International Journal of Computer Architecture and Mobility
(ISSN 2319-9229) Volume 1-Issue 8, June 2013

Concept and techniques of transaction processing of Distributed Database management system

Ganesh kohad, Shikha gupta, Trupti gangakhedkar, Umesh ahirwar, Ajit kumar

gk.dc2012@gmail.com,shikha_gupta25@rediffmail.com,truptiqkhedkar@gmail.com,
uahirwar728@gmail.com,ajitece20@gmail.com

Abstract- **A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network. A distributed database management system (distributed DBMS) is the software system that permits the management of the distributed database and makes the distribution transparent to the users. There are 2 processes in distributed database, replication and duplication. Replication using specialized software to look for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. Duplication will identify one database as master and then duplicate it. Upon those processes, it seems data in database will be very consistent and will have no problem when some transaction is executed. But in fact, if there is no definition on transaction, the database will be in a mess condition. So how does the database stay consistent and stable? The answer for this is ACID properties, which are a set of properties that guarantee the reliability of database transactions. In this paper, we make the case that transactions will cause a fault in a distributed database without ACID properties.**

The transaction-processing model of distributed database includes data, Transaction, Data Manager, and Transaction Manager and also discussed the various operations performed by these components. In this paper we discussed about basic of distributed database, Transaction in Distributed database and ACID Properties. We describe a series of operation of transaction that used in distributed database. The purpose of the paper is to examine the underlying transaction property in distributed database.

Keywords-component: Distributed, Database, Distributed processing, parallel Processing, Fragmentation, Allocation, Replication, Transaction, Transaction Manager, Concurrency Control, Decomposition, Synchronization.

I. INTRODUCTION

Today's business environment has an increasing need for distributed database and client/server applications as the desire for reliable, scalable and accessible information is steadily rising. Distributed database systems provide an improvement on communication and data processing due to its data distribution throughout different network sites. Not only is data access faster, but a single-point of failure is less likely to occur, and it provides local control of data for users. However, there is some complexity when attempting to manage and control distributed database systems.

As distributed networks become more popular, the need for improvement in distributed database management systems becomes even more important. A distributed system varies from a centralized system in one key respect: The data and often the control of the data are spread out over two or more geographically separate sites. A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network. It is necessary to ensure the data in a distributed database should be integrity so that all users who access the database can get the correct data and run their program without any error. To address this issue, we can use ACID properties for distributed database transaction. The definition of ACID properties for distributed database is that it is a set of properties that guarantee the reliability of database transactions. ACID defines properties that traditional transaction must display; they are Atomicity, Consistency, Isolation, and Durability.

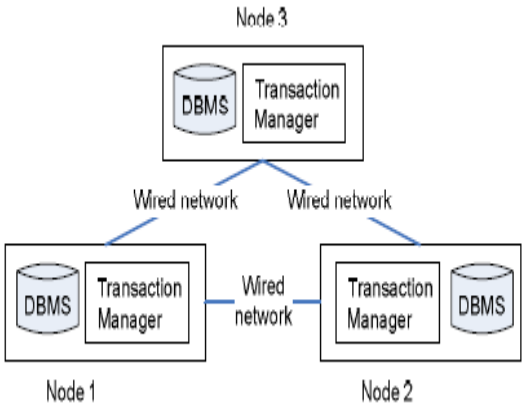
In this paper we describe distributed database system and their transaction process. A major issue of distributed transaction is ACID Property. So we are also describing ACID property of distributed transaction and different operations of

transaction that use in distributed database. Finally, we will compare the relative merits of database and distributed database with respect to transaction.

II. Transaction Processing in a Distributed System

A transaction is a logical unit of work constituted by one or more SQL statements executed by a single user. A transaction begins with the user's first executable SQL statement and ends when it is committed or rolled back by that user. A remote transaction contains only statements that access a single remote node. A distributed transaction contains statements that access more than one node.

A distributed transaction processing system is a collection of sites or nodes that are connected by communication networks.



The communication networks are usually reliable and high speed wired networks, like LANs or WANs. At each node in a distributed system, there is a local database management system and a local transaction processing system (TPS) that operates semi-independently and semi-autonomously. An execution of a transaction in a distributed database system may have to spread to be processed at many sites. The transaction managers at different sites in a distributed transaction system cooperate for managing the transaction execution processes.

Transactions in a distributed system can be categorized into two classes:

1. Local transaction
2. Global transaction.

Local transactions are submitted directly to local transaction managers. Local transactions only access data at one database system at one site, and are managed by the local transaction manager. On the other hand, global transactions are submitted via the global transaction manager. A global transaction can be decomposed into a set of sub-transactions;

We consider a distributed database management system with a data collection of sites interconnected by a network. Each site runs one or more of the following software modules:

1. A transaction manager (TM),
2. A data manager (DM)
3. A network concurrency control scheduler (CCS).

A Database Manager is software responsible for processing a segment of the distributed database. Another main component is the User Request Interface, which is usually a client program that acts as an interface to the Distributed Transaction Manager. "A Distributed Transaction Manager is a program that translates requests from the user and converts them into actionable requests for the database managers, which are typically distributed.

III. Operations on Distributed Transaction

Transactions communicate with TMs, TMs communicate with DMs, and DMs manage the data. TMs supervise transactions. Each transaction executed in the DDBMS is supervised by a single TM, meaning that the transaction issues all of its database operations to that TM. Any distributed computation that is needed to execute the transaction is managed by the TM. Four operations are defined at the transaction-TM interface.

- READ(X): returns the value of X (a logical data item) in the current logical database state.
- WRITE(X, new-value): creates a new logical database state in which X has then specified new value.
- BEGIN and END operations to bracket transaction executions.

DMs manage the stored database, functioning as backend database processors. In response to commands from

transactions, TMs issue commands to DMs specifying stored data items to be read or written.

IV. ACID properties of Distributed Transactions:

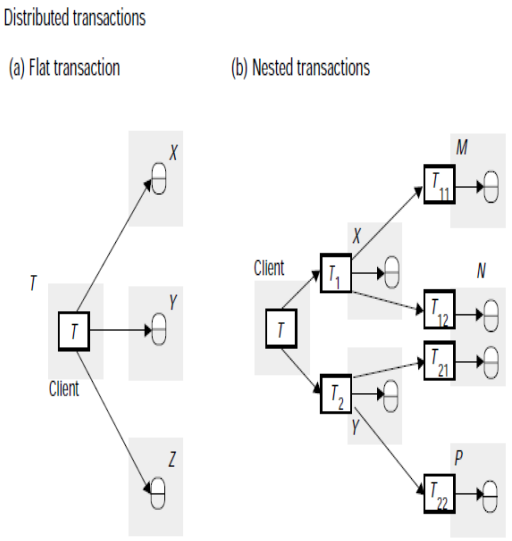
A transaction in a database must have ACID properties to run the program correctly. In a distributed database, transactions are implemented over multiple applications and hosts. Moreover, distributed transactions also enforce the ACID properties over multiple data stores. ACID properties are Atomicity, Consistency, Isolation and Durability. Each transaction must follow these 4 properties.

- **Atomicity:** Atomicity refers to the ability of the DBMS to guarantee that either all of the tasks of a transaction are performed or none of them are. Atomicity states that database modifications must follow an "all or nothing" rule. If some part of a transaction fails, then the entire transaction fails, and vice versa.
- **Consistency:** The Consistency property ensures that the database remains in a consistent state, despite the transaction succeeding or failing and both before the start of the transaction and after the transaction is over.
- **Isolation:** Isolation refers to the requirement that other operations cannot access or see the data in an intermediate state during a transaction. As discussed above, the Isolation property can helps to implement concurrency of database.
- **Durability:** Durability states that once a transaction is committed, its effects are guaranteed to persist even in the event of subsequent failures. That means when users are notified of success, the transactions will be persist, not be undone and survive from system failure.

IV. Nested Transactions

By structure, distributed transaction is dividing into two types

- Traditional flat Transaction
- Nested Transaction



A flat transaction, **FT**, is an operation, performed on a database, which may consist of several simple actions. From the client's point of view the operation must be executed indivisibly. Main disadvantage with **FT**s: If one action fails the whole transaction must abort.

A transaction may contain any number of **subtransactions**, which again may be composed of any number of subtransactions - conceivably resulting in an arbitrarily deep hierarchy of nested transactions. The root transaction, which is not enclosed in any transaction, is called the **top-level transaction (TL-transaction)**. Transactions having subtransactions are called **parents**, and their subtransactions are their **children**. We will also speak of **ancestors** and **descendants**. The ancestor (descendant) relation is the reflexive transitive closure of the parent (child) relation. We will use the term **superior (inferior)** for the non-reflexive version of the ancestor (descendant). The set of descendants of a transaction together with their parent/child relationships is called the transaction's **hierarchy**. In the following, we will use the term 'transaction' to denote both TL-transactions and subtransactions.

A so-called transaction tree can represent the hierarchy of a TL-transaction. The nodes of the tree represent transactions, and the edges illustrate the parent/child relationships between the related transactions.

The properties defined for flat transactions are **atomicity**, **consistency**, **isolated execution**, **and durability** (ACID-properties). In the nested transaction model, the ACID-

properties are fulfilled for TL-transactions, while only a subset of them are defined for subtransactions.

V. COMMITMENT OF DISTRIBUTED TRANSACTIONS

A distributed transaction refers to a flat or nested transaction that accesses objects managed by multiple servers. When a distributed transaction comes to an end either the entire servers commit the transaction or all of them abort the transaction. A transaction may be terminated by command of

- Commit Command, i.e. successfully completed, or
- Rollback Command, i.e. aborted.

Commit makes DB operations effect permanent and the result is visible to other transactions. Rollback undoes all DB operations and restores the DB to the state before the execution of the transaction.

In case nested transaction subtransaction must begin after its parent and finish before its parent. The commit/abort of a subtransaction depends on its parent.

Conclusion

In this paper, we have reviewed the basic concepts of database systems and database transactions, and discussed the architecture of transaction processing systems in distributed environments.

It is really important for database to have the ACID properties to perform

Atomicity, Consistency, Isolation and Durability in transactions. It ensures all data in scientific research and business field to be correct and valid statues, without them, database will be in a mess.

We are in the process of investigating schemes by which the performance of high security level transactions can be improved without compromising with the security. Further we are looking to secure real time distributed systems by which the performance of high security level transactions can be improved without compromising the security.

References

1. Concurrency Control in Distributed Database Systems By PHILIP A. BERNSTEIN AND NATHAN GOODMAN
2. Distributed Transaction Processing on an Ordering Network By Rashmi Srinivasa, Craig Williams, Paul F. Reynolds
3. Transaction Processing in Distributed Databases. By Manpreet Kaur
4. Database Concepts by Elmasri Navathe By Pearson Education
5. Data base Concepts by Colloly. By Pearson Education
6. Introduction to Database Concepts by Rob Coronel.
7. World of DBMS By Jitendra sheetlani, Dhiraj gupta
8. WWW.Citeseer.com