**A database transaction** is an important concept to understand while working in databases and SQL. Transaction in the database is required to protect data and keep it consistent when **multiple users access the database at the same time**.  In this **database transaction tutorial** we will learn what is a transaction in a database, why do you need *transactions in the database*, ACID properties of database transactions, and an example of database transactions along with commit and rollback.     
  
**what is a transaction in the database?**

A database transaction is a collection of SQL queries that forms a logical task. For a transaction to be completed successfully all SQL queries have to run successfully. Database transaction executes either all or none, so for example if your database transaction contains 4 SQL queries and one of them fails then the change made by the other 3 queries will be rolled back.   
  
This way your database always remains consistent whether the transaction succeeded or failed. The transaction is implemented in the database using **SQL keyword transaction, commit, and rollback**. 

Commit writes the changes made by transaction into the database and rollback removes temporary changes logged in transaction log by database transaction.

**Database Transaction tutorial**

**Why transaction is required in database**

The database is used to store data required by real life application e.g. Banking, Healthcare, Finance etc. All your money stored in banks is stored in the database, all your shares of DMAT account is stored in the database and many application constantly works on these data.   
  
In order to protect data and keep it consistent, any changes in this data need to be done in a transaction so that even in the case of failure data remain in the previous state before the start of a transaction.   
  
**Let us understand the concept of a transaction through the following explanation.**

We can understand the concept of a transaction in [MySQL](https://www.javatpoint.com/mysql-tutorial) by considering a **banking database**. Suppose 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:

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

**ACID Properties of database transaction**

**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. It includes features:

* COMMIT statement.
* ROLLBACK statement.
* Auto-commit setting.
* Operational data from the INFORMATION\_SCHEMA tables.

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

* InnoDB doublewrite buffer.
* InnoDB crash recovery.

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

* SET ISOLATION LEVEL statement.
* Auto-commit setting.
* The low-level details of InnoDB locking.

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

* Write buffer in a storage device.
* Battery-backed cache in a storage device.
* Configuration option innodb\_file\_per\_table.
* Configuration option innodb\_flush\_log\_at\_trx\_commit.
* Configuration option sync\_binlog.

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. So if we do not want to commit changes automatically, use the below statement:

SET autocommit = 0; OR SET autocommit = OFF;