SQL NOTES

ACID PROPERTIES

* Database Management Systems are an important part of any organization. Understanding the different properties and functions of DBMS is critical for the effective use of the system. Among the most important properties of DBMS are the ACID (Atomicity, Consistency, Isolation, and Durability) properties.
* ACID properties are central to concurrent transaction processing in a DBMS, data integrity, prioritization of transaction execution, and sharing of data in a multi-user environment.

The database frequently requires updating and modification. For this purpose, a set of queries (called transactions) is written and executed. Any changes in the stored data must be consistent with the database and must be well integrated without affecting the entire database.

The ACID properties of the database management system are responsible for ensuring the successful execution of transactions.   
The ACID full form in DBMS is ‘A’ for Atomicity, ‘C’ for Consistency, ‘I’ for Isolation, and ‘D’ for Durability. Each of these properties ensures the reliability of transactions (a group of operations) in the database.

### 1. Atomicity

This property ensures that all operations in a transaction are either all completed or none of them are completed. This means that any transaction does not execute partially. There are two operations involved to ensure atomicity:

* **Abort:** The changes made in the data are not reflected in the database if the transaction aborts.
* **Commit:**The changes made in the data are reflected and visible in the database when the transaction is fully completed.

Consider the following example that shows the failure and success of atomic transactions.

Let’s consider a scenario involving X and Y’s bank accounts. X has $50 in his account (Account A) and wants to transfer $10 to Y’s account (Account B), which already has a balance of $100.

In this transaction, there are two essential steps:

* Deducting $10 from X’s account (Account A).
* Crediting $10 to Y’s account (Account B).

Now, the following events occur during the transaction:

* The deduction of $10 from Y’s account (Account A) proceeds successfully.
* However, the attempt to credit the same $10 to Y’s account (Account B) encounters an issue and fails to complete it.

As a result of this partial transaction, X’s account (Account A) is left with a balance of $40, while Y’s account (Account B) retains its previous balance of $100. This situation illustrates the challenge of ensuring that both debit and credit operations are treated as a single atomic unit to maintain the integrity and consistency of the accounts.

When both the debit and credit events are completed as a single transaction, it is considered successful.

### 2. Consistency

This ACID property of the DBMS ensures that any transaction made is valid according to all defined rules and does not violate any constraints in the database system. This property also ensures that the data remains consistent and correct across the database, even if multiple transactions are being executed simultaneously.

For example,

When updating the balance of a bank account, it’s crucial to ensure that the new balance value adheres to the bank’s rules. Let’s consider three bank accounts: A, B, and C. Account A conducts a transaction (T) in which it debits and credits Accounts B and C. Suppose Account A has a balance of $500, Account B has $350, and Account C has $200.

Firstly, Account A credits $50 to Account B, which results in Account B having a balance of $400. Secondly, Account A transfers $100 to Account C, which results in Account C having a balance of $300. Before each debit transaction from Account A, both Account B and C check the balance amount in Account A, and it is correct.

This consistency ensures that the transaction is executed successfully and the data remains consistent. Nevertheless, if B and C read the same balance of $500 in Account A, it would indicate a data inconsistency since the debit operation would not be consistent.

### 3. Isolation

This property of ACID ensures that concurrent transactions do not interfere with each other and that all operations in a transaction are treated as if they were executed sequentially. This means if multiple transactions occur simultaneously, each transaction completes separately and does not affect the other transactions. It also ensures that any changes made by a transaction are permanent and are not affected by other transactions.

For example,

Consider a banking system with two accounts, A and B, and two transactions, T1 and T2, executed simultaneously. With T1 transferring money from account A to account B and T2 transferring money from account B to account A, the isolation property ensures that these two transactions run simultaneously, neither transaction is affected by the other.

In other words, the balances on both accounts end up being the same as before the two transactions were executed. This means that until the change is committed to memory, any modifications made by one transaction won’t be visible to other transactions.

### 4. Durability

This property ensures that once a transaction is committed, its effects are permanent and will not be lost due to a system failure. It is achieved by writing the changes to a [log file](https://www.ibm.com/docs/en/imdm/11.6?topic=support-log-files) and to the database itself. The log file serves as a backup to recover from any system failure, and the changes in the database are written in the same order as they are performed in the transaction.

This ensures that the changes are reflected in the database, even after a system failure. Note that the COMMIT command in DBMS must be used every time a change is made in the database.

For example,

Suppose that a customer books a train ticket using the railway management system. After the transaction is committed, the ticket details are written to a log file and stored in the database. Even if the system fails, the ticket details will not be lost because the changes are stored in the log file. When the system is restored, the log file can be used to recover the data, and the customer can still board the train.