**TRANSACTION IN DATABASE**

Transaction is a group of logically related operations that must be executed as a whole. It's an all-or-nothing scenario ensuring consistency and reliability. It is a result of a request made by a user to access the content of a database and perform operations on it. Transactions can consist of various operations and states in a completion process. There should be certain properties that must be followed to keep the data consistent.

**IMPORTANCE OF TRANSACTIONS**

* **Data Integrity:** By enforcing atomicity and consistency, transactions help ensure the data remains accurate and consistent even when multiple users access and modify it simultaneously.
* **Error Handling:** Facilitating [error handling](https://www.dremio.com/wiki/error-handling/) by allowing transactions to be rolled back if a problem occurs.
* **Concurrency Control:** Transactions help manage concurrent access to data by enforcing isolation. This prevents conflicts and ensures that each transaction sees a consistent and valid snapshot of the data.

**OPERATIONS OF TRANSACTIONS**

There are different types of operation that relate to transactions. They are discussed as follows;

1. **Read(X)**: A read operation is used to retrieve the value of X from the database and store it in a buffer in the main memory for further actions, such as displaying that value. This operation is performed when a user only wants to view the content of the database without making any changes. For example, when a user wants to check their account balance, a read operation would be performed to retrieve the account balance from the database.
2. **Write(X):** A write operation modifies data in the buffer and then stores it back in the database. For example, when a user withdraws money, the account balance is fetched using a read operation, modified, and then stored back using a write operation.
3. **Commit:** The commit operation in transactions is used to maintain the integrity of the database. If a transaction is interrupted due to power, hardware, or software failure before all its operations are completed, it can lead to inconsistency in the database. To ensure that other transactions are only performed after the current transaction is completed, the commit operation permanently applies the changes made by a transaction to the database.
4. **Rollback:**  This operation is performed to revert the database to its last saved state when a transaction is interrupted due to power, hardware, or software failure. In simple terms, a rollback operation undoes the transactions that were performed before the interruption to ensure the database is in a safe state and to prevent any ambiguity or inconsistency.

**Transaction Schedule**

When we have multiple requests made at the same time we need to determine their order of execution. Transaction schedule can be defined as a chronological (a way of organizing based on their sequence in time)order of executing multiple transactions.

There are two types of transaction schedule:

1. **Serial Schedule**: In this type of schedule, when multiple transactions are to be executed, they are carried out one by one. This means that at any given time, only one transaction is executed while the others wait for the current transaction to be completed. This approach ensures consistency in the database, as transactions do not occur simultaneously.

But it increases the waiting time of the transactions in the queue, which in turn lowers the throughput of the system, i.e. the number of transactions executed per time.

1. **Non-Serial Schedule:** In order to reduce transaction waiting times and improve system efficiency, we utilize non-serial schedules, which allow multiple transactions to start before a transaction is completely executed. However, this approach can lead to inconsistency and errors in database operation. To address this, specific algorithms are employed to maintain database consistency and enhance CPU throughput. Non-serial schedules are also known as parallel schedules, as transactions execute in parallel under this approach.

**Serializable:** Serializability in DBMS is the property of a non-serial schedule that determines whether it would maintain the consistency of the database or not.

Non-serial schedule that guarantees database consistency after executing transactions in the specified order is called a Serializable Schedule. Serial schedules maintain database consistency by allowing a transaction to start only after the completion of the preceding transaction. Therefore, serial schedules are always serializable.

A transaction is a series of operations, so various states occur during its completion journey. These states are discussed as follows:

* **Active:** The initial stage of any transaction is known as the execution stage. This is when the transaction begins to be carried out. Operations like insertion, deletion, or updating of data records are performed during this stage. It's important to note that the data records are manipulated in the main memory buffer during this stage and are not immediately saved to the database.
* **Partially Committed:** The transaction reaches this state when it has completed most of its operations and is now executing its final operation. This state can serve as a signal to initiate the commit operation. After the final operation of the transaction is completed, the data needs to be saved to the database through the commit operation. If an error occurs during this state, the transaction enters a failed state; otherwise, it transitions to the Committed state.
* **Committed:** The state of a transaction is achieved when all the transaction-related operations have been executed successfully, along with the Commit operation. This means that the data has been saved into the database after the required manipulations, marking the successful completion of the transaction.
* **Failed**: If any of the transaction-related operations result in an error while the transaction is active or partially committed, the transaction is halted and moved to a failed state. In this situation, the database recovery system ensures that the database remains in a consistent state.
* **Aborted**: If an error is not resolved while in a failed state, the transaction is aborted, and a rollback operation is performed to bring the database back to its last saved consistent state. When the transaction is aborted, the database recovery module will either restart the transaction or terminate it.

**Properties of Transaction**

There are four properties which can be denote with the acronym ACID which stand for Atomicity, Consistency, Isolation, Durability

* **Atomicity:**  This concept ensures that all operations within a transaction are either fully executed or completely undone. In other words, a transaction cannot be partially completed. Each transaction is treated as a single unit, similar to an atom. Atomicity is achieved through commit and rollback operations. This means that changes are only made to the database if all operations related to a transaction are successfully completed. If a transaction is interrupted, any changes made are rolled back using the rollback operation to return the database to its previous state.
* **Consistency:**  This property of a transaction ensures that the database remains consistent before and after the transaction is completed. Any transaction must guarantee that after it is executed, the database is either in its previous stable state or a new stable state. In other words, the outcome of a transaction should be the transformation of the database from one consistent state to another consistent state. Consistency, in this context, means that the changes made in the database result only from logical operations that the user intended to perform, without any ambiguity.
* **Isolation:** In a database system where multiple transactions are executed simultaneously and in parallel, the isolation property ensures that each transaction is carried out as if it is the only transaction in the system. This means that no transaction will impact the existence of any other transaction.
* **Durability:** This property ensures that any changes made to the database after a transaction is fully executed are permanent. It signifies that successful execution of a transaction results in lasting changes. In the event of system failures or crashes, the consistent state achieved after the completion of a transaction remains unchanged. The recovery subsystem of the DBMS is responsible for enforcing this property.

**SQL Transaction Control Commands**

* **COMMIT:**  To save changes.
* **ROLLBACK:** To undo changes.
* **SAVEPOINT** − creates points within the groups of transactions in which to ROLLBACK.
* **SET TRANSACTION** − Place a name on a transaction.