Scheduler Anomalies in Concurrency Control

Muzammal Ahmad*, Waqar Raza*, Imran Ahmad*, Sridevi Tharanidharan+, Shamim Kamal Abdul Nazar+
Department of Computer Sciences, COMSATS University Sahiwal Campus, Pakistan*
Department of Computer Science, King Khalid University, K.S.A+

muzammal0@gmail.com, waqarraza54@gmail.com, imran2275@gmail.com, sreeshansujith@gmail.com, shamimasif2011@gmail.com

Abstract—As the distribution of data across network becoming more and more popular, the need for advancement in distributed database management system (DDBMS) is increasing daily. Concurrency Control is among one of the important factors. In terms of DDBMS, concurrency meant to control and managerial time transactions and queries while maintaining data integrity. In this paper, scheduler anomalies are presented.

Keywords—Scheduler, Concurrency Control, Distributed Database, Integrity, Transactions, DBMS, Distributed Database (D-DB), Consistency, ACID

I. INTRODUCTION

As the reliability and scale of digital information is increasing the needs for maintaining and improving distributed data is also increasing[11].

D-DB systems provide an efficient way for data to be processed and communicated over the network. It not only decreases the access time for data transferring but also provides a good probability of system success. D-DB also provides its user the local data controlling. However, there exist a lot of complexities while implementing and handling D-DB systems.

Concurrency control is a technique which is used for the detection and avoidance for real-time query issues in the multi-user system. Concurrency control, when applied on a DB system, is meant to handle multiple real-time transactions while not affecting user data [1].

Yet there exist a bunch of issues which occurs while managing these concurrent transactions. Some of that issues lie under Scheduler in Concurrency control. A simple scheduler handles the incoming concurrent queries by pushing these queries into a queue structure while maintaining the order of these queries it performs transactions on the data. Efficient DBMSs have built-in schedulers that permit the query to perform its desired action in such a way that retains the data integrity. In simple words, we can describe scheduler as a communication channel between queries and the data manager. [20]

Transaction Manager ↔ Scheduler ↔ Data Manager [20]

II. PREVIOUS WORK

As Sang H. Son and Seog Park [10] presented a paper on Scheduler integration. These authors divided the query execution process into two major factors which are read wait and write phases.

Authors introduced the priority-dependent locking protocol, which changes the serialization order of the current action queries.

Mandeep Kaur and Harpreet Kaur [2] presented a paper on concurrency control in D-DBMS. Authors have discussed the reliability of D-DB and compare the reliability with centralized DBMS.

Philip A. Bernstein and Nathan Goodman [17] also presented their survey on Concurrency Control in D-DS. Authors have discussed deeply the problems that occurs in concurrency control and tries to decompose the concurrency control problem into two major sections read-write and write-write synchronization.

In another paper author Lars Frank [18] has discussed the anomalies that will occurs if the isolation property from transaction ACID properties is not followed. Those anomalies includes lost updates, dirty read and non-repeatable read.

III. DISTRIBUTED DATABASE SYSTEMS

A D-DB is a single logically spread DB over multiple computers in different geographical locations connected by means of different communication data links. [12]
D-DB is a type of database which virtually exists in many physically stored components spread over different geographical locations. It gives a user the availability of data to be existed anywhere. The user will think the data they required is present on their local machines but in actual terms, it’s opposite to it.

A DBMS is the software that manages the D-DB and provides such a mechanism that allows this transparency of data from the users.[12, 13] The main purpose of a D-DBMS is to manage D-DB in such a way that it appears centralized and transparent to its users.[12]. In D-DB multiple systems are called site and these sites are connected with each other over a computer network.[15]

There are two types of D-DB.

- Homogenous D-DB.
- Heterogeneous D-DB.

In homogenous D-DB, all sites that are connected to the network uses the same software and each site is aware for all the other sites. A homogeneous DBMS acts as a single user system.

While In a heterogeneous D-DB different sites use different software’s to manage their databases. That difference in sites schema causes major problems for query and transaction processing.

In other words, heterogeneous systems consist of a set of nodes which may have different hardware and software structures. These nodes might not be compatible to each other which arises many complexities in managing heterogeneous D-DB.

IV. CONCURRENCY CONTROL

In D-DB system, a database is mostly used by multiple users. Concurrency Control allows multiple queries to run parallel i.e. at the same time. Concurrency control is a task of handling parallel executions of run-time queries in such a way that it provides transparency to its user or idea that no one other is operating or connected to user’s database[12]. Concurrency control provides accesses to its user to ensure the illusion of being working on a separate system.

Thus to apply this transparency a lot of difficulties arrive as what if the data that is being used by user-1 is not being updated by the user-2 at the same time. To make a DB efficient enough, there is a need to prevent the interference of users at multiple levels, thus concurrency control is introduced into the D-DB to control many problems that makes the system unreliable for data integrity.

V. SCHEDULING

Execution of multiple transitions in an orderly sequence is called a schedule[16]. A schedule may carry multiple queries depending on a system that is going to perform some query action.

The main goal of scheduling in D-DB is based on two major parts:

- To meet timing constraints.
- To enforce data consistency.

Real-time task scheduling protocols can be modified or used for real-time query scheduling. Although concurrency controlling methods are still required for maintaining data consistency.

When parallel queries are executed by the system there exists a possibility that these queries might be overlapped with each other either in terms of resources or data.

To run the query in a cycle of completeness, another type of schedule is available known as Serial Scheduler[16]. It is a scheduler where queries are arranged in such a way that when one queries is being executed no other query will run until that first query has completed its task. Hence, this cycle continues till all the queries are executed. The main problem of the serial scheduler is that it takes a lot of time to perform the actions of queries in sequence.

Thus to make D-DB more efficient parallel execution of queries are used which speeds up query processing.
These parallel executions cause multiple anomalies in the scheduling process.

VI. SCHEDULING ANOMALIES

A. Lost Updates

Lost updates happen when more than one query selects the exact same row and then tries to update the row or performs an operation based on the value that was previously stored in that row. [19]

Each query that is performing its desired task is unaware of other queries that were performing their desired tasks. Hence, if the first query updates the original value then its result will be updated by other queries.

For example, let user A select a row in DB. After this, let suppose that another user B selects the same row that user A has selected. However user B writes updates as soon as it reads, this updating happens before user A writes his update. Then, the update made by user B will be overwritten by the update made by user A. This is known as the lost update problem. As shown in below figure:

![Fig.2 Example for Lost Update](5)

B. Dirty Reads

Dirty reads anomaly happens when a query selects and reads a row that is being updated by another query. [19]

For example: suppose there are two queries Q1, Q2. Q1 is updating and writing any variable in DB and still not commit it. Same time if Q2 reads the exact value that from Q1 before committing, then it is called a Dirty Read. Because there is a chance that Q1 may never commit and might roll back, but Q2 has already used that value.

![Fig.3 Example for Dirty reads](4)

C. Phantom Reads

Phantom reads anomaly happens when a user makes changes in a DB based upon the assumption that only rows that satisfied the condition(s) were returned while neglecting the possibilities that the selected data may be altered by another query hence, the data that the user holding before may not show completeness. [19]

For example: let assume a query Q1 reads and return a set of rows N that satisfy some condition. Query Q2 then executes such action that generates one or more new rows that also satisfy the condition. Hence, Q1 does not have all the rows that satisfied that condition and if Q1 repeats its process it obtains a different set of rows this time.

![Fig.3 Example for Phantom Reads](6)

D. Non-Repeatable Reads

A non-repeatable anomaly occurs when a row is selected more than once during the execution of a single query. This causes the ambiguity between the values of the selected row. [19]

This anomaly may occur in a lock based concurrency control when read locks are not provided while performing a select operation or it occurs when the locks that are applied to
selected rows are released immediately when the select operation is performed.

Sometimes Non-Repeatable read leads to phantom read anomaly. This happens when shared locks are used although that shared lock stops the data being changed while other queries are performing read task on rows, it also protects when other queries are updating the rows, but this does not stop the insert operation which can add new data to the rows, thus leads us straight to phantom read anomaly.

Fig.4 Example for Non-Repeatable Read[7]

VII. CONCLUSION AND FUTURE WORK

In this article, some common anomalies that occur in concurrency control scheduling process have been discussed. These anomalies are common problems that arrive during concurrency control management. These seemingly common anomalies play a major role in data integrity and consistency. Other anomalies that need further research include data access anomalies.

REFERENCES


[7]: [8]: http://www.tutorialspoint.com/dbms/dbms_transaction.html

[9]: https://docs.oracle.com/html/A96524_01/c21cnsis.htm

[10]: Sang H. Son and Seog Park “Scheduling and Concurrency Control for Real-Time Database Systems”


[14]: http://wps.pearsoned.co.uk/wps/media/objects/10977/11240737/Web%20chapters/Chapter%2012_WEB.pdf


[16]: http://www.tutorialspoint.com/dbms/dbms_transaction.htm


[18]: Lars Frank “Transaction Concurrency Methods” from Department of Informatics, Copenhagen Business School


[20]: Distributed Systems By Mrs. Shehal Kamalapur Mrs. Neeta Deshpande.