Variorum Multi-Disciplinary e-Research Journal
Vol.,-05, Issue-II, May 2014

Avertable Occurrence and Horrific Addiction of Lost Update Problems in SQL Server

Mr. Bosco Paul Alapat: Lecturer, Department of Computer Science, Adigrat University, Ethiopia

Abstract

The purpose of the study was to find out the solution for the Avertable Occurrence and Horrific Addiction of lost update problems in SQL server .When two or more transactions select the same row and try to update the row, each transaction is unaware of other transactions. The last update overwrites the updates made by the other transactions, in which the data can be lost or the data can be overwritten. In order to analyze the operation on data, different locking methods were used to find the level of significance. There were significant results in the modification of data, coping bulk of data at the same time , the security during transaction. Microsoft SQL Server uses locking to ensure transactional integrity and database consistency. Locking prevents users from reading data being changed by other users, and prevents multiple users from changing the same data at the same time. When locking is not used, data within the database may become logically incorrect, and queries executed against that data may produce unexpected results.

Keywords

Schema Modification (Sch-M) Locks, Schema Stability (Sch-S) Locks, Shared Locks, Update Locks, Intent Locks.

Introduction

When many people attempt to modify data in a database simultaneously, a system of controls must be implemented so that modifications made by an individual do not adversely affect the same on another individual. Microsoft SQL Server supports a wide range of optimistic and pessimistic concurrency control mechanisms. However, the following problems may occur if two or more transactions use the same data at the same time. The major problem of concurrency control method is the lost, or buried update problem. SQL Server 2000 also uses several different background processes to efficiently manage computer resources. One example is checkpoints. SQL Server periodically generates automatic checkpoints in each database. Checkpoints flush grimy data and log pages from the buffer cache of the current database, minimizing the number of modifications that have to be rolled forward during a recovery. Another example of background process is called Lazy writer which is unique to each instance. The lazy writer process sleeps for an interval of time then wakes to scan through the buffer cache where it checks the size of the free buffer list. If the free buffer list is below a certain point (dependent on the size of the cache) the lazy writer process scans the buffer cache to reclaim unused pages and write dirty pages that have not been recently referenced, while frequently referenced pages remain in memory.

Lost updates occur when two or more transactions select the same row and then update the row based on the value originally selected. Each transaction is unaware of other transactions. The last update overwrites updates made by the other transactions, which results in lost data.

For example, two editors make an electronic copy of the same document. Each editor changes the copy independently and then saves the changed copy, thereby overwriting the original document. The editor who saves the changed copy last overwrites changes made by the first

editor. This problem could be avoided if the second editor could not make changes until the first editor had completely finished.

The Locks object in Microsoft SQL Server provides information about SQL Server locks on individual resource types. Locks are held on SQL Server resources, such as rows read or modified during a transaction, to prevent concurrent use of resources by multiple transactions.

SQL Server 2000 supports the following lock modes:

Shared locks

Shared (S) locks are used for operations that read data, such as a SELECT statement. During Shared (S) locks used, concurrent transactions can read (SELECT) a resource, but cannot modify the data while Shared (S) locks exist on the resource. If you do not use the HOLDLOCK locking hint and your transaction isolation level is not set to REPEATABLE READ or SERIALIZABLE, the Shared (S) locks on a resource are released as soon as the data has been read. If you use the HOLDLOCK locking hint or your transaction isolation level is set to REPEATABLE READ or SERIALIZABLE, the Shared (S) locks on a resource will be held until the end of the transaction.

By the way, when you select database in the Enterprise Manager and then click Tables, the Shared (S) lock will be placed on this database, but you can insert/delete/update rows in the tables in this database.

Update Locks

Update (U) locks are used when SQL Server intends to modify a row or page, and later promotes the update page lock to an exclusive lock before actually making the changes. The Update (U) locks are used to prevent a deadlock. For example, if two transactions intend to update the same row, each of these transactions set the shared lock on this resource and after that tried to set the exclusive lock. Without Update (U) locks, each transaction will wait for the other transaction to release its shared-mode lock, and a deadlock will occur.

To prevent a potential deadlock, the first transaction which tried to update the row will set the Update (U) lock on this row. Because only one transaction can obtain an Update (U) lock to a resource at a time, the second transaction will wait until the first transaction convert the update lock to exclusive lock and release the locked resource.

Exclusive Locks

Exclusive (X) locks are used for data modification operations, such as UPDATE, INSERT, or DELETE. Other transactions cannot read or modify data locked with an Exclusive (X) lock. During the Shared (S) exists, other transactions cannot acquire an Exclusive (X) lock.

Intent Locks

Intent locks are used when SQL Server wants to acquire a shared lock or exclusive lock on some of the resources lower down in the hierarchy.

Intent locks include:

1. Intent Shared(IS)
2. Intent Exclusive(IX)
3. Shared with Intent Exclusive(SIX)
4. Intent Update(IU)
5. Update Intent Exclusive(UIX)
6. Shared Intent Update(SIU)

Intent shared (IS) locks are used to indicate the intention of a transaction to read some resources lower in the hierarchy by placing Shared (S) locks on those individual resources.Intent

exclusive (IX) locks are used to indicate the intention of a transaction to modify some resources lower in the hierarchy by placing Exclusive (X) locks on those individual resources.Shared with intent exclusive (SIX) locks are used to indicate the intention of the transaction to read all of the resources lower in the hierarchy and modify some resources lower in the hierarchy by placing Intent exclusive (IX) locks on those individual resources.

Intent update (IU) locks are used to indicate the intention to place Update (U) locks on some subordinate resource in the lock hierarchy.Update intent exclusive (UIX) locks are used to indicate an Update (U) lock hold on a resource with the intent of acquiring Exclusive (X) locks on subordinate resources in the lock hierarchy.Shared intent update (SIU) locks are used to indicate shared access to a resource with the intent of acquiring Update (U) locks on subordinate resources in the lock hierarchy.

Schema Locks

Schema locks are used when an operation dependent on the schema of a table is executing.

Schema locks include:

1. Schema modification(Sch-M)
2. Schema stability(Sch-S)

Schema modification (Sch-M) locks are used when a table data definition language (DDL) operation is being performed.

Schema stability (Sch-S) locks are used when compiling queries. This lock does not block any transactional locks, but when the Schema stability (Sch-S) lock is used, the DDL operations cannot be performed on the table.

Bulk Update Locks

Bulk Update (BU) locks are used during bulk copying data into a table when one of the following conditions exists:

1. TABLOCK hint is specified
2. Table lock on bulk load table option is set using sp_tableoption

The bulk update table-level lock allows processes to bulk copy data concurrently into the same table while preventing other processes that are not bulk copying data from accessing the table.

Key-Range Locks

Key-Range locks are used by SQL Server to prevent phantom insertions or deletions into a set of records accessed by a transaction. Key-Range locks are used on behalf of transactions operating at the serializable isolation level. Shared Key-Range and Shared Resource (RangeS_S) locks are used to indicate a serializable range scan.Shared Key-Range and Update Resource (RangeS_U) locks are used to indicate a serializable update scan.

Insert Key-Range and Null Resource (RangeI_N) locks are used to test ranges before inserting a new key into an index.Exclusive Key-Range and Exclusive Resource (RangeX_X) locks are used when updating a key in a range.There are also Key-Range conversion locks.

Key-Range conversion locks include:

RangeI_S , RangeI_U , RangeI_X , RangeX_S , RangeX_U

Key-Range conversion locks are created when a Key-Range lock overlaps another lock.

1. RangeI_S locks are used when RangeI_N lock overlap Shared (S) lock.
2. RangeI_U locks are used when RangeI_N lock overlap Update (U) lock.
3. RangeI_X locks are used when RangeI_N lock overlap Exclisive (X) lock.

Variorum Multi-Disciplinary e-Research Journal
Vol.,-05, Issue-II, May 2014

4. RangeX_S locks are used when RangeI_N lock overlap RangeS_S lock.
5. RangeX_U locks are used when RangeI_N lock overlap RangeS_U lock.

Key-Range conversion locks are rarely used and can be observed for a short period of time under complex circumstances.

Methodology

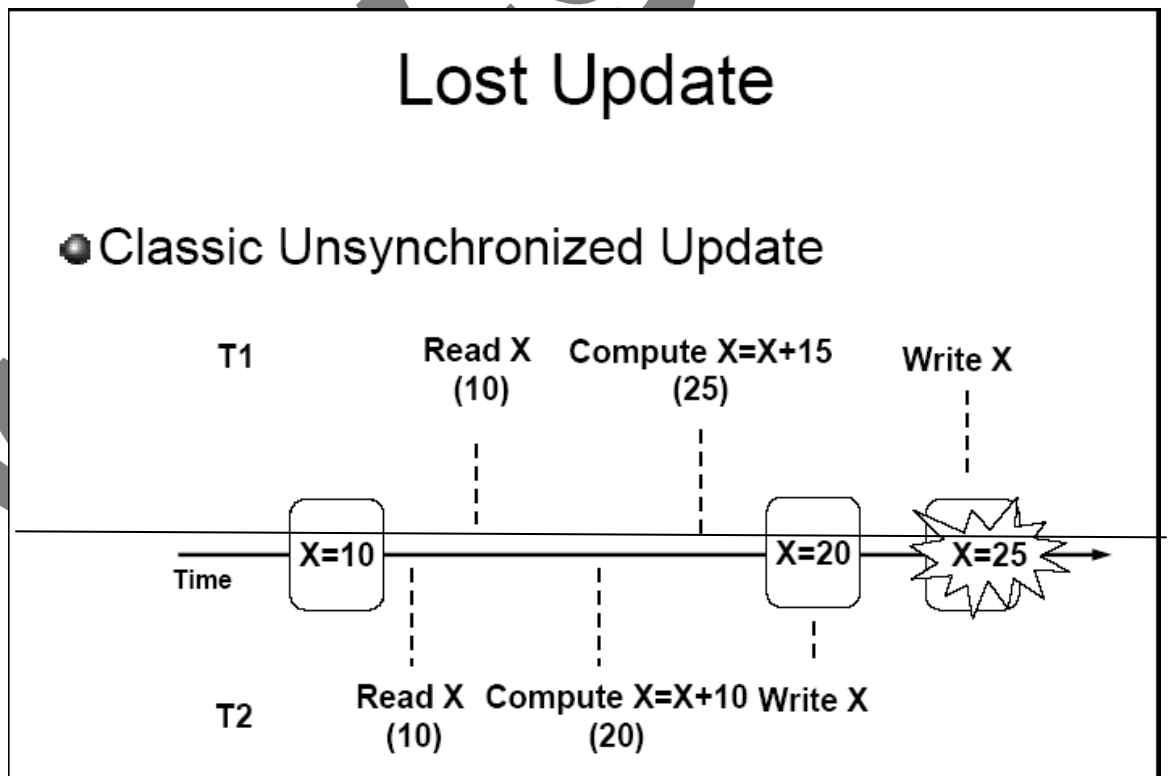
To achieve the purpose of study, different locks methods was used at different transactions to see the integrity of the result that occurs in the database. The locks that were used was shared lock, update lock , intent lock and schema lock.

1. Shared (S) locks are used for operations that read data, such as a SELECT statement.
2. Update (U) locks are used when SQL Server intends to modify a row or page, and later promotes the update page lock to an exclusive lock before actually making the changes.
3. Intent locks are used when SQL Server wants to acquire a shared lock or exclusive lock.
4. Schema locks are used when an operation dependent on the schema of a table is executing.

For every different lock that was used as an example really gave a great difference in the data security and also in the data transaction. With the use of Schema modification(Sch-M) locks, all the data operations was tried during data definition stage and the Schema stability(Sch-S) locks was tired during the time of compiling queries.

Analysis of result

The lost update problem arises when two or more transactions select the same row and then update the row based on the value originally selected. Because each transaction is unaware of other transactions, the last update overwrites the updates made by the other transactions. Therefore, data has been lost.The last update overwrites the updates made by the other transactions. Therefore, data has been lost [KAL01].



Variorum Multi-Disciplinary e-Research Journal
Vol.,-05, Issue-II, May 2014

Fig 1: lost update problem

The above diagram shows that the good example of Lost Update problem. Here there are two transactions T1 and T2 respectively. The row of data is x=10. First T2 took the value of 10 and it computes the evaluation. The evaluation is X is originally 10. With 10 the transaction T2 adds 10. So, the value of X is changed. Its value is 20. After the computation is over, T2 now writes or commit its value to the source place. So the basic value of 10 is changed and now the value of X is 20, because of T2 commit this value. After reads the T2, T1 also reads the value of X, T1 now computes the evaluation. The evaluation is X is originally 10. With 10 the transaction T1 adds 15. So the value of X is changed, X value is 25. But before that T2 committed the value that X is 20, but now T1 is going to commit the vale as 25. Now there is a collision occurred or the first transaction (T2) data has been lost in this circumstance.

Conclusion

The following conclusion where drawn based on the result of the study.

1. Microsoft SQL Server uses locking to ensure transactional integrity and database consistency.
2. Locking prevents users from reading data being changed by other users, and prevents multiple users from changing the same data at the same time.
3. If locking is not used, data within the database may become logically incorrect, and queries executed against that data may produce unexpected results.

Recommendation

1. The same problems can also occurs in the different version of SQL but this method can give a solution for a wide range of problems what come across in the database.
2. By this solution there will not be any data collision in the database during the transactions and users would most feasible of updation , modification and insertion of data.

References

Annop Siritikul, "Microsoft SQL Server 2000 Database Development Overview", http://download.microsoft.com/download/1/e/7/1e7034fd-19ea-4145-a12a-93f2af2421b5/SQL.pdf.

Bill Todd, "Borland Inter Base and SQL Server2000: A Technical comparison", http://www.borland.com/resources/en/pdf/white_papers/ib_vs_SQLServer.pdf.

Christian Plattner, Gustavo Alonso, "PDDBS Exercises SQL Server 2000: Snapshot Isolation", http://www.iks.inf.ethz.ch/education/ss05/PDDBS/u6-snapshot.pdf

Connolly & Begg, "Transaction Management Concurrency control", http://www.csc.liv.ac.uk/~valli/Comp302/COMP302-concurrencycontrol-notes.pdf

David Campbell, "The new Locking, Logging and Recovery Architecture of Microsoft SQL Server 7.0", http://www.vldb.org/conf/1999/P25.pdf.

David Gornshtein, Boris Tamarkin, "Features, Strengths and Weaknesses comparisons between MS SQL Server 2000 and Oracle 10g Databases", http://www.wisdomforce.com/dweb/resources/docs/MSSQL2005_ORACLE10g_compare.pdf.

Gerome Miklau, "Concurrency Control",

http://edlab-www.cs.umass.edu/cs645/lectures/645-Lec22-Concurrency2.pdf.

Jeremy T.Torres, "Concurrency and transaction Management in an Object Database", http://students.depaul.edu/~jtorres4/se690/formal_project_proposal.pdf

Pam Quick, "Transactions Concurrency control", http://www.doc.mmu.ac.uk/STAFF/P.Quick/tran-concurr.ppt.

P.A. Bernstein, V.Hadzilacos, and N.Goodman, "Concurrency control and Recovery in database systems", Addison-Wesley, Reading, Massachusetts, 1987

R.Alonso, H.Garcia-Molina, and K.Salem, "Concurrency control and Recovery for Global Procedures in Federated Database Systems", In IEEE Data Engineering. 1986

boscoallapatt@gmail.com, 09526253375