Preserving Consistency and Security of Data in E-business Applications

CRISTEA BOBOILA, NICOLAE CONSTANTINESCU, COSTIN-RADU BOLDEA
Department of Computer Science
University of Craiova
Bd. A.I. Cuza, no 13, Craiova 200585
ROMANIA
{boboila, nikyc, cboldea}@central.ucv.ro

Abstract: - Databases are a central component of e-business applications. They provide the storage and functional facilities (i.e. accessing transactional information) of commercial systems. This paper analyzes database models for e-commerce applications, and studies access constraints necessary to preserve the consistency of data; in particular, access scenarios in which phantoms appear are investigated.

Key-Words: - Concurrent transactions, Database management systems, Phantoms, Locking, Consistency.

1 Introduction

The most valuable assets of an enterprise are represented by its data, not its offices or factories. Regardless of the nature of the enterprise - either a private business, an educational institution, or a government agency - the actual record of its history and current state is imprinted upon its data. If the data is inconsistent with the state of the enterprise, or if the data is inconsistent with itself, its usefulness is questionable, and problems are certain to arise.

The application programs perform updates on the data in accordance with changes to the state of the enterprise. Application system components share the same set of data, and typically, but not necessarily, share the same computing environment. Today, application systems use many platforms: desktop microcomputers, file servers, and mainframes. With cooperative Web-based computing coming into vogue, sometimes we don't know or even care where the application is running. Although each platform brings its unique benefits and challenges to the science of data management, the fundamental concepts of database management software have been unchanged for more than three decades [1, 13].

Database management systems (DBMSs) were created to enforce order and consistency upon file-based application systems. With database systems, no longer are programmers free to describe and access a data element in any manner they please.

The rest of the paper is organized as follows. Section 2 describes database models for business applications. Section 3 presents experimental aspects, analyzing cases where phantoms occur while running concurrent transactions. Section 4 concludes the paper.

2 Databases for Business Application

A database is software specifically designed for data storing and modifying [14], ideal for most of the e-commerce sites to store information about products and users’ data. A database is one element that can not be absent from an effective and productive website, as it has to correspond to the data type and has to be easily updated. Usually, a database keeps the information from every page of a website and the inventory for each item, but it can also manage websites, keep the links intact, and enhance the security of the site. The easiest way to build a web page through databases is to use a program that queries a database for inventory items and produces an HTML page based on the results of that query [7].

There are two major types of databases:

a) The analytical databases are often referred to as “OnLine Analytical Processing” (OLAP), and store archived, historical data used for analysis. Web pages are generated dynamically by querying the available products in the inventory.

b) The operational databases are an integral part of OLTP (“OnLine Transaction Processing”); they manage more dynamic data by modifying it, deleting or adding data. OLTP is generally used to return the quantity and availability of an item.

2.1 Database Management Systems (DBMS)

Technically, a DBMS is a software program designed for storing and accessing information which supports the functioning of a website.
Most databases are structured in the basic form of “rows” and “columns”. Databases allow the separation of the content - the catalog offers different from the graphical design.

Because most of the companies wish to allow its customers to quickly and optimally search for the wanted item, the database integration will be a challenge for the IT specialists to join it with the website.

There is a single definition of the data elements in a database management system. This definition is a system's database schema. On some systems, a distinction is made between the programmer's view of the database, its logical schema, and the computer system's view of the database, called its physical schema. The database management system integrates the physical and logical views of the database. Application programs employ the logical schema presented by the database management system to read and update data within the physical schema, under control of the database management system and the operating system [6].

The goal of database management systems is to provide timely and easy access to large amounts of data, but to do so in ways that assure database integrity is always preserved [9, 10]. This means that a database management system must allow users to define and manage rules, or constraints, placed on certain critical data elements. The definition and enforcement of security and data integrity constraints are critical to the usefulness of any database management system.

An important component of a database management system is its transaction manager. A transaction manager controls updates to data objects in such a way as to assure that the database is always in a consistent state. Formally, a transaction manager controls changes to the state of data so that each transaction has the following properties [8]:

- **Atomicity** - All related updates take place within the bounds of the transaction or no updates are made at all.
- **Consistency** - All updates comply with the constraints placed on all data elements.
- **Isolation** - No transaction can interfere with the activities or updates of another transaction.
- **Durability** - Successful transactions are written to "durable" media (e.g., magnetic disk) as soon as possible.

These four items are known as the **ACID properties** of transaction management. Transaction managers prevent race conditions through their enforcement of atomicity and isolation.

The phantom phenomenon may occur when one transaction establishes a set of documents as the result of a query and other concurrent transactions insert or delete documents which would have been part of that set (Figure 1) [1].

1. Transaction 1 reads some data that fulfills some search condition.
2. Transaction 2 inserts a document which would have been part of the result set calculated by transaction 1.
3. Transaction 3 deletes a document which was part of the result set calculated by transaction 1.
4. Both transaction 2 and transaction 3 commit.
5. Transaction 1 executes the same query. The results are different.

The database access controls such as security, index management, and lock management consume tremendous system resources. In fact, this overhead was so great on early systems that some people argued successfully to continue using their file-based systems, because their host computers could not handle the database management load. Even with today's enormously powerful systems, throughput can suffer severely if the database system isn't properly tuned and maintained. High-volume transaction environments are often attended to by systems programmers and database analysts whose sole job is to keep the system working at optimal performance [5].

Transaction management is one database component often partitioned from the core data management functions of a database management system. Transaction managers are particularly useful when business transactions span two or more separate databases.

Database system software controls access to data files, often through the services of a transaction processing system. The ACID properties of a database system ensure that the data is always in a consistent state.

2.2 Database Models

There are basically three models used today: the text file, relational and object. Virtually, any information can be stored into a database (image, text, sound files or movies) but not all databases fit for every business situation [4].

**Text File Databases.** Text file databases are very simple and easy to use but are not recommended for a website because of their no concurrent ability (they cannot be use by two people at the same time.)

**Relational Databases.** It is organized in rows and columns and can be used by many people at the same time but all data in the database column must be of the same type and the rows are not ordered. When creating a relational database you have to normalize the structure of the data by eliminating the redundancy and the inconsistency. A RDBMS (Relational Database Management System) is a program that helps you to create, update and administer the database. It takes the SQL statements and makes the operation on the database. Some of the best-known relational database packages include: Microsoft’s SQL Server, Oracle, Sybase, Informix, and IBM’s DB2.

Figure 2 shows a typical Client-Server connection: a user makes a request of a database from his PC and the client software establishes a connection through TCP/IP sockets to the database server.

![Figure 2: A typical Client-Server connection](image)

Once the connection is verified, the client software sends the SQL queries to the server who returns the matching data back to the client who displays them on the user’s screen [2].

![Figure 3: The web server program](image)
An RDBMS is needed if we have data that changes frequently. Relational databases have concurrency control to ensure that the tables won’t get corrupted even if many people simultaneously write to and read from the database.

The term “client/server” was devised to describe how users work with relational databases. Web servers are designed to handle large scale, dynamic websites, i.e. they can deal efficiently with any RDBMS.

Figure 3 illustrates a web server program such as Apache or perhaps a CGI script for a RBDMS-backed website. The user types a URL address or a text in a navigator (Netscape Navigator or Internet Explorer) and the RBDMS server responds to it [2]. The retrieved data travels then from the server to the client and back.

Object-Oriented Databases. An object-oriented database actually provides database management system for the object-oriented programs such as C++ or Smalltalk. Because of its two new features, the heterogeneity of the objects and the inheritance, OBDBMS becomes more and more wanted and well paid. Some examples of object-oriented databases are: GemStone, NeoAccess, ObjectStore, Fast Objects, and so on.

In a relational database each record must have a unique primary key to avoid redundancy. On the other side, in an object-oriented database, objects can contain a variety of attributes and data, and each object implements its own query methods.

The basic difference between the two database models is that the relational one is structured into rows and columns while the object-oriented databases keeps the information from every item on the website in a different object. To use a relational database when dealing with image or audio files can require extra special coding that can bring problems to your website. Object-oriented databases not only manage the images and the sounds from a website very well, but are also optimal for managing a complex interrelationship between the objects. So, generally, object-oriented databases are the right choice for a website which contains audio or image files and can call multiple servers, unlike relational databases. It is also possible to build a website using both types of databases [1].

3 Phantoms. Experimental Aspects

We ran several experiments at REPEATABLE READS isolation level of SQL [11, 12]. At this isolation level, the transaction acquires read locks on all retrieved data, but does not acquire range locks. Our experiments test REPEATABLE READS both with and without delay between transactions, in order to analyze situations in which phantoms appear [3].

We run two transactions in parallel: one transaction updates some price (SetPrice) and the other reads the price two consecutive times (ReadPrice). We observe that our database is prone to phantoms, because the update can take place in-between the database reads.

For REPEATABLE READS without delay, no phantoms were detected in our experiments. Although theoretically they might appear at this isolation level, it seems that the two reads (select operations in the ReadPrice transaction) are performed close enough, not leaving time for the update to happen in-between. The number of occurrences is close (614 for SetPrice, and 615 for ReadPrice).

**REPEATABLE READS without delay:**

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetPrice</td>
<td>614</td>
</tr>
<tr>
<td>ReadPrice</td>
<td>615</td>
</tr>
</tbody>
</table>

For REPEATABLE READS with delay, phantoms appear. Due to the delay, the update (write) in the SetPrice transaction gets interleaved between the two selects (reads) in ReadPrice, generating phantoms. During our experiments, we obtained an average of 20 phantoms.

At this level of isolation, only non-repeatable reads are prevented, but phantoms are not. We can have repeatable reads because there will be intention shared locks and intention exclusive locks at the upper level, and shared and exclusive locks on rows. Therefore, the intention shared locks and intention exclusive locks do not conflict, allowing access to different rows in the table. The row that is updated is not initially among the ones that are read and on which the shared lock is set, allowing a phantom to appear.

**REPEATABLE READS with delay:**

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetPrice</td>
<td>253</td>
</tr>
<tr>
<td>ReadPrice</td>
<td>32</td>
</tr>
</tbody>
</table>

The number of occurrences is significantly different between the two transactions. SetPrice has more occurrences than ReadPrice, because ReadPrice takes more time with the delay. Since they are not serializable, SetPrice does not have to
wait for the other transaction, thus it occurs more often. The average response time for ReadPrice is higher than for SetPrice, because it is called less times that SetPrice during the period of the experiment.

The number of phantoms relative to the total number of times the transaction was run is:

$$\frac{20}{32} = 0.625 = 62.5\%$$

This result shows that phantoms occur rather often. For REPEATABLE READS without delay, the ratio is 0, because we have no phantoms.

In conclusion, the number of phantoms obtained was none for REPEATABLE READS without delay, and approximately 20 for REPEATABLE READS with delay (in 62.5% cases of transaction runs).

4 Conclusion

When building a business website, the type of operating system, web server and databases to use is an important decision. A well chosen database may provide simple, flexible and affordable solutions to a business web site, such as membership login (user feedback systems) and product catalogs. To sum up, a database is the ideal solution and the most important element in a business application site to provide exactly what the customers demand.

References: