Read Skew Explained

How a race condition can corrupt your data, and how Snapshot Isolation saves the day.

The Problem: Read Skew

Imagine you have a database table of the Kardashian family's bank accounts. A critical rule (an "invariant") is that the total balance must always be **$1,000,000**. They can only transfer money among themselves.

Scenario: The Inconsistent Total

You run a long query to calculate the total balance by reading each account one by one. But while your query is running, a transaction occurs!

Diagram: How Read Skew Happens

Initial Balances

Name	Balance
Caitlyn	$100,000
...	...
Kris	$100,000
...	...
Total	**$1,000,000**

→

Timeline of Operations

●’our Query (T-Read): Reads Caitlyn's balance: `$100,000`.

● **Transaction (T-Write):** Kris sends $100,000 to Caitlyn.

- Caitlyn's balance becomes $200,000.
- Kris's balance becomes $0.

●’our Query (T-Read): Continues and reads Kris's **new** balance: `$0`.

The Incorrect Result

Your query saw Caitlyn's **old** value ($100k) but Kris's **new** value ($0). The final calculated sum is wrong: **`$900,000`**!

This is **Read Skew**: your read saw an inconsistent mix of old and new data.

The Solution: Snapshot Isolation

To fix this, we need to ensure our read query sees the database in a single, consistent state. This is achieved with **Snapshot Isolation**, which relies on a technique called **Multi-Version Concurrency Control (MVCC)**.

How It Works: Keeping Old Versions

Instead of overwriting data, the database keeps old versions of rows, each tagged with the transaction ID that created it. All transactions are ordered sequentially (T1, T2, T3...).

When your query begins, it's given a "snapshot" time (e.g., it starts at T15). It will only see data from transactions committed *before* T15.

Diagram: Reading from a Snapshot at T15

Your query starts at Transaction 15. It must ignore any data from T16 onwards.

Account	Version History (Transaction ID → Balance)	Version Seen by T15
Caitlyn	~~T1→$100k~~, ~~T6→$300k~~, T12→$200k, ~~T17→...~~	$200,000
Kendall	T15→$1,000,000, ~~T22→...~~	$1,000,000
Kris	~~T22→...~~	(Row does not exist yet)
Lamar	T3→$100k	$100,000

The Correct Result

Your query reads the latest version of each row that was committed *before or at T15*. Any writes happening after T15 (like at T17 or T22) are invisible.

The result is a **consistent view** of the database, and the total balance will be correct!

Summary

Read Skew (The Problem)

A read query sees a mix of old and new data from different points in time, leading to inconsistent and incorrect results.

Snapshot Isolation (The Solution)

The query is given a "snapshot" of the database from the moment it started. It only sees data versions from that snapshot, guaranteeing a consistent view.

The key takeaway: **Writers don't block readers, and readers don't block writers**, allowing for high concurrency while maintaining data consistency for read queries.

Summary

Read Skew

The main topic is a race condition called "repeatable read" or "read skew." The speaker uses an example of a database table for the Kardashian family's bank accounts, where the total balance must always sum to one million dollars. During a long read query that sums up all balances, a transaction occurs mid-way (e.g., Kris sends Caitlyn $100k). The query reads Caitlyn's old balance but later reads Kris's new, updated balance. This leads to an incorrect total sum ($900k instead of $1M), which is the essence of read skew: seeing an inconsistent state of the database because of a write that happens during a long read.

Right Ahead Log (WAL)

The proposed solution starts with the Write-Ahead Log (WAL). Every write goes into the WAL sequentially, which allows transactions to be ordered with a monotonically increasing sequence number (e.g., T1, T2, T3...). This ordering is key to solving the problem.

Exercise & Snapshot Isolation

how to see a consistent database snapshot at a specific transaction time, like T15. Instead of deleting old data, the database stores multiple versions of each row, tagged with the transaction number that wrote it (this is MVCC). To get the state at T15, the query ignores any data written by transactions after T15. For each row, it selects the most recent version written at or before T15. This process ensures the query sees a consistent "snapshot" of the database, preventing read skew. This is the premise of Snapshot Isolation.