IJCSC International Journal of Computer Science and Communication Vol. 3, No. 1, January-June 2012, pp. 101-104

NEW APPROACHES OF TRANSACTION PROCESSING IN DISTRIBUTED DATABASE SYSTEM

Amita Kumari[1] and Chanderkant[2]

[1]Deptt. of CSE, KITM, Kurukshetra

[2]Deptt. of Computer Science and Applications, K.U., Kurukshetra,
E-mail: vermar.rajesh1974@gmail.com, ckverma@rediffmail.com

ABSTRACT

A transaction processing system is a type of information system. Transaction processing system collect, store, modify, and retrieve the transactions of an organization. A transaction is an event that generates or modifies data that is eventually stored in an information system. For considering a transaction processing system, the computer must pass the ACID test. The essence of a transaction program is that it manages data that must be left in a consistent state. E.g. if an electronic payment is made, the amount must be both withdrawn from one account and added to the other; it cannot complete only one of those steps. Either both must occur, or neither. In case of the failure of a transaction, the partially executed transaction must be 'rolled back' by the transaction processing system. While this type of integrity is particularly important for online processing e.g. if an airline seat reservation system is accessed by multiple operators, after an empty seat inquiry, the seat reservation data must be locked until the reservation is made, otherwise another user may get the impression a seat is still free while it is actually being booked at the time. Without proper transaction monitoring, double bookings may occur. Transaction Processing is not limited to application programs. In this article, we emphasize on important concepts in transaction processing system and explain how transactions access data in a distributed database.

1. INTRODUCTION:

Distributed database systems (DDBS) pose different problems when accessing distributed and replicated databases. Particularly, access control and transaction management in DDBS require different mechanism to monitor data retrieval and update to databases. Current trends in multi-tier client/server networks make DDBS an appropriated solution to provide access to and control over localized databases. Oracle, as a leading Database Management System (DBMS) vendor employs the two-phase commit technique to maintain consistent state for the database. 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. Following Figure 1 shows that process for transaction execution in distributed database system.

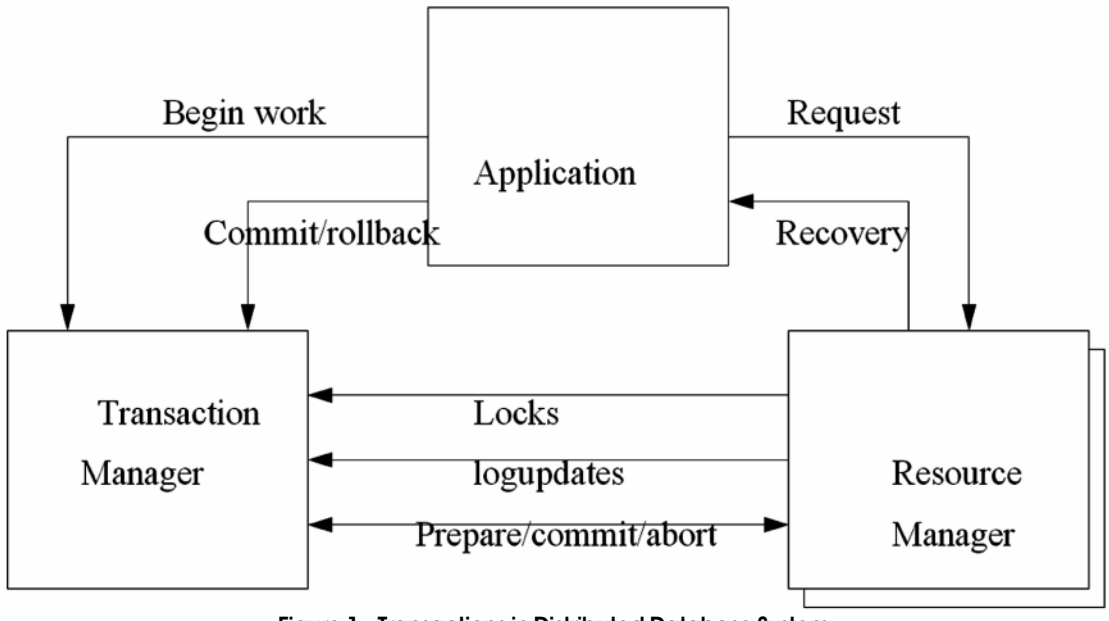

Figure 1: Transactions in Distributed Database System

The following sections define new approaches of transaction processing and explain how transactions access data in a distributed database:

- Remote SQL Statements
- Distributed SQL Statements
- Shared SQL for Remote and Distributed Statements
- Remote Transactions
- Distributed Transactions
- Database Link Name Resolution

2. APPROACHES OF TRANSACTION PROCESSING IN DISTRIBUTED DATABASE SYSTEM

2.1 Remote SQL Statements

A remote query statement is a query that selects information from one or more remote tables. The remote tables reside at the same remote node. For example, the following query accesses data from the dept table in the scott schema of the remote sales database

```
SELECT * FROM scott.dept@sales.us.americas.acme_auto.com;
```

A remote update statement is an update that modifies data in one or more tables, all of which are located at the same remote node. For example, the following query updates the dept table in the scott schema of the remote sales database:

```
UPDATE scott.dept@mktng.us.americas.acme_auto.com
SET loc = 'NEW YORK'
WHERE deptno = 10;
```

Note: A remote update can include a subquery that retrieves data from one or more remote nodes, but the update happens at only a single remote node, the statement is classified as a remote update.

2.2 Distributed SQL Statements

A distributed query statement retrieves information from two or more nodes. For example, the following query accesses data from the local database as well as the remote sales database:

```
SELECT ename, dname
FROM scott.emp e, scott.dept@sales.us.americas.acme_auto.com d
WHERE e.deptno = d.deptno;
```

A distributed update statement modifies data on two or more nodes. A distributed update is possible using a PL/SQL subprogram unit such as a procedure or trigger that includes two or more remote updates that access data on different nodes. For example, the following PL/SQL program unit updates tables on the local database and the remote sales database:

```
BEGIN
  UPDATE scott.dept@sales.us.americas.acme_auto.com
    SET loc = 'NEW YORK'
    WHERE deptno = 10;
 UPDATE scott.emp
    SET deptno = 11
    WHERE deptno = 10;
END;
COMMIT;
```

The database sends statements in the program to the remote nodes, and their execution succeeds or fails as a unit.

2.3 Shared SQL for Remote and Distributed Statements

The mechanics of a remote or distributed statement using shared SQL are essentially the same as those of a local statement. The SQL text must match, and the referenced objects must match. If available, shared SQL areas can be used for the local and remote handling of any statement or decomposed query.

2.4 Remote Transactions

A remote transaction contains one or more remote statements, all of which reference a single remote node. For example, the following transaction contains two statements, each of which accesses the remote sales database:

```
UPDATE scott.dept@sales.us.americas.acme_auto.com
  SET loc = 'NEW YORK'
  WHERE deptno = 10;
UPDATE scott.emp@sales.us.americas.acme_auto.com
  SET deptno = 11
  WHERE deptno = 10;
COMMIT;
```

2.5 Distributed Transactions

A distributed transaction is a transaction that includes one or more statements that, individually or as a group, update data on two or more distinct nodes of a distributed database. For example, this transaction updates the local database and the remote sales database:

```
UPDATE scott.dept@sales.us.americas.acme_auto.com
```

```
    SET loc = 'NEW YORK'
    WHERE deptno = 10;
  UPDATE scott.emp
    SET deptno = 11
    WHERE deptno = 10;
  COMMIT;
```

Note: If all statements of a transaction reference only a single remote node, the transaction is remote, not distributed.

2.6 Database Link Name Resolution

A global object name is an object specified using a database link. The essential components of a global object name are:

- Object name
- Database name
- Domain

The following table shows the components of an explicitly specified global database object name:

Statement	Object	Database	Domain
SELECT * FROM joan.dept@sales.acme.com	dept	Sales	acme.com
SELECT * FROM emp@mktg.us.acme.com	emp	mktg	us.acme.com

Whenever a SQL statement includes a reference to a global object name, the database searches for a database link with a name that matches the database name specified in the global object name. For example, if you issue the following statement:

```
SELECT * FROM scott.emp@orders.us.acme.com;
```

The database searches for a database link called orders.us.acme.com. The database performs this operation to determine the path to the specified remote database. The database always searches for matching database links in the following order:

1. Private database links in the schema of the user who issued the SQL statement.
2. Public database links in the local database.
3. Global database links (only if a directory server is available).

2.6.1 Name Resolution when the Global Database Name is Complete

Assume that you issue the following SQL statement which specifies a complete global database name:

```
SELECT * FROM emp@prod1.us.oracle.com;
```

In this case, both the database name (prod1) and domain components (us.oracle.com) are specified, so the database searches for private, public, and global database links. The database searches only for links that match the specified global database name.

2.6.2 Name Resolution when the Global Database Name is Partial

If any part of the domain is specified, the database assumes that a complete global database name is specified. If a SQL statement specifies a partial global database name (i.e. only the database component is specified), the database appends the value in the DB_DOMAIN initialization parameter to the value in the DB_NAME initialization parameter to construct a complete name. For example, assume you issue the following statements:

```
CONNECT scott@locdb
SELECT * FROM scott.emp@orders;
```

If the network domain for locdb is us.acme.com, then the database appends this domain to orders to construct the complete global database name of orders.us.acme.com. The database searches for database links that match only the constructed global name. If a matching link is not found, the database returns an error and the SQL statement cannot execute.

2.6.3 Name Resolution when no Global Database Name is Specified

If a global object name references an object in the local database and a database link name is not specified using the @ symbol, then the database automatically detects that the object is local and does not search for or use database links to resolve the object reference. For example, assume that you issue the following statements:

```
CONNECT scott@locdb
SELECT * from scott.emp;
```

Because the second statement does not specify a global database name using a database link connect string, the database does not search for database links.

2.6.4 Terminating the Search for Name Resolution

The database does not necessarily stop searching for matching database links when it finds the first match. The database must search for matching private, public, and network database links until it determines a complete path to the remote database. The first match determines the remote schema as illustrated in the following table:

User Operation	*Database Response*	*Example*
Do not specify the CONNECT clause	Uses a connected user database link	CREATE DATABASE LINK k1 USING 'prod'
Do specify the CONNECT TO ... IDENTIFIED BY clause	Uses a fixed user database link	CREATE DATABASE LINK k2 CONNECT TO scott IDENTIFIED BY tiger USING 'prod'
Specify the CONNECT TO CURRENT_USER clause	Uses a current user database link	CREATE DATABASE LINK k3 CONNECT TO CURRENT_USER USING 'prod'
Do not specify the USING clause	Searches until it finds a link specifying a database string. If matching database links are found and a string is never identified, the database returns an error.	CREATE DATABASE LINK k4 CONNECT TO CURRENT_USER

After the database determines a complete path, it creates a remote session, assuming that an identical connection is not already open on behalf of the same local session. If a session already exists, the database reuses it.

3. CONCLUSION

Organizations want to preserve their investments in IT and databases system. They also wish to exploit new database technology if it is cost-effective. Real-world distributed systems inevitably involve interfacing with databases of various types and ages (hierarchical, network, relational, post-relational/object-oriented). However, the database community has a rather different view of how distribution should be handled. In this article, We have tried to highlight the transaction processing systems in distributed database system.

REFERENCE

[1] Wikipedia: Database Management Systems.

[2] Rick Long, Mark Harrington, Robert Hain, Geoff Nicholls, "IMS Primer", *International Technical Support Organization*, IBM Corporation, 2000.

[3] P.A. Bernstein, V. Hadzilacos, N. Goodman, "Concurrency Control and Recovery in Database Systems", *Addison-Wesley*, 1987.

[4] G. Coulouris, J. Dollimore, T. Kindberg, "Disributed Systems, Concepts and Design", *Addison-Wesley*, 1994.

[5] G. Schussel, "Replication, The Next Generation of Distributed Database Technology", via Internet: http://www.dci.com/speakers/archive/replica.html

[6] P. Bajerski, M. and KáRSHN_ ._ +DU*ODN_ +_ -RVL_VNL_ Environment for Distributed Database ManagementAlgorithms Evaluation Based on Virtual Shared Memory". *Proceedings of the 16th IASTED International Conference APPLIED INFORMATICS*, Held February 23-25, 1998, Garmisch-Partenkirchen, Germany.

[7] Ozsu, Tamer M., and Valduriez, Patrick [1991], "Principles of Distributed Database Systems", *Prentice Hall*.

[8] Rob P., Coronel C.; "Database Systems: Design, Implementation, and Management", (7th ed.). Boston, *Massachusetts: Thomson Course Technology*; 2007. **396**, 473-475, 490 pp.

[9] What is Distributed Database? [Article Online] 2002. Available from http://www.webopedia.com/TERM/D/distributed_database.html. Accessed 2009 Oct 2

[10] Database Management Systems by Alexis Leon.