gradeup

Normalization for Relational Database Part-6

Content :-

1. **Dead Lock Handling**
2. **Dead Lock Recovery**
3. **Concurrency Control**
4. **Locking**
5. **Recovery**

DeadLoack Handling :- A system is said to be in deadlock state when in a set of transactions, every transaction is waiting for another transaction to finish its operations.

A deadlock is a state of statement hat may result when two or more transaction are each waiting for locks held by the other to be released .

There are two principal methods of dealing with deadloack problem.

1. Deadlock preventing protocol
2. Deadlock detection and recovery process.

Deadlock prevention Protocol :- This protocol ensures that the system will not go into deadlock state. There are different methods that can be used for deadlock prevention .

- The simplest scheme requires that each transaction locks all its data item before it start execution
- Another method for the prevention of deadlock is to impose a partial ordering of all data items and require that a transaction can lock a data item only in the order specified by partial order.
- Another approach for the prevention of deadlock is to use pre-emption and transaction rollbacks.
- **Wait-die:** If $TS(T_i) < TS(T_j)$, then (T_i older than T_j) T_i is allowed to wait ; otherwise (T_i younger than T_j) abort T_i(T_i dies) and restart it later with the same time stamp.

- **Wound-wait** : if $TS(T_i) < TS(T_j)$, then (T_i older than T_j) abort, T_j (T_i woulds T_j) and restart it later with the same timestamp; otherwise (T_i younger than T_j) T_i is allowed to wait.

Deadlock Detection :- Deadlock can be described precisely in terms of a directed graph called wait for graph . This graph consist of pair G = (V,G) where V is a set of vertices and E is a set of edges . Each elements in the set E of edges is an ordered pair $T_i \rightarrow T_j$ if $T_i \rightarrow T_j$ is in E , then there is a set of edge that transaction $T_i + T_j$ implying that transaction T is waiting for transaction T to release a data item that it needs.

Note: Whenever graph contain cycle only then the deadlock exists in the system.

Recovery From Deadlock :-

When the detection algorithm determines that the deadlock exists, the system must recover from the deadlock . The most common solution is to rollback one or more transaction to break the deadlock . The action need to be taken

1. **Selection of Victim :-** In this we determine which transaction (or transactions) to be rollback to break the deadlock . We should rollback those transactions that will cause the minimum cost .
2. **Rollback:-** once we decide that a particular transaction must be roll back , we must determine how far this transaction should be roll back. The simplest solution is a “total rollback” . Abort the transaction and then restart it . However, for breaking the deadlock roll back transaction was done as far as necessary. Such partial rollback requires the system to maintain additional information about the state of all the running transaction .

3. **Starvation :-** In a system where a selection of transaction for roll back is based on the cost factor, it may happen that the same transaction are always picked up.

Concurrency Control :- It is a management of concurrent transaction execution. DBMS implement concurrency control techniques to ensure serializablity and isolation of transaction in order to guard the consistency and integrity of the database.

Concurrency control can be performed by DBMS , with various methods such as locking methods, timestamp methods , validation based techniques and optimistic method.

Concurrency Control is process of managing simultaneous operations, queries , updates , insert , deletes o the database without having them interfere with one another .

The objective of concurrency control is similar to the objective of multi-user computer system where many users can perform , different operations at the same time. In these system the different users are allowed to perform different actions simultaneously due to concept called multi-programming .

Types of problems we may encounter with the transactions if they run concurrently

1. **Lost Update Problem :-** This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes that value of some database item incorrect.
2. **The Temporary Update(or Dirty Read) Problem :-** This problem occurs when one transaction updates a database item and then the transaction updates a database item and then the transaction fails for some reason. The updated item is accessed by another transaction before it is changed back to its original value.

3. **The Incorrect Summary Problem** :- If one transaction is calculated an aggregate summary function on a number of records while other transactions are updating some of these records , the aggregate function may calculate some values before they are updated and others after they are updated.

Another problem that may occur is called **unrepeatable read,** where transaction T reads an item twice and the item is changed by another transaction T' between the two read. Hence, T receives different values for its two reads of the same item.

Locking Techniques for Concurrency Control:-

Locking is a procedure used to control concurrent access to data. When one transaction is accessing the database , a lock may be deny access to other transactions to prevent incorrect locking is one of the most widely used mechanism to ensure serializability .

Rules of locking are :-

1. If a transaction has a read lock on a data item , it can read the item but not update it.
2. If a transaction has a read lock on a data item , the other transactions can obtain that data item , but not write lock
3. If a transaction has a write lock on a data item , it can both read and update the data item
4. If a transaction has a write lock on a data item, then other transactions cannot be obtain either a read lock or a weak lock on the data item .

Mode of Locking:

There are basically two mode of locking :-

1. **Shared Lock**
2. **Exclusive Lock**

1. **Shared Lock :-** If a transaction T_1 has obtained a shared mode lock on item Q then T_j can both read and write O . It is denoted by (X) .
2. **Exclusive Lock:-**If a transaction T_j has obtained an exclusive-mode lock on item O , then T_j can both read and write Q , it is denoted by (X).

Two-Phase Locking Protocol :- The two phase locking protocol ensures serializability. This protocol requires that each transaction issue lock and unlock requests in two phase:

1. Growing Phase
2. Shrinking Phase

1. **Growing Phase :-** In this phase a transaction may obtain locks but may not release any lock. Here the number of locks increases from zero to the maximum for the transaction.
2. **Shrinking Phase :-** In this phase a transaction ay release locks but may not obtain any new locks. Here the number of locks decreases from maximum to zero.

There are three types of two-phase locking protocol:

- **Conservative Two-phase Locking**
- **Strict Two-phase Locking**
- **Rigorous two-phase Locking**

Conservative two-phase locking: It requires a transaction to pre-declare all the locks it requires before its execution. If any of pre-declared items cannot be locked the transaction waits until all item are locked.

Note :- The locking protocol is deadlock free but it is difficult to use in practice.

Strict two phase locking:- In this protocol a transaction does not release any of its write lock until it commits or aborts.

Rigorous two phase locking:- In this protocol , a transaction does not release any of write lock and read lock until it commits or aborts .

Time Stamping Protocols:- Time stamping is a concurrency control protocol in which the fundamental goal is to order transactions globally in such a way that order transactions get priority in the event of a conflict. In the time stamping method , if a transaction attempts to read or write a data item , then a read or write operation is allowed only if the last update on the data item was carried out by an older transaction, otherwise the transaction , requesting the read or write is re-started and given a new timestamp.

If any conflict is found between transaction using timestamp based system, one of the conflicting transactions is rolled back.

- When a transaction having smaller timestamp value tries to access a data value that is writing by the transaction having larger timestamp value .
- When transaction with a smaller timestamp value tries to update value already read or written by a transaction having larger timestamp value.

Recovery

To be able to recover from failures that affect transactions , the system maintains a log to keep track of all transactions, operations that affect the values of the database item . Permit from recovery failure needs this information. The log is kept on disk, so it is not affected by any type of failure except for the disk or catastrophic failure. In addition , the log is periodically backed up to archival storage (tape) to guard against such catastrophic failure .

The Recovery Algorithm :-

It consists of of : analysis , REDO and UNDO. The **REDO** phase actually replies updates from the log to the database . Generally , the REDO operation is applied only to commit transactions. **UNDO** phase , the log is scanned backward and the operations of transactions that were active at the time of the crash are undone to reverse order.

Log sequence number(LNS)is associated with every log recordthat is monotonically increasing and indicates the address of log record on the disk. Each LNS corresponds to specific changes (action) of some transaction. A log record written any of the following actions: updating a page(write), committing a transaction(commit) , aborting a transaction(abort) , undoing an update(undo), and ending transaction (end) .

Beside the log , two tables are needed for efficient recover : the Transaction Table and the Dirty Page Table, which are maintained by the transaction manager . These tables are rebuilt in the analysis phase of recovery whenever a crash occurs . The Transaction Table contains an entry for each active transaction, with information such as the transaction ID, transaction status , and the LNS of the most recent log record for the transaction . The Dirty Page Table contains an entry for each dirty page in the buffer , which includes the page ID and the LNS corresponding to the earliest update to the page.

Checkingpointing :- in consists of the following : writing a being_checkpoint record to the log , writing an end_checking record to the log , and writing the LNS of the being_checkpoint records to a special file.

Transaction Recovery: A transaction beings with a successful execution of a BEING TRANSACTION statement and ends with the successful execution of a COMMIT and ROLLBACK statement . Thus , a COMMIT establishes a **COMMIT POINT(also called synch-point)** at which database is in state of consistency . A ROLLBACK rolls back the database to previous COMMIT POINT , at which again the database was in state of consistency.

When COMMIT POINT is establish : -

1. All the updates made by the program since the previous COMMIT POINT , are committed i.e. are made PERMANENT
2. All database pointers (i.e. addressability to certain tuples) and all tuple-locks will be released . Some systems provide an option to retain addressability to tuple (and also retain tuple locks) from one commit point to next.

gradeup SUPER

Gradeup UGC NET
Super Superscription

Features:

1. 7+ Structured Courses for UGC NET Exam
2. 200+ Mock Tests for UGC NET & MHSET Exams
3. Separate Batches in Hindi & English
4. Mock Tests are available in Hindi & English
5. Available on Mobile & Desktop

Gradeup Super Subscription, Enroll Now

gradeup.co