ACID PROPERTIES

By Pavani Korada

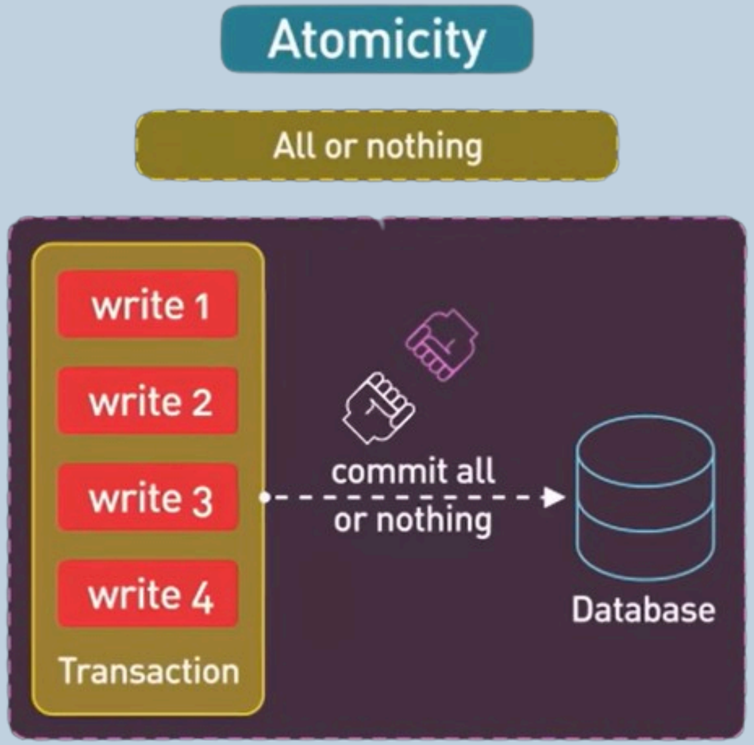
ACID

ACID stands for:

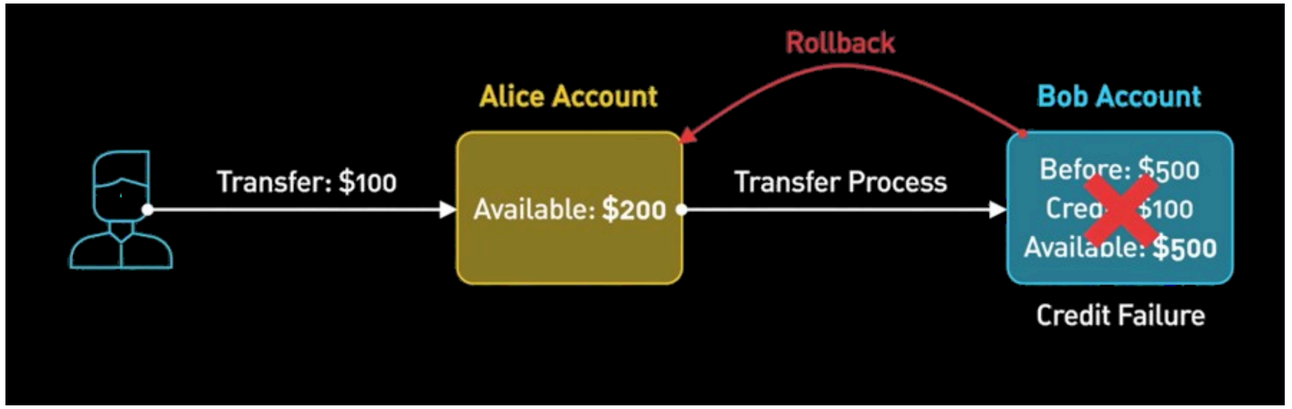
- Atomicity
- Consistency
- Isolation
- Durability

These are the four essential properties that make sure database transactions are correct, complete, and reliable.

ATOMICITY

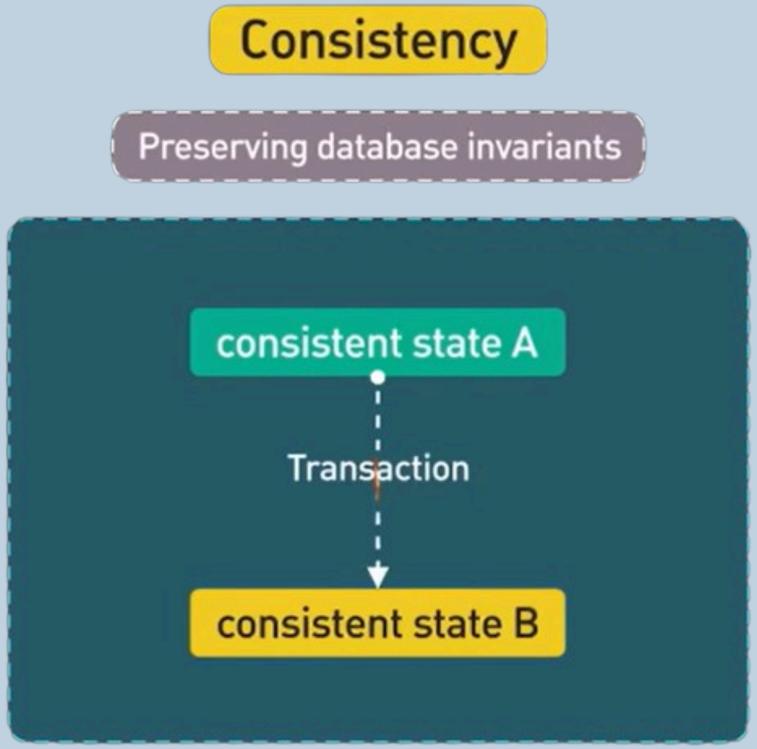


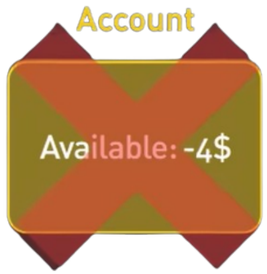
- Ensures a transaction is treated as a single unit.
- Either all operations succeed, or none do.
- No partial changes are left if something fails mid-way.
- **Example**: transferring money from one account to another must debit and credit together, or not happen at all.

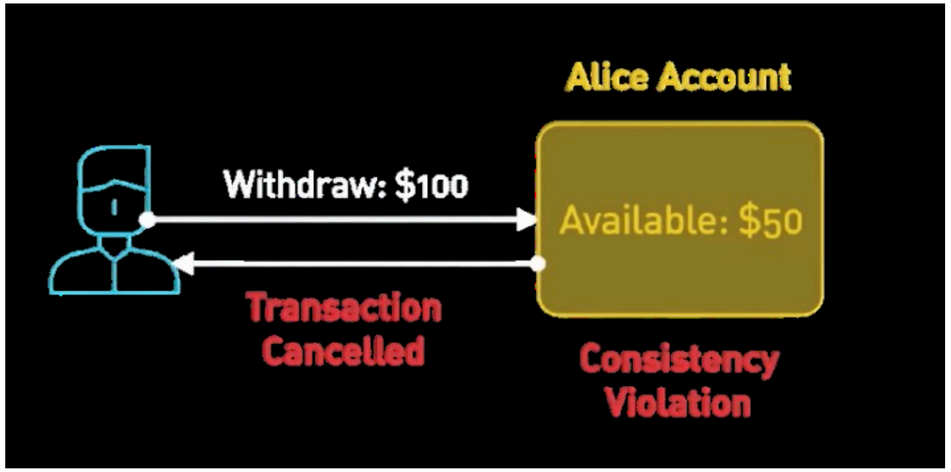


CONSISTENCY

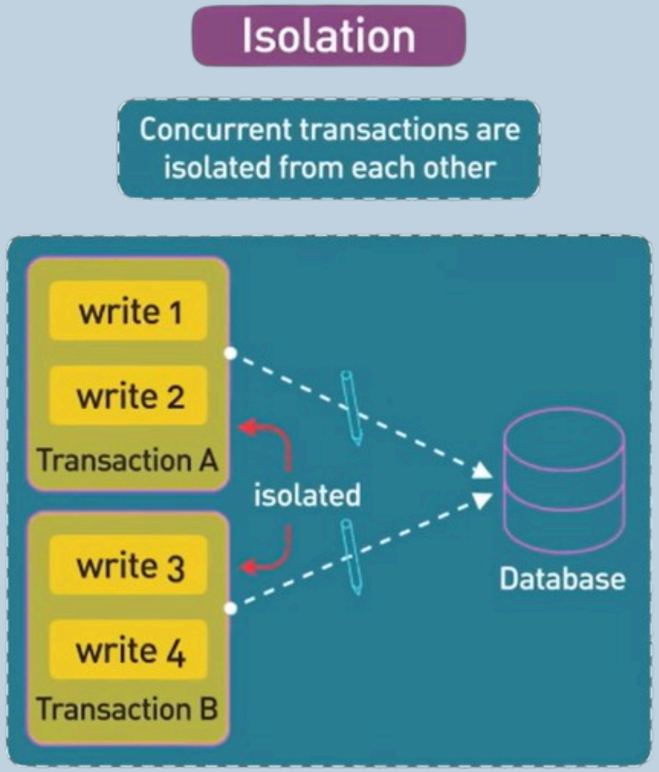
- After a transaction, the database must stay in a valid state.
- All rules (constraints, data types, relationships) must still hold.
- If a transaction breaks a rule, it gets rolled back.
- This keeps data “correct” before and after transactions.







ISOLATION



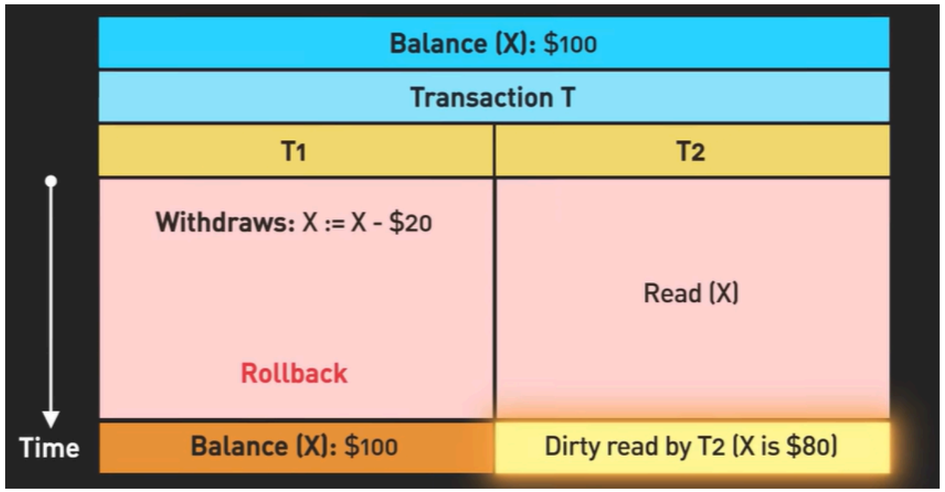
- Even when many transactions run at the same time, each one behaves as if it runs alone.
- One transaction should not see half-done changes from another.
- This prevents issues like dirty reads and inconsistent results.

Example:

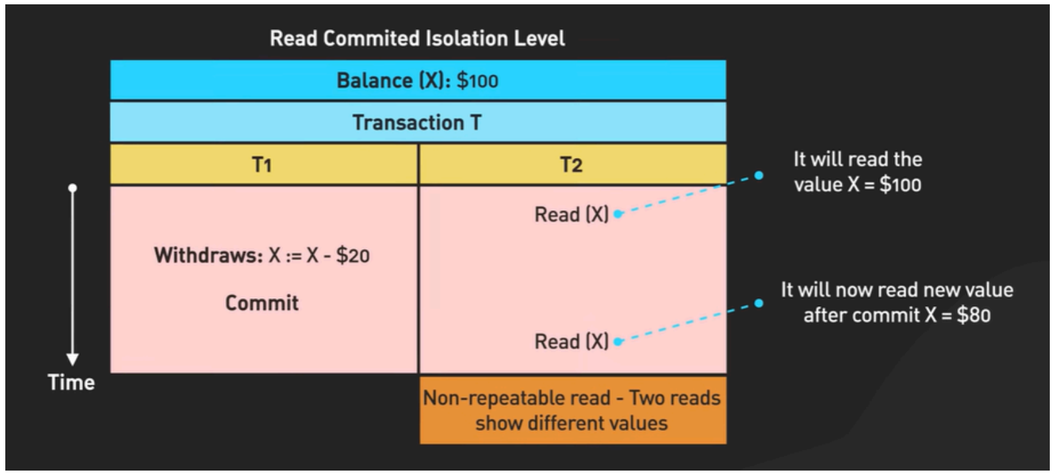
- Two users updating the same record at the same time should not corrupt the data.
- Databases use locking or version control to handle this.

ISOLATION LEVELS

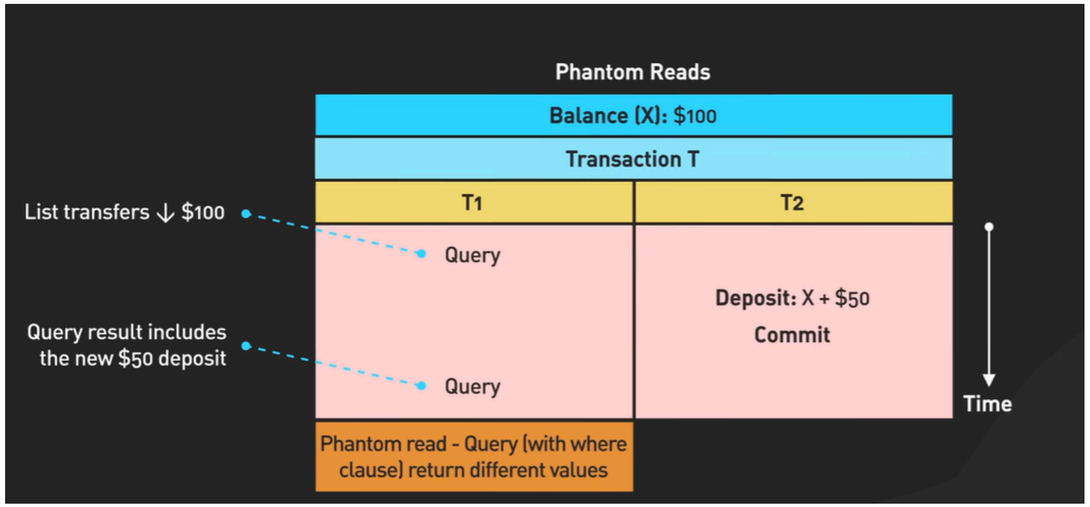
Isolation Level	Violations		
	Dirty Read	Non-repeatable Read	Phantom Read
Serializable	Don't occur	Don't occur	Don't occur
Repeatable	Don't occur	Don't occur	May occur
Read commited	Don't occur	May occur	May occur

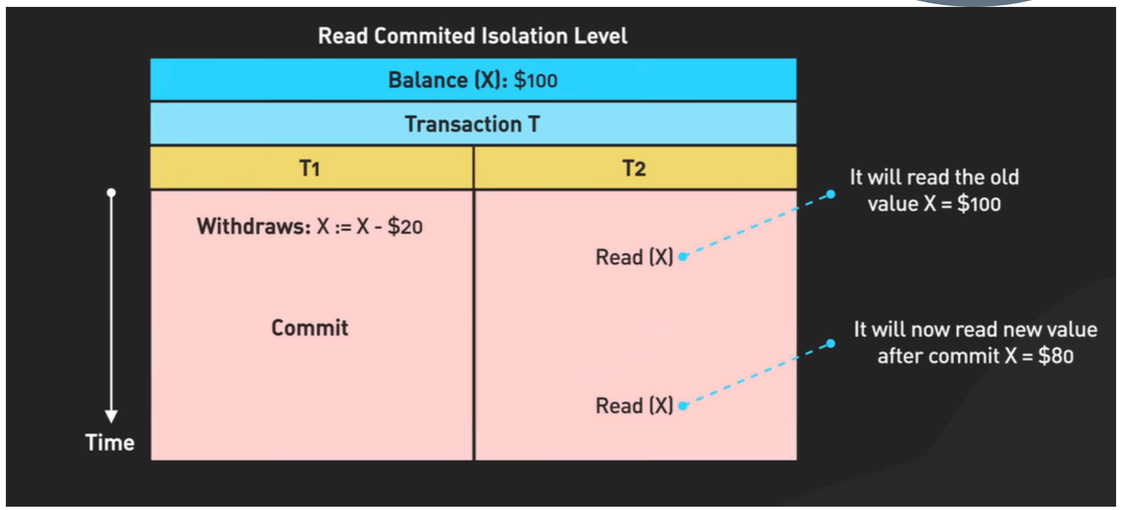


A dirty read happens when a transaction reads data modified by another transaction that hasn't been committed yet.



This happens when a transaction reads the same data twice and gets different results because another transaction modified the data in between.

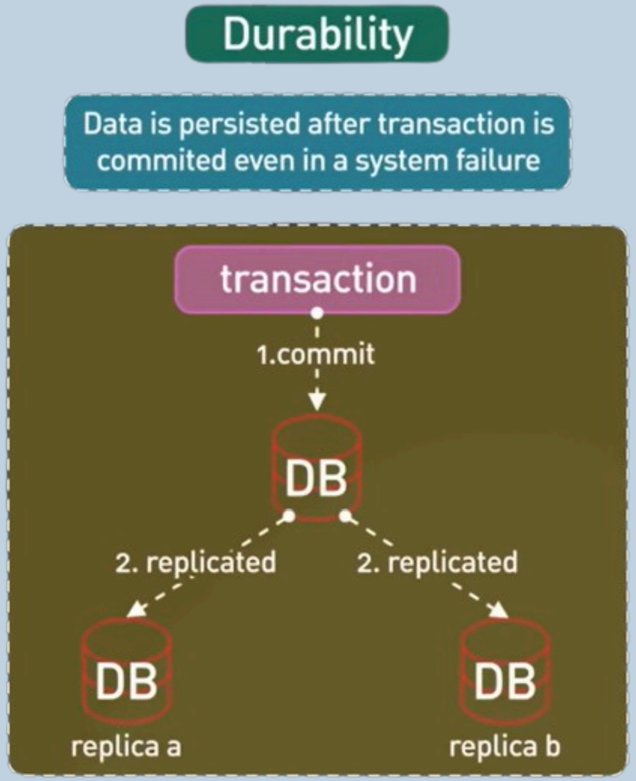




A phantom read occurs when a transaction re-runs a query and gets additional or missing rows because another transaction inserted or deleted matching rows.

Repeatable Read prevents non-repeatable reads. It gives each transaction a consistent snapshot of the data, but Phantom reads can still occur.

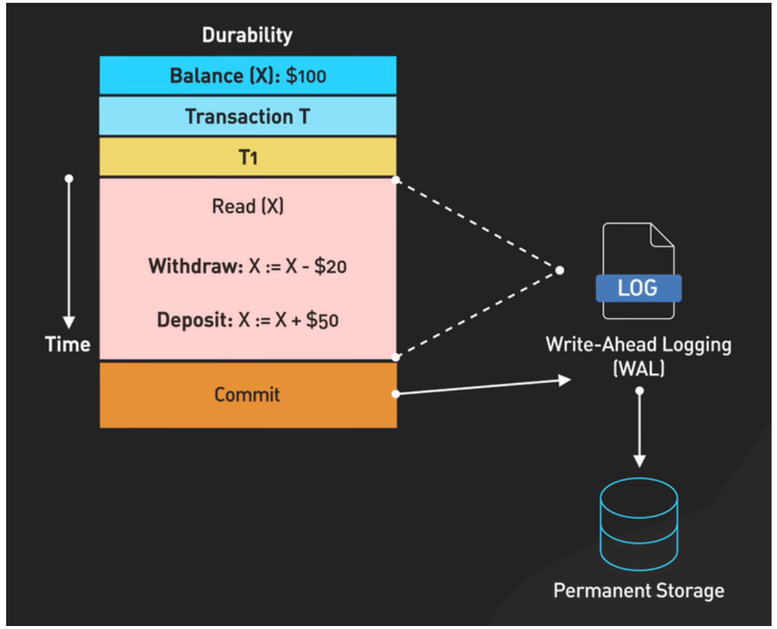
DURABILITY



Once a transaction is committed, it stays committed — even if:

- System crashes
- Power goes off
- Server restarts

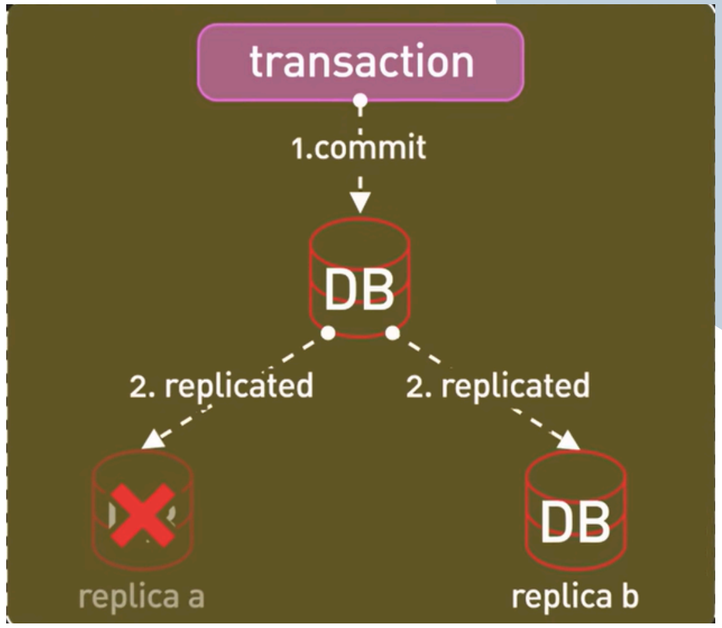
The data is permanently stored.This guarantees that completed work isn’t lost.
This ensures reliability.



- Writing transaction logs
- Using Write-Ahead Logging (WAL)
- Persisting changes to disk before confirming commit

In distributed databases:

- Data is replicated across multiple nodes
- If one node fails, committed transactions are safe on other nodes



WHY ACID IS IMPORTANT

As a data engineer, we deal with:

- ETL/ELT pipelines
- Data ingestion
- Batch & streaming jobs
- Data warehousing

Without ACID

- Pipelines may create duplicate data
- Partial writes may happen
- Corrupted tables can break dashboards
- Analytics results become unreliable

HOW DATABRICKS USES ACID

- Atomic Write
  - If a Spark job fails while writing data to a Delta table, the entire write is rolled back.
- Consistency with Schema Enforcement
  - Delta Lake enforces schema rules.If you try to insert wrong columns, it throws an error.
- Isolation with Optimistic Concurrency Control
  - Multiple users can update the same Delta table safely.
  - Databricks handles conflicts automatically.
- Durability with Transaction Log
  - Delta Lake stores a transaction log (_delta_log folder).
  - This ensures:
    - Every change is tracked
    - Data can be restored
    - No data loss after commit

Thank You