Introduction to TCL & DCL

MySQL Transaction

A transaction in MySQL is a sequential group of statements, queries, or operations such as select, insert, update or delete to perform as a one single work unit that can be committed or rolled back. If the transaction makes multiple modifications into the database, two things happen:

- Either, all modifications are successful when the transaction is committed.
- Or, all modifications are undone when the transaction is rollback.

In a simple words, a transaction cannot be successful without completing each operation available in the set. It means if any statement fails, the transaction operation cannot produce results.

MySQL Transaction

Let us understand the Transaction with a small banking process. A bank customer wants to transfer money from one account to another account. We can achieve this by using the SQL statements that will be divided into the following steps:

1. First, it is required to check the availability of the requested amount in the first account.
2. Next, if the amount is available, deduct it from the first account. Then, update the first account.
3. Finally, deposit the amount in the second account. Then update the second account to complete the transaction.
4. If any of the above processes fails, the transaction will be rolled back into its previous state.

Properties of Transaction

The transaction contains mainly four properties, which referred to as ACID property. Now, we are going to discuss the ACID property in detail. The ACID property stands for:

- Atomicity
- Consistency
- Isolation
- Durability

Atomicity: This property ensures that all statements or operations within the transaction unit must be executed successfully. Otherwise, if any operation is failed, the whole transaction will be aborted, and it goes rolled back into their previous state.

Properties of Transaction

Consistency: his property ensures that the database changes state only when a transaction will be committed successfully. It is also responsible for protecting data from crashes.

Isolation: This property guarantees that each operation in the transaction unit operated independently. It also ensures that statements are transparent to each other.

Durability: This property guarantees that the result of committed transactions persists permanently even if the system crashes or failed.

MySQL Transaction Statement

MySQL control transactions with the help of the following statement:

- MySQL provides a START TRANSACTION statement to begin the transaction. It also offers a "BEGIN" and "BEGIN WORK" as an alias of the START TRANSACTION.
- We will use a COMMIT statement to commit the current transaction. It allows the database to make changes permanently.
- We will use a ROLLBACK statement to roll back the current transaction. It allows the database to cancel all changes and goes into their previous state.
- We will use a SET auto-commit statement to disable/enable the auto-commit mode for the current transaction. By default, the COMMIT statement executed automatically. Hence, if we do not want to commit changes automatically.

  SET autocommit = 0 to turn-off the auto commit and 1 to turn it on to commit automatically.

MySQL Transaction

Example for COMMIT

```
select * from employee;  ## Before transaction state of the table
set autocommit = 0; ## Setting the autocommit off for the session

start transaction; ## Starting the transaction
## Set of statements
select @emp_id = max(id) + 1 from employee;
insert into employee values (@emp_id, 'Arun', 25000);
commit;   ## Committing the transaction

select * from employee; ## After transaction state of the table
```

MySQL Transaction

Example for ROLLBACK

```
select * from employee;  ## Before transaction state of the table

start transaction; ## Starting the transaction
## Set of statements
Delete Employee;
Select * from employee;
rollback;   ## Rollback the transaction

select * from employee; ## After transaction state of the table
```

MySQL Transaction

The **SAVEPOINT** statement creates a special mark with the name of the identifier inside a transaction. It allows all statements that are executed after savepoint would be rolled back. So that the transaction restores to the previous state it was in at the point of the savepoint. If we have set multiple savepoints in the current transaction with the same name, the newly savepoint is responsible for rollback.

The **ROLLBACK TO SAVEPOINT** statement allows us to rolls back all transactions to the given savepoint that was established without aborting the transaction.

The **RELEASE SAVEPOINT** statement destroys the named savepoint from the current transaction without undoing the effects of queries executed after the savepoint was established. After these statements, no rollback command occurs. If the savepoint does not exist in the transaction, it gives an error

MySQL Grant Privilege

GRANT Statement: The grant statement enables system administrators to assign privileges and roles to the MySQL user accounts so that they can use the assigned permission on the database whenever required.

Syntax

```
GRANT privilege_name(s)   ON object  TO user_account_name;
```

privilege_name(s): It specifies the access rights or grant privilege to user accounts. If we want to give multiple privileges, then use a comma operator to separate them.

Object: It determines the privilege level on which the access rights are being granted. It means granting privilege to the table; then the object should be the name of the table.

user_account_name: It determines the account name of the user to whom the access rights would be granted.

MySQL Grant Privilege

Privilege Level	Syntax	Descriptions
Global	GRANT ALL ON *.* TO john@localhost;	It applies to all databases on MySQL server. We need to use *.* syntax for applying global privileges. Here, the user can query data from all databases and tables of the current server.
Database	GRANT ALL ON mydb.* TO john@localhost;	It applies to all objects in the current database. We need to use the db_name.* syntax for applying this privilege. Here, a user can query data from all tables in the given database.
Table	GRANT DELETE ON mydb.employees TO john@localhsot;	It applies on all columns in a specified table. We need to use db_name.table_name syntax for assigning this privilege. Here, a user can query data from the given table of the specified database.

MySQL Grant Privilege

Privilege Level	Syntax	Descriptions
Column	GRANT SELECT (col1), INSERT (col1, col2), UPDATE (col2) ON mydb.mytable TO john@localhost;	It applies on a single column of a table. Here, we must have to specify the column(s) name enclosed with parenthesis for each privilege. The user can select one column, insert values in two columns, and update only one column in the given table.
Stored Routine	GRANT EXECUTE ON PROCEDURE mydb.myprocedure TO john@localhost;	It applies to stored routines (procedure and functions). It contains CREATE ROUTINE, ALTER ROUTINE, EXECUTE, and GRANT OPTION privileges. Here, a user can execute the stored procedure in the current database.
Proxy	GRANT PROXY ON root TO peter@localhost;	It enables one user to be a proxy for other users.

MySQL Grant Privilege

CREATING an user:

```
CREATE USER shan@localhost IDENTIFIED BY ’shan12345’;
```

DISPLAY USER Privileges

```
SHOW GRANTS FOR shan@localhost;
```

```
## To assign all privileges on all objects to a database;
GRANT ALL ON tempshan.* TO shan@localhost;
```

```
## To assign EXECUTE privilege on stored functions.
GRANT EXECUTE ON FUNCTION grade_fn TO shan@localhost;
```

MySQL Grant Privilege

We can choose access right from the below list on which privileges can be applied.

SELECT: It enables us to view the result set from a specified table.
INSERT: It enables us to add records in a given table.
DELETE: It enables us to remove rows from a table.
CREATE: It enables us to create tables/schemas.
ALTER: It enables us to modify tables/schemas.
UPDATE: It enables us to modify a table.
DROP: It enables us to drop a table.
INDEX: It enables us to create indexes on a table.
ALL: It enables us to give ALL permissions except GRANT privilege.
GRANT: It enables us to change or add access rights.

MySQL REVOKE Privilege

REVOKE Statement: The revoke statement enables system administrators to revoke privileges and roles to the MySQL user accounts so that they cannot use the assigned permission on the database in the past.

Syntax

```
REVOKE privilege_name(s)   ON object   FROM user_account_name;
```

privilege_name(s): It specifies the access rights or grant privilege that we want to revoke from user accounts.

Object: It determines the privilege level on which the access rights are being granted. It means granting privilege to the table; then the object should be the name of the table.

user_account_name: It determines the account name of the user from which we want to revoke the access rights.

MySQL REVOKE Privilege

Privilege Level	Syntax	Descriptions
Global	REVOKE ALL, GRANT OPTION FROM john@localhost;	It applies to remove all access rights from the user on MySQL server.
Database	REVOKE ALL ON mydb.* FROM john@localhost;	It applies to revoke all privileges from objects in the current database.
Table	REVOKE DELETE ON mydb.employees FROM john@localhsot;	It applies to revoke privileges from all columns in a specified table.

MySQL REVOKE Privilege

Privilege Level	Syntax	Descriptions
Column	REVOKE SELECT (col1), INSERT (col1, col2), UPDATE (col2) ON mydb.mytable FROM john@localhost;	It applies to revoke privileges from a single column of a table.
Stored Routine	REVOKE EXECUTE ON PROCEDURE/FUNCTION mydb.myprocedure FROM john@localhost;	It applies to revoke all privileges from stored routines (procedure and functions).
Proxy	REVOKE PROXY ON root FROM peter@localhost;	It enables us to revoke the proxy user.

MySQL REVOKE Privilege

Privilege Level	Syntax	Descriptions
Column	REVOKE SELECT (col1), INSERT (col1, col2), UPDATE (col2) ON mydb.mytable FROM john@localhost;	It applies to revoke privileges from a single column of a table.
Stored Routine	REVOKE EXECUTE ON PROCEDURE/FUNCTION mydb.myprocedure FROM john@localhost;	It applies to revoke all privileges from stored routines (procedure and functions).
Proxy	REVOKE PROXY ON root FROM peter@localhost;	It enables us to revoke the proxy user.

MySQL REVOKE Privilege

```
## Let us display the existing grants for the user
SHOW GRANTS FOR shan@localhost;

## Removing the privilege for the user
REVOKE ALL, GRANT OPTION FROM shan@localhost;

## Removing the execute privilege
REVOKE EXECUTE ON FUNCTION grade_fn TO shan@localhost;
```

MySQL REVOKE Privilege

We can revoke privileges from the below list on which privileges can be applied.

CREATE: It enables the user account to create databases and tables.

DROP: It allows the user account to drop databases and tables.

DELETE: It enables the user account to delete rows from a specific table.

INSERT: It allows the user account to insert rows into a specific table.

SELECT: It enables the user account to read a database.

UPDATE: It enables the user account to update table rows.

thank
you