentiu