IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-5, Issue-12 E-ISSN: 2347-2693

An Approach Towards Designing Relational Database With Transaction Operations

Kunal Kumar[1], Sachindra Kumar Azad[2*]

[1]Department of Statistics and Computer Applications, T. M. Bhagalpur University, Bhagalpur, India
[2*]Department of Statistics and Computer Applications, T. M. Bhagalpur University, Bhagalpur, India

[*]*Corresponding Author: skazad@rediffmail.com*

Available online at: www.ijcseonline.org

Received: 05/Nov/2017, Revised: 14/Nov/2017, Accepted: 13/Dec/2017, Published: 31/Dec/2017

***Abstract*—** A transaction processing on relational database perform as single logical unit of work, which either executed completely or not at all on database system. A transaction processing on database perform mainly two basic operations read and write operation on the database system. Read operations fetch the content of the database using some standard SQL query and retrieve the data from database to the local logical buffer in main memory. In other hand, write operations, update of the local logical buffer and send back the data from the local main memory to database system for permanent storage. In this paper, we are going to know various applications of transactions schedules and how to resolve the various issues that implicitly generated during the transaction schedule generation. Every transaction has many operations on it; all these operations may or may not be of same duration.

***Keywords*—** Serial Schedule, Relational Database, Normalization, Shared Lock, Exclusive Lock.

I. INTRODUCTION

Database transaction execute on the principle of "All or nothing". Transaction is always considered as single logical unit of work and this single logical unit of works is which divided into collection of operations. The operations are either processor based operations or Input output operations. Transaction always works with large number of databases, where more than one user can access the same database at the same time [1]. Example of such types of system may includes, railway reservation, online banking transactions, stock market and etc. For, this type transaction where collection of operations executed one after another, the transaction system always needed high speed processing system and very fast response from client as well as server side. Sometimes, it is happen that a single user can execute more than one transaction at the same time this causes incorrect result and failure of transactions [2]. In this paper, we are going to discussed various schedules of transaction operations and how to resolve the problems arises during the transaction processing. Let us suppose, Person X has a bank account number A and X wants to transfer $\$100$ to person Y account, whose account number is B [3]. The transaction begins with the following operations.

$$begin;$$
$$R(A)$$
$$A := A - \$100$$
$$W(A)$$
$$R(B)$$
$$B := B + \$100$$
$$W(B)$$
$$commit;$$

Figure 1: Transaction operation

In Fig 1, the transaction begin and the read operation is executed first and this operation reads the account balance of person X, then the second operation of transaction deducted \$100 from the account number A. The third operation, write the balance of A back to the database. After that, fourth operation is going to read the balance of B, where the account of \$100 to be transferred. Now, fifth operation executed and amount \$100 is to be added to account number B. The last write back the raised amount to the database [4].

These are all a sequence of operations which are actually executed on a *single logical unit of work*. Transfer of money from one person account to another person account. For theses we need to execute above Fig. 1 transaction operations. Suppose KAZAD, is the bank database system and many end users will sending many transaction. For example, some people want to transfer the money, some want to online shopping and many more transactions are submitted, at the same time on KAZAD database system. All these different transaction have lots of operations on it, all these operations may be or may be not be of same execution time [5]. The operations are mainly processor oriented and Input output operations. Some section of the transaction operations are processor oriented and same section of transaction operations are input output. So, it can say that transaction operations are mix of both. Executing lots of transaction may slow down the performance of the system [6].

The well-known transaction property of database system is A.C.I.D. property. This property of the transaction is also enforced by the concurrency control mechanism and recovery methods. In A.C.I.D. property A stands for *Atomicity*, which assured that the complete operation should execute or nothing should execute i.e. it should not stop in the middle. In Fig 1, if we are trying to transfer money from account A to B, either transaction execute completely or it should not stop in the middle. So, that either everything has to be execute or nothing has to be execute. *C* stands for *Consistency* when more than one transaction executes the same data at the same time; the access is controlled by concurrency control mechanisms. The next is *I* stand for *isolation*, even though a transaction is executed along with the some other transaction, it should get an affect or give an affect as if it is running isolated from other. The last one *D* stands for *Durability*, it indicates after executing the *commit* command, made some changes, all this changes or update are should be made permanently. There are various modules in the Database management System which allow taking care of each one the software modules in DBMS. There is a module called Transaction Manager which will monitoring on Atomicity and the consistency is monitored by the User or application program, which means it is the responsibility of application program to see the consistency. Isolation is taken care by the concurrency control manager and finally durability it is taken care by the recovery manager [7].

While transaction is executing in the database it must be in one of the following states. Active state, when the transaction begin its execution or new born state, for the sometimes transaction will be in the active state and after that transaction will become partially committed, which means while it is executing when it is not yet over.

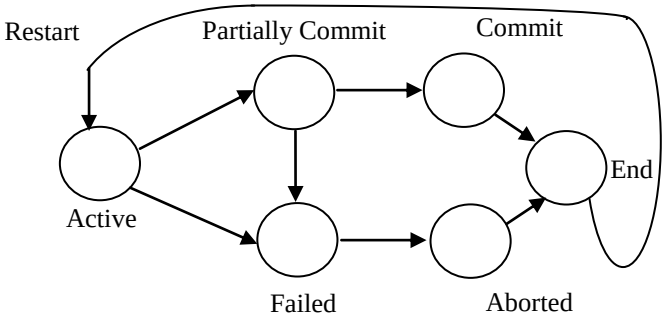

Figure 2: Transaction States

It means it is still executing and after executed if it turns commit, Commit is command, which says, transaction is executed successfully. Sometimes it happens because of some internal issues, some failures i.e. logical error or concurrency control enforcement is down because of some failures transaction might not continue further. It's done half the way and its stop and after stop it comes to failure state.Aborted means forcefully terminated, after abort the end state is executed, then transaction will be rollback to their initial state and the currently executed transaction is killed or restarts.

II. RELATED WORK

E.F Codd, 1970 was the first, who introduced the concept of relational database normalization. Deigning a table for relational database and to the reach at the highest level of normalization form is a step by step procedure. After reaching the table at highest level of the normalization the table should be able to handle or manage large number of transaction submitted by the user form remote site. Transactions on database usages more than one table to retrieve and store information at the same time. So, the designer of the database must apply the best concurrency control mechanisms to handle this type of situation [8]. There are many kind of transaction schedule possible, out of this entire schedule only

the schedule which is serializable i.e. which are equal to the serial schedule are generally logical. In other hand, other schedules which are not serializable are not going to present correct result. How can we make sure that when we are getting so many transactions as a request from the user and we are executing them concurrent manner, how can we make sure that the way they executing the transaction is always going to generate a serializable schedule?. It means what results we are getting after executing is correct or not conflict. If they have to be correct then they have to be serializable schedule. How can we guarantee it? So, we execute the schedule randomly and then it is supposed to be a serializable schedule. So, in order to do this there are some predefined rules or protocols [9].

Concurrency control is a mechanisms in a database management system, what it will try to do is, it will just follow some protocols or it will make sure that transaction will follow some protocols and if it follows these protocols then the transaction is automatically a serializable schedule and such rules are named as follows Lock based protocols, Graph Based protocols and Time stamp based protocol. When we follow three protocols and apply some rule of these protocols we can guarantee that all the schedule going to be serializable.

A. Lock Based Protocol

The only operation that user wants to execute on some data item is either read or write operations. Users are compulsory to get the lock before they start accessing it. In other words, the entire data item should be access in mutually exclusive manner. So, that when one transaction can access these data item no other transaction operations are executing on it in middle. Generally two types of lock are implemented I. Shared Lock (Lock –S) II**.** Exclusive Lock (Lock- X).Shared Lock (Lock –S), This lock only used for read something from the data item, then user can go for shared lock, the lock is shareable means more than one transaction can get read only lock on it [10]. Exclusive Lock (Lock- X), when it takes exclusive lock, the purpose of this lock is used to write on data item. This lock can be taken for both reading and writing. The major difference between these two locks is if any one getting shared lock on a data item and this shared lock can be can be taken by someone else . If Exclusive Lock (Lock- X) is taken on any data item by any of the transaction, no other transaction can get Exclusive Lock (Lock- X) again [10].

B. Graph Based Protocol

First all other additional information should be collected before implementing graph based protocol on any database transaction. The simplest way is that, we must have prior knowledge that how database item will be accessed. To collect such prior information, then apply partial ordering on the all the data items. Partial ordering has directed acyclic graph based representation sometimes it is also known as database graph. The simplest protocol that graph based protocol usages is tree protocol [7]. The only lock allow in tree protocol is Exclusive Lock (Lock- X),each transaction T_i can lock a data item at most once at must observe that the fist lock T_i may be on any data item.

C. Timestamp Based Protocol

In a schedule it is not possible that it have only one transaction it may have more than one transaction, Timestamp based protocol assign all the transaction with some number according to their arrival time. The Timestamp protocol will assign the number on each transaction that the transaction comes earlier will get lesser number and the transaction which arrived later will get higher number respectively. After that if we found any conflict on transactions action the, timestamp number will solve this problem. If I have two conflicting action then the action of the earlier or smaller number timestamp processed. It means the transactions which had smaller timestamp number will they get the CPU first and the transaction will execute first, during the execution of transaction operations if any abort or failure occur then the transaction rollback to its initial state. Timestamp ordering always assign a unique number to each transaction, created by DBMS. Read timestamp is highest timestamp number that perform read operation successfully [11]. Write timestamp is highest timestamp number that performs write operation. Generally, system clock is represented the timestamp number or logical counter is used to represent the timestamp ordering this may incremented after new timestamp has been assigned.

III. TRANSACTION APPROACH FOR DESIGNING RELATIOANL DATABASE

Let's suppose that, we have m number of transaction and each transaction have n numbers of operations on it is represented as

T_1	T_2	T_3	T_4	.....	T_m
a_1	b_1	c_1	d_1	.....	z_1
a_2	b_2	c_2	d_2	.....	z_2
a_3	b_3	c_3	d_3	.....	z_3
.	.	.	.	.....	.
.	.	.	.	.....	.
.	.	.	.	.....	.
a_{n_1}	b_{n_2}	c_{n_3}	d_{n_4}	.....	z_{nm}

The total number of permutations of the all transaction operations are arranged in vertical column is

$$= (n_1 + n_2 + n_3 + n_4 + ... + n_{nm})!$$

A. Restriction on Transaction Operations

The transaction scheduling only follow one rule that is the operations of transaction will not altered their order of execution at any cost otherwise it is not possible to maintain consistency in the database, but the interleaving between the operations among various transactions is allowed. In serial schedule, there is no interleaving means it doesn't allow any other transaction come in the middle, while one transaction is executing. In any column of T_1 operation a_1 must come before a_2, and a_2 should come before a_3 and so on for any other schedule in S, the same order rule is applied for b, c and d operations. Consider only one vertical column of $(n_1 + n_2 + n_3 + n_4 + ... + n_{nm})!$ operations $a_1, a_2, a_3, a_4, ..., a_{nm}$ can be arranged as $n_1!$ ways but we must follow the restriction rule and then after one set of transaction $a_1, a_2, a_3, a_4, ..., a_{nm}$ is acceptable. Similarly set if transaction operations $b_1, b_2, b_3, b_4, ..., b_{nm}$ can be arranged in $n_2!$ ways and S represent the all possible schedules for the transaction operations can be arranged in

$$S \times (n_1!n_2!n_3!.....n_m!) = (n_1 + n_2 + n_3 + n_4 + ... + n_m)!$$

$$S = \frac{(n_1 + n_2 + n_3 + n_4 + ... + n_m)!}{n_1!n_2!n_3!.....n_m!}$$

Let Suppose, there are m transactions which are represented as $T_1, T_2, T_3, T_4, ..., T_m$ and each transaction has n numbers of operations on it $n_1, n_2, n_3, n_4, ..., n_{nm}$. So, the required number of permutation for schedules S is

$$S = \frac{(n_1 + n_2 + n_3 + n_4 + ... + n_m)!}{n_1!n_2!n_3!.....n_m!}$$

A serial schedule is a schedule which performs their executing one after the other transactions. Suppose, we have two transactions T_1 and T_2 it may perform like $T_1 \to T_2$ i.e. T_1 is over then T_2 start or it may perform like $T_2 \to T_1$ i.e. T_2 is over then T_1 start. Whenever the schedules are serial the output of the transaction is always consistence. Serial schedule does not perform interleaving between the other transactions.

Then the number of serial schedule possible for the transactions $T_1, T_2, T_3, T_4, ..., T_m$ is $m!$. Wolfram mathmatica is used for the simulation purpose, the built symbol of *permutations [list, {n}]* are available in mathmatica.

IV. AN ILLUSTRATIVE ANALYSIS

Let say, the two transactions T_1 and T_2, each transaction have two operations. It is not possible that a transaction have only two operations it may have more than two operations.

T_1	T_1
X_1	Y_1
X_2	Y_2

Transaction T_1 has two operations X_1 and X_2, Transaction T_2 has two operations Y_1 and Y_2. There is various ways to combine this Transaction and form the schedule. One schedule S_1 may be like perform T_1 operations first and then

perform T_2 operations. Another schedule S_2 may perform T_2 operations first and then perform T_1 operations.

S_1	S_2
X_1	Y_1
X_2	Y_2
Y_1	X_1
Y_2	X_2

And other schedule can be mix up of both the transaction operation as follows.

S_3	S_4	S_5	S_6
X_1	Y_1	X_1	Y_1
Y_1	X_1	Y_1	X_1
Y_2	X_2	X_2	Y_2
X_2	Y_2	Y_2	X_2

From the following permutations of schedule is clear that the entire above schedule holds the consistency state i.e. in all the above S_1 to S_6 schedule only after X_1, X_2 should occur and after Y_1 , Y_2 should occur and interleaving can be done in any fashion. Form the arrangements of schedules, It is clear that between X_1, X_2 or between Y_1 , Y_2 , any number of operations can be execute that is perfectly a valid schedule. Now, form the permutations of schedule S_1 to S_6 all may not give correct output. Some of the schedules may hold the consistency problems. So, it is compoulsory to allow only the schedule which is guaranteed the database result going to be consistence and there will be no errors. Now the main problems are how we have to find the serial schedule form above permutations of schedule which gives correct output. Transaction S has two operations X_1 and X_2 , Transaction T_2 has two operations Y_1 and Y_2. Total number of schedules possible from both the transaction

$$S = \frac{(2+2)!}{2!.2!} = \frac{4!}{2\times 2} = 6 \text{ Schedule Possible}$$

In the above six schedule all are not going to give the correct results. To get the correct result we should find out serial schedule. So, the total number of serial schedule possible is $m!$ always.

Total number of serial schedule $= 2! = 2$ Schedule

Total number of non- serial schedule $= 6 - 2 = 4$ Schedules

SQL Server Management Studio is using for designing the relational database and implement the transaction operations.

V. CONCLUSION

Relational Database Design and normalization approach is the way to standardize the table, after reaching the database table at specific level of the normal form. The number of transaction is submitted by the client side to server database continuously at the real time system fashion. All these transaction are independent to each other are all have different types of operations on it. The transaction operations cannot overlapped with other transaction operation for this we should maintain the consistency on transaction system. Serial schedule is the best ways that don't allows interleaving of transaction operation with other transaction operations and also maintain the consistency in the transaction system. The permutation based mathematical calculation derives the all possible schedules for transactions operations and also find the serial schedule.

VI. REFERENCES

[1]. A.Waqas, A.W. Mahessar, N. Mahmood, Z. Bhatti, M..Karbasi, A. Shah, February 2015 , "Transaction management techniques and practices in current cloud computing environments: a survey", IJDMS) Vol.7, No.1,

[2]. Deng, Yuetang Frankl, Phyllis Chays, David, "Testing database transaction consistency", Technical Report TRC-IS-2003, Page No .184-195, 2003

[3] Elmasri, R. and Navathe, S.B.2007: Fundamentals of Database Systems, 5th Ed., Addison Wesley.

[4] R. G. Tiwari, S. B. Navathe, and G. J. Kulkarni, September 2010. "Towards Transactional Data Management over the Cloud," 2010 Second Int. Symp. Data, Privacy, E-Commerce, pp. 100–106

[5] S. Das and A. El Abbadi, 2010 "G-Store: A Scalable Data Store for Transactional Multi key Access in the Cloud," in SoCC', Indianapolis, Indiana, USA, , pp. 163–174.

[6] Codd E. F., 1970, "A relational model of data for large shared data banks", Communications of the ACM. vol. 13, No.6, pp. 377–387.

[7] Ravula, R ,2014-2015 "Database : Transactions and concurrency control", online Lecture series , E- Classes by R Ravula. ravindrababuravula.com

[8] Connolly, Thomas, Carolyn Begg, 2005, : Database Systems. A Practical Approach to Design, Implementation, and Management, Pearson Education, 3rd edition.

[9] R. Kaula, and J. Chin, April 1993 "A database approach towards flexiblemanufacturing: A conceptual framework, " in Computers and Industrial Engieering, Vol 24, No 2, pp. 131-141,1993

[10] Elmagarmid, Ahmed K, "Database Transaction Modelsfor Advanced Applications", Database, 611, 1992

[11] Gray, Jim Lamport, Leslie, "Consensus on transaction commit", ACM Transactions on Database Systems, 133-160, Vol – 31, Issue -1 , 2006

Authors Profile

Mr. Kunal Kumar received the M.C.A and B.C.A degree from the Birla Institute of Technology, Mesra, Ranchi, India in 2012 and 2008. He is currently working as a Research Scholar toward the PhD degree at the University Department of Statistics and Computer Applications. Tilka Manjhi Bhagalpur University, Bhagalpur, India. His current research interests areas include database design ,Big Data and database theory.

Dr. Sachindra Kumar Azad is an Associate Professor and Head in the University Department of Statistics and Computer Applications at Tilka Manjhi Bhagalpur University, Bhagalpur, India. His current research interests areas include database design and database theory and Big Data..

