WCETR 2011

The Replication Technology in E-learning Systems

Iacob (Ciobanu) Nicoleta – Magdalena

Abstract

The distributed systems are attractive for the users with various, interactive activities that require large processing capabilities, resource sharing and high availability. This paper presents the advantages of e-Learning education and focus on the importance of data replication in the design of a distributed database system that an e-learning portal uses. In addition, we analyze the available methods and strategies of data replication depending on needs, for improving performance and increasing data availability. The purpose of this paper is to present the advantages of using data replication technology in e-learning systems by universities with geographically distributed locations.

Keywords: e-Learning; distributed databases; fragmentation; replication; allocation; methods; strategies

1. Introduction

The present society is an information society, a various set of software applications being implemented day by day for helping people to make different tasks. Most areas of activities are based on information systems that led to the creation of computer networks for the rapid transmission of information and development of computer applications for the operation of various institutions, many of them distributed geographically. Thus, in many branches of activity (industry, economy, finance and banking, universities, telecommunications) the distributed systems are used. Both in research and in production, distributed technologies are growing fast. Despite the complexity and higher cost (compared with centralized databases), distributed databases are the solution to many of the new technical requirements: fast access, availability and reliability of data, large amounts of data.

A distributed database (DDB) is a collection of logically interrelated data which are physically distributed on computers (nodes) over a network. Every computer in the network has the autonomy to process local applications. Also, each computer participates in the execution stage of at least one global application that requires accessing data from multiple computers (Iacob, 2010).
A distributed database management system (DDBMS) is a set of programs that allows the management of distributed database and makes the distribution transparent to the users. The objective of transparency is to make a distributed system appear similar to a centralized system. This is usually called the fundamental principle of DDBMS.

E-learning incorporates traditional or modern learning methods and techniques and use computers to deliver them to the students - multimedia processing and asynchronous or synchronous communication. E-learning courses include both content (information) and instructional methods (techniques) that help people learn the content (Clark & Mayer, 2008).

Asynchronous e-learning can be defined as a form of computer based training, in which the teacher (trainer) and the course participants do not use educational resources at the same time, hence the name asynchronous e-learning.

Synchronous e-learning can be defined as a form of computer based training, in which teacher (trainer) and the course participants use educational resources at the same time hence the name synchronous e-learning. The students can log into the system and can interact with the teacher and other students that are in the same location or different locations (as far as the teacher allows it) in real time.

2. Distributed database

Distributed database design: The methodology used for the logical design of a centralized database applies to the design of the distributed one as well. However, for a distributed database three additional factors have to be considered: data fragmentation, data replication and data allocation.

A distributed database system is a database system which is fragmented or replicated on the various configurations of hardware and software, located usually at different geographical sites within an organization (Beynon-Davies, 2004).

2.1. Data fragmentation

Before we decide how to distribute the data we must determine the logical units of distribution. The database may be split up into logical units called fragments which will be stored at different sites. The simplest logical units are the tables themselves.

Fragmentation aims to improve:
- Reliability;
- Performance;
- Balanced storage capacity and costs;
- Communication costs;
- Security.

The fragmentation is the partitioning of a global relation \( R \) in fragments \( R_1, R_2, \ldots, R_n \) containing enough information to reconstruct the original relation \( R \).

There are some basic rules to be followed in defining the fragments:
- **Completeness.** Decomposition of relation \( R \) into fragments \( R_1, R_2, \ldots, R_n \) is complete if each data item in \( R \) can also be found in some \( R_i \).
- **Reconstruction.** If relation \( R \) is decomposed into fragments \( R_1, R_2, \ldots, R_n \), then there should exist some relational operator \( \sqcup \) that reconstructs \( R \) from its fragments, i.e., \( R = R_1 \sqcup \ldots \sqcup R_n \).
- **Disjointness.** If relation \( R \) is decomposed into fragments \( R_1, R_2, \ldots, R_n \) and data item \( d_i \) appears in fragment \( R_j \), then \( d_i \) should not appear in any other fragment \( R_k \), \( k \neq j \) (exception: primary key attribute for vertical fragmentation).

The following information is used to decide fragmentation:
- Quantitative information: frequency of queries, site, where query is run, selectivity of the queries, etc;
- Qualitative information: types of data access, read/write, etc.

There are two types of fragmentation: horizontal fragmentation and vertical fragmentation.
- **Horizontal fragmentation:** A horizontal fragment of a table is a subset of rows in it. So horizontal fragmentation divides a table 'horizontally' by selecting the relevant rows and these fragments can be
assigned to different sides in the distributed system. The relation is split horizontally, i.e., tuples are distributed among fragments.

- **Vertical fragmentation:** a vertical fragment of a table keeps only certain attributes of it. It divides a table vertically by columns. It is necessary to include the primary key of the table in each vertical fragment so that the full table can be reconstructed if needed.
- **Mixed fragmentation:** is achieved by successively applying the operations for horizontal and vertical fragmentation. In this way, mixed fragments can be obtained by applying horizontal fragmentation to a vertical fragment, respectively by applying a vertical fragment to a horizontal fragment. In practice it is recommended to have at most two levels of fragmentation.

### 2.2. Data replication

A copy of each fragment can be maintained at several sites. Data replication is the design process of deciding which fragments will be replicated. **Replication** is the operation of storing portions from a database, as copies, on multiple nodes on a network. If a user updates a local copy then DDBMS automatically updates all copies of that data. The fragmentation and the replication can be combined: A relationship can be partitioned into several pieces and can have multiple replicas of each fragment (Silberschatz, Korth & Sudarshan, 2010).

The purposes of replication are multiple:

- **System availability.** Distributed DBMSs may remove single points of failure by replicating data, so that data items are accessible from multiple sites. Consequently, even when some sites are down, data may be accessible from other sites.
- **Performance.** One of the major contributors to response time is the communication overhead. Replication enables us to locate the data closer to their access points, thereby localizing most of the access that contributes to a response time reduction.
- **Scalability.** As systems grow geographically and in terms of the number of sites (consequently, in terms of the number of access requests), replication allows for a way to support this growth with acceptable response times.
- **Application requirements.** Finally, replication may be dictated by the applications, which may wish to maintain multiple data copies as part of their operational specifications (Ozsu & Valduriez, 2011).

Although data replication has clear benefits, it poses the considerable challenge of keeping different copies synchronized.

Replication is a process that consists in making and distributing copies of data between different places, at remote or mobile users, by using Internet. In addition, allows changes to be propagated consistently to the relevant copies. Distribution of these replicas must ensure that the data processing is done locally, in order to maximize efficiency and reduce communication costs. In addition, replication provides more reliability, minimizes the chance of total data loss, and greatly improves disaster recovery (Rahimi & Haug, 2010).

**Epsilon serializability (ESR)** allows a query to see inconsistent data while replicas are being updated, but requires that the replicas converge to a one-copy serializable state once the updates are propagated to all of the copies. It bounds the error on the read values by an epsilon ($\varepsilon$) value (hence the name), which is defined in terms of the number of updates (write operations) that a query “misses”. Given a read-only transaction (query) $T_Q$, let $T_U$ be all the update transactions that are executing concurrently with $T_Q$. If $RS(T_Q) \cap WS(T_U) \neq \emptyset$ ($T_Q$ is reading some copy of some data items while $T_U$ is updating (possibly a different) copy of those data items) then there is a read-write conflict and $T_Q$ may be reading inconsistent data. The inconsistency is bounded by the changes performed by $T_U$. Clearly, ESR does not sacrifice database consistency, but only allows read-only transactions (queries) to read inconsistent data. For this reason, it has been claimed that ESR does not weaken database consistency, but “stretches” it.

#### 2.2.1. Replication methods

The methods that can be used for replication (Table 1) are:

- **not replicated** data means that DDBMS allocates space for some data on a single node from a computer network. The characteristics of this method are: minimal redundancy, data access concurrency is maximum, update time is small and retrieval time is great.
- **partitioned**: each fragment resides at only one site. Rule of thumb: If \(\frac{\text{read only queries}}{\text{update queries}} \geq 1\), then replication is advantageous, otherwise replication may cause problems.

- **partially replicated** data means that DDBMS allocates for a part of data a single copy on a computer, and for another part of data multiple copies on multiple computers on the network. The characteristics of this method are: redundancy is increased, concurrency access to data is decreased, update and retrieval time is medium.

- **fully replicated** data means that DDBMS allocates for the entire database multiple copies on different computers on the network. The characteristics of this method are: redundancy is high, concurrency access to data is minimal, the time update is great and retrieval time is small.

<table>
<thead>
<tr>
<th>Method of data replication</th>
<th>Not replicated</th>
<th>Fully replicated</th>
<th>Partially replicated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data consistency</td>
<td>Large</td>
<td>Small</td>
<td>Small</td>
</tr>
<tr>
<td>Storage costs</td>
<td>Small</td>
<td>Large</td>
<td>Medium</td>
</tr>
<tr>
<td>Reliability and availability</td>
<td>Small</td>
<td>Large</td>
<td>Medium</td>
</tr>
<tr>
<td>Update speed</td>
<td>Medium</td>
<td>Small</td>
<td>Small</td>
</tr>
<tr>
<td>Communication costs</td>
<td>Large</td>
<td>Large</td>
<td>Small</td>
</tr>
<tr>
<td>Redundancy</td>
<td>Small</td>
<td>Large</td>
<td>Medium</td>
</tr>
<tr>
<td>Concurrency access to data</td>
<td>Large</td>
<td>Small</td>
<td>Medium</td>
</tr>
<tr>
<td>Update time</td>
<td>Small</td>
<td>Large</td>
<td>Medium</td>
</tr>
<tr>
<td>Retrieval time</td>
<td>Large</td>
<td>Small</td>
<td>Medium</td>
</tr>
</tbody>
</table>

The replication of a fragment \(R_i\) at \(j\) station, formula (1), is justified if the fragment allocation cost is less than the cost of remote accessing the applications \(k\) from \(j\) station to other copies of the fragment found on other stations \(j'\) \((j' \neq j)\):

\[
C_{ij} < \sum_k d_{ij'}
\]  

(1)

The replication is a solution for a distributed database environment when we need: to copy and distribute the data to one or more places; to distribute the copies of the data on a scheduled basis; when needed, to allow more users to make changes, then send data changes together, and even to resolve potentially conflicts; to build applications with data which can be used in online or offline environments; to build Web applications in which the users can access large volumes of data.

### 2.2.2. Replication database strategies

There are two basic parameters to set in the design process of a replication strategy, "where" and "when" the updates are propagated. So, depending on when parameter, the replication can be: **synchronous** and **asynchronous**.

1) **In synchronous replication**, replicas are kept in sync at all times. Synchronous replication means that an update should be propagated and applied immediately in all replicas, so this way database consistency is assured all the time. In this approach, a transaction may access any copy because the accessed data are the same in all replicas from different locations. Some applications, for example financial or reservation of places, are well served by this technique. This process is usually done using a protocol called "two phase commit" and refers to the transaction that performs the update.

2) Since synchronous replication is often related to distributed databases and involves a very high level of complexity, which often translates into very high costs (both for communication, design and operation), **asynchronous replication** is used as an alternative, because is more cheaper. In asynchronous replication, unlike synchronous replication, the replicas are not always kept synchronized and this means that synchronization process of databases occurs only periodically. Two or more replicas of the same data can sometimes have different values and in a transaction these different values may be seen. This is acceptable for some applications that don’t require the updates to be made in real time. Depending on where the updates can take place we have: **primary copy (master)** and **update everywhere (group)**.

3) With an **update everywhere** approach, changes can be initiated at any replicas. This means that any of the sites that have a copy of the information can update its value. Each location may make the data
updates, and changes are replicated to all other nodes. It is obvious that this configuration is the most exposed to integrity loss, due to conflicting updates, also called "conflicts".

4) With a **primary copy** approach, there is only one copy that can be updated (master copy), all others (secondary copies) being updated in order to reflect changes in the master copy.

### 2.3. Data allocation

Each fragment has to be allocated to one or more sites, where it will be stored.

There are three *strategies* regarding the allocation of data:

- **Centralized**: Consists of a single DB and DBMS stored at one site with users distributed across the network
- **Fragmented** (or **Partitioned**): The database is partitioned into disjoint fragments, with each fragment assigned to one site (no replication). This is also called 'non-redundant allocation'.
- **Complete replication**: A complete copy of the database is maintained at each site (no fragmentation). Here, storage costs and communication costs for updates are most expensive. To overcome some of these problems, snapshots are sometimes used. A snapshot is a copy of the data at a given time. Copies are updated periodically.
- **Selective replication**: A combination of fragmentation and replication.

### 3. Conclusion

Distributed databases have appeared as a necessity, because they improve availability and reliability of data and offer high performance in data processing by allowing parallel processing of queries, and also reduce processing costs.

The use of distributed databases in e-Learning systems improves access to information and offer rapid data collection.

Modern universities, with geographically distributed locations, must assume the responsibility to introduce new technologies in the educational process and must adopt learning methods based on the new technologies that are more efficient than the traditional ones. In other words, in order to be competitive, any university must take these technologies into account.

### References