Design and Implementation of a Client Server Distributed Database for Student Results Processing.

Djam Xaveria Youh, M.Sc.

Department of Mathematics, Gombe State University, PMB 127, Gombe State, Nigeria.

E-mail: dxaveria@yahoo.com
Telephone: +235-8068993095

ABSTRACT

Today's university environment has an increasing need for distributed database systems as the desire for easy, reliable, scalable, and accessible information is steadily increasing. The inadequacies of centralized database systems and manual systems in handling students' result necessitated the use of distributed database systems in this paper. The proposed system is a relational database designed in a way that each academic department in the university has its own database including the Central Record Processing Unit (CRPU), Exams and Record Unit, Student Affair Division, Dean's Offices, and Senate. The master database is hosted at CRPU. Microsoft Visual Basic 6.0® and Structured Query Language (SQL) were used to design a prototype of a client server distributed database for processing student records.

(Keywords: distributed database system, student results processing, relational database, client server system)

INTRODUCTION

Today's university environment has an increasing need for distributed database systems as the desire for easy, reliable, scalable, and accessible information is steadily increasing. A distributed database is a collection of databases that can be stored at different computer network sites and a distributed database management system (DDBMS) is the software that manages a distributed database while providing an access mechanism that makes the distribution transparent to the users [1].

From the above definition, the main unit of a distributed database system is a computer that is networked with other computers. Data in a distributed database is divided into fragments. Each fragment is a set of data items that are managed by a computer, or rather a local database management system in the computer.

It has been proven that, in universities and other higher institutions, information is highly essential for correct student record keeping. University environments have an increasing need for distributed database and client server applications. Distributed database systems provide an improvement in communications and data processing due to their data distribution throughout different network sites. Not only is data access faster, but a single-point of failure is less likely to occur, and it provides local control of data for users [1].

A distributed database system offers many advantages compared to centralized database systems and manual systems of record keeping. A distributed database system has been proposed for various reasons ranging from organization decentralization and economical processing to greater autonomy. Some of the advantages of distributed database system are location transparency, performance transparency, copy transparency, transaction transparency, fragmentation transparency, schema change transparency, and local DBMS transparency.

These seven transparencies are believed to incorporate the desired functions of distributed database systems [2]. Other goals of a successful distributed database system include:

- **Reflects organizational structure** — database fragments are located in the departments to which they are related.

- **Local autonomy** — a department can control their own data (as they are the ones most familiar with it.)

- **Improved availability** — a fault in one database system will only affect one fragment, instead of the entire database.

- **Improved performance** — 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.)

- **Economics** — it costs less to create a network of smaller computers with the power of a single large computer.

- **Modularity** — systems can be modified, added and removed from the distributed database.

Despite the above advantages of a distributed database, it is unfortunate that educational institutions in the developing world, such as the universities in Nigeria, still operate under the centralized systems and some still operate under the manual system of record keeping which is highly prone to errors. This has resulted in problems during course registration and in the computation of students' Grade Point Average (GPA) scores.

The present system consists of a relational database of student records which could be accessed by the Central Record Processing Unit (CRPU), Exams and Record Unit, Student Affair Division, Dean's offices, Senate and various academic departments in the university. Each of these is considered as a site in the distributed database. The master database hosted at the CRPU. The system is intelligent and capable of checking to detect authenticity of student results especially when it involves interfaculty or interdepartmental transfer of results.

Interfaculty or interdepartmental transfer of results can only be viewed but cannot be modified by other lecturers except the course examiner. Furthermore, the system also checks to detect falsification of data. For example, age, because at the point of entering/registration, a student might submit his date of birth while at the point of graduation, the student might change his date of birth in order to go for NYSC. Such information is contained in a secured distributed database. Security is of paramount importance when designing a distributed database for processing information. Policies and standards are needed to ensure appropriate security and privacy of student results.

Recent statistics have shown that distributed data systems turn out to perform better than their manual or centralized counterparts. In this present system, the distribution is transparent — users must be able to interact with the system as if it were one logical system. This applies to the system's performance, and methods of access among other things. Also, transactions are transparent — each transaction must maintain database integrity across multiple databases in the system.

The client server architecture is adopted in this paper because it is a platform for database application development [3]. A client is an application that requests information from a server.

Inadequacies the of Centralized System and Manual System of Record Keeping

- Whenever resources are centralized, there is an increased security risk. The integration of files also makes failure of any component a risk to bring the operation to a standstill.

- The computation of students' result is done manually. This could be cumbersome and error prone leading to delay in the release of results for a large number of student.

- Because of the manual processing of students' result, the process of retrieving missing results takes longer, especially when it has to do with courses from another department. Sometimes, supplementary and amended results are not updated immediately to reflect the true status of students' record, thereby resulting in inconsistent records.

- Much time is taken to process students' transcript and sometimes, there are discrepancies in the results.

In this paper, an attempt is made to design and implement a client server distributed database system for processing students' results thereby eliminating most of the setbacks associated with the centralized record keeping system and the manual system of student record keeping.

MATERIALS AND METHODS

Preliminary investigations about the existing systems (centralized system and manual record system) were carried out. A detailed study of the student information handbook of Gombe State University was also carried out [4]. The inadequacies of the existing systems were identified and a prototype of a distributed database system was proposed, designed and implemented using Microsoft Visual Basic 6.0®, Microsoft SQL Server and a web browser with Java Script compatible with Internet Explorer 5.0 or later versions made available at every client workstation.

DISTRIBUTED DATABASE ARCHITECTURE (DDBA)

The distributed database system consist of multiple Database Management Systems running on multiple servers (sites or nodes) connected by a network. The distributed databases are made up of homogeneous (similar) nodes. Each site or node may have some degree of autonomy - that is, it can accept transactions locally while still participating as a node in the distributed database system. Figure 1 shows each department with its own local applications and databases in the DDBA.

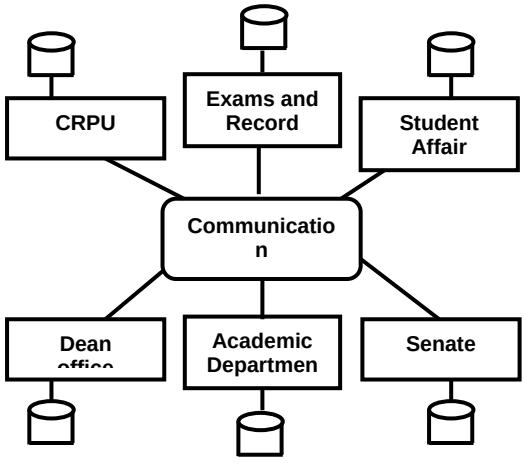

Figure 1: Distributed Database Architecture

From the above distributed database architecture, the system can offer more flexibility, higher performance and greater levels of independence over centralized systems.

COMPONENTS OF THE DISTRIBUTED DATABASE SYSTEM

The distributed database system must include (at least) the following components:

- Computer Workstations (sites or nodes) that form the network system. The distributed database system must be independent of the computer system hardware.

- Network Hardware and Software Components that reside in each workstation. The network components allow all sites to interact and exchange data. Network system independence is a desirable distributed database system attribute.

- Communication Media that carry the data from one workstation to another. The distributed database management system must be communications-media-independent; that is, it must be able to support several types of communication media.

- The Transaction Processor (TP), which is the software component found in each computer that requests data. The transaction processor receives and processes the application's data requests (remote or local). The TP is also known as the application processor (AP) or the transaction manager (TM).

- The Data Processor (DP), which is the software component residing on each computer that stores and retrieves data located at the site. The DP is known also known as the data manager (DM) .

DESIGN AND IMPLEMENTATION OF A CLIENT SERVER DISTRIBUTED DATABASE FOR STUDENT RESULT PROCESSING

The essence of automation is to enhance the efficiency and productivity of an existing. In view of this, the proposed system is fully automated with the aim of eliminating the deficiencies inherent in manual and centralized systems. The

system is designed in such a way that each academic department has its own database including all Dean's Offices, CRPU, Exams and Records, Student Affair Division, and Senate. The master database is expected to be hosted by the CRPU. The system is design with some security features such data unauthorized modification is not allow.

OVERALL SYSTEM STRUCTURE

The system consists of a relational database of student records which could be accessed by all the departments. The proposed system is a secured wed-based distributed student information system. The database contains CREATE TABLE statements for students' records including all courses, clearance procedures, registration procedures, add/drop course procedures, entering of scores procedures, and great point average (GPA) procedures.

Each table in the database file is design with a specific format for each field. Input is processed against the file to produce the necessary output. For security reasons, accessing mode for each file in the database fragments is strictly departmental-based (i.e., records in other departments can be viewed (read) but cannot be modified by other departments).

A relational database model is based on the concept that data is organized and stored in two dimensional tables called relations.

A relational database consists of a collection of tables, each of which is assigned a unique name and a row in a table represents a relationship among a set of values [4]. A table consists of a heading defining the table name, column names and a body containing rows of data.

Some of the ***relations*** and ***CREATE TABLES*** statements in the database are presented below:

STUDENT (StdMatNo, StdFirstName, StdLastNmame, StdAge, StdCity, StdState, StdZip, StdFaculty, StdDept, StdLevel,StdScore, StdGPA)

COURSE (CourseCode, CourseTitle, CreditUnit, Score)

STAFF (StaffID, StaffFirstName, StaffLastName, StaffAge, StaffCity, StaffState, StaffZip, StaffPhoneNumber)

DEPARTMENT (DeptName, DeptHead, DeptLoc)

Table 1: Course Table.

Course Code	StdMatNo	Course Title	Credit Unit	Score
COSC203	SCN001	Database1	3	20
COSC201	SCN002	C++	3	40

Table 2: Course Database Structure.

Field Name	Field Type	Field Size
CourseCode	Char	7
StdMatNo	Char	6
Course Title	Char	25
CreditUnit	Char	3
Score	Decimal	3,2

Form the above ***Course Table,*** the attribute ***StdMatNo,*** is a foreign key in the ***Course Table*** because it saves as a primary key in ***Student Table*** and as none key attribute in ***Course Table*****.**

SQL **CREATE TABLE** statement for the above course database structure is shown below:

```
CREATE TABLE Course
(CourseCode     CHAR(7)       NOT NULL,
StdMatNo        CHAR(6)       NOT NULL,
CourseTitle     CHAR(20)      NOT NULL,
CreditUnit      CHAR(1)       NOT NULL,
Score           DECIMAL(3,2)  NULL,
Primary Key(CourseCode))
```

Table 3: Student Table.

StdMatNo	StdFirstName	StdLastName	StdAge	StdCity	StdState	StdZip	StdFaculty	StdDept	StdLevel	StdScore	StdGPA
SCN001	John	Ann	23	Gombe	Gombe	234	Science	Computer	200	20	3.3
SCN002	Mary	Bnn	25	Benin	Gombe	237	Science	Computer	200	25	2.7

Table 4: Student Database Structure

Field Name	Field Type	Field Size
StdMatNo	Char	6
StdFirstName	Varchar	15
StdLastName	Varchar	15
StdAge	Integer	3
StdCity	Varchar	13
StdState	Char	13
StdZip	Char	5
StdFaculty	Char	25
StdDept	Char	15
StdLevel	Varchar	3
StdScore	Decimal	3,2
StdGPA	Decimal	3,2

SQL **CREATE TABLE** statement for the above student database structure is shown below:

```
CREATE TABLE Student
(StdMatNo       CHAR(6)       NOT NULL,
StdFirstName    VARCHAR(15)   NOT NULL,
StdLastName     VARCHAR(15)   NOT NULL,
StaAge          INTEGER(3)    NOT NULL,
StdCity         VARCHAR(13)   NOT NULL,
StdState        CHAR(13)      NOT NULL,
StdZip          CHAR(5)       NOT NULL,
StdFaculty      CHAR(25)      NOT NULL,
StdDept         CHAR(15)      NOT NULL,
StdLevel        VARCHAR(3)    NOT NULL,
StdScore        DECIMAL(3,2)  NULL,
StdGPA          DECIMAL(3,2)  NOT NULL,
Primary Key(StdMatNo))
```

CREATE TABLE is a statement in SQL used to create the initial tables in the database. Because SQL is an industry standard language, the **CREATE TABLE** statement including other Data Definition Language (DDL) and Data Manipulation Language (DML) commands were used to create restructure and manipulate the database [5].

- **Primary Key:** A column (or columns) in a table that makes the rows in the table distinguish from every other row in the same table. The role of the primary key is to uniquely identify each record in a table. By its very nature, the primary key cannot be empty. A primary key could consist of one or more columns (fields) i.e., though some fields may contain duplicate values, their combination (set) is unique throughout the entire table. A key that consist of two or more fields is called a composite key [4].

- **Foreign Key:** A foreign key is a column (or columns) in a table that draws its values from a primary in another table. A foreign key assists in ensuring the data integrity of a table [4].

VERIFICATION AND TESTING

The following tests were performed after coding and implementation with a view to bringing out the bottlenecks, errors and inefficiencies in the new system.

- **Unit Testing:** Every independent unit (register courses, show registered courses, update score, etc) in this system was tested separately for internal validation.

- **Integration Testing:** Using the relationship between the units, the interface each unit presents, was tested with another unit for its validity.

- **System Test:** Having been able to validate the interactions between the different units, the system was tested as a whole.

HARDWARE REQUIREMENTS

The following minimum hardware requirements have to be met for the software to operate properly:
Pentium III processor speed, 256 Megabytes of RAM (512 Megabytes recommended), 200 Megabytes free hard disk space, a 48x CD Re-writable drive for installing and back-up of the software and a printer for printing results and other reports including transcripts.

SOFTWARE REQUIREMENTS

Apart from the hardware requirements listed above, the following minimum requirements have to be installed for the software to be operated properly: A Microsoft SQL Server Database, a JSP/Server Container e.g. Apache Tomcat, which was used in the testing of the proposed system, a web browser with Java Script (for example, Internet Explorer 5.0 or later versions) must be available at every client workstation. Macromedia Flash 5.0 plug-in or later version must be available on the browser. However, it comes with recent versions of internet explorer (IE 5.5 and above) and window 98 or higher operating system.

FLOWCHARTS FOR THE SYSTEM

There are different types of tools that can be used to achieve the design of the present systems. The top down approach has been used to achieve the design [7]. Some of the flowcharts for the system are shown below:

CONCLUSION AND RECOMMENDATION

The advantages of the proposed system cannot be over emphasized. Recent statistics have shown that distributed database systems turn out to perform better that their manual/centralized counterparts.

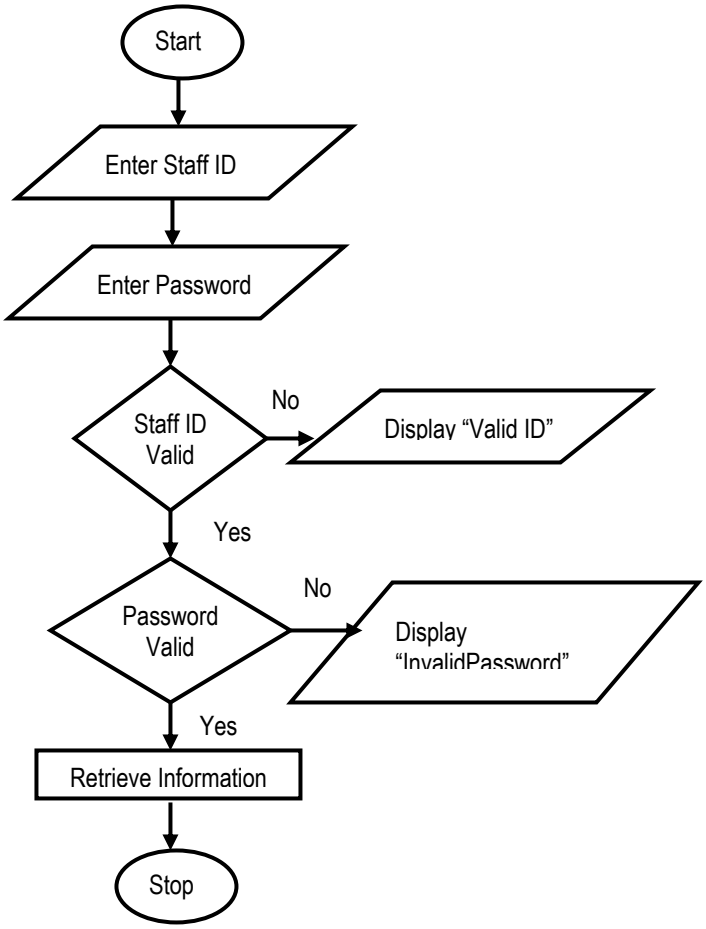

Figure 1: Login Flowchart for Staff.

The inadequacies of the manual/centralized systems in handling our students' results necessitated the use of a client server distributed database system in this paper.

Distributed database system provides an improvement on communication and data processing due to its data distribution throughout different network sites. Embracing this technology is very important for any organization that wants to be effective, efficient and innovation.

For a system as complex as this, adequate training has to be employed so that the goal of this paper is not defeated. It is our natural human nature not to accept what we do not understand. Users (members of staff) will not accept the new system if the change-over method employed does not cater for detail user training.

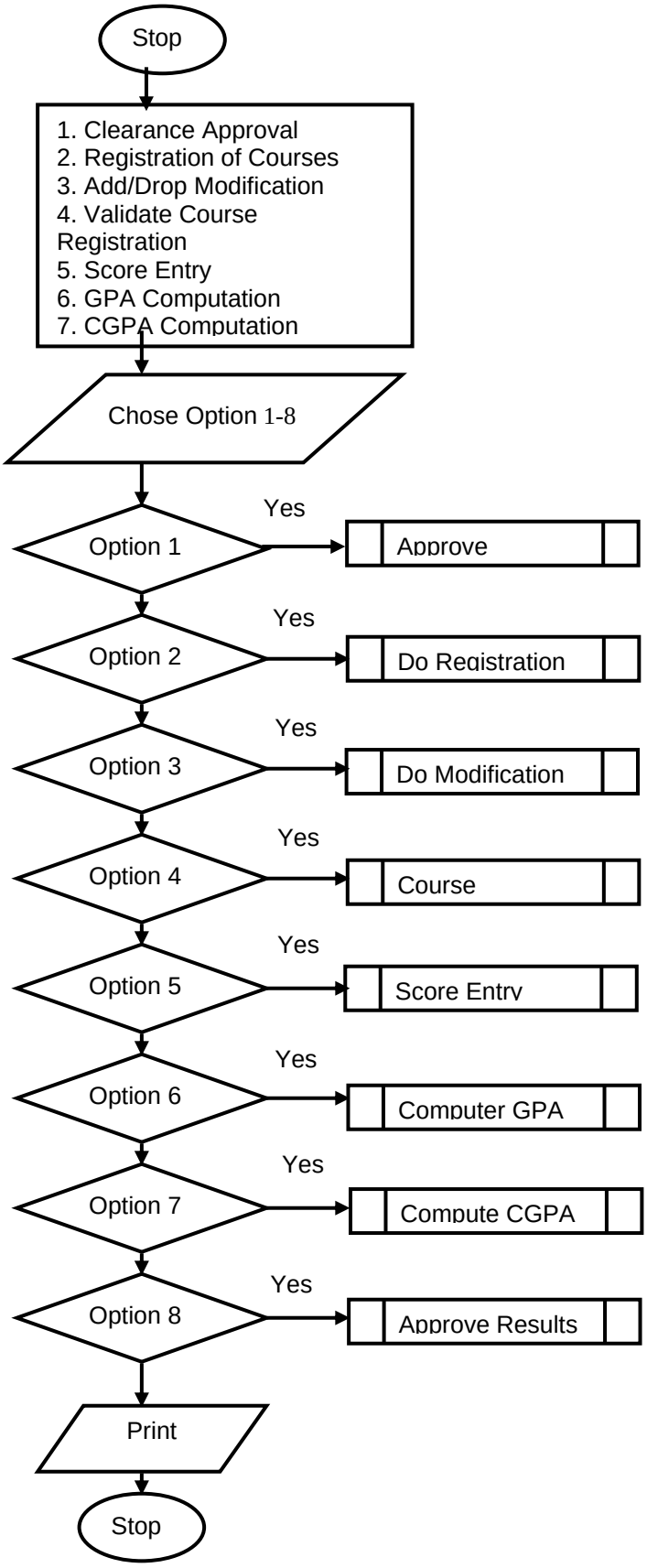

Figure 2: Flowchart for System Information Processing.

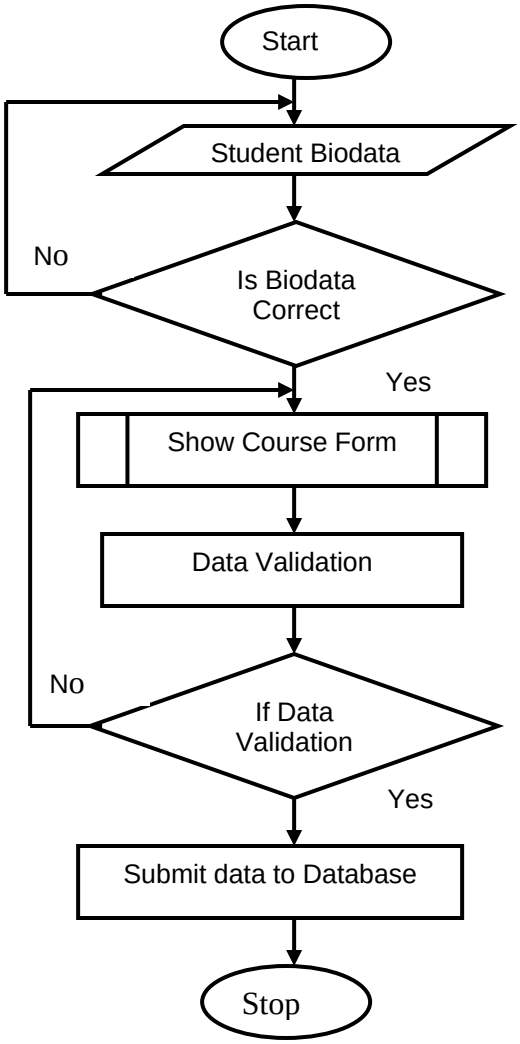

Figure 3: Flowchart for Student Records.

Parallel change over method is recommended where the new system is used simultaneously with the manual/centralized system and results/efficiency are compared.

Security is a predominant issue in software operations. It should be remembered that, no system is more secure that the users allow it. Therefore, a system as this that uses password authentication, users should realized that passwords should be very confidential as no matter the level of encryption employed in protecting passwords at the database level, if users reveals passwords intentionally or otherwise, the security of the system will be heavily compromised.

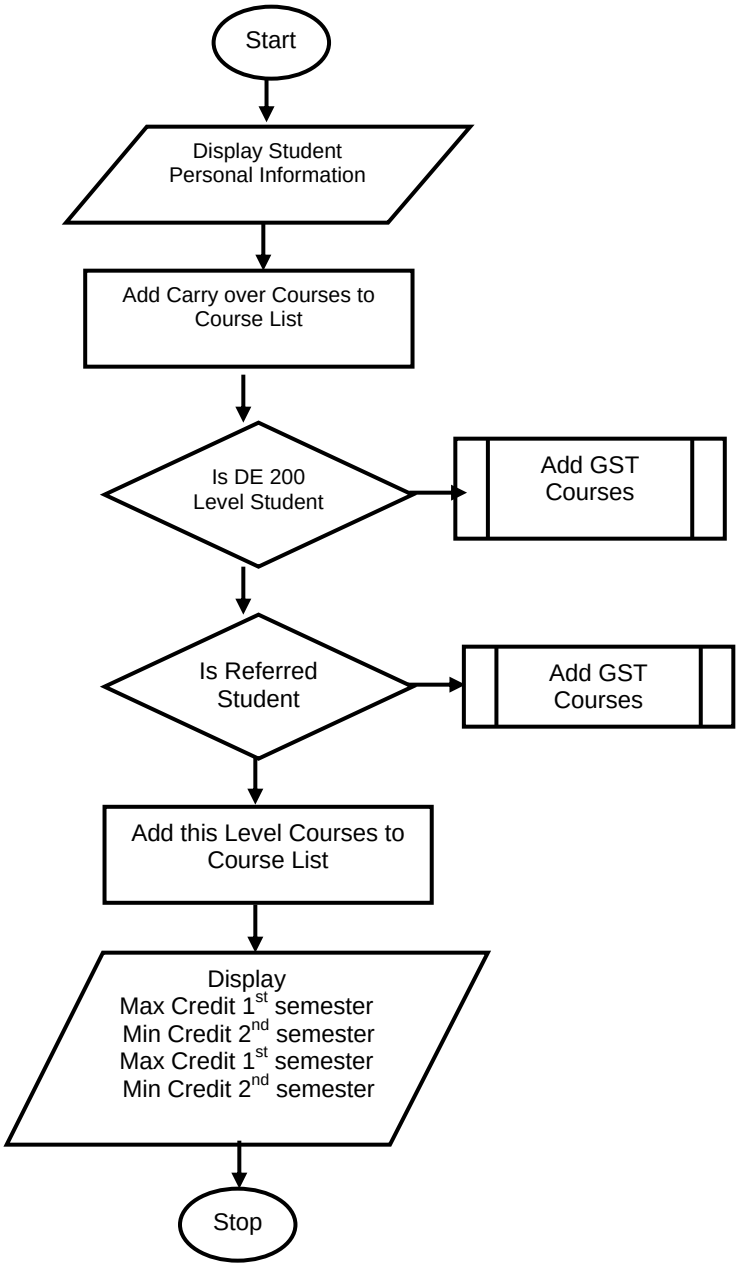

Figure 4: Course Form Flowchart.

Distributed database systems however, has yet a long way to go before it will yield the full flexibility and power of which it is theoretically capable. In this light, I recommend that further research work should be carried out to examine more security issues for distributed database.

REFERENCES

1. Mitchell, C. 2004. "Components of a Distributed Database." http://www.cs.nsu.edu/research/techdocs/troo5_caralyn_mitchell.pdf

2. Navathe, E. 2000. *Fundamental of Database Systems. Third Edition*. Teturo Sawada, Exclusive Publisher and Distributor.

3. Ramakrishnan, R. and Gehrke, J. 2002. *Database Management System. Second Edition*. McGraw Hill: New York, NY.

4. Gombe State University. 2007. "Handbook for Undergraduate Students". Faculty of Science. Gombe State University, Gimbe: Gombe State, Nigeria.

5. Silberschatz, A., Korth, H.F., and Sudarshan, S. 2006. *Database System Concepts. Fifth Edition.* McGraw Hill: New York, NY.

6. Carter, J. 2002. *Database Design and Programming with Access, SQL, Visual Basic, and ASP.Net.* McGraw Hill: New York, NY.

7. Stonebraker, J.M. 1996. "Object-Relational Database Management Systems." *The Next Great Wave*. Morgan Kaufmann: New York, NY.

ABOUT THE AUTHOR

Djam Xaveria Youh, is currently an Assistant Lecturer in the Mathematical Science Department, Gombe State University, Gombe State, Nigeira. She obtained her B.Sc. degree in Computer Science in 2004 from University of Benin (UNIBEN), Benin City, Nigeria and M.Sc. degree in Computer Science in 2007 from University of Benin (UNIBEN), Benin City, Nigeria. Her research interests are in databases, information systems, software engineering, and internet programming. She is an affiliate member of Computer Professionals in Nigeria (CPN).

SUGGESTED CITATION

Youh, D.X. 2010. "Design and Implementation of a Client Server Distributed Database for Student Results Processing". *Pacific Journal of Science and Technology*. 11(2):288-295.

Pacific Journal of Science and Technology