IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555
Vol. 3, No.1, February 2013

Advanced Database Management Systems

Review of Transaction Management in Distributed Database

Praseeda Manoj
Dept. of Computer Science
Muscat College
Muscat, Oman.

***Abstract*— Data Base Management System consists of a collection of interrelated data and a set of programs to access those data. The collection of data is database. These are designed to manage large bodies of information. This paper describes advanced aspects of database management system including Query processing, data modeling, data warehouse, mining, fuzzy dimensions, distributed database, XML and web application database. Aim of this paper is to introduce various ways of designing and implementing database systems, features and distributed databases.**

Keywords-Database, DBMS, Distributed DB, Advanced DBMS, Data

I. Introduction

A database can be defined as a structured collection of records or data that is stored in a computer so that a program can consult it to answer queries. The records retrieved in answer to queries become information that can be used to make decisions. The computer program used to manage and query a database is known as a database management system (DBMS). The term "database" originated within the computing discipline. Although its meaning has been broadened by popular use, even to include non-electronic databases, this article is about computer databases. Database-like records have been in existence since well before the industrial revolution in the form of ledgers, sales receipts and other business related collections of data. The central concept of a database is that of a collection of records, or pieces of knowledge. The term database refers to the collection of related records, and the software should be referred to as the database management system or DBMS. When the context is unambiguous, however, many database administrators and programmers use the term database to cover both meanings. Database management systems are usually categorized according to the data model that they support: relational, object-relational, network, and so on. The data model will tend to determine the query languages that are available to access the database. A great deal of the internal engineering of a DBMS, however, is independent of the data model, and is concerned with managing factors such as performance, concurrency, integrity, and recovery from hardware failures.

The earliest known use of the term 'data base' was in June 1963, when the System Development Corporation sponsored a symposium under the title Development and Management of a Computer-centered Data Base. Database as a single word became common in Europe in the early 1970s and by the end of the decade it was being used in major American newspapers. The first database management systems were developed in the1960s. The relational model was proposed by E. F. Codd in 1970. however, For a long while, the relational model remained of academic interest only. During the 1980s, research activity focused on distributed database systems and database machines, but these developments had little effect on the market. Another important theoretical idea was the Functional Data Model, but apart from some specialized applications in genetics, molecular biology, and fraud investigation, the world took little notice. In the 1990s, attention shifted to object-oriented databases. These had some success in fields where it was necessary to handle more complex data than relational systems could easily cope with, such as spatial databases, engineering data (including software engineering repositories), and multimedia data. In the 2000s, the fashionable area for innovation is the XML database. As with object databases, this has spawned a new collection of startup companies, but at the same time the key ideas are being integrated into the established relational products. XML databases aim to remove the traditional divide between documents and data, allowing all of an organization's information resources to be held in one place, whether they are highly structured or not.

II. Query Processing

The aim of query processing is to find information in one or more databases and deliver it to the user quickly and efficiently. Traditional techniques work well for databases with standard, single-site relational structures, but databases containing more complex and diverse types of data demand new query processing and optimization techniques. Most real-world data is not well structured. Today's databases typically contain much non-structured data such as text, images, video, and audio, often distributed across computer networks. In this complex milieu (typified by the World Wide Web), efficient and accurate query processing becomes quite challenging.

The Query Optimizer is the component of a database management system that attempts to determine the most efficient way to execute a query. The optimizer considers the possible query plans (discussed below) for a given input query, and attempts to determine which of those plans will be the most efficient. Cost-based query optimizers assign an estimated "cost" to each possible query plan, and choose the plan with the least cost. Costs are used to estimate the runtime cost of evaluating the query, in terms of the number of I/O operations required, the CPU requirements, and other factors.

A Query Plan (or Query Execution Plan) is a set of steps used to access information in a SQL relational database management system. This is a specific case of the relational model concept of access plans. Since SQL is declarative, there are typically a large number of alternative ways to execute a given query, with widely varying performance. When a query is submitted to the database, the query optimizer evaluates some of the different, correct possible plans for executing the query and returns what it considers the best alternative. Because query optimizers are imperfect, database users and administrators sometimes need to manually examine and tune the plans produced by the optimizer to get better performance. The set of query plans examined is formed by examining the possible access paths (e.g. index scan, sequential scan) and join algorithms (e.g. sort-merge join, hash join, nested loops). The search space can become quite large depending on the complexity of the SQL query. The query optimizer cannot be accessed directly by users. Instead, once queries are submitted to database server, and parsed by the parser, they are then passed to the query optimizer where optimization occurs.

III. DATA MANAGEMENT SYSTEMS

Most current data management systems (DMS), have been built on the assumption that the data collection, or database, to be administered consists of a single media type – structured tables of "fact" data or unstructured strings of bits representing such media objects as text documents, images, or video. The result is that most DMS store and index a specific type of media data and provide a query (data access) language that is specialized for efficient access to and retrieval of this data type. The primary objective for a DMS is to provide efficient and effective management of the database. This includes providing functions for data storage, retrieval, secure modification, DB integrity and maintenance. There are two principle quality measures for a DMS; efficiency and effectiveness.

- DMS efficiency is typically measured in the time and machine capacity used for data retrieval and storage, respectively. Here, low time or storage requirements indicate high efficiency. Since these are somewhat conflicting objectives, trade-offs are necessary.
- DMS effectiveness is typically measured in the quality of service, for example; the correctness or relevance of retrieval or modification results, smooth or seamless presentation of multiple media data, particularly for video, or the security levels supported.

Techniques used to reach these goals include:

- Index generation is used to increase the efficiency of data retrieval, by speeding data location and thereby reducing retrieval time.
- Data compression is used to reduce storage and transmission requirements
- User interfaces can enhance system effectiveness by supporting formulation of 'complete' information needs.
- Similarity algorithms seek to locate only data/ documents that are relevant to the user query.

Figure 1 gives a generic DMS architecture, highlighting principal components.

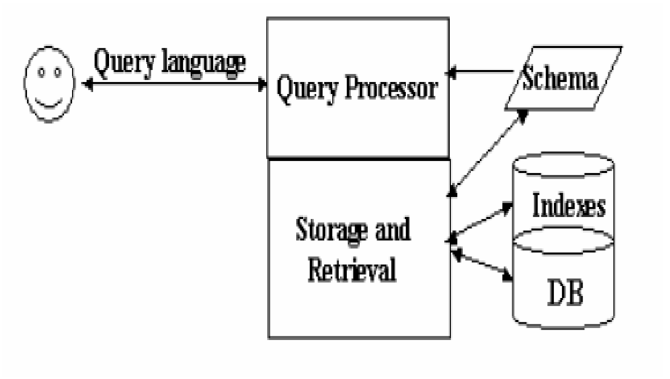

Figure 1 - DMS Components

IV. DATA MODELING

Data Modeling is the activity in information system analysis and design that identifies and develops a model of the information requirements for a proposed computer-based application. An output of data modeling is a set of viable database structure proposals. The data modeling process consists of:

1. Identifying and describing the information requirements for an information system,
2. Specifying the data to be maintained by the data management system, and
3. Specifying the data structures to be used for data storage that best support the information requirements by providing efficient and effective information retrieval.

A fundamental tool used in this process is the data model, which is used both for specification of the information requirements at the user level and for specification of the data structures for the database. During implementation of a database, the data model guides construction of the schema or data catalog which contains the metadata that describe the DB structure and data semantics that are used to support database implementation and data retrieval.

Basically, a data model type should consist of 3 parts containing concepts to be used to specify:

- Data Structure – data types and inter-relationships,
- Constraints – allowable values, relationships, and cardinalities, and
- Basic Operations – for data storage, retrieval, modification, maintenance, and control.

The goal of a data modeling process is to convert information requirements, as perceived by the user community, to an efficient database structure. The modeling process consists of 3 iterative phases, each converting an input model to a more formal model with the final objective to specify a DB model for implementation.

V. DATA WAREHOUSE & DATA MINING

Data Warehouses became a distinct type of computer database during the late 1980s and early 1990s. They were developed to meet a growing demand for management information and analysis that could not be met by operational systems. Operational systems were unable to meet this need for a range of reasons:

- The processing load of reporting reduced the response time of the operational systems,
- The database designs of operational systems were not optimized for information analysis and reporting,
- Most organizations had more than one operational system, so company-wide reporting could not be supported from a single system, and
- Development of reports in operational systems often required writing specific computer programs which was slow and expensive

As a result, separate computer databases began to be built that were specifically designed to support management information and analysis purposes. These data warehouses were able to bring in data from a range of different data sources, such as mainframe computers, minicomputers, as well as personal computers and office automation software such as spreadsheet, and integrate this information in a single place. This capability, coupled with user-friendly reporting tools and freedom from operational impacts, has led to a growth of this type of computer system. As technology improved (lower cost for more performance) and user requirements increased (faster data load cycle times and more features), data warehouses have evolved through several fundamental stages:

- Offline Operational Databases – Data warehouses in this initial stage are developed by simply copying the database of an operational system to an off-line server where the processing load of reporting does not impact on the operational system's performance.
- Offline Data Warehouse – Data warehouses in this stage of evolution are updated on a regular time cycle (usually daily, weekly or monthly) from the operational systems and the data is stored in an integrated reporting-oriented data structure
- Real Time Data Warehouse – Data warehouses at this stage are updated on a transaction or event basis, every time an operational system performs a transaction (e.g. an order or a delivery or a booking etc.)
- Integrated Data Warehouse – Data warehouses at this stage are used to generate activity or transactions that are passed back into the operational systems for use in the daily activity of the organization.

Data Mining, or Knowledge Discovery in Databases (KDD) as it is also known, is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data. This encompasses a number of different technical approaches, such as clustering, data summarization, learning classification rules, finding dependency networks, analysing changes, and detecting anomalies.

VI. FUZZY DIMENSION TO DATABASES

The vast expansion of the Internet gives rise to a significant growth of the number of huge online databases. Therefore, data mining with its goal of reducing the complexity and the number of computations in very large databases attracts more and more attention from the information industry. At the stage of the typical data classification in information systems, there appear some types of uncertainty, for instance, when the boundaries of a class of objects are not sharply defined. In this case, the most common, useful and widely accepted solution is the introduction of fuzzy sets. Fuzzy sets provide mathematical meanings to the natural language statements and become an effective solution for dealing with uncertainty. In particular, only one property or measure seldom defines a business process or the quality of a service. In most practical classification situations, more than one attribute or criteria must be considered simultaneously. There is no simple procedure for combining the different criteria into one general performance measure because the criteria are measured with different scales, the relative significance of different criteria differs and for some criteria the objective is maximization, but for others it is minimization or a specific target. The approach of fuzzy set theory with its membership functions is widely used to form a realistic description of the evaluation. Different criteria with separate scales and optimization objectives can be combined into a joint response measure – the aggregated value of the membership.

VII. DISTRIBUTED DATABASE

A Distributed Database appears to a user as a single database but is, in fact, a set of databases stored on multiple computers. The data on several computers can be simultaneously accessed and modified using a network. Each database server in the distributed database is controlled by its local DBMS, and each cooperates to maintain the consistency of the global database. Figure 2 illustrates a representative distributed database system.

A Database Server is the software managing a database, and a Client is an application that requests information from a server. Each computer in a system is a Node. A node in a distributed database system can be a client, a server, or both. For example, in Figure 2, the computer that manages the HQ database is acting as a database server when a statement is issued against its own data (for example, the second statement in each transaction issues a query against the local DEPT table), and is acting as a client when it issues a statement against remote data (for example, the first statement in each transaction is issued against the remote table EMP in the SALES database).

Oracle supports heterogeneous client/server environments where clients and servers use different character sets. The character set used by a client is defined by the value of the NLS_LANG parameter for the client session. The character set used by a server is its database character set. Data conversion is done automatically between these character sets if they are different.

Distributed Database is a collection of data which belong to the same system but are spread over the sites of a computer network. This definition implies:

- Distribution: that data are not resident at the same site.
- Logical Correlation: that data have some properties which tie them together.

A. *Differences in Distributed & Centralized Databases*

In centralized control one "database administrator" ensures safety of data whereas in distributed control, it is possible to use hierarchical control structure based on a "global database administrator" having the central responsibility of whole data along with "local database administrators", who have the responsibility of local databases.

B. *Direct and Indirect Connections*

A client can connect directly or indirectly to a database server. In Figure 2, when the client application issues statements for each transaction, the client is connected directly to the intermediate HQ database and indirectly to the SALES database that contains the remote data.

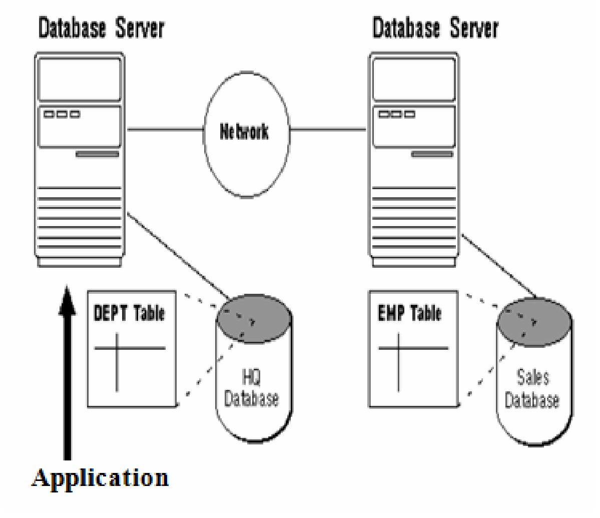

Figure 2 - An Example of a Distributed DBMS Architecture

C. *Data Independence*

In central databases it means the actual organization of data is transparent to the application programmer. The programs are written with "conceptual" view of the data (called "Conceptual schema"), and the programs are unaffected by physical organization of data. In Distributed Databases, another aspect of "distribution dependency" is added to the notion of data independence as used in Centralized databases. Distribution Dependency means programs are written assuming the data is not distributed. Thus correctness of programs is unaffected by the movement of data from one site to another; however, their speed of execution is affected.

D. *Reduction of Redundancy*

Redundancy was reduced for two reasons in centralized databases: (a) inconsistencies among several copies of the same logical data are avoided, (b) storage space is saved. Reduction of redundancy is obtained by data sharing. In distributed databases data redundancy is desirable as (a) locality of applications can be increased if data is replicated at all sites where applications need it, (b) the availability of the system can be increased, because a site failure does not stop the execution of applications at other sites if the data is replicated. With data replication, retrieval can be performed on any copy, while updates must be performed consistently on all copies.

E. *Complex Physical Structures and Efficient Access*

In centralized databases complex accessing structures like secondary indexed, interfile chains are used. All these features provide efficient access to data. In distributed databases efficient access requires accessing data from different sites. For this an efficient distributed data access plan is required which can be generated either by the programmer or produced automatically by an optimizer. Problems faced in the design of an optimizer can be classified in two categories:

- Global optimization consists of determining which data must be accessed at which sites and which data files must consequently be transmitted between sites.
- Local optimization consists of deciding how to perform the local database accesses at each site.

F. Integrity, Recovery and Concurrency Control

A transaction is an atomic unit of execution and atomic transactions are the means to obtain database integrity. Failures and concurrency are two dangers of atomicity. Failures may cause the system to stop in midst of transaction execution, thus violating the atomicity requirement. Concurrent execution of different transactions may permit one transaction to observe an inconsistent, transient state created by another transaction during its execution. Concurrent execution requires synchronization amongst the transactions, which is much harder in all distributed systems.

G. Privacy and Security

In traditional databases, the database administrator, having centralized control, can ensure that only authorized access to the data is performed. In distributed databases, local administrators face the same as well as two new aspects of the problem; (a) security (protection) problems because of communication networks is intrinsic to database systems. (b) In certain databases with a high degree of "site autonomy" may feel more protected because they can enforce their own protections instead of depending on a central database administrator.

H. Distributed Query Processing

The DDBMS should be capable of gathering and presenting data from more than one site to answer a single query. In theory a distributed system can handle queries more quickly than a centralized one, by exploiting parallelism and reducing disc contention; in practice the main delays (and costs) will be imposed by the communications network. Routing algorithms must take many factors into account to determine the location and ordering of operations. Communications costs for each link in the network are relevant, as also are variable processing capabilities and loadings for different nodes, and (where data fragments are replicated) trade-offs between cost and currency. If some nodes are updated less frequently than others there may be a choice between querying the local out-of-date copy very cheaply and getting a more up-to-date answer by accessing a distant location. The ability to do query optimization is essential in this context - the main objective being to minimize the quantity of data to be moved around. With single-site databases one must consider both generalized operations on internal query representations and the exploitation of information about the current state of the database.

I. Distributed Directory (Catalog) Management

Catalogs for distributed databases contain information like fragmentation description, allocation description, mappings to local names, access method description, statistics on the database, protection and integrity constraints (consistency information) which are more detailed as compared to centralized databases.

VIII. RELATIVE ADVANTAGES OF DISTRIBUTED DATABASES OVER CENTRALIZED DATABASES

Many organizations are decentralized, and a distributed database approach fits more naturally the structure of the organization. The organizational and economic motivations are amongst the main reasons for the development of distributed databases. In organizations already having several databases and feeling the necessity of global applications, distributed databases is the natural choice. In a distributed environment, expansion of the system in terms of adding more data, increasing database size, or adding more processors is much easier. Many applications are local, and these applications do not have any communication overhead. Therefore, the maximization of the locality of applications is one of the primary objectives in distributed database design. Data localization reduces the contention for CPU and I/O services and simultaneously reduces access delays involved in wide are networks. Local queries and transactions accessing data at a single site have better performance because of the smaller local databases. In addition, each site has a smaller number of transactions executing than if all transactions are submitted to a single centralized database. Moreover, inter-query and intra-query parallelism can be achieved by executing multiple queries at different sites, or breaking up a query into a number of sub queries that execute in parallel. This contributes to improved performance. Reliability is defined as the probability that a system is running (not down) at a certain time point. Availability is the probability that the system is continuously available during a time interval. When the data and DBMS software are distributed over several sites, one site may fail while other sites continue to operate. Only the data and software that exist at the failed site cannot be accessed. This improves both reliability and availability. Further improvement is achieved by judiciously replicating data and software at more than one site. This refers to freedom for the user from the operational details of the network. It may be divided into location and naming transparency. Location transparency refers to the fact that the command used to perform a task is independent of the location of data and the location of the system where the command was issued. Naming transparency implies that once a name is specified, the named objects can be accessed unambiguously without additional specification. Copies of the data may be stored at multiple sites for better availability, performance, and reliability. Replication transparency makes the user unaware of the existence of copies. Two main types of fragmentation are Horizontal fragmentation, which distributes a relation into sets of tuples (rows), and Vertical Fragmentation which distributes a relation into sub relations where each sub relation is defined by a subset of the column of the original relation. A global query by the user must be transformed into several fragment queries. Fragmentation transparency makes the user unaware of the existence of fragments.

IX. CONCLUSION

Database management systems are usually categorized according to the data model that they support: relational, object-relational, network, and so on. The data model will tend to determine the query languages that are available to access the database. A great deal of the internal engineering of a DBMS, however, is independent of the data model, and is concerned with managing factors such as performance, concurrency, integrity, and recovery from hardware failures. Furthermore Applications in domains such as Multimedia, Geographical Information Systems, and digital libraries demand a completely different set of requirements in terms of the underlying database models.

X. REFERENCES

[1] Codd, E.F. (1970)."A Relational Model of Data for Large Shared Data Banks". In: Communications of the ACM 13 (6): 377–387.

[2] Bradley, Neil The XML Companion. Harlow, UK: Addison-Wesley, 2000.

[3] Data Base System Concepts, Navathe, PHI

[4] Database Management Systems, Raghu Ramakrishnan (Author), Johannes Gehrke, Tata McGraw Hill

[5] An Introduction Database System, 7th ed.; Addison-Wesley 2000.

[6] Database System Concepts, Avi Silberschatz, Henry F. Korth, S. Sudarshan, McGraw-Hill

[7] Gray. J. [1981] “The Transaction Concept: Virtues and Limitations”, in VLDB [1981].

[8] http://www.nou.edu.ng/noun/NOUN_OCL/pdf/pdf2/CIT%20744.pdf

[9] M. Atkinson, R. Morrison. Orthogonally Persistent Object Systems. The VLDB Journal 4(3), 319-401, 1995

[10] Gray., J., and Reuter, A [1983] Transaction Processing: Concepts and Techniques, Morgan Kaufmann, 1993.

[11] Atzeni, P., and De Antonellis, V. [1993] Relational Database Theory Benjamin / Cummings, 1993.

[12] Eswaran, K., Gray, J., Lorie, R., and Traiger, I. [1976] “The Notions of Consistency and Predicate Locks in a Data Base System”, CACM, 19 : 11 November 1976.

[13] R.G.G.Cattell, D.K.Barry (Eds.): The Object Data Standard: ODMG 3.0. Morgan Kaufmann 2000.

[14] Boyce, R., Chamberlin, D., King, W., and Hammer, M. [1975] “Specifying Queries as Relational Expressions”, CACM, 18 : 11, November 1975.

[15] www.w3schools.com

[16] Rob, P., and Coronel, C. [2000] Database Systems, Design, Implementation, and Management,4th ed., Course Technology, 2000.

[17] Feasibility of a Set-Theoretic Data Structure : A General Structure Based on a Reconstituted Definition of Relation, D. L. Childs, 1968, Technical Report 6 of the CONCOMP (Research in Conversational Use of Computers) Project, University of Michigan, Ann Arbor, Michigan, USA

[18] http://en.wikipedia.org

[19] http://www.doc.ic.ac.uk/~pjm/adb/

[20] Gray. J. [1981] “The Transaction Concept: Virtues and Limitations”, in VLDB [1981].

[21] Advanced Database Applications

http://www.cs.bu.edu/fac/gkollios/ada05/

[22] G. Slivinskas, C. S. Jensen, and R. T. Snodgrass. Query Plans for Conventional and Temporal Queries Involving Duplicates and Ordering. In Proceedings of the 16th International Conference on Data Engineering, San Diego, California, February/March 2000

[23] Development of an object-oriented DBMS; Portland, Oregon, United States; Pages: 472 – 482; 1986; ISBN 0-89791-204-7

[24] Distributed and Parallel Databases

http://www.springer.com/computer/database+management+%26+information+retrieval/journal/10619

AUTHORS PROFILE

Praseeda Manoj received M.Sc degree in Statistics in 1996 from Calicut University, India, Master of Computer Applications from Kerala University, India in 2000 and M.Phil Computer Science degree in 2012 from Madurai Kamaraj University, India. She is now senior lecturer in Muscat College, Sultanate of Oman and pursuing P.hD in Computer Science.