ACID properties refer to a set of fundamental guarantees provided to ensure the reliability and consistency of data transactions. ACID stands for Atomicity, Consistency, Isolation, and Durability.

ACID Properties in DBMS

Atomicity ensures that a transaction is treated as a single indivisible unit, either executing all its operations or none at all.

Consistency ensures that the database remains in a valid state before and after a transaction.

Isolation ensures that concurrent transactions do not interfere with each other, maintaining data integrity.

Durability guarantees that once a transaction is committed, its effects are permanent and survive any system failures. Together, these properties ensure reliability and maintain data integrity in DBMS operations.

What is Database Transaction?
A transaction is a logical unit of work that accesses and updates the contents of a database. Read and write operations are used by transactions to access data. A transaction has several states:

Transaction State of Acid properties in DBMS

State Description
Active Transactions are in progress.
Partially Committed Operations completed, but data not yet saved.
Failed Transaction fails database recovery system checks.
Committed Successful completion, changes permanently saved.
Aborted Transaction fails tests, rolled back or aborted.
Terminated Transaction terminated, system ready for new transactions.
What are ACID Properties in DBMS?

ACID properties are a set of properties that guarantee reliable processing of transactions in a database management system (DBMS). Transactions are a sequence of database operations that are executed as a single unit of work, and the ACID properties ensure that transactions are processed reliably and consistently in a DBMS.

The Atomicity property ensures that a transaction is either executed completely or not at all.
The Consistency property ensures that the database remains in a consistent state before and after a transaction.
The Isolation property ensures that multiple transactions can run concurrently without interfering with each other.
The Durability property ensures that the results of a committed transaction are permanent and cannot be lost due to system failure.
Together, these properties ensure that transactions are processed reliably and consistently in a DBMS, which is essential for the integrity and accuracy of data in a database.

1. Atomicity in DBMS
The term atomicity is the ACID Property in DBMS that refers to the fact that the data is kept atomic. It means that if any operation on the data is conducted, it should either be executed completely or not at all. It also implies that the operation should not be interrupted or just half completed. When performing operations on a transaction, the operation should be completed totally rather than partially. If any of the operations aren't completed fully, the transaction gets aborted.

Example Sometimes, a current operation will be running and then, an operation with a higher priority enters. This discontinues the current operation and the current operation will be aborted.

In the given scenario, if two users simultaneously try to book the only available seat on a train, the transaction is considered incomplete. According to atomicity, the first user who successfully clicks the booking button will

reserve the seat and receive a notification, while the second user's transaction will be rolled back, and they will be notified that no more seats are available.

In a simpler example, if a person tries to book a ticket, selects a seat, and proceeds to the payment gateway but encounters a failure due to bank server issues, their booked seat will not be reserved for them. A complete transaction involves reserving the seat and completing the payment. If any step fails, the operation is aborted, and the user is brought back to the initial state without their seat being reserved.

Atomicity in DBMS is often referred to as the ‘all or nothing’ rule.

2. Consistency in DBMS
This ACID Property will verify the total sum of seats left in the train + sum of seats booked by users = total the number of seats present in the train. After each transaction, consistency is checked to ensure nothing has gone wrong.

Example Let us consider an example where one person is trying to book a ticket. They are able to reserve their seat but their payment hasn’t gone through due to bank issues. In this case, their transaction is rolled back. But just doing that isn’t sufficient. The number of available seats must also be updated. Otherwise, if it isn’t updated, there will be an inconsistency where the seat given up by the person is not accounted for. Hence, the total sum of seats left in the train + the sum of seats booked by users would not be equal to the total number of seats present in the train if not for consistency.

3. Isolation in DBMS
Isolation is defined as a state of separation. Isolation is an ACID Property in DBMS where no data from one database should impact the other and where many transactions can take place at the same time. In other words, when the operation on the first state of the database is finished, the process on the second state of the database should begin. It indicates that if two actions are conducted on two different databases, the value of one database may not be affected by the value of the other. When two or more transactions occur at the

same time in the case of transactions, consistency should be maintained. Any modifications made in one transaction will not be visible to other transactions until the change is committed to the memory.

Example Suppose two people try to book the same seat simultaneously. Transactions are serialized to maintain data consistency. The first person's transaction succeeds, and they receive a ticket. The second person's transaction fails as the seat is already booked. They receive an error message indicating no available seats.

4. Durability in DBMS
The ACID Property durability in DBMS refers to the fact that if an operation is completed successfully, the database remains permanent in the disk. The database's durability should be such that even if the system fails or crashes, the database will survive. However, if the database is lost, the recovery manager is responsible for guaranteeing the database's long-term viability. Every time we make a change, we must use the COMMIT command to commit the values.

Example Suppose that there is a system failure in the railway management system resulted in the loss of all booked train details. Millions of users who had paid for their seats are now unable to board the train, causing significant financial losses and eroding trust in the company. The situation is particularly critical as these trains are needed for important reasons, causing widespread panic and inconvenience.

Properties of Transaction in DBMS
There are four major properties that are vital for a transaction to be successful. These are used to maintain state consistency in the database, both before and after the transaction. These are called ACID properties.

Atomicity: This property means that either the transaction takes place completely at once or doesn't happen at all. There is no middle option, i.e.,

transactions do not occur partially. Each transaction is considered as one single step which either runs completely or is not executed at all.

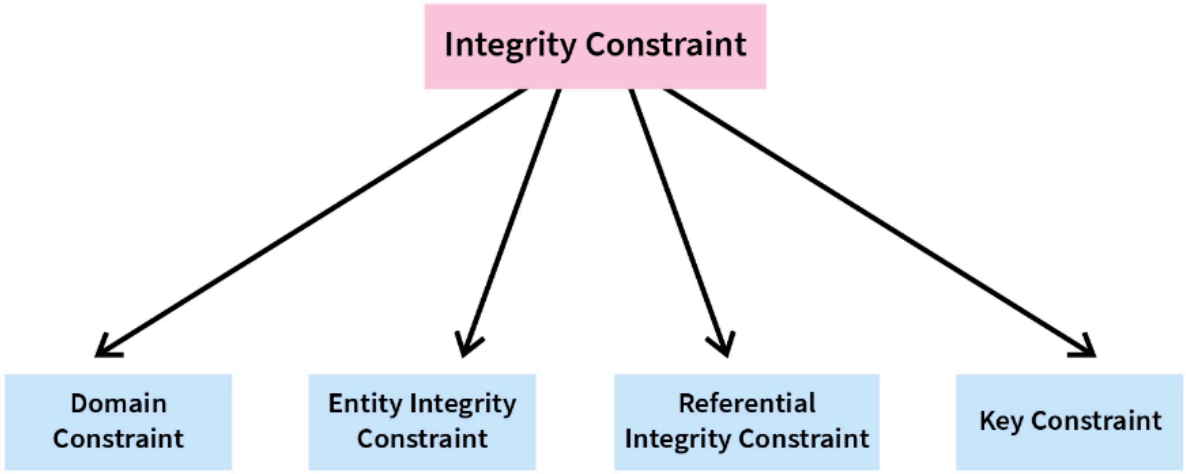
Example) Transactions taking place in a bank (Credit or Debit of Money)

If Marino has an account namely A with $50 in it and wants to send $20 to Amanda who has an account namely B. An amount of $200 is already existing in account B. When $20 is deposited to account B, the total becomes $220.

Two procedures are now scheduled to take place. One is that the $20 that Marino wishes to send will be deducted from his account A and would be credited to account B, i.e., into Amanda's account. What happens now is that the initial debit operation succeeds, but the crediting operation fails.

As a result, the value in Marino's account A becomes 30$, while the value in Amanda's account remains $200 as it was earlier.

Consistency: This property means that the integrity constraints of a database are maintained so that the database is consistent before and after the transaction. It refers to the correctness of a database.
In Database Management Systems, integrity constraints are pre-defined set of rules that are applied on the table fields(columns) or relations to ensure that the overall validity, integrity, and consistency of the data present in the database table is maintained. Evaluation of all the conditions or rules mentioned in the integrity constraint is done every time a table insert, update, delete, or alter operation is performed. The data can be inserted, updated, deleted, or altered only if the result of the constraint comes out to be True. Thus, integrity constraints are useful in preventing any accidental damage to the database by an authorized user.



Isolation in DBMS

Isolation is defined as a state of separation. Isolation is an ACID Property in DBMS where no data from one database should impact the other and where many transactions can take place at the same time. In other words, when the operation on the first state of the database is finished, the process on the second state of the database should begin. It indicates that if two actions are conducted on two different databases, the value of one database may not be affected by the value of the other. When two or more transactions occur at the same time in the case of transactions, consistency should be maintained. Any modifications made in one transaction will not be visible to other transactions until the change is committed to the memory.

Example Suppose two people try to book the same seat simultaneously. Transactions are serialized to maintain data consistency. The first person's transaction succeeds, and they receive a ticket. The second person's transaction fails as the seat is already booked. They receive an error message indicating no available seats.

Durability in DBMS

The ACID Property durability in DBMS refers to the fact that if an operation is completed successfully, the database remains permanent in the disk. The database's durability should be such that even if the system fails or crashes, the database will survive. However, if the database is lost, the recovery

manager is responsible for guaranteeing the database's long-term viability. Every time we make a change, we must use the COMMIT command to commit the values.

Example Suppose that there is a system failure in the railway management system resulted in the loss of all booked train details. Millions of users who had paid for their seats are now unable to board the train, causing significant financial losses and eroding trust in the company. The situation is particularly critical as these trains are needed for important reasons, causing widespread panic and inconvenience.