Concurrency Control

Northeastern University
College of Professional Studies

ITC6000 80036 Database Management Systems
Presentation Assignment
Date: 6 May 2024

Submitted by Nazar Mammedov

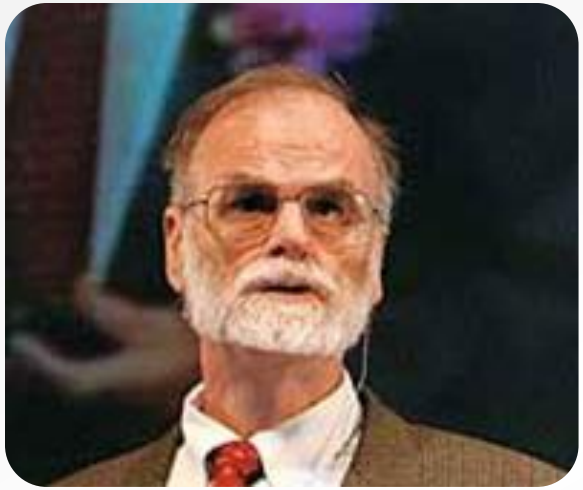
Presentation Outline

- What is concurrency control?
- Three problems of concurrency control
- Locking Methods
- Time Stamping method
- Concurrency Control in Oracle Database
- Conclusion

What is concurrency control?

- In real-world transactions, such as paying for an electric bill or your credit card company withdrawing money from your account can take place at the same time.
- Databases need to be able to execute these **concurrent transactions**.
- Managing the execution of concurrent transactions is called **concurrency control**.
- Companies consider concurrency control characteristics of DBMS when choosing vendors. Example: Uber's migration from PostgreSQL

Scholars of concurrency control in DBMS



Source: https://amturing.acm.org

James Gray

Source: https://www.microsoft.com/

Phil Bernstein

What is a transaction?

- Most of practically-useful real world actions correspond to transactions.

- A **transaction** is a sequence of database operations that need to be entirely completed or aborted.

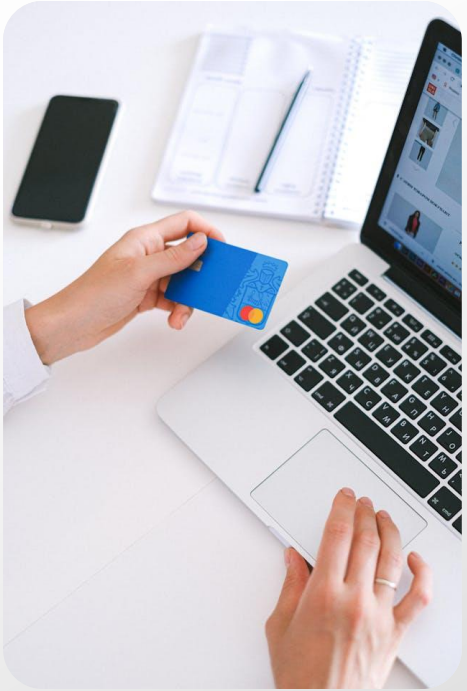
- All transactions must have these properties:
  - **Atomicity**: all or none of operations must be completed.
  - **Consistency**: Transactions must start and finish with consistent database
  - **Isolation:** Data operations in transactions must be isolated from each other
  - **Durability:** The results of completed transactions must be permanent.

Three problems of concurrency control

- **Lost updates**
  - Changes made by one transaction are lost because of overwriting by another transaction.
- **Uncommitted data**
  - A transaction uses uncommitted (temporary, draft) changes generated by a previous transaction as if they are real change.
- **Inconsistent retrievals**
  - A transaction uses inconsistent data due to reading the data before and after other transactions work on the data.

Lost Updates

- What happens when two people want to deposit money in the same account?
- John has $100 in his account.
- Alice wants to send $20 to John.
  - Read John's balance and add $10 and update the balance $120
- Bob wants to send $50 to John.
  - Read John's balance and add $50 and update the balance $150
- If Alice cannot complete her transfer before Bob initiates his transfer Alice's money is lost.



Source: pexels.com

How to deal with concurrent transactions?

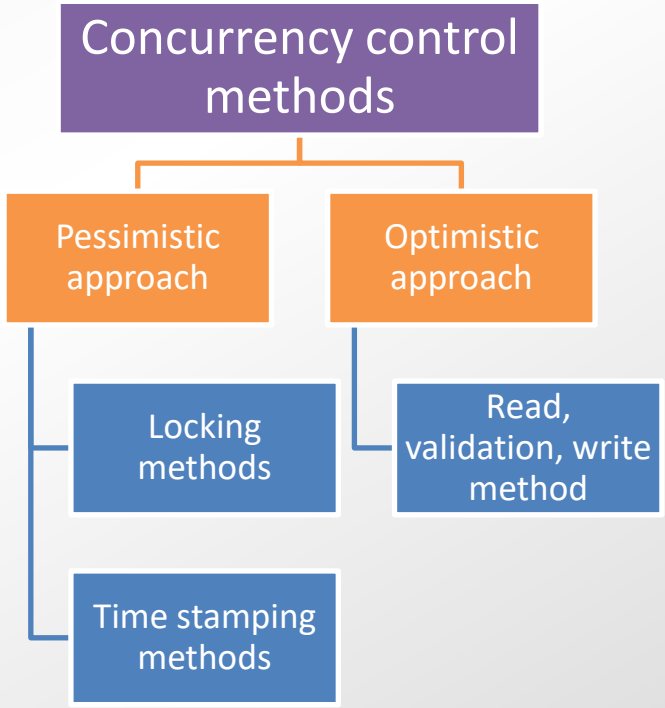
- A special DBMS process called **scheduler** determines the order in which the operations of concurrent transactions are executed.

- **First-come, first-served** – Transactions are executed in the order they were received by the DBMS

- **Concurrency conflicts** occur when two transactions run concurrently and at least one of them **writes** to the database.

Concurrency control approaches and methods

- Based on assumption regarding the likelihood of conflict between transactions, two approaches are used:
  - **Pessimistic** – conflict is **likely**
    - Used with frequently updated data
  - **Optimistic** – conflict is **unlikely** (or rare)
    - Used with data with few update transactions



Locking Methods

- **Locking methods** aim to prevent concurrent use of data item that can produce data inconsistency.
- Commonly used technique in concurrency control.
- To guarantee exclusive use of a data item by a transaction the DBMS provides a **lock**.

Source: https://dbaparadise.com/

Lock Granularity

Lock granularity level	What is locked?	Use case	Good for multiuser?
Database level	The entire database is locked	Good for batch processes, *database backup process*	**No**
Table level	Table(s) used by a transaction are locked	Good for operations reading the entire table, *generating reports from the entire table*	**No**
Page level	Locks diskpages used by a transaction	Good for multi-row operations that don't need to change the entire table, *payroll calculations*	**Yes, most frequently used method**
Row level	Rows used by transaction are locked	Transactions operating on different rows, *bank accounts*	**Yes, frequently used**
Field level	Only fields accessed by transaction are locked	Locking fields for sensitive data, *hiding SSN field from view*	**Yes, but rarely used due to high overhead**

Lock Types

Type	What happens?	Use case
Binary Locks	Locks are either locked or unlocked	Rarely used due to their restrictive nature
Exclusive locks	Access is reserved for the transaction locking the object	Updating a data item
Shared locks	Two read transactions share access to the data	Used for read-only operations

Addressing locking problems

- Locks can lead to two problems:
  - Serialization is impossible
  - Deadlocks – two transactions blocking each other indefinitely

- Serializability is achieved through **two-phase locking (2PL) protocol**.

- Deadlocks are addressed using the following:
  - **Deadlock prevention:** A transaction is aborted if a deadlock is likely
  - **Deadlock detection:** The DBMS monitors and rolls back the deadlocked transaction
  - **Deadlock avoidance:** get all locks necessary for the transaction, roll back if cannot

Deadlock example

- Row at employee_id = 100 is locked by Transaction 1
- Row at employee_id = 200 is locked by Transaction 2

- **Transaction 1:** SQL> UPDATE employees SET salary = salary*1.1 WHERE employee_id = 200;

- **Transaction 2:** SQL> UPDATE employees SET salary = salary*1.1 WHERE employee_id = 100;

- Deadlock occurs.

Source: https://docs.oracle.com/en/database/oracle/oracle-database/23/cncpt/

Time Stamping Methods

- The time stamping method attaches a **time stamp** to each transaction and uses the time stamp to determine the **order** in which conflicting operations are executed.

- There are two requirements for time stamps:
  - **Uniqueness** – they must be unique in the DBMS
  - **Monotonicity** – their values must always increase

- Each value in the database must have two additional time stamps:
  - One for the **last read** operation
  - One for the **last update** operation

- The main disadvantage of the time stamping method is that it requires a lot of memory and processing resources.

Time Stamping Schemes

- If two transactions conflict, one is rolled back and issued a new time stamp value and reenters the execution queue.

- There are two schemes for time-stamping:

  - **Wait/die scheme** – the older transaction waits for and executes after the younger transaction. Example: *If Customer A has an appointment number lower than the Customer B who is already receiving the service, Customer A must wait*

  - **Wound/wait scheme** – the older transaction rolls back the younger transaction and assigns a new higher time stamp. Example: *If Customer A has an appointment number lower than the Customer B who is already receiving the service, the desk stops servicing Customer B, sends to waiting room, proceeds to serve Customer A*

- In the time stamping method, deadlocks are avoided by assigning **a time-out value** for each lock request. After the time-out expires, the lock request is cancelled and the transaction is rolled back.

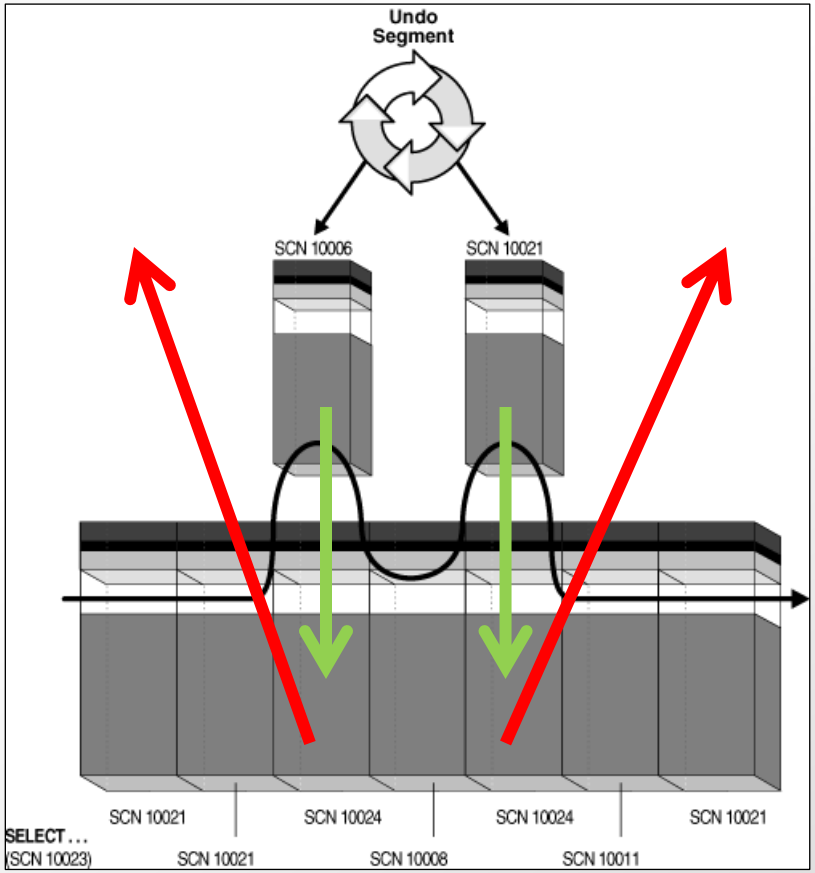
Concurrency Control in ORACLE Database

Source: oracle.com

- The latest version is Oracle Database release 23
- Oracle uses the following mechanisms to manage concurrency control:
  - Multiversion Read Consistency
  - Locking mechanisms
  - ANSI/ISO Transaction Isolation Levels
- Oracle Database automatically manages locking. Users can manually lock if necessary.
- Oracle uses row-level locking (rather than page-level)
- Oracle never escalates locks because it can increase the likelihood of deadlocks.

Multiversion Read Consistency in Oracle Database

- Multiple versions of data are maintained that are consistent for a single point in time.
- Never allows dirty reads (reading of uncomitted data in another transaction).
- Provides statement-level and transaction-level read consistency.
- Oracle uses SCN (System Change Number) to determine versions of data at different points in time.
- Changes after SCN 10023 are ignored.



Source: https://docs.oracle.com/en/database/oracle/oracle-database/23/cncpt/

Key Takeaways

- Concurrency control is a **fundamental feature** required for modern multi-user DBMS.
- Real-world concurrency control requires **compromise** between performance and flexibility.
- The more granular and flexible concurrency control **the more computing resources** we need.
- **Pessimistic concurrency control** is good for simultaneous manipulation of data accessed by multiple users where consistency is more important than the speed.
- **Optimistic concurrency control** is good and fast if restarting a transaction happens seldom and is not costly for business.
- Real-world DBMS-s use complex and diverse methods of transaction control that go beyond the textbook content.

References

- Coronel, C., & Morris, S. (2016). *Database Systems: Design, Implementation, & Management*. Cengage Learning.

- ACM. James Gray. Retrieved May 5, 2024
- https://amturing.acm.org/award_winners/gray_3649936.cfm

- Microsoft. Phil Bernstein. Retrieved May 5, 2024. https://www.microsoft.com/en-us/research/people/philbe/book/

- Oracle. Database Concepts. Retrieved May 5, 2024. https://docs.oracle.com/en/database/oracle/oracle-database/23/cncpt/data-concurrency-and-consistency.html

- Uber. Why Uber Engineering Switched from Postgres to MySQL. Retrieved May 5, 2024. https://www.uber.com/blog/postgres-to-mysql-migration/

Thank you!

Q & A