Serializability and Concurrency Control

Ensuring Correctness and Consistency in Concurrent Transactions

Presented to Computer Science Students and Junior Database Engineers

Teammates:

1. Almas M
2. Aman Jha
3. Amirthavarshini P
4. Anish S

Understanding Serializability

Serializability ensures that the results of running transactions concurrently are identical to running them sequentially, one after another. This fundamental principle maintains data correctness even when multiple users access the database simultaneously.

Without serializability guarantees, databases suffer from inconsistent states, lost updates, and unreliable query results. It provides the foundation for reliable, predictable transaction outcomes in multi-user environments.

Why Serializability Matters

Multi-User Access

Modern databases serve thousands of concurrent users, making coordination essential for maintaining consistency.

Data Integrity

Prevents corruption, lost updates, and anomalous reads that could compromise critical business information.

Reliability Guarantee

Users depend on predictable, correct results regardless of how many transactions execute simultaneously.

Types of Serializability

Conflict Serializability

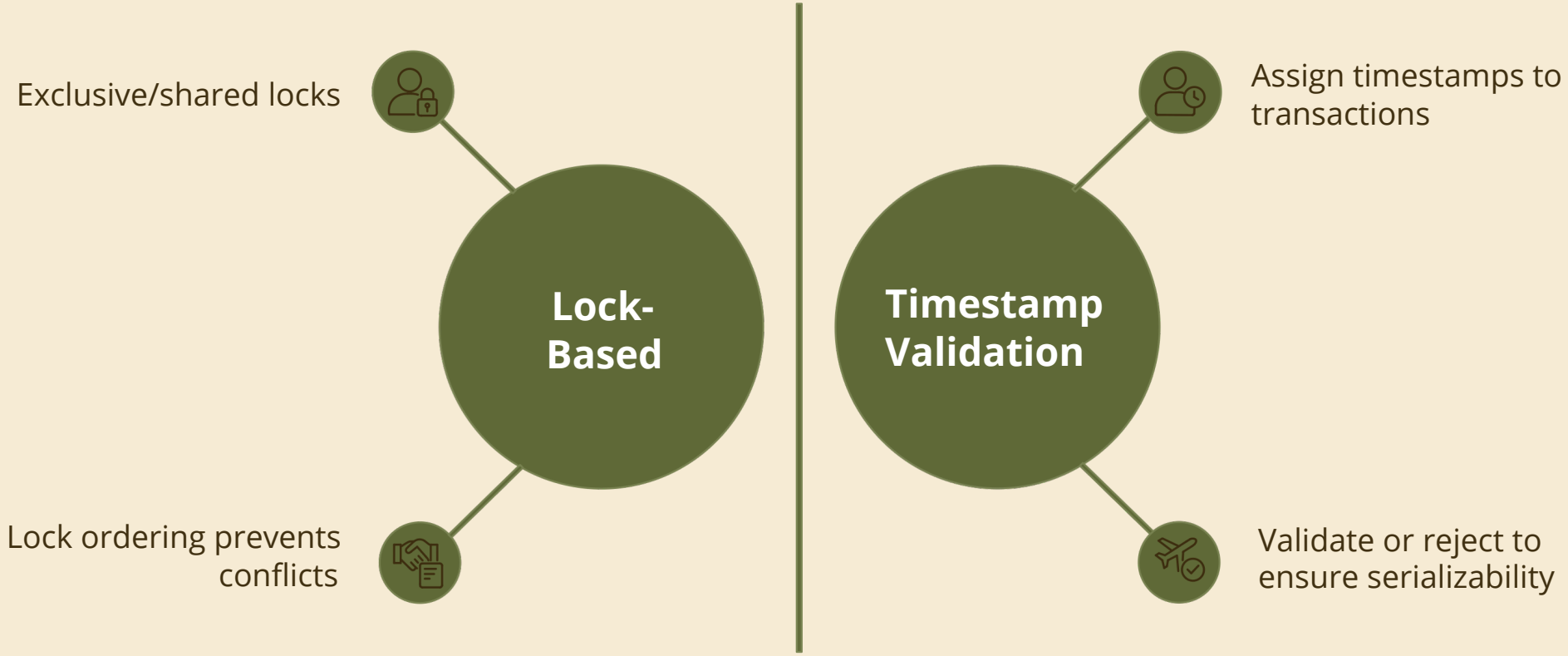
Examines conflicting operations reads and writes on identical data. Uses precedence graphs to detect cycles. No cycles indicate serializability. Most databases use this approach for efficiency.

View Serializability

Verifies that the final database state matches some serial execution order. More permissive than conflict serializability but computationally expensive to verify. Rarely used in practice.

Concurrency Control: Keeping Transactions Safe

Concurrency control mechanisms allow multiple transactions to execute safely and simultaneously without compromising data consistency. They prevent the over lapping operations of multiple users from creating incorrect or corrupted database states.



Problems Without Concurrency Control

Lost Update

Two transactions read the same value, modify it independently, and write back one update is overwritten and lost entirely.

Dirty Read

A transaction reads uncommitted changes from another transaction, then that transaction rolls back, leaving incorrect data used by the first transaction.

Unrepeatable Read

A transaction reads the same data twice and gets different values because another transaction modified it between reads.

Inconsistent Retrieval

Reading partial or mixed data during multi-step updates, violating database integrity constraints and business logic rules.

Lock-Based Concurrency Control

Shared Lock (S-Lock)

Permits multiple transactions to read data simultaneously. No transaction can write while shared locks exist. Used for read-only access, enabling parallelism.

Exclusive Lock (X-Lock)

Grants sole access to one transaction for writing. Prevents all other transactions from reading or writing. Ensures isolation and atomicity of modifications.

Two-Phase Locking (2PL) Protocol

Growing Phase	Shrinking Phase
Acquire all necessary locks	Releaselocks;nonewlocks permitted

Key Takeaways

Serializability is Essential

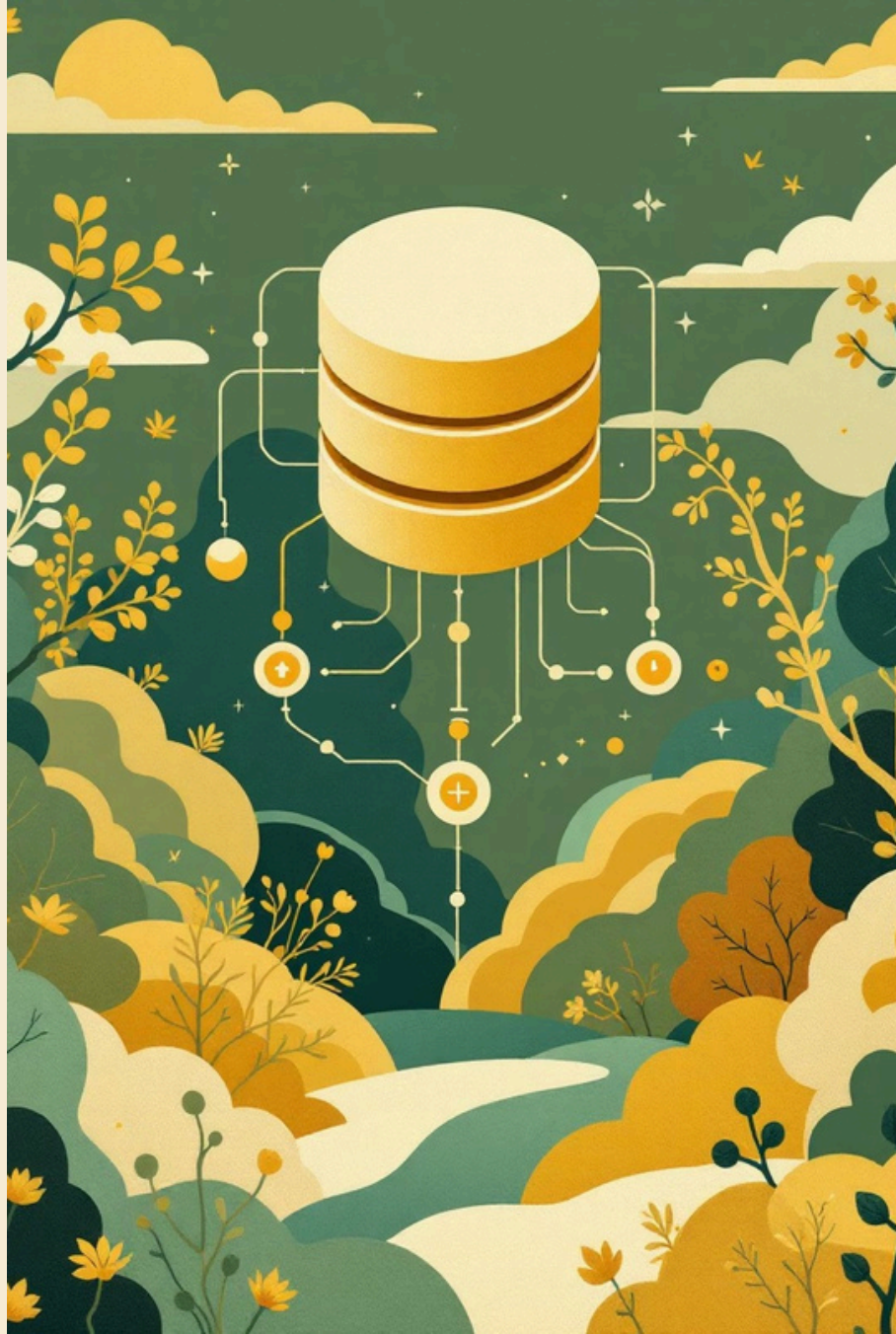
It guarantees thatconcurrent transactions produce results identical to sequential execution, maintaining database correctness and reliability across multi-user systems.

Concurrency Control is Necessary

Lock-based,timestamp-based,and validation-based protocols prevent anomalies like lost updates, dirty reads, and inconsistent retrievals.

2PL is Practical and Proven

Two-Phase Locking remains the industry standard, balancing serializability guarantees with acceptable performance for most database systems.



Thank You