NAVIGATING CONCURRENCY CONTROL & DATABASE RECOVERY

Take home Assignment II

Index No : 21000034
Name :M.K.P.Ahinsa

CONCURRENCY CONTROL BASED ON TIMESTAMP ORDERING

WHAT IS TIMESTAMP

- **Timestamp** is a unique identifier created by the DBMS to identify a transaction.
- They are usually assigned in the order in which they are submitted to the system.
- Refer to the timestamp of a transaction *T* as ***TS(T)***.

TIMESTAMP ORDERING PROTOCOL

- The Timestamp Ordering Protocol is used to order the transactions based on their Timestamps. The order of transaction is nothing but the ascending order of the transaction creation.
- The priority of the older transaction is higher that's why it executes first. To determine the timestamp of the transaction, this protocol uses system time or logical counter.
- The lock-based protocol is used to manage the order between conflicting pairs among transactions at the execution time. But Timestamp based protocols start working as soon as a transaction is created.

TIMESTAMP ORDERING PROTOCOL CONT.

- Let's assume there are two transactions T1 and T2. Suppose the transaction T1 has entered the system at 007 times and transaction T2 has entered the system at 009 times.

  T1 has the higher priority, so it executes first as it is entered the system first.

- The timestamp ordering protocol also maintains the timestamp of last 'read' and 'write' operation on a data.

BASIC TIMESTAMP ORDERING PROTOCOL

- Every transaction is issued a timestamp based on when it enters the system.
- Suppose, if an old transaction T_i has timestamp $TS(T_i)$, a new transaction T_j is assigned timestamp $TS(T_j)$ such that **$TS(T_i) < TS(T_j)$**.
- The protocol manages concurrent execution such that the timestamps determine the serializability order. The timestamp ordering protocol ensures that any conflicting read and write operations are executed in timestamp order.
- Whenever some Transaction *T* tries to issue a R_item(X) or a W_item(X), the Basic TO algorithm compares the timestamp of *T* with **R_TS(X) & W_TS(X)** to ensure that the Timestamp order is not violated.
- This describes the Basic TO protocol in the following two cases.

Case 1: Whenever a Transaction T issues W_item(X) operation , Check the following Conditions:

- If ***R_TS(X) > TS(T)*** and if ***W_TS(X) > TS(T)***, then abort and rollback T and reject the operation. else,
- Execute W_item(X) operation of T and set W_TS(X) to TS(T).

```
if(W_TS(X) >TS(T) OR R_TS(X)  >TS(T))
{
        abort T AND Rollback;
}else{
        W(X);
        W_TS(X)=TS(T);
}
```

Case 2: Whenever a Transaction T issues R_item(X) operation , Check the following Conditions:

- If ***W_TS(X) > TS(T)*** , then abort and reject T and reject the operation. else,
- If W_TS(X) <= TS(T), then execute the R_item(X) operation of T and set R_TS(X) to the larger of TS(T) and current R_TS(X).

```
if(W_TS(X) >TS(T) )
{
        abort T AND Rollback;
}else{
        R(X);
        R_TS(X)=TS(T);
}
```

Advantages and Disadvantages of Basic TO protocol

- Timestamp Ordering protocol ensures serializability since the precedence graph will be of the form:

- Timestamp protocol ensures freedom from deadlock as no transaction ever waits.
- But the schedule may *not be cascade free* , and may not even be recoverable.

STRICT TIMESTAMP ORDERING PROTOCOL

- A variation of Basic TO is called Strict TO ensures that the schedules are both Strict and Conflict Serializable.

- In this variation, a Transaction T that issues a R_item(X) or W_item(X) such that TS(T) > W_TS(X) has its read or write operation delayed until the Transaction T' that wrote the values of X has committed or aborted.

- Every transaction is assigned with a unique integer sequence number.

- TS(T9) is the timestamp of T9 which is the sequence number of the transaction = 9 .
- Consider two transactions T5 and T8, where T8 has started after T5. Hence, T8 is younger to T5. Also, TS(T8) > TS(T5).

T9 requests write (X):

```
If ((R_TS(X) > TS(T9)) OR (W_TS(X) > TS(T9))) {

        T9 will abort; //some younger trans. has
        read/written X
} Else {

        T9 will wait till the youngest transaction
        (W_TS(X) ) that has written X, is
        committed/aborted.

        T9 performs write (X);
        W_TS(X) = TS(T9);
 }
```

T9 requests read(X):

```
If (W_TS(X) > TS(T9)) {

        T9 will abort; //some younger trans. has
        written X
} Else {

        T9 will wait till the youngest transaction
        (W_TS(X) ) that has written X, is
        committed/aborted.

        T9 performs Read (X);
        R_TS(X) = max(R_TS(X),TS(T9));
 }
```

THOMAS'S WRITE RULE

- **Thomas Write Rule** allows such operations and is a modification of the Basic Timestamp Ordering protocol. In Thomas Write Rule users ignore outdated writes.

- Moreover, of all the Concurrency Protocols that have been discussed, Concurrency is imposed on Schedules that are Conflict Serializable, in Thomas Write Rule, the most important improvement is a user can achieve Concurrency with View Serializable schedules.

THOMAS'S WRITE RULE CONT.

Thomas Write Rule does not enforce Conflict Serializability but rejects fewer Write Operations by modifying the check Operations for W (X) .

- If **R_TS(X) > TS(T)**, then abort and roll back T and reject the operation.
- If **W_TS(X) > TS(T)**, then don't execute the Write Operation and continue processing. This is a case of *Outdated or Obsolete Writes*. Remember, outdated writes are ignored in Thomas Write Rule but a Transaction following the Basic TO protocol will abort such a Transaction.
- If neither the condition in 1 or 2 occurs, then and only then execute the **W (X)** operation of T and set **W_TS(X) to TS(T).**

DIFFERENCE BETWEEN BASIC TO PROTOCOL & THOMAS WRITE RULE

BASIC TO PROTOCOL	THOMAS WRITE RULE
• Not allowed R1(X) W2(X) W1(X)R2(X) W1(X)W2(X) • Allowed All Operations where T2 occurs before T1. R1(X)R2(X)	• Not allowed R1(X) W2(X) W1(X)R2(X) • Allowed All Operations where T2 occurs before T1. Outdated Writes: $W_1(X)$ $W_2(X)$ R1(X)R2(X)

DATABASE RECOVERY TECHNIQUES

Types of Recovery Techniques in DBMS

- Database recovery techniques are used in database management systems (DBMS) to restore a database to a consistent state after a failure or error has occurred.

- The main goal of recovery techniques is to ensure data integrity and consistency and prevent data loss.

- There are mainly two types of recovery techniques used in DBMS
  - **Rollback/Undo Recovery Technique**
  - **Commit/Redo Recovery Technique**

UNDO, REDO operations

- To maintain atomicity, a transaction's operations are redone or undone.
  - Undo: Restore all BFIMs on to disk (Remove all AFIMs).
  - Redo: Restore all AFIMs on to disk.

- Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two.

- These operations are recorded in the log as they happen.

WHAT IS ROLLBACK?

- If transaction is failed then we have recovery management system.

- With the help of rollback , transaction recovery will take place. The rollback will take place with the help of transaction logs. logs are the file that keeps a record of all the activity which is done by the transaction.

- The rollback undo all the modifications that has taken place during a transaction.

CASCADING ROLLBACK

- It is formed using two different words which are cascade and Rollback.
- The word cascade means, “waterfall” and rollback means, “ an act of making an action to change back to what it was before”.

- Due to the failure of a single transaction a cascade of transaction rollbacks. This is known as cascading rollback. For instance we can refer to the below mentioned transaction.

CASCADING ROLLBACK

T1	T2	T3
Read(A)		
Read(B)		
Write(A)		
	Read(A)	
	Write(A)	
		Read(A)
abort		

DEFERRED UPDATE

Updates are recorded in the log file and applied to the database only upon transaction commit. If a rollback occurs, changes are discarded from both the database and the log file. Upon system restart, committed changes in the log file are applied to the database.

IMMEDIATE UPDATE

Updates are immediately applied to the database, and the log file contains both old and new values. Upon commit, changes are permanently stored in the database. Rollbacks discard changes from both the database and the log file. Committed changes are permanently stored in the database upon system restart.

Difference between Deferred update and Immediate update:

DEFFERED UPDATE	IMMEDIATE UPDATE
• In a deferred update, the changes are not applied immediately to the database.	• In an immediate update, the changes are applied directly to the database.
• The log file contains all the changes that are to be applied to the database.	• The log file contains both old as well as new values.
• In this method once rollback is done all the records of log file are discarded and no changes are applied to the database.	• In this method once rollback is done the old values are restored to the database using the records of the log file.
• Concepts of buffering and caching are used in deferred update method.	• Concept of shadow paging is used in immediate update method.
• The major disadvantage of this method is that it requires a lot of time for recovery in case of system failure.	• The major disadvantage of this method is that there are frequent I/O operations while the transaction is active.

Difference between Deferred update and Immediate update cont:

FEATURES	DEFFERED UPDATE	IMMEDIATE UPDATE
Update timing	Updates occur after instruction execution	Updates occur during instruction execution
Processor speed	May be faster: update occurs after instruction execution, allowing for multiple updates to be performed at once	May be slower: update occurs during instruction execution, potentially causing the processor to stall
Complexity	More complex: requires additional instructions or mechanisms to handle updates after instruction execution	Less complex: updates occur immediately during instruction execution

Difference between Deferred update and Immediate update Cont:

FEATURES	DEFFERED UPDATE	IMMEDIATE UPDATE
Consistency	May result in temporary inconsistency between data in registers and memory	Data in registers and memory are always consistent
Flexibility	May be more flexible: allows for more complex data manipulations and algorithms	Less flexible: immediate updates can limit the range of data manipulations and algorithms that can be performed

IN PLACE UPDATING

- In in-place updating, a page of the DBMS cache is associated with a single, unique page on the permanent secondary storage.
- A cache flush always writes each page of the disk cache to the associated page in secondary storage.

SHADOW UPDATING

- In shadow updating, a page of the DBMS cache may be associated with several (usually two) pages on the permanent secondary storage, all associated with the same logical data item(s).
- Thus, there may be several permanent copies of a data item.
- A cache flush may write the page of the disk cache to a new location on the secondary storage, thus preserving the original data on that store.

Algorithm for Recovery and Isolation Exploiting Semantics (ARIES)

- Algorithm for Recovery and Isolation Exploiting Semantics (ARIES) is based on the Write Ahead Log (WAL) protocol. Every update operation writes a log record which is one of the following :

  - **Undo-only log record:** Only the before image is logged. Thus, an undo operation can be done to retrieve the old data.
  - **Redo-only log record:** Only the after image is logged. Thus, a redo operation can be attempted.
  - **Undo-redo log record:** Both before images and after images are logged.

Advantages of ARIES Algorithm

- **Robust Recovery:** ARIES ensures robust recovery from system failures, minimizing data loss and maintaining database consistency.
- **Efficient Logging:** By employing before-images and after-images, ARIES optimizes logging and recovery operations, reducing overhead.
- **Concurrency Control:** ARIES supports efficient concurrency control mechanisms, allowing for high-performance transaction processing.

ARIES Algorithm Cont.

•ARIES is a powerful algorithm for database recovery and concurrency control, providing essential mechanisms to maintain data integrity and recover from system failures.

•Its adoption in many modern database management systems highlights its significance in ensuring reliability and consistency in data management.

THANK YOU