International Journal of Database Theory and Application
Vol.8, No.2 (2015), pp. 259-266
http://dx.doi.org/10.14257/ijdta.2015.8.2.24

Relational Database's Transaction Operation and the Concurrent Control

Na Liu and Jianfei Zhou

School of Information Engineering, Chongqing Industry Polytechnic College
na814@qq.com
Information Center, Chongqing Industry Polytechnic College
fn219@qq.com

Abstract

This paper puts forward the strategy of concurrent control that would solve the concurrent operation, which results from the transaction object in order to ensure the data integrity.

Keywords: *relational database, transaction, ACID, concurrent operation, concurrent control*

1. Introduction

Database technology is a new technology using computer to manage the data. In the fields of science and technology, culture, economy and military affairs, we will encounter a lot of data, These data are complex, and the large amount of data, so how to scientific management of data is a very important topic, especially when multiple objects in the same data access at the same time, it may bring a series of data error.

Transaction and Lock are two closely related concepts. Transaction uses Lock to prevent other users from modifying the data has not yet completed in another transaction. There are many locks in a relational database, allowing the transaction to lock different resources, The lock is to protect the specified resource, not be operated by other transactions, the Lock is to ensure concurrency control methods, locking in the process is called the concurrency control mechanism.

2. Transaction

2.1. Concept

The transaction in the database logical unit of work is a set of operation sequences of user defined; these operations do, or not do. Such as transfer transactions, including funds transferred in and out of the two operations, only when the two operations are completed, the transfer can be successful, or two operations do not, it will not affect the original funds, assuming only completed out of an operation, it will cause the loss of funds.

2.2. ACID Feature

Database transaction must have ACID characteristics, ACID is the Atomic, Consistency, Isolation and Durability of English abbreviations.

The ACID feature of transaction is achieved by RDBMS (Relational Database Management System). DBMS (Database management system) uses the log to guarantee the atomicity, consistency and durability of the transaction. The log records the update that the transaction made to the database, if one transaction error occurs during execution, you can undo the update that the transaction made to the database according to the log, so the database back to the initial state of transaction before the execution.

DBMS uses the lock mechanism to achieve transaction’s isolation. When multiple transactions simultaneously update the same data in the database, allowing only the transaction that holding the lock can update the data, other transactions must wait until the lock is released by previous transaction, and then other transactions have the opportunity to update the data.

2.2.1. Atomic: A transaction is an indivisible unit of work. Transactions must be atomic unit of work; for the modification of data, either executed for all or none. Usually, the operations that associated with a transaction have the same goal, and are interdependent. If the system performs only a subset of these operations, it may undermine the overall objective of the transaction. Atomic feature eliminates the possibility of the system processing the subset.

2.2.2. Consistency: Consistency feature refers to the consistent state that all the data must be made, when the transaction is completed. In the relational database, all the rules must be applied to the transaction of the modification, in order to maintain the integrity of all data. All internal data structures must be ensured the accuracy at the end of the transaction. The result of transaction execution must make the database from one consistent state to another consistent state.

2.2.3. Isolation: The execution of one transaction is not interfered by other transactions, internal transaction operation and the use of data are separated from other concurrent transactions. It refers to the modification of each concurrent transaction must be independent of another. One transaction can see the data that either before another transaction modify it, or after it has been modified, but this transaction cannot see being modified data, this is also known as serial feature.

2.2.4. Durability: Once the transaction submitted, the change of data in the database is permanent. Durability refers to the transaction’s effect permanently generated in the system, when it is completed, that is this modification been wrote in the database. After the transaction has completed, it is permanent for system effects, even if the modification cause the fatal system failure.

2.3. Statement Definition

In the relational database, a transaction can be a SQL statement, a set of SQL statements or the entire program. In the SQL statement that defines the transaction statement has three:

Begin statement: Begin transaction

End statement: (1) Commit means all operations of committing the transaction, updating database, and the normal end of the transaction.

(2) Rollback means the occurrence of certain fault in the transaction process, the transaction cannot be executed continuous, cancel all data operation, rollback the transaction to started state.

2.4. Execute Process (Figure 1)

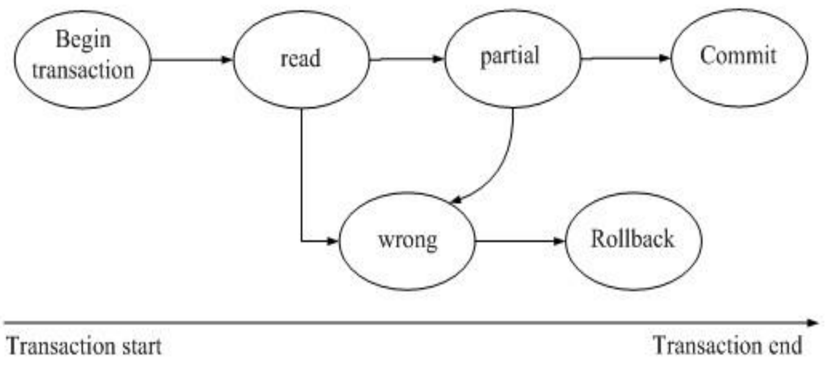


Figure 1. The Transaction's Executing Process

3. Concurrent Operation

3.1. Concept

In order to make full use of database resources, give play to resource sharing of the database, allows multiple users to access the database concurrently that is multiple transactions operate the same data object at the same time, called the concurrent operation.

3.2. Data Inconsistency

If there is no locking and multiple users access a database simultaneously, it may occur some problem when their transactions use the same data at the same time, lead to the data inconsistently in the database. Concurrent operation the most common example is the booking operation in the train/airline reservation system.

For example, a sequence of activities in the system:

1. Conductor A read out a flight ticket number of balance X, set X=10;
2. Conductor B read out the same flight ticket number of balance X, also 10;
3. Conductor A sell a ticket, modify the ticket number of balance X=X-1=9, and write X back to the database;
4. Conductor B also sell a ticket, modify the ticket number of balance X=X-1=9, and write X back to the database.

The results clearly sold two tickets, ticket balance only 1 reduction in database.

This is called the inconsistency of the database. This inconsistency is caused by A and B two conductors' concurrent operations. In the case of concurrent operation, it randomly scheduled A and B two transactions' operating sequence. If the scheduling sequence was executed for the above, the transaction A's modifications would be lost. This is due to the transaction B modifies X and write back to cover transaction A modifications in the Step 4.

Concurrent operation brings the database Inconsistency can be divided into three categories: Lost update, dirty read, and non-repeatable read.

3.2.1. Lost Update (Figure 2): Lost Update refers to the transaction A and B read and modify the same data from the database, the submitted result of transaction B destroyed the submitted result of transaction A, and causes the modification of transaction A was lost. Such as the air ticket booking system, at the same time in different places, read the same flight information simultaneously, respectively in the operation of the database, and submit a ticket balance information, finally caused the balance data error.

Time	Transaction A	Transaction B
T1	Read N	Read N
T2	N←N-M (Write back to N)	
T3	COMMIT	N←N-H (Write back to N)
T4		COMMIT

Figure 2. Lost Update

3.2.2. Dirty Read (Figure 3): Dirty Read refers to the transaction A modify a data, and writes the result back to the database, after the transaction B reads the same data, transaction A is cancelled due to some fault, then the data that transaction A has modified recover the original value, the data that the transaction B has read are inconsistent with that in the database, said as the Dirty Read. For example, in the air ticket booking process, after selling a ticket, you submit the information of balance, and do the ticketing operation according to the updated balance, and then the tickets which were sold a moment ago are failed to issue due to system failure, resulting in failure of the booking, eventually lead to balance data error.

Time	Transaction A	Transaction B
T1	Read N N←N-M (Write back to N)	
T2		Read N
T3	ROLLBACK (Retrieve N)	

Figure 3. Dirty Read

3.2.3. Non-Repeatable Read: Non-Repeatable Read refers to after the transaction A reads the data, transaction B performs an update operation, it makes transaction A unable to reproduce the results last time. Such as the air ticket booking system, the ticket balance is in the update, but the balance data which were read out last time will be unable to know after the update.

4. Concurrency Control

4.1. Concept

Concurrency control refers to use the correct method of scheduling concurrent operation, access to data queued to avoid data inconsistency, so that a user transaction execution without interference of other transactions.

The purpose of concurrency control is to ensure that a user's work will not generate Influence of unreasonable to another user's work. In some cases, these measures ensure that when the user and other users operate together, the results are the same with the separate operation of the result. In other cases, this means that the user working in a predetermined manner affected by other users.

The method and technique to avoid data inconsistency is concurrency control, the most commonly used concurrency control technique is the lock mechanism.

4.2. Lock Mechanism

Lock refers to before the transaction A operate a data objects such as tables, records, send the request of locking this data to the system in advance. After locking the transaction A has a certain control to the data object, other transactions can't update this data object until transaction A release its lock. There are two basic types of lock:

4.2.1. Exclusive Locks (X Locks for Short): X Lock is also known as the write lock, if transaction A give a X Lock to the data object N, only allows the transaction A to read and modify the data object N, any other transactions can not give any type of lock to the data object N, until the transaction A release the X Lock on the data object M, it ensures that the other transactions can't read and modify the data object N until the transaction A releases the lock on the data object M before. (Figure 4)

Time	Transaction A	Transaction B
T1	XLOCK N Read N N←N-M (Write back to N)	
T2		Wait
T3	ROLLBACK (Retrieve N)	
T4		Read N

Figure 4. With X Lock Solve Dirty Read Error

4.2.2. Share Locks (S Lock for Short): S Lock is also known as the read lock, if transaction A give a S Lock to the data object N, the transaction A can read the information of N, but cannot modify N, other transactions only can give the S Lock to the data object N, but not X lock, until the transaction A release the S Lock on the data object N, it ensures that the other transactions can read the data object N, but cannot make any changes on the data object M before the transaction A releases the S Lock on the data object N. (Figure 5).

Time	Transaction A	Transaction B
T1	SLOCK N Read N	
T2		SLOCK N Read N
T3	Wait	Wait
T4	Unlock N	

Figure 5. With S Lock Resolve Lost Update Error

4.3. Locking Protocol

The purpose of the locking is to guarantee that proper scheduling of concurrent operations. Therefore, in the use of X Lock and S Lock the two basic locks on a certain size of data object lock, also need to agree some rules, for example, when they should apply for X lock or S Lock, the lock holding time, when to released etc., we refer to these rules as the locking protocol.

By setting the different rules to each locking ways, formed a variety of locking protocol, they are in different extent to provide a guarantee for the proper scheduling of the concurrent operations. The incorrect scheduling of the concurrent operations may bring about three kinds of data inconsistency: Lost Update, Dirty Read, and Non-Repeatable Read. The locking protocol for ensuring data consistency use three levels locking protocol, it respectively in different degree solved this problem.

4.3.1. Level 1 Locking Protocol: The level 1 locking protocol content is: The transaction T must give the X Lock to data R at first before modifying it, until the end of the transaction before the release. The ends of the transaction include normal end (Commit) and non-normal end (Rollback). Level 1 locking protocol can prevent Lost Update, and ensure the transaction T can be restored. The use of level 1 locking protocol solved the Lost Update problem of the ticket reservation example (Figure 6).

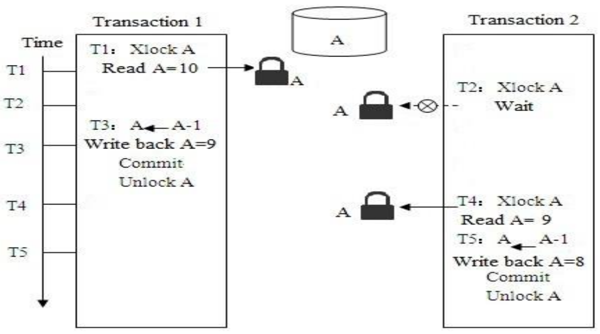


Figure 6. Level 1 Locking Protocol

As shown in Figure 6, the transaction 1 gave the X Lock to data A before modifying it, the transaction 2 was refused to give the X Lock to data A, can only wait for transaction 1 released the lock on the data A. After transaction 1 modified the value of A=9, written back to disk, and released the X Lock on data A, transaction 2 obtained the X Lock on data A, the data A what was read by transaction 2 at this time was the value 9 that had been updated by transaction 1, then in computing according to this new data A value, and written the result value A=8 back to disk. This avoided the loss of transaction 1's update.

In the level 1 locking protocol, it is not necessary to lock if only to read data without modifications, so it cannot guarantee repeatable read and Dirty Read.

4.3.2. Level 2 Locking Protocol: The Level 2 locking protocol content is: Level 1 locking protocol and the transaction T must give the S Lock to the data R at first before reading it, the S Lock can be released after reading. In addition to prevent the Lost Update, level 2 locking protocol can further prevent Dirty Read. The use of 2 protocols solves the problem of reading "dirty" data (see Figure 5). The use of level 2 locking protocol solved the problem of Dirty Read (Figure 7).

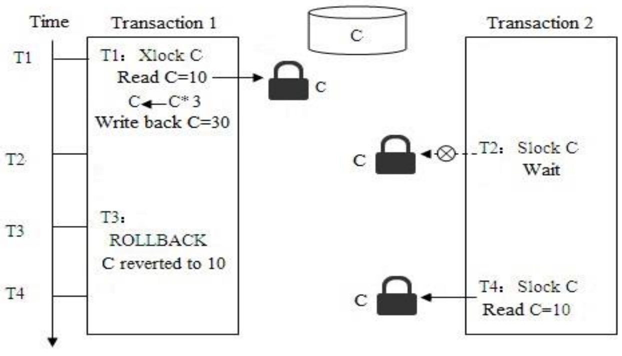


Figure 7. Level 2 Locking Protocol

As shown in Figure 7, before the transaction 1 modified the data C, it gave the X Lock to the data C at first, then modified its value and written back to disk. The transaction 2 requested to give the S Lock to data C at this time, but as the data C had been given the X Lock by the transaction 1, the transaction 2 can only waited until the transaction 1 release it. After the transaction 1 was revoked for some reasons, data C reverted to the original value of 10, and released the X Lock. The transaction 2 got the S Lock on data C, and read value C=10. This avoided the transaction 2 reading dirty data.

In the level 2 locking protocol, the S Lock can be released after reading the data, so it cannot guarantee repeatable read.

4.3.3. Level 3 Locking Protocol: The Level 3 locking protocol content is: Level 1 locking protocol and the transaction T must give the S Lock to the data at first before reading it, and not release until the end of the transaction. In addition to prevent the Lost Update and not read “dirty” data, the level 3 locking protocol can further prevent the Non-Repeatable Read. The use of level 3 locking protocol solved the problem of the Non-Repeatable Read (Figure 8).

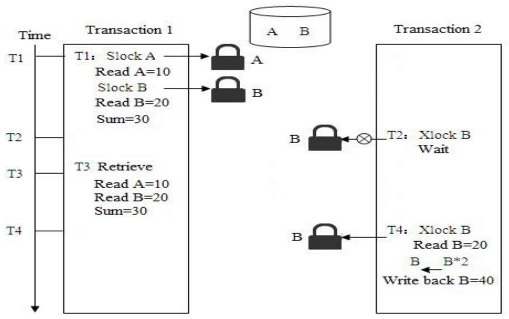


Figure 8. Level 3 Locking Protocol

As shown in Figure 8, the transaction 1 gave the S Lock to both data A and B before reading them, so other transactions could only give the S Lock to both data A and B, but not X Lock, that is to say other transactions can only read both data A and B, but cannot modify them. So the Transaction 2 was rejected when it asked for X Lock on data B which was to be modified, and the other modifications cannot be performed, only wait for the transaction 1 to release the lock on the data B. Then the transaction 1 read both data A and B again for checking, the value of data B is still 20, and the summation of the result is still 30, that is repeatable read.

The main difference between these three protocols is what operations need to apply for the locking and when to release the lock (i.e. the lock holding time). The three levels locking protocol can be summarized in the table below (Figure 9).

Locking Protocol	X Lock		S Lock		Consistency of data		
	Release after operation	Release after transaction	Release after operation	Release after transaction	Non-Lost Update	Non-Dirty Read	Repeatable Read
Level 1		√			√		
Level 2		√	√		√	√	
Level 3		√		√	√	√	√

Figure 9. The Summarization of the Three Levels Locking Protocol

Before the Database management system performs the read and writes operations on data, you should perform the lock operation on data, according to some locking protocols, to control the concurrent operation, enable multiple concurrent operation implementations orderly, and thus avoid Lost Update, Non-Repeatable Read and Dirty Read, such as data inconsistency.

5. Summary

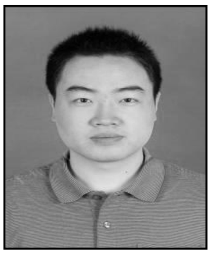
This paper through the description of the transaction, and the problems that the transaction's concurrent operation may bring about, put forward the corresponding concurrency control strategy, provides the control mechanism guarantee for the management and maintenance of data.

References

[1] S. Shixuan and W. Shan, "Database System Overview", Higher Education Press, **(2000)**, no. 3, pp. 264-267.
[2] L. Qifen, "SQL Server practical tutorial", Electronic Industry Press, **(2001)**.
[3] H. Yujie, "Database foundation and application technology", Tsinghua University Press, **(2003)**.
[4] J.D. Ullman, "Principles of database and knowledge-base systems", Stanford University, **(1999)**.

Authors

Na Liu, received the Bachelor degree in Engineering in College of computer and Information Science from Southwestern Normal University, and the Master's degree of Engineering in Computer Technology field From College of Computer Science of Chongqing University, China in 2004 and 2013 respectively. She is currently researching on Database technology, Graphics and Image Processing.



Jianfei Zhou, received the Bachelor degree in Engineering in College of computer and Information Science from Southwestern Normal University, and the Master's degree of Engineering in Computer Technology field From College of Computer Science of Chongqing University, China in 2004 and 2012 respectively. He is currently researching on Computer network, Information security, Graphics and Image Processing.