**ACID PROPERTIES**

In computer science, ACID (Atomicity, Consistency, Isolation, Durability) is a set of properties of database transactions intended to guarantee validity even in the event of errors, power failures, etc. In the context of databases, a sequence of database operations that satisfies the ACID properties (and these can be perceived as a single logical operation on the data) is called a transaction. For example, a transfer of funds from one bank account to another, even involving multiple changes such as debiting one account and crediting another, is a single transaction.

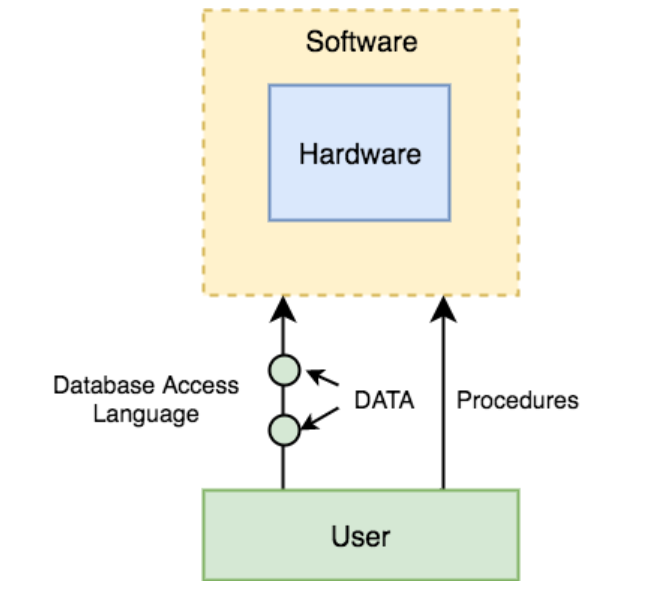
**Atomicity**: Transactions are often composed of multiple statements. Atomicity guarantees that each transaction is treated as a single "unit", which either succeeds completely, or fails completely: if any of the statements constituting a transaction fails to complete, the entire transaction fails, and the database is left unchanged. An atomic system must guarantee atomicity in each and every situation, including power failures, errors and crashes.

**Consistency**: Consistency ensures that a transaction can only bring the database from one valid state to another, maintaining database invariants: any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof. This prevents database corruption by an illegal transaction but does not guarantee that a transaction is correct.

**Isolation**: Transactions are often executed concurrently (e.g., reading and writing to multiple tables at the same time). Isolation ensures that concurrent execution of transactions leaves the database in the same state that would have been obtained if the transactions were executed sequentially. Isolation is the main goal of concurrency control; depending on the method used, the effects of an incomplete transaction might not even be visible to other transactions.

**Durability**: Durability guarantees that once a transaction has been committed, it will remain committed even in the case of a system failure (e.g., power outage or crash). This usually means that completed transactions (or their effects) are recorded in non-volatile memory.

**COMPONENTS OF DBMS**



The database management system can be divided into five major components, they are:

DBMS Components: **Hardware**

When we say Hardware, we mean computer, hard disks, I/O channels for data, and any other physical component involved before any data is successfully stored into the memory. When we run Oracle or MySQL on our personal computer, then our computer's Hard Disk, our Keyboard using which we type in all the commands, our computer's RAM, ROM all become a part of the DBMS hardware.

DBMS Components: **Software**

This is the main component, as this is the program which controls everything. The DBMS software is more like a wrapper around the physical database, which provides us with an easy-to-use interface to store, access and update data. The DBMS software is capable of understanding the Database Access Language and interpret it into actual database commands to execute them on the DB.

DBMS Components: **Data**

Data is that resource, for which DBMS was designed. The motive behind the creation of DBMS was to store and utilize data. In a typical Database, the user saved Data is present and meta data is stored. Metadata is data about the data. This is information stored by the DBMS to better understand the data stored in it.

For example: When I store my Name in a database, the DBMS will store when the name was stored in the database, what is the size of the name, is it stored as related data to some other data, or is it independent, all this information is metadata.

DBMS Components: **Procedures**

Procedures refer to general instructions to use a database management system. This includes procedures to setup and install a DBMS, to login and logout of DBMS software, to manage databases, to take backups, generating reports etc.

DBMS Components**: Database Access Language**

Database Access Language is a simple language designed to write commands to access, insert, update and delete data stored in any database. A user can write commands in the Database Access Language and submit it to the DBMS for execution, which is then translated and executed by the DBMS. User can create new databases, tables, insert data, fetch stored data, update data and delete the data using the access language.

**TRIGGERS IN DATABASE.**

In a DBMS, a trigger is a SQL procedure that initiates an action (i.e., fires an action) when an event (INSERT, DELETE or UPDATE) occurs. Since triggers are event-driven specialized procedures, they are stored in and managed by the DBMS. A trigger cannot be called or executed; the DBMS automatically fires the trigger as a result of a data modification to the associated table. Triggers are used to maintain the referential integrity of data by changing the data in a systematic fashion. Each trigger is attached to a single, specified table in the database.

Triggers can be viewed as similar to stored procedures in that both consist of procedural logic that is stored at the database level. Stored procedures, however, are not event-drive and are not attached to a specific table as triggers are. Stored procedures are explicitly executed by invoking a CALL to the procedure while triggers are implicitly executed. In addition, triggers can also execute stored procedures.

A trigger can also contain INSERT, UPDATE and DELETE logic within itself, so when the trigger is fired because of data modification it can also cause another data modification, thereby firing another trigger. A trigger that contains data modification logic within itself is called a nested trigger.

