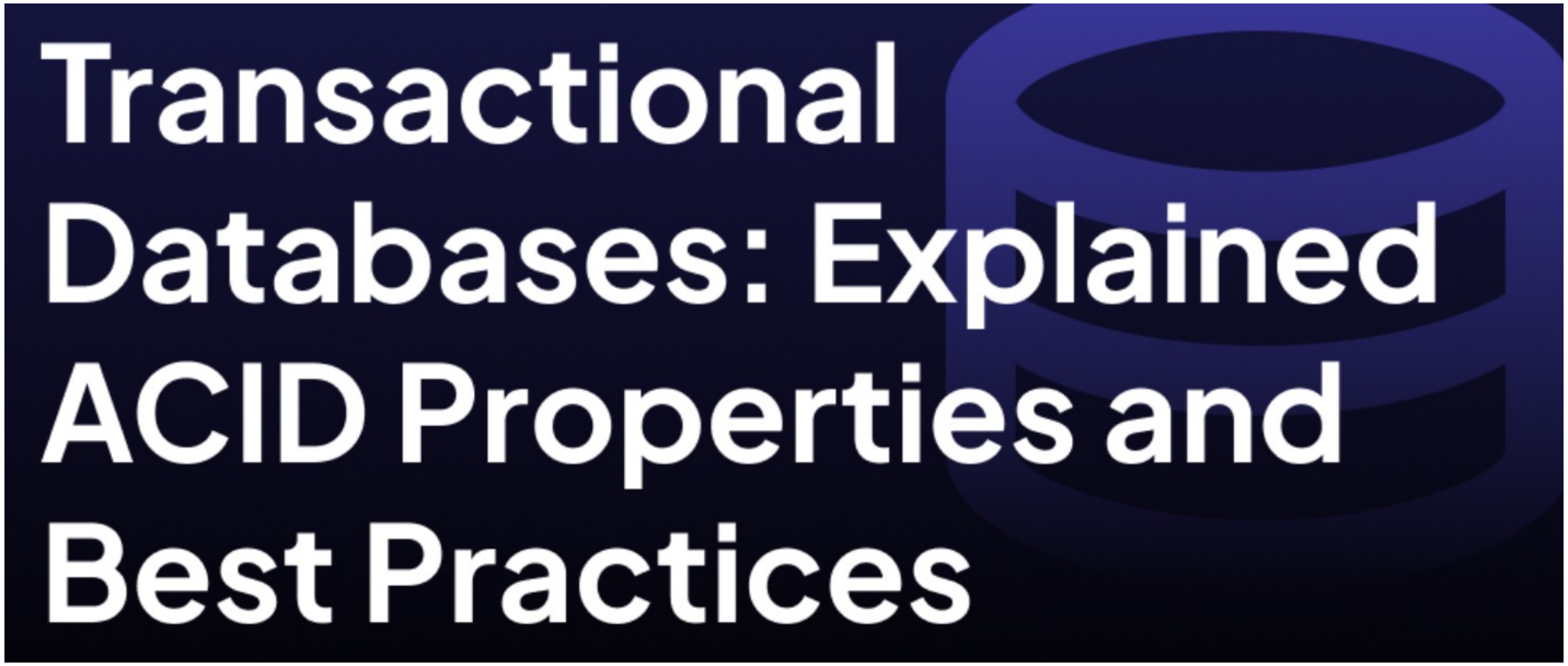
NATIONAL UNIVERSITY

TRANSACTION MANAGEMENT & CONCURRECY CONTROL



Transactional Databases: Explained ACID Properties and Best Practices

TRANSACTIONAL DATABASE OVERVIEW

- Transactional databases are foundational components of modern information systems. A transactional database, also known as a transactional system, is a type of database designed to manage and process transactions. The transaction refers to the sequence of one or more operations performed on a database. These operations include data retrieval, insertion, updation, and deletion.
- In a transactional database, data is organized and stored in rows, with each row representing a record within a table. Transactional databases are optimized for OLTP (Online Transaction Processing) workloads, which involve frequent, small-scale transactions such as inserting, updating, or deleting records. These databases provide mechanisms for ensuring data integrity, concurrency control, and transaction management, making them suitable for applications that require real-time data processing and high transaction throughput.
- At the heart of transactional databases lie the ACID properties: Atomicity, Consistency, Isolation, and Durability.

WHAT ARE ACID PROPERTIES IN A DATABASE?

- ACID (Atomicity, Consistency, Isolation, Durability) is a set of essential properties that defines and preserves the reliability of transactions in database systems after executing a group of operations.
- ACID properties ensure the integrity of data operations, safeguarding against errors and concurrent access.

ATOMICITY

- Atomicity states that a transaction is atomic, meaning it is a single unit of work.
- This property ensures that all the operations within the transactions are successfully completed, and the changes are committed to the database, or none of them take effect (all-or-nothing).
- If any part of the transaction fails due to a network issue, system crash, or an error, the entire transaction is rolled back, ensuring the database remains in a consistent state

CONSISTENCY

- Before committing a transaction, consistency checks are essential to ensure the database maintains its integrity, adhering to both business rules and database constraints.
- Consistency verifies only valid data is written to the database.
- This approach acts as a safeguard against potential anomalies, data corruption, and invalid entries, maintaining data accuracy.

ISOLATION

- Isolation ensures that one transaction cannot impact the outcome of another transaction. In other words, each transaction appears to execute in isolation, without interference from other transactions.
- This property becomes crucial in scenarios where multiple transactions are executed simultaneously. Without proper isolation, issues such as data inconsistency and integrity violations can arise.
- Usually, this is achieved by using locking mechanisms to keep uncommitted transactions isolated.

DURABILITY

- Once a transaction is successfully completed (all operations are executed and committed), the changes made by the transaction are durable.
- This means that the changes persist even in the event of a system failure, power loss, or database crash.
- Often, mechanisms like transaction logs or disk storage are used to record changes and ensure durability.
- This property guarantees that the database remains in a reliable state over time.

IMPORTANCE OF ACID PROPERTIES IN TRANSACTIONAL DATABASES

- Prevents data corruption by maintaining atomicity, ensuring transactions are fully completed or fully aborted.
- Guarantees database validity before and after transactions, upholding data integrity.
- Prevents interference between concurrent transactions, ensuring isolation and avoiding concurrency-related issues.
- Guarantees committed transactions are permanently saved, ensuring data recoverability and durability.
- Crucial for the reliability and trustworthiness of transactional databases, essential for operational continuity and business success.
- Ensures data consistency and integrity throughout transactional processes.

BEST PRACTICES FOR MAINTAINING ACID PROPERTIES

1. **Transaction Management**: Implement robust transaction management systems to ensure atomicity and consistency in database operations. Properly manage transaction boundaries, rollback mechanisms, and commit protocols to maintain data integrity.
2. **Concurrency Control**: Employ effective concurrency control mechanisms to preserve isolation and prevent data interference between concurrent transactions. Use techniques such as locking, multiversion concurrency control (MVCC), and optimistic concurrency control to manage concurrent access to shared resources.
3. **Backup and Recovery Strategies**: Develop comprehensive backup and recovery strategies to ensure the durability and recoverability of committed transactions. Regularly back up database files, transaction logs, and system snapshots to prevent data loss and facilitate timely recovery in the event of system failures or disasters.

CONCURRENCY CONTROL:

Concurrency control mechanisms ensure that multiple transactions can access and modify the database concurrently without corrupting data or causing inconsistencies.

- **Locking:** Transactions can acquire locks on specific data items (records or fields) to prevent other transactions from modifying them while the lock is held.
- **Optimistic** Concurrency Control (OCC): Transactions proceed without locks, but any conflicts are detected and resolved when a transaction tries to commit.
- **Timestamp Ordering**: Transactions are assigned timestamps, and their order of execution or validation is determined based on these timestamps.
- Multiversion Concurrency Control (MVCC): MVCC maintains multiple versions of data to allow concurrent transactions to access data without blocking each other. Each transaction sees a consistent snapshot of the database based on its start time.

HOW TO MAINTAIN ACID PROPERTIES IN TRANSACTIONAL DATABASES?

- **Transaction Logs:** Maintain transaction logs that could be used not only for rollbacks but also for recovery purposes. Verify that these logs are backed up regularly and stored securely.
- **Explicit Transaction:** Define transactions using explicit control statements to ensure all-or-nothing transaction behavior. This can be implemented using some SQL statements like BEGIN TRANSACTION to mark the start of the transaction, COMMIT to make all changes permanent, and ROLLBACK to undo the entire transaction in case of an error or other conditions.
- **Error Handling:** Implement robust bug-managing mechanisms to handle exceptions. This includes using try-catch block or error handling parameters depending on the database management system you are using. You can also reverse-perform the entire transaction to double-check if any part encounters an error.
- **Nested Transactions:** Use nested transactions to break down complex transactions into smaller units. This will allow you to perform partial rollbacks if needed.

- **Define Constraints:** Always tailor the constraints to your specific data model. For this, you must define and enforce integrity constraints to ensure data consistency. Some of the constraints that you can include:
  ~Unique Constraint: This constraint ensures that a column or a combination of columns contains unique values for each row in a table.
  ~Check Constraint: You can define a condition that each row in a table must satisfy.
  ~Primary Key: This constraint allows you to define a unique and non-null identifier key for each record in a table.
  ~Not-null Constraint: With this constraint, you can ensure that a column does not contain any null values.
- **Business Rules:** Implement business rules and logic at the database level. You can use triggers, application-level checks or constraints, and stored procedures. By incorporating these techniques, you fix business rules directly into the database structure. This allows you to enhance consistency and reduce the risk of rule violations.

- **Regular Audits & Testing:** Once you define business rules and constraints, you need to perform regular audits to ensure the data complies with the defined rules. For instance, you can implement scheduled jobs or scripts to automate the audit process and track changes in the database over time.
- **Concurrency Control:** You can execute effective concurrency control mechanisms to manage concurrent access to data. Here are some examples of how you can achieve isolation property:
  ~Use exclusive locks to prevent multiple transactions from accessing the same data simultaneously.
  ~Implement a deadlock detection mechanism to identify and resolve scenarios where more than two transactions are waiting for each other to release their state.
  ~Define appropriate and fixed timeout values for transactions. This will enable it to terminate if certain conditions are not met within the specific time frame. Timeouts are essential for managing resource contention, preventing long-running transactions, and avoiding potential deadlocks.
- **Backup and Recovery:** Plan a robust backup and recovery strategy. Schedule regular database backups and test the recovery process to ensure data durability.
- **Failover Checks:** You can apply failover mechanisms or redundancy checks to verify if the data remains accessible even in the event of hardware failure or system crash.
- **Monitoring and Alerts:** You can conduct regular maintenance tasks to ensure optimal performance and database durability. In addition, you can also implement monitoring tools to track the health of the database and set up alerts to quickly notify potential issues that may impact durability.

ADVANTAGES OF CONCURRENCY

In general, concurrency means, that more than one transaction can work on a system. The advantages of a concurrent system are:

- **Waiting Time:** It means if a process is in a ready state but still the process does not get the system to get execute is called waiting time. So, concurrency leads to less waiting time.
- **Response Time:** The time wasted in getting the response from the cpu for the first time, is called response time. So, concurrency leads to less Response Time.
- **Resource Utilization:** The amount of Resource utilization in a particular system is called Resource Utilization. Multiple transactions can run parallel in a system. So, concurrency leads to more Resource Utilization.
- **Efficiency:** The amount of output produced in comparison to given input is called efficiency. So, Concurrency leads to more Efficiency.

DISADVANTAGES OF CONCURRENCY

- **Overhead:** Implementing concurrency control requires additional overhead, such as acquiring and releasing locks on database objects. This overhead can lead to slower performance and increased resource consumption, particularly in systems with high levels of concurrency.
- **Deadlocks:** Deadlocks can occur when two or more transactions are waiting for each other to release resources, causing a circular dependency that can prevent any of the transactions from completing. Deadlocks can be difficult to detect and resolve, and can result in reduced throughput and increased latency.
- **Reduced concurrency:** Concurrency control can limit the number of users or applications that can access the database simultaneously. This can lead to reduced concurrency and slower performance in systems with high levels of concurrency.
- **Complexity:** Implementing concurrency control can be complex, particularly in distributed systems or in systems with complex transactional logic. This complexity can lead to increased development and maintenance costs.
- **Inconsistency:** In some cases, concurrency control can lead to inconsistencies in the database. For example, a transaction that is rolled back may leave the database in an inconsistent state, or a long-running transaction may cause other transactions to wait for extended periods, leading to data staleness and reduced accuracy.

REVIEW QUESTIONS?

Q1: What are the ACID properties of a transaction?

Q2: Which database is best for ACID transactions?

Q3: What makes a sequence of database operations a transaction?

ANSWERS:

Ans1: The ACID properties of a transaction in a database management system are Atomicity, Consistency, Isolation, and Durability.

Ans2: A sequence of database operations becomes a transaction when it satisfies the properties of ACID (Atomicity, Consistency, Isolation, Durability).

Ans3: Several relational database management systems (RDBMS) are known for their robust support of ACID transactions. Some popular databases known for ACID compliance are Microsoft SQL Server, MySQL, SQLite, and Oracle database.

THE END.

Thank You