Automated Stock Control System for Bookshops in Tertiary Institutions

Oladimeji Adegbola Isaac[1], Boniface Ekechukwu[2] and Chiamaka Ijioma Chukwuneke[3]

[1]Department of Computer science, College of Education, Azare, Bauchi State.
[2]Department of Computer Science, Nnamdi Azikiwe University, PMB5025 Awka, Anambra State, Nigeria
boni_eke@yahoo.com

Abstract

The recent need for automated stock control system for bookshops in tertiary institutions was generated by unequal availability of books and stiff scarcity of books in some areas while in other areas books are being wasted or unsold. This research has made use of distributed database systems in developing stock control systems for bookshops in tertiary institutions. Stock information is generally accessible from various terminals (Institution Bookshops) and stock queries are made easy and fast through any participating terminal (institution Bookshop). A distributed database can normally be applied by business units, companies, or geographical regions. To improve data availability and reliability, replication of data is applied in this work. This approach provides for faster response times for users because the database is local to each business unit within the organization. The business unit is typically smaller than the entire organization, which reduces the overall load on the each server. A higher degree of availability can be achieved by storing multiple copies of the same information in different sites of Bookshops in tertiary institutions. Object Oriented Analysis and Design Methodology was used in this research. The major objective of this research is to design and implement the distributed database system for automated control system for bookshops in tertiary institutions.

Keywords: Distributed Database, higher availability, bookshops, OOADM

1.0 Introduction

When an organization is geographically dispersed, it may choose to store its databases on a central database server or to distribute them to local servers (or a combination of both). Distributed database is a database in which the data is contained within a number of separate subsystems, usually in different physical locations. If the constituent subsystems are essentially similar, the system is said to be homogeneous, otherwise it is said to be heterogeneous. A distributed database is a single logical database that is spread physically across computers in multiple locations that are connected by a data communications network. We emphasize that a distributed database is truly a database, not a loose collection of files. The distributed database is still centrally administered as a corporate resource while providing local flexibility and customization. The network must allow the users to share the data; thus a user (or program) at location A must be able to access (and perhaps update) data at location B. The sites of a distributed system may be spread over a large area (e.g., the United States or the world) or over a small area (e.g., a building or campus). The computers may range from PCs to large-scale servers or even supercomputers. Distributed database is a single logical database that is spread physically across computers in multiple locations that are connected by a data communication link. [6]

A distributed database requires multiple instances of a database management system (or several DBMSs), running at each remote site. The degree to which these different DBMS instances cooperate, or work in partnership, and whether there is a master site that coordinates requests involving data from multiple sites distinguish different types of distributed database environments. It is important to distinguish between distributed and decentralized databases. A decentralized database is also stored on computers at multiple locations; however, the computers are not interconnected by network and database software that make the data appear to be in one logical database. Thus, users at the various sites cannot share data. A decentralized database is best regarded as a collection of independent databases, rather than having the geographical distribution of a single database.

2.0 Background Information

Centralized data processing was dominant from the late 1960s until the mid 1980s[3]. In the 1980s lower priced PC became available (widespread now). PCs were placed at various sites within an organization and connected to a network. This allowed users to access data from anywhere along the network. This was the beginning of distributed processing. [1] Distributed database (DDB) is the database component of distributed processing. A DDB is a single logical DB that is physically distributed to computers at several sites in a computer network. A distributed database management system (DDBMS) is needed to support and manipulate DDBs. A communications network allows computers at different sites to communicate with each other. Computers communicate by sending messages. Messages increase traffic on the network. Usually better to sent a small number of lengthy messages rather than a larger number of short messages. [7]

A DDBMS can be either homogeneous (same DBMS at all sites) or heterogeneous. Heterogeneous systems are more complex and difficult to manage. Users should be unaware that the database is not all together in one location[2]. In a distributed system, if all sites use the same DBMS product, it is called a homogenous distributed database system. However, in reality, a distributed database has to be constructed by linking multiple already-existing database systems together, each with its own schema and possibly running different database management software. Such systems are called heterogeneous distributed database systems. In a heterogeneous distributed database system, sites may run different DBMS products that need not be based on the same underlying data model, and thus, the system may be composed of relational, network, hierarchical, and object-oriented DBMSs. Homogeneous distributed DBMS provides several advantages such as simplicity, ease of designing and incremental growth. It is much easier to design and manage a homogeneous distributed DBMS than a heterogeneous one.

In a homogeneous distributed DBMS, making the addition of a new site to the distributed system is much easier, thereby providing incremental growth. These systems also improve performance by exploiting the parallel processing capability of multiple sites.

3.0 Materials And Methods

This research used Object Oriented Analysis and Design Methodology for the design and development of automated control systems for bookshops in tertiary institutions. We live in a world of objects. These objects exist in nature, in man-made entities, in business, and in the products that we use. They can be categorized, described, organized, combined, manipulated and created. Therefore, an object-oriented view has come into picture for creation of computer software. An object-oriented approach to the development of software was proposed in late 1960s[4]. Object-Oriented development requires that object-oriented techniques be used during the analysis, and implementation of the system. This methodology asks the analyst to determine what the objects of the system are, how they behave over time or in response to events, and what responsibilities and relationships an object has to other objects[8]. Object-oriented analysis has the analyst look at all the objects in a system, their commonalties, difference, and how the system needs to manipulate the objects[5]. Based on this system study, the analyst prepares a model of the desired system.

4.0 System Design

System design is the process of designing the architecture, components, modules, interfaces and data for a computer system to satisfy specified requirements. It is the process of defining and developing a system to satisfy specified requirements of the market or customer as the case may be. Database specification is included in the system designed which allows replication of data to achieve reliability since it is possible to continue the normal operations of a particular site in case of site failure, by referencing the copy of the same information from other sites.

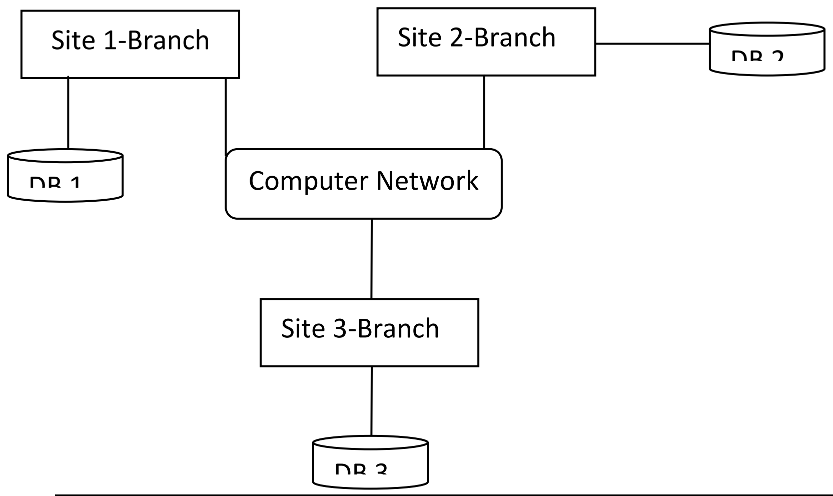


Fig.1: Distributed Database for bookshops in tertiary institutions

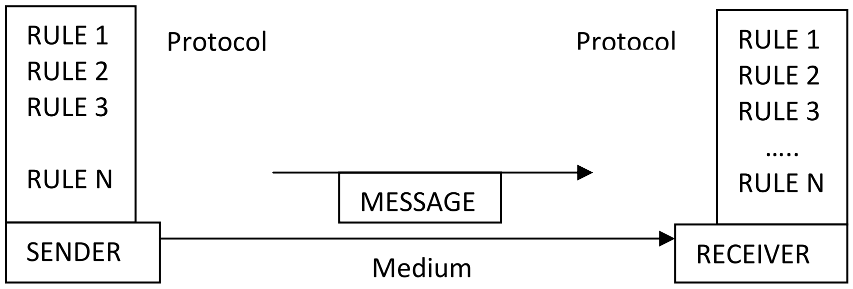


Fig 2: Data communication

There are five components in data communication. The components include, message, sender, receiver, protocol, and transmission medium. Tertiary institutions are located in many places throughout the country. Each branch has its own local system that maintains information regarding all the clients, projects and employees in that particular branch. Each such individual branch is termed a site or a node. All sites in the system are connected via a communication network. Each site maintains three relational schema: Project for project information, Client for client information and Employee for employee information, which are listed in the following. Project = (project-id, project-name, project-type, project-leader-id, branch-no, amount)
Client = (client-id, client-name, client-city, project-id)
Employee = (emp-id, emp-name, designation, salary, emp-branch, project-no)

Physical Topology for bookshops in tertiary institutions

[9]The term physical topology refers to the way in which a network is laid out physically. Two or more devices connect to a link; two or more links form a topology. Type of connection employed in this research is multipoint as shown below.

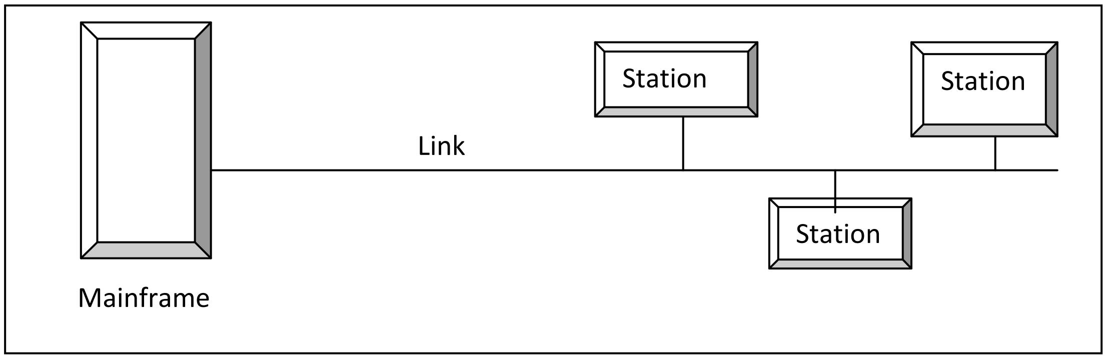
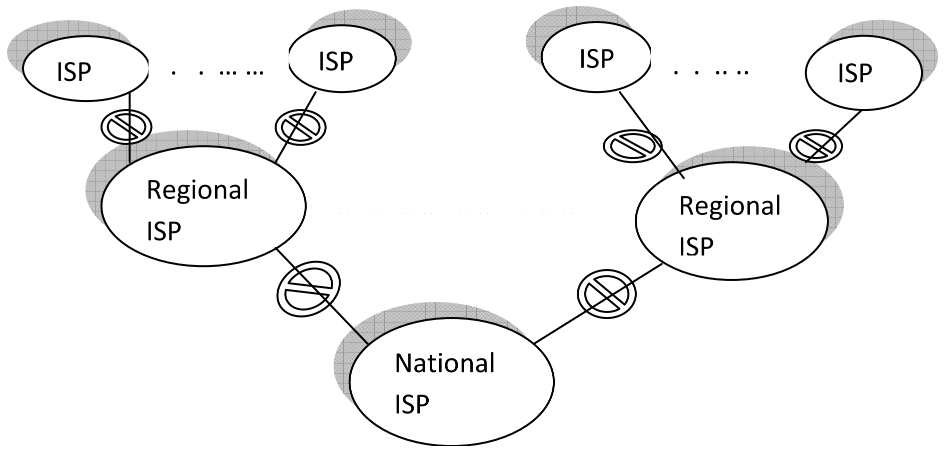


Fig.3: **Multipoint connections**

The topology of a network is a geometric representation of the relationship of all the links and linking devices (usually called nodes) to one another. System Design is part of development stage where the overall architecture of the desired system is decided. The system is organized as a set of sub systems interacting with each other. While designing the system as a set of interacting subsystems, the analyst takes care of specifications as observed in system analysis as well as what is required out of the new system by the end user. [10]As the basic philosophy of Object-Oriented method of system analysis is to perceive the system as a set of interacting objects, a bigger system may also be seen as a set of interacting smaller subsystems that in turn are composed of a set of interacting objects.

Hierarchical Organization of the Internet



Structure of national ISP

Fig 4: ***Hierarchical organization of the internet***

While designing the system, the stress lies on the objects comprising the system and not on the processes being carried out in the system as in the case of traditional Waterfall Model where the processes form the important part of the system.

The phase of Object Design follows after the details of the system analysis and system design are implemented. The Objects identified in the system design phase are designed. Here the implementation of these objects is decided as the data structures get defined and also the interrelationships between the objects are defined. Object Oriented Philosophy is very much similar to real world and hence is gaining popularity as the systems here are seen as a set of interacting objects as in the real world. To implement this concept, the process-based structural programming is not used; instead objects are created using data structures. Just as every programming language provides various data types and various variables of that type can be created, similarly, in case of objects certain data types are predefined. If we can define a data type called pen and then create and use several objects of this data type. This concept is known as creating a class. A class is a collection of similar objects. It is a template where certain basic characteristics of a set of objects are defined. The class defines the basic attributes and the operations of the objects of that type. Defining a class does not define any object, but it only creates a template. For objects to be actually created instances of the class are created as per the requirement of the case. In the case of abstraction**:** Classes are built on the basis of abstraction, where a set of similar objects are observed and their common characteristics are listed. Of all these, the characteristics of concern to the system under observation are picked up and the class definition is made. The attributes of no concern to the system are left out. This is known as abstraction. The abstraction of an object varies according to its application.

For instance, while defining a pen class for a stationery shop, the attributes of concern might be the pen color, ink color, pen type etc., whereas a pen class for a manufacturing firm would be containing the other dimensions of the pen like its diameter, its shape and size etc. Inheritance is another important concept in this regard. This concept is used to apply the idea of reusability of the objects. A new type of class can be defined using a similar existing class with a few new features. For instance, a class vehicle can be defined with the basic functionality of any vehicle and a new class called car can be derived out of it with a few modifications. This would save the developers time and effort as the classes already existing are reused without much change. Coming back to our development process, in the Object Designing phase of the Development process, the designer decides onto the classes in the system based on these concepts. The designer also decides on whether the classes need to be created from scratch or any existing classes can be used as it is or new classes can be inherited from them. During the phase of implementation, the class objects and the interrelationships of these classes are translated and actually coded using the programming language decided upon. The databases are made and the complete system is given a functional shape. The complete Object oriented (OO) methodology revolves around the objects identified in the system. When observed closely, every object exhibits some characteristics and behavior.

The objects recognize and respond to certain events. For example, considering a Window on the screen as an object, the size of the window gets changed when resize button of the window is clicked. Here the clicking of the button is an event to which the window responds by changing its state from the old size to the new size. While developing systems based on this approach, the analyst makes use of certain models to analyze and depict these objects. The methodology supports and uses three basic Models: Object Model describes the objects in a system and their interrelationships. This model observes all the objects as static and does not pay any attention to their dynamic nature. Dynamic Model depicts the dynamic aspects of the system. It portrays the changes occurring in the states of various objects with the events that might occur in the system. Functional Model basically describes the data transformations of the system. This describes the flow of data and the changes that occur to the data throughout the system. While the Object Model is most important of all as it describes the basic element of the system, the objects, all the three models together describe the complete functional system. As compared to the conventional system development techniques, Object Oriented (OO) modeling provides many benefits. Among other benefits, there are all the benefits of using the Object Orientation. Some of these include Reusability, Inheritance and Data Hiding. In the case of reusability - The classes once defined can easily be used by other applications. This is achieved by defining classes and putting them into a library of classes where all the classes are maintained for future use.

Whenever a new class is needed the programmer looks into the library of classes and if it is available, it can be picked up directly from there. In the case of Inheritance - The concept of inheritance helps the programmer use the existing code in another way, where making small additions to the existing classes can quickly create new classes. Programmer has to spend less time and effort and can concentrate on other aspects of the system due to the reusability feature of the methodology. In the case of Data Hiding - Encapsulation is a technique that allows the programmer to hide the internal functioning of the objects from the users of the objects. Encapsulation separates the internal functioning of the object from the external functioning thus providing the user flexibility to change the external behavior of the object making the programmer code safe against the changes made by the user. The systems designed using this approach are closer to the real world as the real world functioning of the system is directly mapped into the system designed using this approach.

5.0 Results and Discussions

Management of distributed data with different levels of transparency like network transparency, fragmentation transparency, replication transparency, facilitates the transaction systems for bookshops in tertiary institutions. The transaction systems for bookshops in tertiary institutions have become reliable especially in the aspect of stock availability using this new system. Even when there is interconnectivity of transaction systems for bookshops in tertiary institutions the Local autonomy or site autonomy still exist for the profit control. The distributed database system in bookshops transactions has avenue for protection of valuable data. It follows that if there were ever a catastrophic event such as a fire, all of the data would not be in one place, but distributed in multiple locations. This new transaction system for bookshops in tertiary institutions improved performance of the bookshops as data is located near the site of greatest demand, and the database systems themselves are parallelized, allowing load on the databases to be balanced among servers. (A high load on one module of the database won't affect other modules of the database in a distributed database.). On economic level the transaction system for bookshops in tertiary institutions costs less to create a network of smaller computers with the power of a single large computer. The new system has advantage of modularity. Systems can be modified, added and removed from the distributed database without affecting other modules (systems). The replication effect makes the new system reliable for transactions due to replication of database. The new system supports continuous operation, distributed query processing and distributed transaction management. Single site failure does not affect performance of system. All transactions follow A.C.I.D property: A stands for Atomicity, the transaction takes place as a whole or not at all; C represents Consistency, - it maps one consistent DB state to another; I denotes Isolation, each transaction sees a consistent DB; D for Durability which emphasis the need for, the results of a transaction to survive system failures.

References

[1] O'Brien, J. & Marakas, G.M.(2008) Management Information Systems (pp. 185- 189). New York, NY: McGraw-Hill Irwin.

[2] Bell, D., and J. Grimson. 1992. Distributed Database Systems. Reading, MA: Addison-Wesley.

[3] Date, C. J. 2003. An Introduction to Database Systems, 8th ed.Reading, MA: Addison-Wesley.

[4] http://en.wikipedia.org/wiki/Distributed_database. Retrieved on the 19th December, 2012.

[5] Lightstone, S. Teorey, T. Nadeau, T. (2007). Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more. Morgan Kaufmann Press.

[6] Alkhatib G., Labban R. S. (2002): "Transaction Management in Distribute Database Systems: the Case of Oracle's Two-Phase Commit" *Journal of Information Systems Education*, Vol. 13(2), pp 95-104

[7] Borysowich C. (2007): Overview of Distributed Databases. Observations from a Tech Architect: Enterprise Implementation Issues & Solutions *http://it.toolbox.com/blogs/enterprise-solutions/overview-of-distributed-databases-16228.htm Accessed July 5, 2010*

[8] Breck L. (2010): *An Introduction to Client/Server Database Development*. MacTech – A Journal of Apple Technology Vol. 11 No 6. Accessed July 6, 2010

[9] Ceri S. and Pelagatti G. (1984): *Distributed Database Design: Principles and Systems*. McGraw-Hill.

[10] Codd E. F. (1970): "A Relational Model for Large Shared Data Banks," *Communications of the ACM*, Vol. 13, No 6, PP 377 – 387