MySQL: Index, ACID and Storage Engines

Tushar B. Kute,
http://tusharkute.com

Index

- An index is a data structure that allows us to add indexes in the existing table. It enables you to improve the faster retrieval of records on a database table.
- It creates an entry for each value of the indexed columns. We use it to quickly find the record without searching each row in a database table whenever the table is accessed.
- We can create an index by using one or more columns of the table for efficient access to the records.

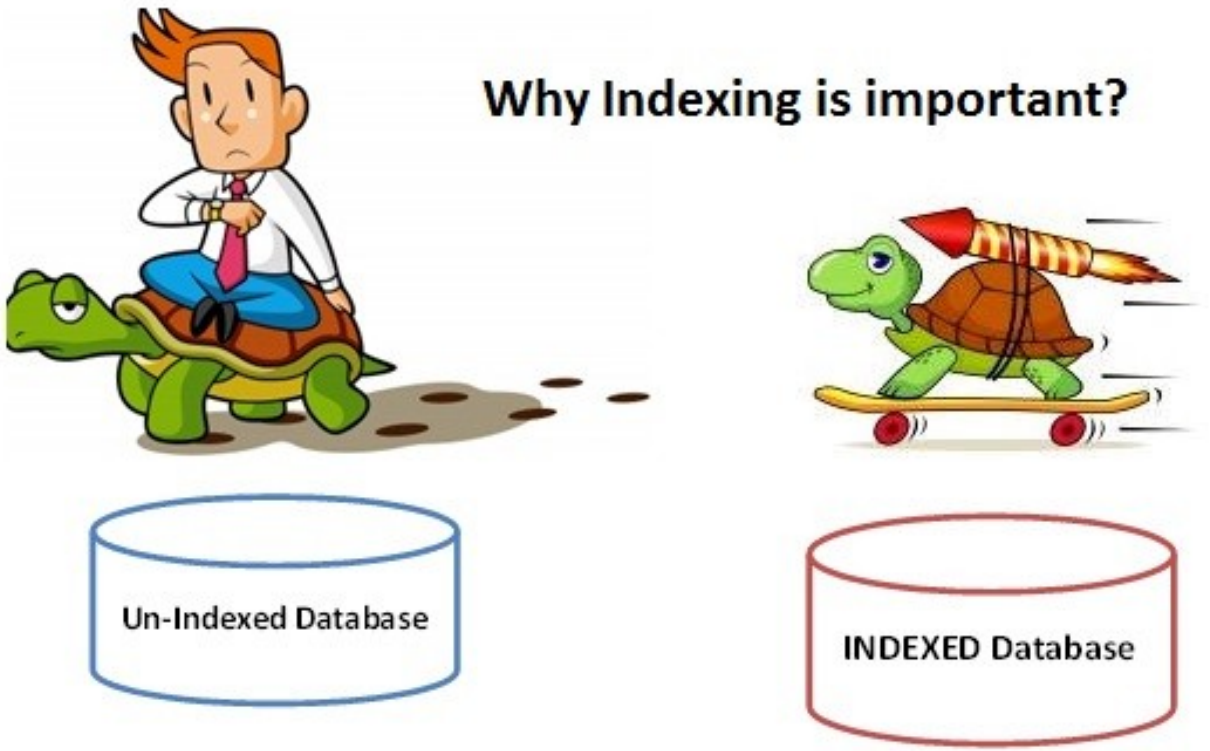
Index

- When a table is created with a primary key or unique key, it automatically creates a special index named PRIMARY.
- We called this index as a clustered index. All indexes other than PRIMARY indexes are known as a non-clustered index or secondary index.

Indexing need

- Suppose we have a contact book that contains names and mobile numbers of the user. In this contact book, we want to find the mobile number of Martin Williamson.
- If the contact book is an unordered format means the name of the contact book is not sorted alphabetically, we need to go over all pages and read every name until we will not find the desired name that we are looking for.
- This type of searching name is known as sequential searching.

Why indexing ?



Syntax:

- Generally, we create an index at the time of table creation in the database. The following statement creates a table with an index that contains two columns col2 and col3.

```
mysql> CREATE TABLE t_index(
    col1 INT PRIMARY KEY,
    col2 INT NOT NULL,
    col3 INT NOT NULL,
    col4 VARCHAR(20),
    INDEX (col2,col3)
);
```

Syntax:

- If we want to add index in table, we will use the CREATE INDEX statement as follows:

```
mysql> CREATE INDEX [index_name] ON
[table_name] (column names)
```

- In this statement, index_name is the name of the index, table_name is the name of the table to which the index belongs, and the column_names is the list of columns.
- Let us add the new index for the column col4, we use the following statement:

```
mysql> CREATE INDEX ind_1 ON t_index(col4);
```

Syntax:

- If you want to show the indexes of a table, execute the following statement:

```
mysql> SHOW INDEXES FROM student;
```

- If we want to get the index information of a table in a different database or database to which you are not connected, MySQL allows us to specify the database name with the Show Indexes statement. The following statement explains it more clearly:

```
mysql> SHOW INDEXES FROM table_name IN
database_name;
```

Syntax:

- Table: It contains the name of the table.
- Non_unique: It returns 1 if the index contains duplicates. Otherwise, it returns 0.
- Key_name: It is the name of an index. If the table contains a primary key, the index name is always PRIMARY.
- Seq_in_index: It is the sequence number of the column in the index that starts from 1.
- Column_name: It contains the name of a column.

Syntax:

- Collation: It gives information about how the column is sorted in the index. It contains values where A represents ascending, D represents descending, and Null represents not sorted.
- Cardinality: It gives an estimated number of unique values in the index table where the higher cardinality represents a greater chance of using indexes by MySQL.
- Sub_part: It is a prefix of the index. It has a NULL value if all the column of the table is indexed. When the column is partially indexed, it will return the number of indexed characters.
- Packed: It tells how the key is packed. Otherwise, it returns NULL.

Syntax:

- NULL: It contains blank if the column does not have NULL value; otherwise, it returns YES.
- Index_type: It contains the name of the index method like BTREE, HASH, RTREE, FULLTEXT, etc.
- Comment: It contains the index information when they are not described in its column. For example, when the index is disabled, it returns disabled.
- Index_column: When you create an index with comment attributes, it contains the comment for the specified index.
- Visible: It contains YES if the index is visible to the query optimizer, and if not, it contains NO.

Drop index

- MySQL allows a DROP INDEX statement to remove the existing index from the table. To delete an index from a table, we can use the following query:

```
mysql>DROP INDEX index_name ON table_name
[algorithm_option | lock_option];
```

- If we want to delete an index, it requires two things:
  - First, we have to specify the name of the index that we want to remove.
  - Second, name of the table from which your index belongs.

Summary of index

- Indexes are very powerful when it comes to greatly improving the performance of MySQL search queries.
- Indexes can be defined when creating a table or added later on after the table has already been created.
- You can define indexes on more than one column on a table.
- The SHOW INDEX FROM table_name is used to display the defined indexes on a table.
- The DROP command is used to remove a defined index on a given table.

Temporary Tables

- MySQL has a feature to create a special table called a Temporary Table that allows us to keep temporary data. We can reuse this table several times in a particular session.
- It is available in MySQL for the user from version 3.23, and above so if we use an older version, this table cannot be used.
- This table is visible and accessible only for the current session. MySQL deletes this table automatically as long as the current session is closed or the user terminates the connection.
- We can also use the DROP TABLE command for removing this table explicitly when the user is not going to use it.

Temporary Tables

- If we use a PHP script to run the code, this table removes automatically as long as the script has finished its execution.
- If the user is connected with the server through the MySQL client, then this table will exist until the user closes the MySQL client program or terminates the connection or removed the table manually.
- A temporary table provides a very useful and flexible feature that allows us to achieve complex tasks quickly, such as when we query data that requires a single SELECT statement with JOIN clauses.
- Here, the user can use this table to keep the output and performs another query to process it.

Temporary Tables

- MySQL uses the CREATE TEMPORARY TABLE statement to create a temporary table.
- This statement can only be used when the MySQL server has the CREATE TEMPORARY TABLES privilege.
- It can be visible and accessible to the client who creates it, which means two different clients can use the temporary tables with the same name without conflicting with each other.

Temporary Tables

- A temporary table in MySQL will be dropped automatically when the user closes the session or terminates the connection manually.
- A temporary table can be created by the user with the same name as a normal table in a database.

Temporary Tables

- In MySQL, the syntax of creating a temporary table is the same as the syntax of creating a normal table statement except the TEMPORARY keyword.
- Let us see the following statement which creates the temporary table:
  - `mysql> CREATE TEMPORARY TABLE table_name ( column_1, column_2, ..., table_constraints);`

Temporary Tables

- If the user wants to create a temporary table whose structure is the same as an existing table in the database, then the above statement cannot be used. Instead, we use the syntax as given below:
  - `mysql> CREATE TEMPORARY TABLE temporary_table_name SELECT * FROM original_table_name LIMIT 0;`

Temporary Tables

- Let us understand how we can create a temporary table in MySQL.
- Execute the following statement that creates a temporary table in the selected database:
  - `mysql> CREATE TEMPORARY TABLE Students( student_name VARCHAR(40) NOT NULL, total_marks DECIMAL(12,2) NOT NULL DEFAULT 0.00, total_subjects INT UNSIGNED NOT NULL DEFAULT 0);`

Temporary Tables

- Next, we need to insert values in the temporary table:

```
mysql>INSERT INTO Students(student_name,
total_marks, total_subjects) VALUES
('Joseph', 150.75, 2), ('Peter', 180.75,
2);
```

Temporary Tables

- Here, the structure of a temporary table is created by using the SELECT statement and merge two tables using the INNER JOIN clause and sorts them based on the price.
- Write the following statement in the MySQL prompt:

```
CREATE TEMPORARY TABLE temp_customers
SELECT c.cust_name, c.city, o.prod_name, o.price
FROM orders o
INNER JOIN customer c ON c.cust_id = o.order_id
ORDER BY o.price DESC;
```

Drop Temporary Tables

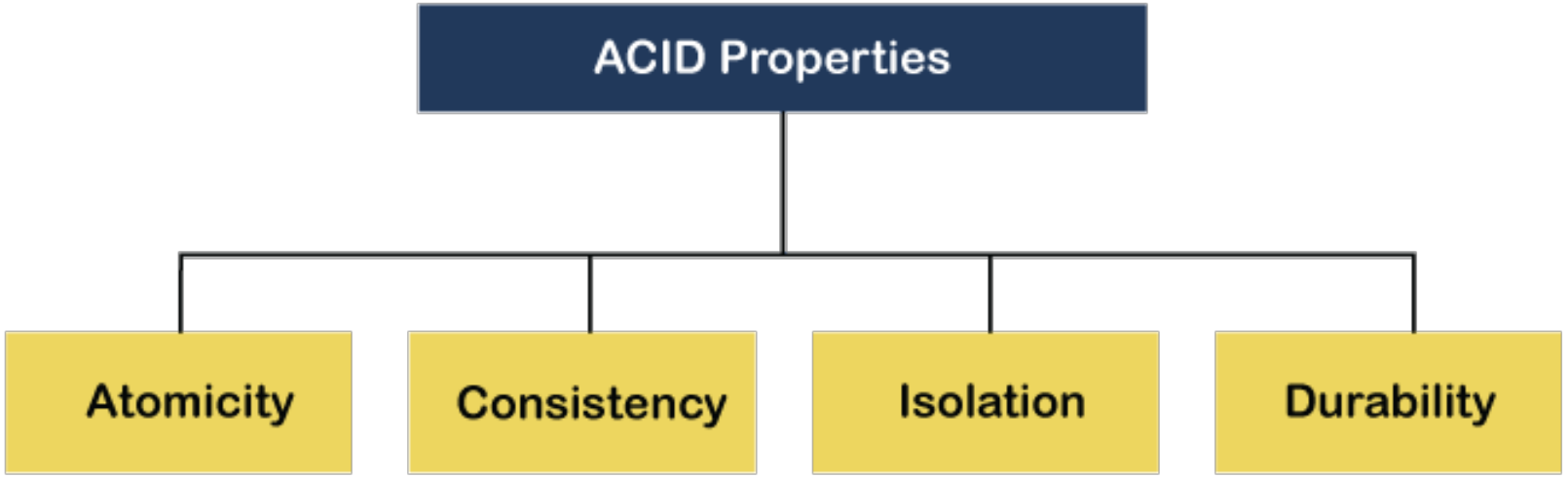
```
mysql> DROP TEMPORARY TABLE
table_name;
```

- This query will not remove a permanent table of the database that means it only deletes a temporary table.

ACID

- DBMS is the management of data that should remain integrated when any changes are done in it.
- It is because if the integrity of the data is affected, whole data will get disturbed and corrupted.
- Therefore, to maintain the integrity of the data, there are four properties described in the database management system, which are known as the ACID properties.
- The ACID properties are meant for the transaction that goes through a different group of tasks, and there we come to see the role of the ACID properties.

ACID


ACID Properties
Atomicity
Consistency
Isolation
Durability

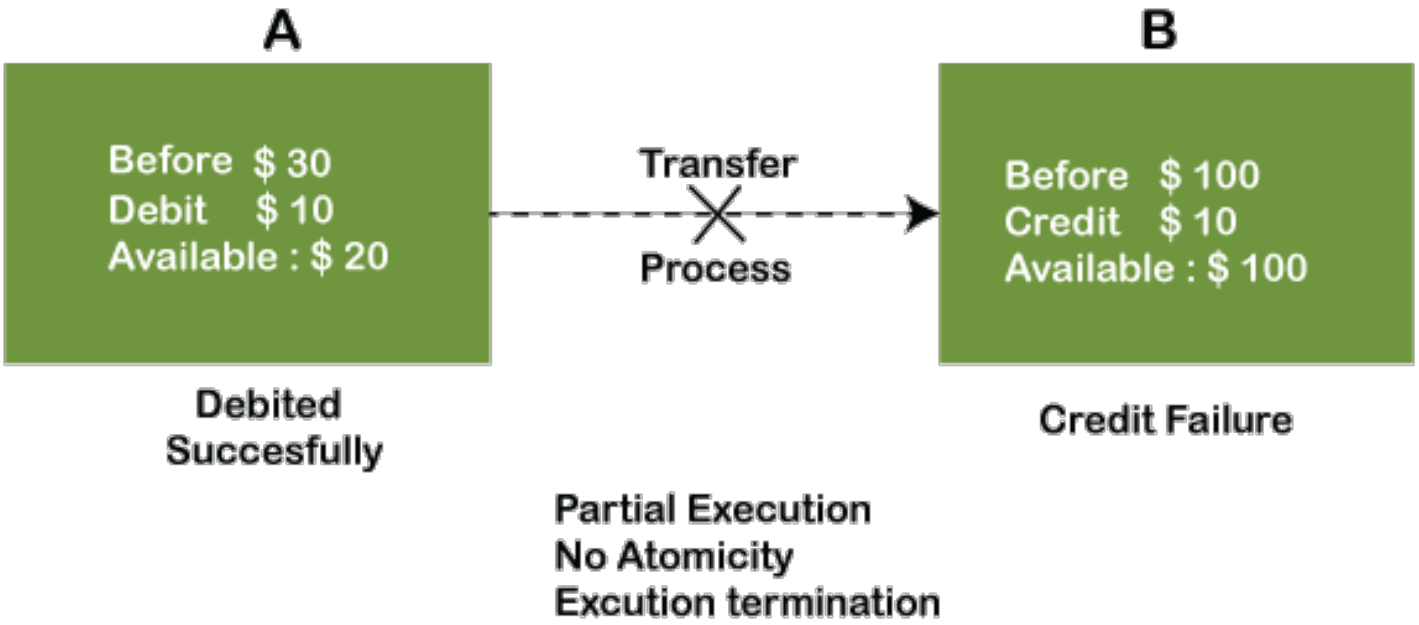
Atomicity

- The term atomicity defines that the data remains atomic.
- It means if any operation is performed on the data, either it should be performed or executed completely or should not be executed at all.
- It further means that the operation should not break in between or execute partially.
- In the case of executing operations on the transaction, the operation should be completely executed and not partially.

Atomicity

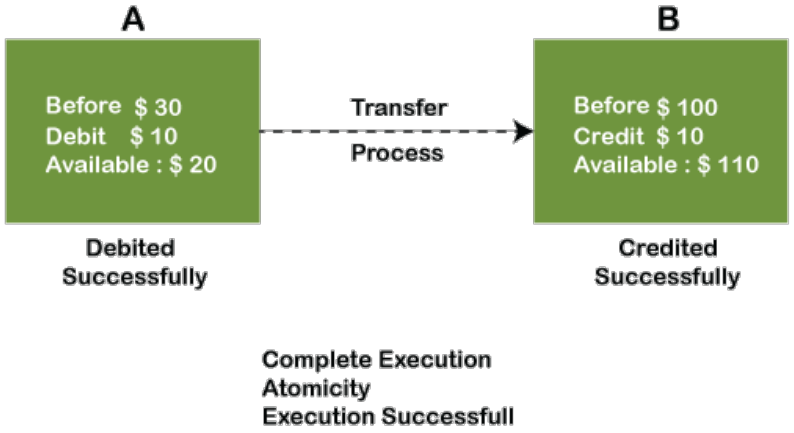
- If Remo has account A having $30 in his account from which he wishes to send $10 to Sheero's account, which is B. In account B, a sum of $ 100 is already present.
- When $10 will be transferred to account B, the sum will become $110. Now, there will be two operations that will take place.
- One is the amount of $10 that Remo wants to transfer will be debited from his account A, and the same amount will get credited to account B, i.e., into Sheero's account.
- Now, what happens - the first operation of debit executes successfully, but the credit operation, however, fails.
- Thus, in Remo's account A, the value becomes $20, and to that of Sheero's account, it remains $100 as it was previously present.

Atomicity


A
Before $ 30
Debit $ 10
Available : $ 20
Transfer
Process
B
Before $ 100
Credit $ 10
Available : $ 100
Debited
Succesfully
Credit Failure
Partial Execution
No Atomicity
Excution termination

Atomicity

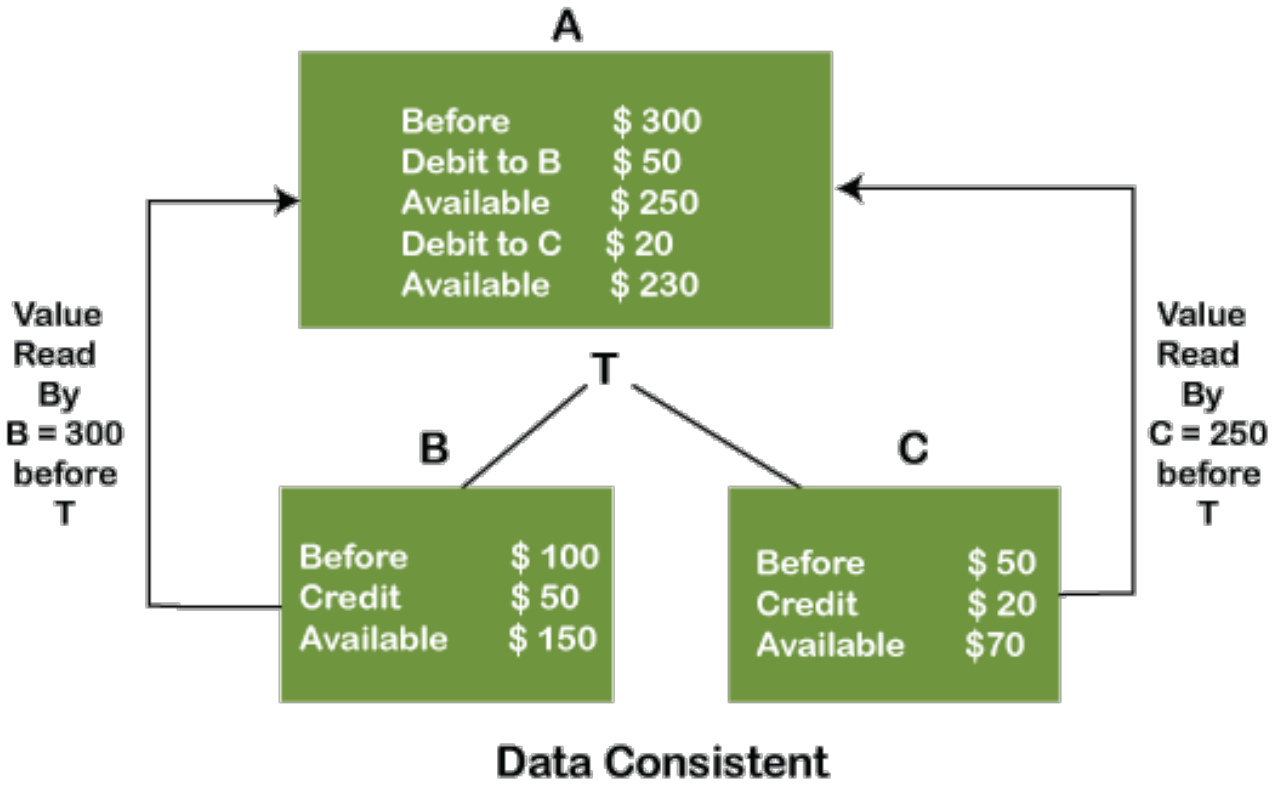
- it can be seen that after crediting $10, the amount is still $100 in account B. So, it is not an atomic transaction.
- The below image shows that both debit and credit operations are done successfully. Thus the transaction is atomic.



Consistency

- The word consistency means that the value should remain preserved always.
- In DBMS, the integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always.
- In the case of transactions, the integrity of the data is very essential so that the database remains consistent before and after the transaction. The data should always be correct.

Consistency



Consistency

- there are three accounts, A, B, and C, where A is making a transaction T one by one to both B & C. There are two operations that take place, i.e., Debit and Credit.
- Account A firstly debits $50 to account B, and the amount in account A is read $300 by B before the transaction. After the successful transaction T, the available amount in B becomes $150.
- Now, A debits $20 to account C, and that time, the value read by C is $250 (that is correct as a debit of $50 has been successfully done to B). The debit and credit operation from account A to C has been done successfully.
- We can see that the transaction is done successfully, and the value is also read correctly. Thus, the data is consistent. In case the value read by B and C is $300, which means that data is inconsistent because when the debit operation executes, it will not be consistent.

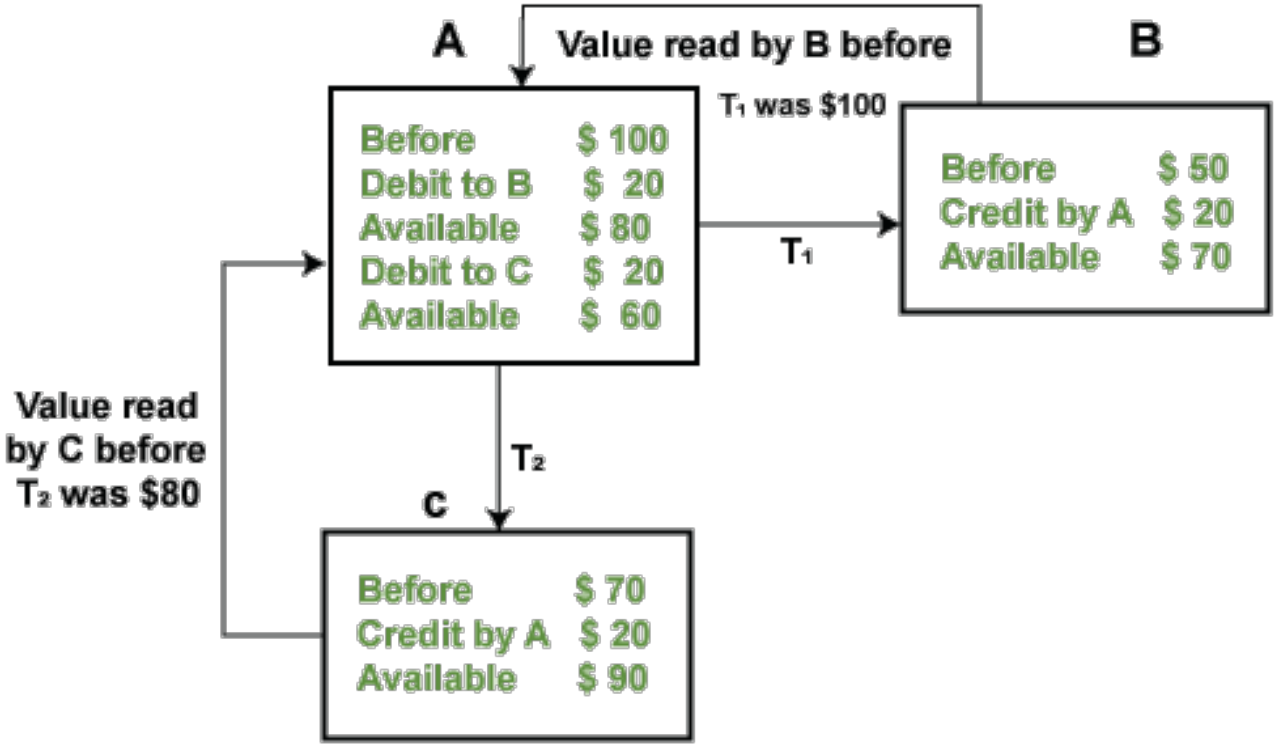
Isolation

- The term 'isolation' means separation. In DBMS, Isolation is the property of a database where no data should affect the other one and may occur concurrently.
- In short, the operation on one database should begin when the operation on the first database gets complete. It means if two operations are being performed on two different databases, they may not affect the value of one another.
- In the case of transactions, when two or more transactions occur simultaneously, the consistency should remain maintained.
- Any changes that occur in any particular transaction will not be seen by other transactions until the change is not committed in the memory.

Isolation

- If two operations are concurrently running on two different accounts, then the value of both accounts should not get affected.
- The value should remain persistent. As you can see in the below diagram, account A is making T1 and T2 transactions to account B and C, but both are executing independently without affecting each other. It is known as Isolation.

Isolation



Isolation - Independent execution of T_1 & T_2 by A

Durability

- Durability ensures the permanency of something. In DBMS, the term durability ensures that the data after the successful execution of the operation becomes permanent in the database.
- The durability of the data should be so perfect that even if the system fails or leads to a crash, the database still survives.
- However, if gets lost, it becomes the responsibility of the recovery manager for ensuring the durability of the database.
- For committing the values, the COMMIT command must be used every time we make changes.

Data model Schema and Instance

- The data which is stored in the database at a particular moment of time is called an instance of the database.
- The overall design of a database is called schema.
- A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
- A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.

Data model Schema and Instance

- A database schema can be represented by using the visual diagram.
- That diagram shows the database objects and relationship with each other.
- A database schema is designed by the database designers to help programmers whose software will interact with the database.
- The process of database creation is called data modeling.

Data model Schema and Instance

- A schema diagram can display only some aspects of a schema like the name of record type, data type, and constraints. Other aspects can't be specified through the schema diagram.
  - For example, the given figure neither show the data type of each data item nor the relationship among various files.
- In the database, actual data changes quite frequently.
  - For example, in the given figure, the database changes whenever we add a new grade or add a student. The data at a particular moment of time is called the instance of the database.

Data model Schema and Instance

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Storage Engines

- MySQL table types/storage engines are essential features that can be used effectively for maximizing the database's performance.
- It handles create, read, and update operations for storing and managing the information in a database.

Storage Engines

- The following are various table types/storage engines supports in MySQL:
  - ISAM
  - MyISAM
  - MERGE
  - InnoDB
  - MEMORY (HEAP)
  - ARCHIVE
  - BDB
  - CSV
  - FEDERATED

Storage Engines

- We can use the below query that determines which table types/storage engines our MySQL server supports.

  mysql> SHOW ENGINES;

ISAM Table

- It is abbreviated as Indexed Sequential Access Method.
- This table type/storage engine has been deprecated and removed from MySQL version 5.x.
- MyISAM now replaces the functionalities of this table.
- The size of an ISAM table is 4 GB, which requires expensive hardware. It is not portable.

MyISAM Table

- It is an extension of the ISAM storage engine. The MyISAM table types are optimized for compression and speed and can be easily portable between system to system.
- Before version 5.5, if we do not specify the table type during table creation, it was the default storage engine. From version 5.x, InnoDB is used as the default table type/storage engine.
- The MyISAM table size is dependent on the OS and can be up to 256 TB. It can be compressed into read-only tables that save spaces in memory.
- MyISAM table type can store 64 keys per table and contains 1024 bytes maximum key length.
- The MyISAM tables work very fast, but they are not transaction-safe.

MyISAM Table

- Advantages of MyISAM
  - If you are new, it will be best to start with MyISAM because it is simple to design and create.
  - It is faster than other storage engines in general conditions.
  - It provides full-text indexing/searching.
- Disadvantages of MyISAM
  - MyISAM tables are not transactions-safe.
  - It has poor data integrity and crash recovery.
  - When we lock the entire table, in that case, MyISAM is slower than InnoDB.

InnoDB Table

- The InnoDB tables in MySQL fully support transaction-safe storage engine with ACID-compliant.
- It is the first table type that supports foreign keys. The InnoDB tables also provide optimal performance. Its size can be up to 64TB.
- InnoDB tables are also portable between systems to systems similar to MyISAM. InnoDB tables can also be checked and repairs by MySQL whenever necessary.

InnoDB Table

- Advantages of InnoDB
  - InnoDB provides optimal performance while processing a large amount of data.
  - InnoDB tables arrange our data on the disk based on the primary key.
- Disadvantages of InnoDB
  - InnoDB tables take more space on disk in comparison with MyISAM.

MERGE Table

- MERGE table is also known as MRG_MyISAM. This table combines multiple MyISAM tables with a similar structure (identical column and index information with the same order) into a single table.
- This table uses indexes of the component tables because it does not have its own indexes. When we join multiple tables, it can also be used to speed up the database's performance.
- We can perform only INSERT, SELECT, DELETE, and UPDATE operations on the MERGE tables.
- If we use the DROP TABLE query in this storage engine, MySQL only removed the MERGE specification, and the underlying tables cannot be affected.

MERGE Table

- Advantages of MERGE
  - The main advantage of this table is to remove the size limitation from MyISAM tables.
  - It performs more efficient searches and repairs.
  - It manages the set of log tables easily.
- Disadvantages of MERGE
  - MySQL allows us to use only identical (similar structure) MyISAM tables for the MERGE table.
  - It cannot support all MyISAM features, such as we cannot create FULLTEXT indexes on MERGE tables.
  - It reads indexes slower.

Memory Table

- The memory table type/storage engine creates tables, which will be stored in our memory. It is also known as HEAP before MySQL version 4.1.
- This table type is faster than MyISAM because it uses hash indexes that retrieve results faster. We already know that data stored in memory can be crashed due to power issues or hardware failure.
- Therefore, we can only use this table as temporary work areas or read-only caches for data pulled from other tables.
- Hence, the memory/heap tables will be lost whenever the MySQL server halts or restarts. The memory/heap table's data life depends on the uptime of the database server.

Memory Table

- Advantages of Memory
  - The main advantage of this table type is its speed, which is very fast. It is because it uses hash indexing that retrieves results faster.
- Disadvantages of Memory
  - It is not a good idea to use the MEMORY storage for the long term because the data would be lost easily as soon as the power failure or hardware crash.

Memory Table

- The CSV table type/storage engine stores data in comma-separated values in a file.
- It provides a convenient way to migrate data into many different software packages, such as spreadsheet software.
- This table type is not as good as a general database engine; however, it enables us to exchange our data most effectively and easily.
- In addition, it will scan the whole table during the read operation.

Memory Table

- Advantages of CSV
  - This table type/storage engine is advantageous when we need to export complex data from one application to a CSV file and then import it into another application.
- Disadvantages of CSV
  - It is not good to store a large volume of data or larger data types like BLOB, although such types are supported.
  - It makes data retrieval slow because there is no indexing.

FEDERATED Table

- The FEDERATED table type/storage engine supports MySQL from version 5.03 that allows access data from a remote MySQL server without using the cluster/replication technology.
- The federated storage engine located in local storage does not store any data. If we will query data from a federated table stored in local memory, MySQL automatically pulled data from the remote federated tables.
- It is to note that it's a way for a server, not for a client, for accessing a remote database.
- It is an effective way to combine data from more than one host or copy data from remote databases into local tables without using the data import and export method.

Archieve Table

- This table type/storage engine allows us to store a large volume of data in a compressed format to save disk space and cannot be modified.
- Thus, it is the perfect storage engine to store log data that is no longer in active use, such as the old invoice or sales data.
- It compresses the data during the insertion and can decompress it using the Zlib library.
- The archive tables only support INSERT and SELECT queries. It does not support most of the data types, such as index data type, without which we need to scan a full table for reading rows.

BDB Table

- BDB stands for the Berkeley DB engine, which is developed by SleepyCat software. It is similar to InnoDB in the transaction-safe.
- It is based on the hash storage mechanism that makes the recovery of information very quickly.
- It supports page-level locking, but the data file is not portable.

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com