

DBMS Concurrency Control: Two Phase, Timestamp, Lock-Based Protocol

What is Concurrency Control?

Concurrency control is the procedure in DBMS for managing simultaneous operations without conflicting with each another. Concurrent access is quite easy if all users are just reading data. There is no way they can interfere with one another. Though for any practical database, would have a mix of reading and WRITE operations and hence the concurrency is a challenge.

Concurrency control is used to address such conflicts which mostly occur with a multi-user system. It helps you to make sure that database transactions are performed concurrently without violating the data integrity of respective databases.

Therefore, concurrency control is a most important element for the proper functioning of a system where two or multiple database transactions that require access to the same data, are executed simultaneously.

In this tutorial, you will learn

- [What is Concurrency Control?](#)

DBMS Tutorial

[3\) DBMS Schemas](#)[4\) Relational Data Model in DBMS](#)[5\) ER Diagram in DBMS](#)[6\) Relational Algebra in DBMS](#)[7\) DBMS Transaction Management](#)[8\) DBMS Concurrency Control](#)[9\) DBMS Keys](#)[10\) DBMS Functional Dependency](#)[11\) Data Independence in DBMS](#)[12\) Hashing in DBMS](#)[13\) SQL Joins \(EXAMPLES\)](#)

- [Potential problems of Concurrency](#)
- [Why use Concurrency method?](#)
- [Concurrency Control Protocols](#)
- [Lock-based Protocols](#)
- [Two Phase Locking \(2PL\) Protocol](#)
- [Timestamp-based Protocols](#)
- [Characteristics of Good Concurrency Protocol](#)

Potential problems of Concurrency

Here, are some issues which you will likely to face while using the Concurrency Control method:

- **Lost Updates** occur when multiple transactions select the same row and update the row based on the value selected
- Uncommitted dependency issues occur when the second transaction selects a row which is updated by another transaction (**dirty read**)
- **Non-Repeatable Read** occurs when a second transaction is trying to access the same row several times and reads different data each time.
- **Incorrect Summary issue** occurs when one transaction takes summary over the value of all the instances of a repeated data-item, and second transaction update few instances of that specific data-item. In that situation, the resulting summary does not reflect a correct result.

Why use Concurrency method?

Reasons for using Concurrency control method is DBMS:

- To apply Isolation through mutual exclusion between conflicting transactions
- To resolve read-write and write-write conflict issues
- To preserve database consistency through constantly preserving execution obstructions
- The system needs to control the interaction among the concurrent transactions. This control is achieved using concurrent-control schemes.
- Concurrency control helps to ensure serializability

Example

Assume that two people who go to electronic kiosks at the same time to buy a movie ticket for the same movie and the same show time.

However, there is only one seat left in for the movie show in that particular theatre. Without concurrency control, it is possible that both moviegoers will end up purchasing a ticket. However, concurrency control method does not allow this to happen. Both moviegoers can still access information written in the movie seating database. But concurrency control only provides a ticket to the buyer who has completed the transaction process first.

Concurrency Control Protocols

Different concurrency control protocols offer different benefits between the amount of concurrency they allow and the amount of overhead that they impose.

- Lock-Based Protocols
- Two Phase
- Timestamp-Based Protocols

- Validation-Based Protocols

Lock-based Protocols

A lock is a data variable which is associated with a data item. This lock signifies that operations that can be performed on the data item. Locks help synchronize access to the database items by concurrent transactions.

All lock requests are made to the concurrency-control manager. Transactions proceed only once the lock request is granted.

Binary Locks: A Binary lock on a data item can either locked or unlocked states.

Shared/exclusive: This type of locking mechanism separates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is called an exclusive lock.

1. Shared Lock (S):

A shared lock is also called a Read-only lock. With the shared lock, the data item can be shared between transactions. This is because you will never have permission to update data on the data item.

For example, consider a case where two transactions are reading the account balance of a person. The database will let them read by placing a shared lock. However, if another transaction wants to update that account's balance, shared lock prevent it until the reading process is over.

2. Exclusive Lock (X):

With the Exclusive Lock, a data item can be read as well as written. This is exclusive and can't be held concurrently on the same data item. X-lock is requested using lock-x instruction.

Transactions may unlock the data item after finishing the 'write' operation.

For example, when a transaction needs to update the account balance of a person. You can allow this transaction by placing X lock on it. Therefore, when the second transaction wants to read or write, exclusive lock prevents this operation.

3. Simplistic Lock Protocol

This type of lock-based protocols allows transactions to obtain a lock on every object before beginning operation. Transactions may unlock the data item after finishing the 'write' operation.

4. Pre-claiming Locking

Pre-claiming lock protocol helps to evaluate operations and create a list of required data items which are needed to initiate an execution process. In the situation when all locks are granted, the transaction executes. After that, all locks release when all of its operations are over.

Starvation

Starvation is the situation when a transaction needs to wait for an indefinite period to acquire a lock.

Following are the reasons for Starvation:

- When waiting scheme for locked items is not properly managed
- In the case of resource leak
- The same transaction is selected as a victim repeatedly

Deadlock

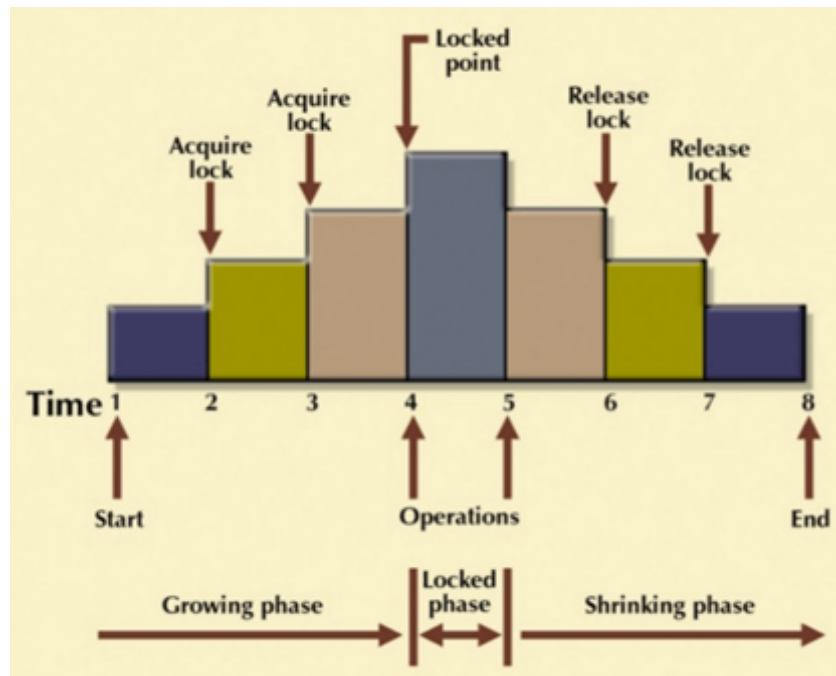
Deadlock refers to a specific situation where two or more processes are waiting for each other to release a resource or more than two processes are waiting for the resource in a circular chain.

Two Phase Locking (2PL) Protocol

Two-Phase locking protocol which is also known as a 2PL protocol. It is also called P2L. In this type of locking protocol, the transaction should acquire a lock after it releases one of its locks.

This locking protocol divides the execution phase of a transaction into three different parts.

- In the first phase, when the transaction begins to execute, it requires permission for the locks it needs.
- The second part is where the transaction obtains all the locks. When a transaction releases its first lock, the third phase starts.
- In this third phase, the transaction cannot demand any new locks. Instead, it only releases the acquired locks.



The Two-Phase Locking protocol allows each transaction to make a lock or unlock request in two steps:

- **Growing Phase:** In this phase transaction may obtain locks but may not release any locks.
- **Shrinking Phase:** In this phase, a transaction may release locks but not obtain any new lock

It is true that the 2PL protocol offers serializability. However, it does not ensure that deadlocks do not happen.

In the above-given diagram, you can see that local and global deadlock detectors are searching for deadlocks and solve them with resuming transactions to their initial states.

Strict Two-Phase Locking Method

Strict-Two phase locking system is almost similar to 2PL. The only difference is that Strict-2PL never releases a lock after using it. It holds all the locks until the commit point and releases all the locks at one go when the process is over.

Centralized 2PL

In Centralized 2 PL, a single site is responsible for lock management process. It has only one lock manager for the entire DBMS.

Primary copy 2PL

Primary copy 2PL mechanism, many lock managers are distributed to different sites. After that, a particular lock manager is responsible for managing the lock for a set of data items. When the primary copy has been updated, the change is propagated to the slaves.

Distributed 2PL

In this kind of two-phase locking mechanism, Lock managers are distributed to all sites. They are responsible for managing locks for data at that site. If no data is replicated, it is equivalent to primary copy 2PL. Communication costs of Distributed 2PL are quite higher than primary copy 2PL

Timestamp-based Protocols

The timestamp-based algorithm uses a timestamp to serialize the execution of concurrent transactions. This protocol ensures that every conflicting read and write operations are

executed in timestamp order. The protocol uses the **System Time or Logical Count as a Timestamp**.

The older transaction is always given priority in this method. It uses system time to determine the time stamp of the transaction. This is the most commonly used concurrency protocol.

Lock-based protocols help you to manage the order between the conflicting transactions when they will execute. Timestamp-based protocols manage conflicts as soon as an operation is created.

Example:

```
Suppose there are there transactions T1, T2, and T3.  
T1 has entered the system at time 0010  
T2 has entered the system at 0020  
T3 has entered the system at 0030  
Priority will be given to transaction T1, then transaction T2 and lastly Transaction T  
3.
```

Advantages:

- Schedules are serializable just like 2PL protocols
- No waiting for the transaction, which eliminates the possibility of deadlocks!

Disadvantages:

Starvation is possible if the same transaction is restarted and continually aborted

Characteristics of Good Concurrency Protocol

An ideal concurrency control DBMS mechanism has the following objectives:

- Must be resilient to site and communication failures.
- It allows the parallel execution of transactions to achieve maximum concurrency.
- Its storage mechanisms and computational methods should be modest to minimize overhead.
- It must enforce some constraints on the structure of atomic actions of transactions.

Summary

- Concurrency control is the procedure in DBMS for managing simultaneous operations without conflicting with each another.
- Lost Updates, dirty read, Non-Repeatable Read, and Incorrect Summary Issue are problems faced due to lack of concurrency control.
- Lock-Based, Two-Phase, Timestamp-Based, Validation-Based are types of Concurrency handling protocols
- The lock could be Shared (S) or Exclusive (X)
- Two-Phase locking protocol which is also known as a 2PL protocol needs transaction should acquire a lock after it releases one of its locks. It has 2 phases growing and shrinking.
- The timestamp-based algorithm uses a timestamp to serialize the execution of concurrent transactions. The protocol uses the **System Time or Logical Count as** a Timestamp.

Guru99 is Sponsored by AQUAFOLD



Aqua Data Studio is proud to sponsor Guru99's informative website for database professionals. Take your productivity to new heights with query tools, visual analytics, and ER diagramming for multiple platforms.

[LEARN MORE ABOUT AQUA DATA STUDIO](#)

Top Tutorials



About

[About Us](#)
[Advertise with Us](#)
[Write For Us](#)
[Contact Us](#)

Career Suggestion

[SAP Career](#)
[Suggestion Tool](#)
[Software Testing as a Career](#)

Interesting

[Books to Read!](#)
[Blog](#)
[Quiz](#)
[eBook](#)

Execute online

[Execute Java Online](#)
[Execute Javascript](#)
[Execute HTML](#)
[Execute Python](#)



Selenium



Testing



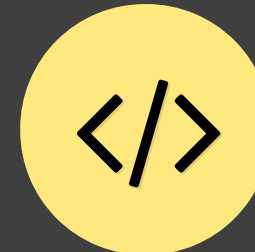
Hacking



SAP



Java



Python



Jmeter



Informatica



JIRA

