| **Atomicity** | **All changes to data are performed as if they are a single operation**. That is, **all the changes are performed, or none of them are**.  For example, in an application that transfers funds from one account to another, the atomicity property ensures that, if **a debit is made successfully from one account, the corresponding credit is made to the other account**. |
| --- | --- |
| **Consistency** | **Data is in a consistent state when a transaction starts and when it ends**.  For example, in an application that transfers funds from one account to another, the consistency property ensures that the **total value of funds in both the accounts is the same at the start and end** of each transaction. |
| **Isolation** | **The intermediate state of a transaction is invisible to other transactions**. As a result, transactions that run concurrently appear to be serialised.  For example, in an application that transfers funds from one account to another, the isolation property ensures that **another transaction sees the transferred funds in one account or the other, but not in both, nor in neither**. |
| **Durability** | **After a transaction successfully completes, changes to data persist and are not undone, even in the event of a system failure**.  For example, in an application that transfers funds from one account to another, the durability property ensures that the **changes made to each account will not be reversed**. |