

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

An object database (also known as object-oriented database) is a database model in which information is represented in the form of objects as used in object-oriented programming. Object-oriented databases are designed and built according to the object-oriented paradigm in which everything is modeled as objects including the data (Hibatullah, 2016). This type of data model helps in tackling complex data structures, for instance multimedia content, in a more natural way and provides a seamless transition from design to conception. According to object-oriented database system manifesto (Atkinson et. al. 1989), an Object-Oriented Database Management System (OODBMS) must satisfy two criteria:

- i) It should be a Database Management System (DBMS)
- ii) It should be an object-oriented system

When database capabilities are combined with Object-Oriented (OO) programming language capabilities, the result is an Object Database Management System (ODBMS).

Today's trend in programming languages is to utilize objects, thereby making Object Oriented Database Management System (OODBMS) ideal for Object-Oriented programmers because they can develop the product, store them as objects, and can replicate or modify existing objects to make new objects within the OODBMS. Information today includes not only text data but video, audio, graphs, spatial data and photos which are considered complex data types.

In many cases non-experts confuse Object-based Databases (OBD), Object-Relational Databases (ORD) with Object-Oriented Databases (OODB) the three may sound similar but

they are different and must not be mixed. According to Ogunlere and Idowu (2015) the different is always there in their features and each have distinct implementation objective. Many data stored in data warehouses contain mixture of simple and complex data. A warehouse is a subject-oriented, integrated, non-volatile, time-variant collection of data in support of management's decisions. It is a collection of decision support technologies, aimed at enabling the knowledge worker, such as executive, manager, and analyst, to arrive at better and faster decisions. Data warehouses provide access to data for complex analysis, knowledge discovery, and decision-making. They support high performance demands on an organization's data and information. It provides an enormous amount of historical and static data from three tiers:

1. Relational databases
2. Multidimensional Online Analytical Processing (OLAP) applications
3. Client analysis tools

A data warehouse by itself does not create value, but value comes from the use of the data in the warehouse.

Advances in hardware and software capability have enabled these complex data to be captured and manipulated in a digital format. Almost any kind of complex data such as images, audio, video, geographic maps and three-dimensional graphics can be digitally stored. For instance an image can be represented as a two-dimensional array of pixels (picture element) where each pixel contains a numeric property representing its colour or shade of gray. Digital storage is usually cheaper and more reliable than traditional means such as paper, film or slides. In addition, digital storage allows easier retrieval and manipulation. Digital images can be manipulated in an image editor with operations such as cropping, texturing and colour funning.

The ability to store and manipulate complex data does not by itself drive the demand for object database technology. Rather the need to store large amounts of complex data and integrate complex data with simple data drives the demand for object database technology and its communication with object-oriented codes in various applications. Many modern and computerized organizations and business applications require large amounts of complex data. For instance JAMB, WAEC, NECO, INEC and even universities records whether online or off-line require large amount of complex data. Insurance claims, medical records, real estates and other identity based records can involve large amount of image data. The ability to simultaneously retrieve complex and simple data is becoming important in many business applications. For instance to review patient's condition, a physician may request x-ray along with vital statistics. Without integration two separate programs are required to display the data; an image editor to display the x-rays and a Database management system (DBMS) to retrieve the vital statistics. The ability to retrieve both image and vital statistics in a single query is a large improvement. Besides retrieving complex data, new operations also may be necessary. For instance a physician may want to compare the similarity of a patient's x-rays with x-rays that show abnormalities. In this work new design model will be presented for use in the development of systems that provides readily solution to the issues of complex and simple data integration in an organizational system.

1.2 Statement of the Problem

The availability of complex data and the need to integrate it with simple data in a single DBMS is the main problem calling for the study. The problem of mismatch between the data types in a Relational Database Management System (RDBMS) and the data types of modern programming languages makes software more complex and difficult to develop. This mismatch is more prominent in complex data. Object-oriented programming languages often

have richer and dynamic type systems than most Relational Database Management System (RDBMS) which often have fixed types. Data stores or warehouses need direct matches to enable it support speedy retrieval required in decision making data content systems. The problem of the present textual representation of data and file storage of complex data creates challenge of delayed fetch and reinterpretation of data before use in today's data warehouses. How can this be resolved? How can we design a system for storing complex data to allow communication between object-oriented DBMS and object-oriented programming data. Many research have been going on and some progress seem to have been made but there is no any database that have provided solution to these challenge as at the time of this research.

1.3 Aim and Objectives of the Study

The aim of the research is to design models for object-oriented database and class object communication in data storage system. The objectives of this research include to:

- i) design a formal model for object-oriented database systems
- ii) design an Object-Role Model for object-oriented database systems capable of handling complex data problems in information representation in data store or warehousing and easy data fetch and management.
- iii) implement the Object-Role Model designed using Conference Management System (CMS) as a Use Case and PHP Programming Language to simplify the integration of the system.
- iv) To evaluate the performance of the model in the conference Management system use case

1.4 Significance of the study

Many modern business applications require large amounts of complex data. So there is need to store large amounts of complex data and integrate the complex data with simple data in a

single application. There is equally a need to make databases easy to fit in the way modern programs are developed so that data can be inherited, extended, polymorphed and hiding (secured) from external programs. There have been attempts at achieving that but it seems that the lack of a solid theoretical base for the development have made it difficult for the development of fully object-oriented databases. Most attempts ended in relational database improvement, object-relational databases and document-oriented databases. The gap that full object-oriented database can fill still remains open due to the complex nature of modern objects. Images for instance has moved from uneditable files (.gif), to editable files (.jpg) and then now to programmable files (.svg) and serialization is changing the way objects and data can be manipulated. Many developers know the benefits of efficiency in programming that the combination of complex and simple data offer especially in today's networked computing environment but most current database still separate them and manipulate them separately or by using their properties. The major challenge in the development is the theoretical base in which the system will be developed. The scarcity of foundation for development of fully object-oriented database system is what the research intend to fill. The models developed in this project offers clear road map which can be followed in the process of building a fully object-oriented database. This proposes new ways of storing, manipulating and retrieving complex and simple data simultaneously and also using Object-Role Modeling to simplify the design and integration of such system. It also creates awareness of new expectations which will spur DBMS vendors to create new DBMS's that can integrate the formal model developed in this research in their products. The significance of data in modern life can be likened to importance of water or air to human life. The biggest concern remains that the nature of modern data are gradually increasing in images, videos, serialized files, encrypted documents and other forms of complex data. This is a shift from simple text which older

databases have been handling efficiently. In recent time the objects have been handled separately simply because they can not be integrated. The closet approach to integration have been in the object-relational databases but the rise of MongoDB a document-oriented database management system have simply proven that continued separation still holds. Modern database development researchers and those working in the field of databases management system development will find the work in this project as a sound base for the improvement on their research in an attempt to overcome the challenges of complex data handling via object-oriented database to object-oriented programming seamless communication. Developers will also benefit by using the concept in the work in solving the new changes which are bound to come with time so that their application should be compliant to future trends in object-oriented developments, especially in the area of databases.

1.5 Scope of the Study

The research develops models through system conceptualization that can handle object (complex) data in a database management environment. It is important to observe that the work is not on object-relational databases, nor in object databases rather it is on object-oriented database. The models will be implemented using available tools with an object-oriented programming language PHP. The research work does not cover the development of any commercial Object-Oriented Database Management System (OODBMS) or any language for its communication need. It covers design of a formal model and an Object-role model which will be the foundation of a real Object-Oriented database that meets with the object-oriented database specified in the work.

1.6 Limitation of the Study

Object-oriented database management targets complex applications which are often a subject of research than enterprenuer development; this hindered the sponsorship we would have

gotten from local enterprises. The challenge of implementation of the system using DBMS that have our specification is there, since enterprise databases are not often made up of complex data content and developing a new DBMS is out of the scope of the research. This made us to extend functionalities of available DBMS to accommodate the new specifications made in the database schema. There are no pure Object-oriented DBMS for use, most are still research work which we believe that our research will complement and also improve.

1.7 Definition of terms:

Data: are the values of subjects with respect to qualitative or quantitative variables, measured, collected, analyzed and reported and can be visualized using graphs, images or other analysis tools. Data as a general concept refers to the fact or pieces of information generated as activities are carried out in organizations.

Raw Data: is a collection of numbers or characters before it has been "cleaned" and corrected by researchers. Raw data needs to be corrected to remove outliers or obvious instrument or data entry errors.

System Data: This is data represented or coded in some form suitable for better usage or processing by the computer or any electronic system.

Database: Database is a collection of persistent data that can be shared and interrelated, it can also be generally stored and accessed electronically from a computer system. Where databases are more complex they are often developed using formal design and modelling techniques.

Database Management System: A database management system (DBMS) is a software that interacts with end users, applications, the database itself to capture and analyze the data and provides facilities to administer the database. DBMS are grouped based on the database model that they support.

Data Warehouse: a central repository for summarized and integrated data from operational database and external data sources. A data warehouse is a system used for reporting and data analysis and is considered a core component of business intelligence. Data warehouse is a central repositories of integrated data from one or more disparate sources. They store historic, current, simple and complex data in one single place. When a different data model is developed, the aggregation system or schema of the data warehouse is expected to change.

Simple Data: Simple data are often referred to as primitive data types, they are mostly in form of the normal text data, number and the data form that need no further definition. It also refers to an arrangement of data in a data base or file in which each grouping of data, such as a record, is in a form that does not require further definition.

Complex Data: Complex data is often referred to as composite data type. They are any Data type which can be constructed in a program using the programming language primitive (simple) data types and other composite types. It is sometimes called a structure or Aggregate data type, even a class or struct in some programming languages like Java or C. The act of constructing a complex type is known as composition.

Object: An object can be a variable (data), a data structure, a function or a method. It is a value in memory referenced by an identifier. It is an instance of a class which is a combination of variables, functions and data structures. In relational database management, an object can be a table or column or an association between data and a database entity.

Object database: An object database is a database management system that represent data using the basic capabilities for an object: identity, properties, and attributes inherent in object-oriented paradigm but does not integrate methods with the attributes nor implements data

extension (inheritance) and polymorphism. This is often confused with object-oriented databases.

Object-Oriented Database: An object-oriented database is a database management system that represents data using both the basic capabilities for an object: identity, properties, and attributes that are inherent in object-oriented paradigm and also integrate methods with the attributes and also implements data extension (inheritance) and polymorphism. Data extensions easily allow developers to build complex data defined as data classes which is not possible with object databases.

Document-oriented Databases

A document-oriented database is a database management system designed for storing, retrieving and managing document-oriented information (semi-structured data). They are often referred to as NoSQL database management Systems. One of the most popular of NoSQL include key-value store (KVS) and MongoDB. The difference between KVS and MongoDB lies in the way the data is processed; in a key-value store, the data is considered to be inherently opaque to the database, whereas MongoDB relies on internal structure in the *document* in order to extract metadata that the database engine uses for further optimization.

Object-relational Database:

Object-relational database is a hybrid of object database management system and relational database management system that represent data in a table-oriented format using the basic capabilities for an object: identity, properties, and attributes inherent in object-oriented paradigm.

Programming Language is a vocabulary and set of grammatical rules for instructing a computer or computing device to perform specific tasks.

Object-Oriented programming is a programming language model organized around objects rather than “actions” and data rather than logic.

Object Role Modeling (ORM) is a powerful method for designing and querying database models at the conceptual level, where the application is described in terms easily understood by non-technical users.

Integration is the act of bringing together smaller components into a single system that functions as one.

Conceptualization is the process of development and clarification of concepts. In other words, clarifying one’s concepts with words and examples and arriving at precise verbal definitions.

Data integration is the combination of technical and business processes used to combine data from disparate sources into meaningful and valuable information.

CHAPTER TWO

LITERATURE REVIEW

2.1 Theoretical Review

In this section, the theoretical concepts that are related to the research is reviewed. This provides a clear understanding of both the main terminologies and the theoretical concept which serve as the foundation upon which the research is based. These areas include data, databases and their past and present mode of design and the database management system that drives their operations in academics and in the industry.

2.1.1 Data

Data can be defined as values of subjects with respect to qualitative or quantitative variables, measured, collected, analyzed and reported and can be visualized using graphs, images or other analysis tools. However, Checkland and Helwell (1998) in their work defined data as Numbers, characters, images, or other method of recording, in a form which can be assessed by a human or input into a computer, stored and processed there, or transmitted on some digital channel. Computers nearly always represent data in binary. Data on its own has no meaning, only when interpreted by some kind of data processing system that it takes on meaning and becomes information. In computing, data is information that has been translated into a form that is more convenient to move or process. Relative to today's computers and transmission media, data is information converted into binary digital form. A common way of representing or displaying a set of correlated data is through table type structures comprised of rows and columns. In such structures, the columns of the table generally signify attributes or characteristics or features and the rows (tuple) signify a set of co-related features belonging to one single item (Hibatullah, 2016). This table type structures are often referred to as Databases.

2.1.1.1 Simple Data: Simple data types are mostly in form of the normal text data, number and the data often referred to in programming languages as primitive data types. Most relational database management systems (RDBMS) support only few data types. Common data types supported in Structure Query Language (SQL) 2 include whole numbers (integer), real numbers, fixed precision numbers (currency), dates, times and text. These data types are “*simple*” data. They are sufficient for many business applications or at least significant parts of many applications. Many business databases contain fields for names, prices, addresses and transaction dates that readily conform to the primitive data types (Garvey and Jackson, 2010).

2.1.1.2 Complex Data: Complex data types are in form of video, spatial types or the data types referred to in programming languages as “struct”, user defined or class types. It can also be the combination of simple data types with the core complex data types. There are several modern applications that require blobs, images, video clips, maps and other spatial data formats which are “*complex*” compared to the textual data. Presently many applications store these complex data as files instead of keeping them as part of the database. Some developers argue that it is alright keeping them as files simply because the present databases do not make full provision for them to be store as part of the databases. This group of developers quickly forgot that text and dates where formally stored in files and PC clock marks before the advent of relational databases and that some developers in those days might have argued for allowing things to remain in the classical file system. Today all users and developers appreciate the relational database compared to the file system (Garvey and Jackson, 2010).

2.1.2 Database

A database is a system intended to organize, store, and retrieve large amounts of data easily. It consists of an organized collection of data for one or more uses, typically in digital form. One way of classifying databases involves the type of their contents, for example: bibliographic,

document-text, statistical (Aljanaby et al., 2005). Advances in data format and application development have called for development of various databases that suit varying development conditions and real world application needs. Some of these databases become more popular than others or even look similar to others but they are unique in many respects in their development and target usage. Some of these include: relational databases, document-oriented databases, object databases, object-relational databases and object-oriented databases. The databases have their benefits based on the need and on the area of application. Small organization and large offline companies may prefer relational databases because of its ease of use and ability to handle large data offline. In contrast online businesses would prefer document oriented databases due largely to their scalability and robustness. Geospatial companies may prefer object-relational databases due to its emphasis on objects and query of data stored with the objects. Users who want to integrate system operations to data need object abstraction and operation bundling which has made research into field of object-oriented databases an interesting and relevant one to database developers, software experts and the academic community.

2.1.2.1 Relational Databases

In 1970, Edgar F. Codd published a paper “A Relational Model of Data for Large Shared Data Banks” (Codd, 1970), where he proposed a relational database model. This model was aimed to solve problems that appeared with the evolution of technology, applications and the amount of data they were dealing with. New innovative concepts were proposed like:

- i) Concept of table with structured data by type (columns) and tuples or registers (rows).
- ii) Table relationships as primary key - foreign key.
- iii) Formal foundation based on set theory with algebraic and calculus-based techniques for querying data.

With all the new possibilities relational databases offered, applications with new domains, demands and requirements appeared. This led to the development of Structured Query Language (SQL) a declarative language used in manipulating the relational databases. These were very much adequate for its time due to the fact that the languages that were driving them, some of which include C, Pascal, Basic, Perl and many other programming languages used then, were also declarative programming languages. Languages like C++, PHP and Java at its earlier years also joined the bandwagon of declarative programming languages. There are serious changes now and the classical declarative languages are improved to be highly object-oriented and functional programming languages. In addition, interacting with SQL from modern Java is not straightforward neither from the developer's point of view (approximately 30% of the code and effort is used in conversion (Henderson-Sellers and Edwards, 1994) nor from the machine's (SQL cannot be interpreted by the compiler). These challenges have continued to widen as the Java language progresses from Java 5 to Java 11.

2.1.2.2 Document-oriented Databases

A document-oriented database is a database management system designed for storing, retrieving and managing semi-structured data. They store data as binary JavaScript Object Notation - JSON (BJSON) formatted documents. Unlike the relational databases which manages structured data in tables, document-oriented databases handle semi structured documents usually collected in streams over the internet without using structured query (Jaspreet et al., 2013). They are often referred to as No Structured Query Language (NoSQL) database management Systems. One of the most popular of document-oriented databases is MongoDB. MongoDB relies on internal structure in the *document* in order to extract metadata that the database engine uses for further optimization (Adam, 2015). It is believed that NoSQL databases are faster than relational databases because they don't typically follow

a predefined schema or enforce data consistency as strictly as relational databases. In addition, JSON is easily readable by both human and machine and use minimal storage space.

MongoDB is an open-source, document-oriented, NoSQL database program. Developers involved with the traditional, relational databases for long, the idea of a document-oriented, NoSQL database might indeed sound peculiar. MongoDB was developed by MongoDB Inc, and it is scalable and flexible. MongoDB stores data in JSON-like documents that can vary in structure. A document in a NoSQL database corresponds to a row in an SQL database. A group of documents together is known as a collection, which is roughly synonymous with a table in a relational database (Adam, 2015).

Apart from being a NoSQL database, MongoDB has a few qualities of its own which have been listed below:

- i) it's easy to install and set up
- ii) it uses a BSON (a JSON-like format) to store data
- iii) it's easy to map the document objects to a developers application code
- iv) it claims to be highly scalable and available, and includes support for out-of-the-box replication
- v) it supports MapReduce operations for condensing a large volume of data into useful aggregated results
- vi) it's free and open source

2.1.2.3 Object Databases

An object database is often confused with object-oriented database but there is a major difference in the two databases though many tend to confuse the two databases. An Object database is a database model in which data is represented using the basic capabilities for an object: identity, properties, and attributes inherent in object-oriented paradigm but does not

integrate methods with the attributes nor implements data extension (inheritance) and polymorphism. This is often confused with object-oriented databases which uses similar model but its data representation integrates the operational part. An object database stores complex data and relationships between data directly, without mapping to relational rows and columns, and this makes them suitable for applications dealing with very complex data (Gupta, 2009).

Objects have a many-to-many relationship and are accessed by the use of pointers. Pointers are linked to objects to establish relationships.

Object database management system (ODBMS) is the database management system used by object databases. As the usage of web-based technology increases with the implementation of Intranets and extranets, companies have a vested interest in ODBMS to display their data. Using an ODBMS that has been specifically designed to store data as objects gives an advantage to those companies that are geared towards multimedia presentation or organizations that utilize computer-aided design (CAD) (O'Brien and Marakas, 2009).

Some object databases are designed to work well with object-oriented programming languages such as Ruby, Python, Perl, Java, C#, Visual Basic .NET, C++, Objective-C and Smalltalk; others have their own programming languages.

Object database management systems grew out of research during the early to mid-1970s into having intrinsic database management support for graph-structured objects. The term "object database system" first appeared around 1985 (Maier et al., 1985). Notable research projects included Encore-Ob/Server (Brown University), EXODUS (University of Wisconsin–Madison), IRIS (Hewlett-Packard), ODE (Bell Labs), ORION (Microelectronics and

Computer Technology Corporation or MCC), Vodak (GMD-IPSI), and Zeitgeist (Texas Instruments). The ORION project had more published papers than any of the other efforts. Won Kim of MCC compiled the best of those papers in a book published by The MIT Press (Kim, 1990).

Early commercial products included Gemstone (Servio Logic, name changed to GemStone Systems), Gbase (Graphael), and Vbase (Ontologic). The early to mid-1990s saw additional commercial products enter the market. These included ITASCA (Itasca Systems), Jasmine (Fujitsu, marketed by Computer Associates), Matisse (Matisse Software), Objectivity/DB (Objectivity, Inc.), ObjectStore (Progress Software, acquired from eXcelon which was originally Object Design), ONTOS (Ontos, Inc., name changed from Ontologic), O₂ (Bancilhon et al., 1992) (O₂ Technology, merged with several companies, acquired by Informix, which was in turn acquired by IBM), POET (now FastObjects from Versant which acquired Poet Software), Versant Object Database (Versant Corporation), VOSS (Logic Arts) and JADE (Jade Software Corporation). Some of these products remain on the market and have been joined by new open source and commercial products such as IntersystemCACHÉ.

Object database management systems added the concept of persistence to object programming languages. The early commercial products were integrated with various languages: GemStone (Smalltalk), Gbase (LISP), Vbase (COP) and VOSS (Virtual Object Storage System for Smalltalk). For much of the 1990s, C++ dominated the commercial object database management market. Vendors added Java in the late 1990s and more recently, C#.

Starting in 2004, object databases have seen a second growth period when open source object databases emerged that were widely affordable and easy to use, because they are entirely

written in OOP languages like Smalltalk, Java or C#, such as db4o (db4objects), DTS/S1 from Obsidian Dynamics and Perst (McObject), available under dual open source and commercial licensing (Gupta,2009).

2.1.2.4 Adoption of Object Databases

Object databases based on persistent programming acquired a niche in application areas such as engineering and spatial databases, telecommunications, and scientific areas such as high energy physics and molecular biology. They have made little impact on mainstream commercial data processing, though there is some usage in specialized areas of financial services (Gupta, 2009). It is also worth noting that object databases held the record for the World's largest database (being the first to hold over 1000 terabytes at Stanford Linear Accelerator Center) and the highest ingest rate ever recorded for a commercial database at over one Terabyte per hour (Garvey and Jackson, 2010).

Another group of object databases focuses on embedded use in devices, packaged software, and real-time systems.

2.1.2.5 Technical Features of Object Databases

Most object databases also offer some kind of query language, allowing objects to be found by a more declarative programming approach. It is in the area of object query languages, and the integration of the query and navigational interfaces, that the biggest differences between products are found. An attempt at standardization was made by the ODMG with the Object Query Language, (OQL).

Access to data can be faster because joins are often not needed (as in a tabular implementation of a relational database). This is because an object can be retrieved directly without a search, by following pointers. (It could, however, be argued that "joining" is a higher-level abstraction of pointer-following.)

Another area of variation between products is in the way that the schema of a database is defined. A general characteristic, however, is that the programming language and the database schema use the same type definitions.

Multimedia applications are facilitated because the class methods associated with the data is responsible for its correct interpretation.

Many object databases, for example VOSS, offer support for versioning. An object can be viewed as the set of all its versions. Also, object versions can be treated as objects in their own right. Some object databases also provide systematic support for triggers and constraints which are the basis of active databases.

The efficiency of such a database is also greatly improved in areas which demand massive amounts of data about one item. For example, a banking institution could get the user's account information and provide them efficiently with extensive information such as transactions, account information entries etc. The Big O Notation for such a database paradigm drops from $O(n)$ towards $O(1)$, greatly increasing efficiency in these specific cases (Gupta, 2009).

2.1.2.6 Object-Relational Databases

Object-relational database is a hybrid of object database management system and relational database management system. The object-relational database model represent data in a table format using the basic capabilities for an object: identity, properties, and attributes inherent in object database management system format. The problems and inconvenients that integrating OOPLs and SQL produced, lead to a new group of research in databases which were aimed at improving the integration between objects and SQL. The result of this research gave rise to what we known as Object-Relational Databases (ORDB) (Eric et al.,2003).

There are needs that the ORDB are expected to provide solution for which includes

- i) Provide support for rich object structures and rules

Rich object structures are complex data elements such as arrays, enumeration, sets and maps but also refer to text data and spatial data. On the other side, developer should be able to create a set of rules on data elements, records and collections, e.g. constraints for referential integrity.

- ii) Must subsume second generation DBMSs

The term second generation DBMSs refers to classic relational databases. This tenet refers to the fact that second generation DBMSs made a major contribution in two main aspects: non-procedural access and data independence, hence those should be maintained in ORDBs.

- iii) Must be open to other subsystems

Prospect to new functionalities: like decision support tools, access from many programming languages, interfaces to business graphic packages, ability to run application from different machine from the database and distribution of databases (Klettke and Meyer, 2000)

The challenges lead to the need for new database systems that can fit the needs and integration of OOPs, but keeping the same base as it was in relational model. In other words, there was a need of new tools to "hide" all the complexities this integration lead to.

The need call for several research in the area of Object-oriented databases but till now non seem to be widely accepted and deployed by the enterprise world who are the largest consumers of database products. Some of the attempts that are worthy of mention include: Db4o, ObjectDB and Objectivity just to mention a few and many other research that are still ongoing.

There is a need to point out certain gray issues here, the similarity and differences between Object-Relational Databases which was an attempt to marry object-orientation with relational database features and Object-oriented Databases which is aimed at a total shift from relational data modeling to entirely object-oriented data model. Many database experts often confuse an Object-relational database research with Object-oriented database research may be due to the fact that the two are all ‘Object’ inclined. The similarity however is that both Object-relational database and Object-oriented database research are aiming to solve similar problems identified in Relational databases.

Object-relational database research is ahead of Object-oriented database research probably because major DB distributors and enterprise users already had a fully functional relational database (second generation), and adding new object support over it was easier than building a new database from scratch.

In db4o, objects are stored directly as seen by the program, this means that the code (classes) is the database schema, there's no need for mappings. This means that elements like object identity, associations, inheritance and complex objects are automatically stored without need of further configuration, this is known as orthogonal or transparent persistence. In other to retrieve objects from the database Db4o use Object Identifier (OID) or Query. In OID each object has associated an unique OID, same as RID on relational databases, and it is possible to directly retrieve an object by its OID. In query retrieval Db4o attempts to query attributes of its class, using Query By Example (QBE) and recently developing its own Native Queries (NQ) (Han, et al.,2003).

2.2 Review of Related Works

According to Hibatullah, (2016) an object-oriented database is a database management system that represent data using both the basic capabilities for an object: identity, properties, and attributes that is inherent in object-oriented paradigm and also integrate methods with the attributes and also implements data extension (inheritance) and polymorphism. Data extension easily allow developers to build complex data defined as data classes which is not possible with object databases.

On the early 1990, Radding (1995) posited that since Object Oriented Programming Languages (OOPL) began to develop with new functionalities like: user-defined classes, methods, encapsulation, polymorphism and inheritance there is a need to also point databases . Those OOPL have the ability to map very complex conceptual models into classes and relationships between them, the main problem comes when all this objects in memory have to be made persistent in the relational model (also known as impedance mismatch).

The main objective of Object-Oriented Database Management Systems, commonly known as OODBMS, is to provide consistent, data independent, secure, controlled and extensible data management services to support the object-oriented model. They were created to handle big and complex data that relational databases could not.

There are important characteristics involved with object-oriented databases. The most important characteristic is the joining of object-oriented programming with database technology, which provides an integrated application development system. Object-oriented programming results in four main characteristics: inheritances, data encapsulation, object identity, and polymorphism. Inheritance allows one to develop solutions to complex problems incrementally by defining new objects in terms of previously defined objects (ODBMS, 2013).

When database capabilities are combined with object-oriented (OO) programming language capabilities, the result is an object database management system (ODBMS).

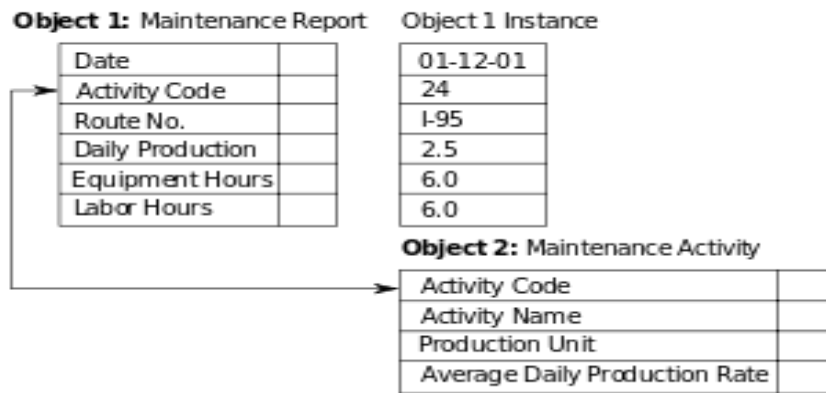


Fig2.1: A simple Object-oriented Database Model (Data Integration Glossary , 2001)

Today's trend in programming languages is to utilize objects, thereby making OODBMS ideal for OO programmers because they can develop the product, store them as objects, and can replicate or modify existing objects to make new objects within the OODBMS (ODBMS, 2013). Information today includes not only data but video, audio, graphs, and photos which are considered complex data types. Relational DBMS aren't natively capable of supporting these complex data types. By being integrated with the programming language, the programmer can maintain consistency within one environment because both the OODBMS and the programming language will use the same model of representation. Relational DBMS projects using complex data types would have to be divided into two separate tasks: the database model and the application.

Data encapsulation or simply encapsulation allows the hiding of the internal state of the objects. Encapsulated objects are those objects that can only be assessed by their methods

instead of their internal states. There are three types of encapsulated objects users and developers should recognize. The first is full encapsulation, in which all the operations on objects are done through message sending and method execution. The second is write encapsulation, which is where the internal state of the object is visible only for reading operations. The third is partial encapsulation, which involves allowing direct access for reading and writing for only a part of the internal state.

Object identity allows objects of the database to be independent of each other. Polymorphism and dynamic binding allow one to define operations for one object and then to share the specification of the operation with other objects. This allows users and/or programmers to compose objects to provide solutions without having to write code that is specific to each object.

The language important to OODBMS is Data Definition and Manipulation Language (DDML). The use of this language allows persistent data to be created, updated, deleted, or retrieved. An OODBMS needs a computational versus a relational language because it can be used to avoid impedance mismatch. DDML allows users to define a database, including creating, altering, and dropping tables and establishing constraints. DDMLs are used to maintain and query a database, including updating, inserting, modifying, and querying data (ODBMS,2013).

The OODBMS has many advantages and benefits. First, object-oriented is a more natural way of thinking. Second, the defined operations of these types of systems are not dependent on the particular database application running at a given moment. Third, the data types of object-oriented databases can be extended to support complex data such as images, digital and

audio/video, along with other multi-media operations (Hibatullah, 2016). Different benefits of OODBMS are its reusability, stability, and reliability. Another benefit of OODBMS is that relationships are represented explicitly, often supporting both navigational and associative access to information. This translates to improvement in data access performance versus the relational model.

Another important benefit is that users are allowed to define their own methods of access to data and how it will be represented or manipulated. The most significant benefit of the OODBMS is that these databases have extended into areas not known by the Relational Database Management System (RDBMS). Medicine, multimedia, and high-energy physics are just a few of the new industries relying on object-oriented databases.

As with the relational database method, object-oriented databases also have disadvantages or limitations. One disadvantage of OODBMS is that it lacks a common data model. There is also no current standard, since it is still considered to be in the development stages.

The main benefit of creating a database with objects as data is speed. OODBMS are faster than relational DBMS because data isn't stored in relational rows and columns but as objects (Radding, 1995). Objects have a many to many relationship and are accessed by the use of pointers. Pointers are linked to objects to establish relationships. Another benefit of OODBMS is that it can be programmed with small procedural differences without affecting the entire system (Burleson, 1994). This is most helpful for those organizations that have data relationships that aren't entirely clear or need to change these relations to satisfy the new business requirements.

Benchmarks between OODBMSs and RDBMSs have shown that an OODBMS can be clearly superior for certain kinds of tasks. The main reason for this is that many operations are performed using navigational rather than declarative interfaces, and navigational access to data is usually implemented very efficiently by following pointers.

Compared to relational databases another major advantage of OODBMSs is that they do not need any object relational mapping layer and object marshaling to map the application object model to the database object model. In RDBMS, this mapping is also source of the impedance mismatch, which does not occur when using OODBMS. Avoiding this layer also improves performance and saves effort for implementation and maintenance.

Critics of navigational database-based technologies like ODBMS suggest that pointer-based techniques are optimized for very specific "search routes" or viewpoints; for general-purpose queries on the same information, pointer-based techniques will tend to be slower and more difficult to formulate than relational. Thus, navigation appears to simplify specific known uses at the expense of general, unforeseen, and varied future uses. However, with suitable language support, direct object references may be maintained in addition to normalized, indexed aggregations, allowing both kinds of access; furthermore, a persistent language may index aggregations on whatever its content elements return from a call to some arbitrary object access method, rather than only on attribute value, which allows a query to 'drill down' into complex data structures.

Other things that work against ODBMS seem to be the lack of interoperability with a great number of tools/features that are taken for granted in the SQL world, including but not limited to industry standard connectivity, reporting tools, OLAP tools, and backup and recovery

standards (Garvey and Jackson,2010). Additionally, object databases lack a formal mathematical foundation, unlike the relational model, and this in turn leads to weaknesses in their query support. However, this objection is offset by the fact that some ODBMSs fully support SQL in addition to navigational access, e.g. Objectivity/SQL++, Matisse, and InterSystems CACHE. Effective use may require compromises to keep both paradigms in sync.

In fact there is an intrinsic tension between the notion of encapsulation, which hides data and makes it available only through a published set of interface methods, and the assumption underlying much database technology, which is that data should be accessible to queries based on data content rather than predefined access paths. Database-centric thinking tends to view the world through a declarative and attribute-driven viewpoint, while OOP tends to view the world through a behavioral viewpoint, maintaining entity-identity independently of changing attributes. This is one of the many impedance mismatch issues surrounding OOP and databases.

2.2.1 Object Database Technology

As advances in computer-related technology improve, increasingly larger files are able to be created, transmitted, and stored electronically. It thus becomes more apparent that object-oriented technology and object-oriented databases in particular, are needed to warehouse the files or “objects.” About 88 percent of organizations use relational databases, yet about 55 percent plan to acquire object-oriented databases at some point in the future (Betts, 1997). Management should therefore plan carefully and understand the benefits of object-oriented technology.

According to Martin and Leben (1995) Most widely used database software uses some form of client/server technology with relational databases. The relational database paradigm, whether centralized or distributed, maintains that only data, and not procedures, should be stored. A major objective of conventional database technology is to make the data completely independent from the procedures. A recently developed form of database, called an object-oriented database, is becoming more frequently used. In contrast to the relational database model, an object-oriented database stores objects, which consist of data as well as procedures (methods) that are used to perform operations on data.

In general terms, an object has a group of characteristics that can be stored as data, which can then be processed as information in a number of ways (Hernandez, 1997). Large-scale database management systems (DBMS) are increasingly in demand to support groups of users in collaborative work environments (Huh et al, 1999). Since database models change along with the reality that is captured in them, their dependent models and views should also evolve.

Object-oriented approaches to programming were introduced as early as 1966 with the emergence of the simulation language Simula67 (Schach, 1996). At the time Simula67 was introduced, the technology was considered too radical for practical use. It basically lay dormant until the early 1980's when it was essentially resurrected within the context of modularity.

2.2.2 List of Object-Oriented Technology

Object-oriented technologies in use today include object-oriented programming languages (e.g., C++ and Smalltalk), object-oriented database systems,

Object-oriented user interfaces (e.g., Macintosh and Microsoft window systems, Frame and Interleaf desktop publishing systems), etc. An object-oriented technology is a technology that makes available to the users facilities that are based on “object-oriented concepts”. To define “object-oriented concepts”, we must first understand what an “object” is.

The term “object” means a combination of “data” and “program” that represent some real-world entity. For example, consider an employee named Tom; Tom is 25 years old, and his salary is N25,000. Then Tom may be represented in a computer program as an object. The “data” part of this object would be (name: Tom, age: 25, salary: N25,000). The “program” part of the object may be a collection of programs (hire, retrieve the data, change age, change salary, fire). The data part consists of data of any type. For the Tom object, String is used for the name, integer for age, and monetary for Salary; but in general, even any user-defined type, such as employee may be used. In the Tom object, the name, age, and salary are called attributes of the object.

Often, an object is said to “encapsulate” data and program. This means that the users cannot see the inside of the object “capsule”, but can use the object by calling the program part of the object. This is not much different from procedure calls in conventional programming; the users call a procedure by supplying values for input parameters and receive results in output parameters.

The term object-oriented roughly means a combination of object encapsulation and inheritance. The term “inheritance” is sometimes called “reuse”. Inheritance means roughly that a new object may be created by extending an existing object. Now let us understand the term “inheritance” more precisely. An object has a data part and a program part. All objects

that have the same attributes for the data part and same program part are collectively called a class (or type). The classes are arranged such that some class may inherit the attributes and program part from some other classes.

Tom, Dick, and Harry are each an Employee object. The data part of each of these objects consists of the attributes Name, Age and Salary. Each of these Employee objects has the same program part (hire, retrieve the data, change age, change salary, fire). Each program in the program part is called a “method”. The term “class” refers to the collection of all objects that have the same attributes and methods. In our example, the Tom, Dick, and Harry objects belong to the class Employee, since they all have the same attributes and methods. This class may be used as the type of an attribute of any object. At this time, there is only one class in the systems, namely, the class Employee; and three objects that belong to the class, namely, Tom, Dick, and Harry objects.

Suppose that a user wishes to create two sales employees, John and Paul. But sales employees have an additional attribute, namely, Commission. The sales employees cannot belong to the class Employee. However, the user can create a new class, sales-Employee, such that all attributes and methods associated with the class Employee may be reused and the attribute Commission may be added to Sales-Employee. The user does this by declaring the class Sales-Employee to be a subclass of the class Employee. The user can now proceed to create the two sales employees as objects belonging to the class Sales Employee. The users can create new classes as subclasses of existing classes. In general, a class may inherit from one or more existing classes, and the inheritance structure of classes becomes a directed acyclic graph (DAG); but for simplicity, the inheritance structure is called an inheritance hierarchy or class hierarchy.

The power of object –oriented concepts is delivered when encapsulation and inheritance work together.

- Since inheritance makes it possible for different classes to share the same set of attributes and methods, the same program can be run against objects that belong to different classes. This is the basis of the object-oriented user interface that desktop publishing systems and windows management systems provide today. The same set of programs (e.g., open, close, drop, create, move, etc.) apply to different types of data (image, text file, audio, directory, etc.).
- If the users define many classes, and each class has many attributes and methods, the benefit of sharing not only the attributes but also the programs can be dramatic. The attributes and programs need not be defined and written from scratch. Adding attributes and methods of existing classes, thereby reducing the opportunity to introduce new errors to existing classes, can create new classes.
- The concept of inheritance was first introduced in Simula67. Inheritance is supported by most object-oriented programming languages, such as C++. The benefit of the concept of inheritance is that new data types can be defined as extensions of previously defined types, thereby avoiding the need to have to define new data types from scratch for each new project. To put it simply, inheritance derives a new data type from an existing data type.
- The two main reasons for the rapid increase in interest in object-oriented programming over the last several years for applications software development have been (a) the wider availability of these languages plus supporting environments for the object-oriented

paradigm, and (b) the emergence of increasingly higher powered hardware (Henderson-Sellers and Edwards, 1994).

- A formal proposal for what constitutes an object-oriented database was developed in 1989 at the First International Conference on Deductive and Object-oriented Databases (Henderson-Sellers and Edwards, 1994). What was agreed upon at and subsequent to the conference was a list of the features that an object-oriented database should support.

According to Martin and Leben (1995), object-oriented databases will probably coexist, rather than replace, relational databases, because each has features that are beneficial. The important features of relational databases include security and integrity, but businesses cannot operate while relying only on the data types available in relational databases as they represent only about 10 percent of the data that is available for storage (Betts, 1997). Some companies may choose to hybridize by putting pure object front-ends on relational databases, thereby gaining the development benefits of objects and the security benefits of relational databases.

The technology for object-oriented databases first evolved from a need to support object-oriented programming. C++ programmers needed a store for data that remained after a process terminated. According to Martin and Leben (1995), object-oriented databases became important for certain types of applications with complex data, such as computer-aided design (CAD) and computer-aided engineering (CAE). Understanding the benefits of object-oriented technology over older technologies (Martin and Odell, 1992) can help management understand the potential strengths of object-oriented databases.

1. Designers think in terms of behavior of objects, not small details. Encapsulation allows the small details to be hidden and makes complex classes easy to use.

2. Classes are designed so that they can be reused again and again. To make the most or reuse, classes can be built so that they can be customized.
3. Classes designed for repeated reuse are more stable.
4. Software built from stable classes is likely to have fewer bugs than software built from scratch
5. Applications are created from preexisting parts, thereby improving the speed of design time.
6. Designs are often of higher quality because they are created from already-proven components.
7. Programs built of smaller pieces are easier to create.
8. Object-oriented analysis may model the enterprise or application area in a way that is closer to reality than conventional analysis.
9. Object-oriented methodologies encourage better communication between programmers and lay clients, because clients think in terms of objects and events rather than in traditional programming structures.
10. A graphical user interface is beneficial because it is easier to click on an icon than to remember numerous commands.
11. Classes are designed to be independent of platforms, hardware, and software environments.
12. It may be easier to share software from numerous vendors. Also, software developed independently by different vendors should be able to work together and appear as a single unit to the end user.

13. Benefits can be realized in client-server computing: a server class may be used by many different clients, and these clients access server data with the class methods, thereby helping ensure that the data is not corrupted.
14. Object-oriented design is the key to large-scale distributed computing. Classes in one machine will interact with classes in other machines without needing to know where the classes reside.
15. Parallel computing means that objects on different processors will be able to execute at the same time, each acting independently.
16. There is a higher level of database automation with an object-oriented design. The data structures in object-oriented databases are linked to methods that take automatic actions. In a sense, an object-oriented database has intelligence built into it in the form of methods, whereas a basic relational database does not.
17. Object-oriented databases have demonstrated significantly higher performance than relational databases for certain applications with very complex data structures.
18. Businesses can conveniently create their own libraries of classes that reflect their company standards and application needs, and can add to these libraries as policies and needs change.

Despite the benefits of object-oriented technology, it may take some businesses longer to adapt than others. Consider companies in the insurance industry. According to representatives of Miller Freeman, Inc. (1999), many insurance companies would like to convert from COBOL mainframe-based applications to component-based object-oriented frameworks, but are indecisive due to the complexity and costs of such a project. One strategy, perhaps, is to externalize certain business functions such as billing, rating, reporting, and underwriting, by

building these as stand-alone components through object-oriented development technology and object-oriented programming.

2.2.3 Data Warehouse

Information Technology (IT) has historically influenced organizational performance and competitive standing. The increasing processing power and sophistication of analytical tools and techniques have put the strong foundation for the product called data warehouse. There are a number of reasons that any organization should consider a data warehouse, which can be the critical tool for maximizing the organization's investment in the information it has collected and stored throughout the enterprise. IT managers need to understand the rationale and benefits of data warehouses because they may need to design and implement, or procure this kingpin of business intelligence.

A data warehouse is a repository (collection of resources that can be accessed to retrieve information) of an organization's electronically stored data, designed to facilitate reporting and analysis (Inmon, 1995).

A data warehouse is also defined as a subject-oriented, integrated, nonvolatile, time-variant collection of data in support of management's decisions. A common way of introducing data warehousing is to refer to the characteristics of a data warehouse as set forth by William Inmon (1995).

This definition of the data warehouse focuses on data storage. The main source of the data is cleaned, transformed and cataloged and is made available for use by managers and other business professionals for data mining, online analytical processing, market research and decision support (O'Brien and Marakas, 2009). However, the means to retrieve and analyze data, to extract, transform and load data, and to manage the data dictionary are also considered essential components of a data warehousing system.

The data warehouses are supposed to provide storage, functionality and responsiveness to queries beyond the capabilities of today's transaction-oriented databases. Also data warehouses are set to improve the data access performance of databases. Traditional databases balance the requirement of data access with the need to ensure integrity of data. In present day organizations, users of data are often completely removed from the data sources. Many people only need read-access to data, but still need a very rapid access to a larger volume of data than can conveniently be downloaded to the desktop. Often such data comes from multiple databases. Because many of the analyses performed are recurrent and predictable, software vendors and systems support staff have begun to design systems to support these functions. Currently there comes a necessity for providing decision makers from middle management upward with information at the correct level of detail to support decision-making. Data warehousing, Online Analytical Processing (OLAP) and data mining provide this functionality.

2.2.3.1 Subject Oriented

Data warehouses are designed to help you analyze data. For example, to learn more about your company's sales data, you can build a warehouse that concentrates on sales. Using this warehouse, you can answer questions like "Who was our best customer for this item last year?" This ability to define a data warehouse by subject matter, sales in this case makes the data warehouse subject oriented.

2.2.3.2 Integrated

Integration is closely related to subject orientation. Data warehouses must put data from disparate sources into a consistent format. They must resolve such problems as naming conflicts and inconsistencies among units of measure. When they achieve this, they are said to be integrated.

2.2.3.3 Nonvolatile

Nonvolatile means that, once entered into the warehouse, data should not change. This is logical because the purpose of a warehouse is to enable you to analyze what has occurred.

2.2.3.4 Time Variant

In order to discover trends in business, analysts need large amounts of data. This is very much in contrast to online transaction processing (OLTP) systems, where performance requirements demand that historical data be moved to an archive. A data warehouse's focus on change over time is what is meant by the term time variant.

More generally, data warehousing is a collection of decision support technologies, aimed at enabling the knowledge worker, such as executive, manager, and analyst, to arrive at better and faster decisions (Oracle, 2012). Data warehouses provide access to data for complex analysis, knowledge discovery, and decision-making. They support high performance demands on an organization's data and information. It provides an enormous amount of historical and static data from three tiers:

1. Relational databases
2. Multidimensional OLAP applications
3. Client analysis tools

Several types of applications such as online analytical processing (OLAP), decision-support systems (DSS) and data mining are being supported. OLAP is a term used to describe the analysis of complex data from the data warehouse.

OLAP is a software technology that allows users to easily and quickly analyze and view data from multiple points-of-view. OLAP provides dynamic and multi-dimensional support to

executives and managers who need to understand different aspects of the data. Activities that are supported include:

- Analyzing financial trends
- Creating slices of data
- Finding new relationships among the data
- Drilling down into sales statistics
- Doing calculations through different dimensions where each category of data (that is, product, location, sales numbers, time period, etc.) is considered a dimension.

There are OLAP tools that use distributed computing capabilities for analyses that require more storage and processing power than can be economically and efficiently located on an individual desktop.

DSS support an organization's leading decision makers with higher-level data for complex and critical decisions. A DSS queries a data warehouse or an OLAP database for relevant information that can be compared in order to make a business decision and predict the impact of that decision.

Finally, data mining is being used for knowledge discovery, the process of searching data for unanticipated new knowledge. Knowledge workers and decision makers use tools ranging from parametric queries to ad hoc queries to data mining. Thus, the access component of the data warehouse must provide support of structured queries (both parametric and ad hoc). These together make up a managed query environment.

2.2.4 Databases and Data Warehouses

A database is a collection of related data and a database system is a database and database software together (Gupta, 2009). A data warehouse is also a collection of information as well as a supporting system. Databases are transactional such as relational, object-oriented, network or hierarchical. Traditional databases support on-line transaction processing (OLTP), which includes insertions, updates, and deletions, while also supporting information query requirements. Traditional databases are optimized to process queries that may touch a small part of the database and transactions that deal with insertions or updates of a few tuples per relation to process.

Thus databases must strike a balance between efficiency in transaction processing and supporting query requirements (ad hoc user requests), That is, they can't further optimized for the applications such as OLAP, DSS and data mining.

But a data warehouse is typically optimized for access from a decision maker's needs. Data warehouses are designed specifically to support efficient extraction, processing and presentation for analytic and decision-making purposes. In contrast to databases, data warehouses generally contain very large amounts of data from multiple sources that may include databases from different data models and sometimes files acquired from independent systems and platforms.

Multidatabases provide access to disjoint and usually heterogeneous databases and are volatile. Whereas a data warehouse is frequently a store of integrated data from multiple sources, and processed for storage in a multidimensional model and nonvolatile. Data

warehouses also support time-series and trend analysis, both of which require more historical data.

In transactional systems, transactions are the unit and are the agent of change to the database, but data warehouse information is much more coarse-grained and is refreshed according to a careful choice of incremental refresh policy. Warehouse updates are handled by the warehouse's acquisition component that provides all required processing. As data warehouses encompass large volumes of data, they are more or less double the size of source databases.

The sheer volume of data likely to be in terabytes is an issue that has been dealt with through enterprise-wide data warehouses, virtual data warehouses and data marts. Enterprise-wide data warehouses are huge projects in need of massive investment of time and resources. Virtual data warehouses are bound to provide views of operational databases that are materialized for efficient access. A data mart is an easy-to-access repository of a subset of highly focused data for a single function or department (i.e., finance, sales, and marketing) and is considerably smaller than a data warehouse. The data comes from operational information that is needed by a particular group of employees for analysis, content, presentations all in terms that are familiar to them. Data for a data mart is derived from a data warehouse or from more specialized access.

Data warehouses exist to facilitate complex, data-intensive and frequent ad hoc queries. Data warehouses must provide far greater and more efficient query support than is demanded of transactional databases. The data warehouse access component supports enhanced spreadsheet functionality, efficient query processing, structured queries, and ad hoc queries, data mining and materialized views. Particularly enhanced spreadsheet functionality includes support for

state-of-the art spreadsheet applications as well as for OLAP applications programs. These provide pre-programmed functionalities such as the following:

Roll-up: Data is summarized with increasing generalization

Drill-down: Increasing levels of detail are revealed

Pivot: Cross tabulation that is, rotation, performed

Slice and dice: Performing projection operations on the dimensions

Sorting: Data is sorted by ordinal value

Selection: Data is available by value or range

Derived or computer attributes: Attributes are computed by operations on stored and derived values.

2.2.5 Query and Query Systems

Query is the request made on data for the purpose of manipulating it to get results based on the need of the user. It may be a request to fetch information, add information, and update information or to manipulate information in a certain manner (Tang et al, 2010). Aquery is a request for information. A query tells IQ/Objects what to look for in your database, how to format the data and where to send it. You construct queries by using different tools to choose fields, apply filters and format the output (Robert and David, 2007).

2.2.5.1 Query System.

On a client server architecture in large systems, a query is a pair of messages, or prompt of a message, transmitted from client to server ,which encodes a question; and the reply ,which is

the answer to the request returned to the client. When no logical reply exists, the server may return an error instead (Tang et al, 2010).

Queries to the database may be posed either interactively or in a batch mode. When queries are posed interactively, the system can only hope to optimize processing of each query separately. In batch environment, where an application program may include a number of queries, it may be desirable to attempt global optimization (Gupta, 2009).

2.2.5.2 Query Languages

A query language is a language in which a user requests information from a database. These are typically higher-level than programming languages. Query languages, according to Gupta (2009), may be one of:

- *Procedural, where the user instructs the system to perform a sequence of operations on the database. This will compute the desired information.

- * Non-procedural, where the user specifies the information desired without giving a procedure for obtaining the information.

He further stated that a complete query language also contains facilities to insert and delete tuples as well as to modify parts of existing tuples. Most commercial relational-database systems, according to him, offer a query language that includes elements of both the procedural and the non-procedural approaches.

Query cost refers to the estimated elapsed time, in seconds required to execute a query on a specific hardware configuration. On other hardware configurations, there is a correlation between cost units and elapsed time, but cost units do not equal seconds. The query governor lets you specify an upper cost limit for a query; a query that exceeds this limit is not run (<http://techmet.microsoft.com/en-us/library/>). Because it is based on estimated query cost rather than actual elapsed time, the query governor does not have any run-time overhead. It also

stops' long running queries before they start, rather than running them until they reach a predefined limit.

2.2.5.3 Query processing

Query processing is the process of translating a query expressed in a high-level language such as SQL into low-level data manipulation operations (Aljanaby et al, 2005). A relational database consists of many parts, but at its heart are two major components: The storage engine and the query processor. The storage engine writes data to disk and reads data from the disk. It manages records, controls concurrency, and maintains log files.

The query processor accepts SQL syntax, selects a plan for executing the syntax, and then executes the chosen plan. The user or program interacts with the query processor in turn interacts with the storage engine. The query processor isolates the user from the details of execution: The user specifies the result, and the query processor determines how this result is obtained.

2.2.5.4 Component of the Query Processor

i. **Parser:** In the first phase, the query is parsed and translated into an internal representation (e.g., a query graph (Jenq et al 1990; Pirahesh et al 1992) that can be easily processed by the later phases. The development of parsers is well understood (Aho et al, 1987) and tools like flex and bison can be used for the construction of SQL or OQL Parsers just as for most other Programming Languages. The same parser can be used for a centralized and distributed database system.

ii. **Query Rewrite** Query rewrite transforms a query in order to carry out optimizations that are good regardless of the physical state of the system (e.g., the size of tables, presence of indices, locations of copies of tables, speed of machines, etc) (Pirahesh et al., 1992). Typical transformations are the elimination of redundant predicates, simplifications of expressions,

and unnesting of subqueries and views. In a distributed system, query rewrite also selects the partitions of a table that must be considered to answer a query (Ceri and Pelagatti, 1984; Ozsu and Valduriez, 1999). Query rewrite is carried out by a sophisticated rule engine (Pirahesh et al, 1992).

iii. Query Optimizer: This component carries out optimizations that depend on the physical state of the system. The optimizer decides which indices to use to execute a query, which methods (e.g; hashing or sorting) to use to execute the operations of a query (e.g; joins and group-bys), and in which order to execute the operations of a query. The query optimizer also decides how much main memory to allocate for the execution of each operation. In a distributed system, the optimizer must also decide at which site each operation is to be executed. To make these decisions, the optimizer enumerates alternative plans using a cost estimation model. Almost all commercial query optimizers are based on dynamic programming in order to enumerate plans efficiently (Kossmann, 2000).

iv. Plan Refinement/Code Generation: This component transforms the plan produced by the optimizers into an executable plan. In System R, for example, this transformation involves the generation of an assembler like code to evaluate expressions and predicates efficiently (Lorie and Wade, 1979). In some systems, plan refinement also involves carrying out sample optimizations which are not carried out by the query optimizer in order to simplify the implementation of the query optimizer (Kossmann, 2000).

v. Query Execution Engine: This component provides generic implementations for every operator (e.g., send, scan, or NLT). All state-of-the-art query execution engines are based on an iterator model (Graefe, 1993). In such a model, operators are implemented as iterators and all iterators have the same interface. As a result, any two iterators can be plugged together (as specified by the consumer- producer relationship of a plan), and thus any plan can

be executed. Another advantage of the iterator model is that it supports the pipelining of results from one operator to another in order to achieve good performance (Kossmann, 2000).

vi. Catalog: The catalog stores all the information needed in order to parse, rewrite and optimize a query. It maintains the schema of the database (i.e, definitions of tables, views, user-defined types and functions, integrity constraints, etc), the partitioning. Schema (ie, information about what global tables have been partitioned and how they can be reconstructed), and physical information such as the location of copies of partitions of tables, information about indices, and statistics that are used to estimate the cost of a plan. In most relational database systems, the catalog information is stored like all other data in tables. In a distributed database system, the question of where to store the catalog arises. The simplest approach is to store the catalog at one central site. In wide-area networks, it makes sense to replicate the catalog at several sites in order to reduce communication cost. It is also possible to cache catalog information at sites in a wide area network (Williams et al, 1981). Both replication and caching of catalog information are very effective because catalogs are usually quite small (Hundreds of Kilobytes rather than gigabytes) and catalog information is rarely updated in most environments. In certain environments however, the catalog can become very large and be frequently updated. In such environments, it makes sense to partition the catalog data and store catalog data where it is most needed. For example, catalogs of distributed object databases need to know where copies of all the objects (Potentially millions) are stored, and they need to update this information every time an object is migrated or replicated. Such catalogs can be implemented in a hierarchical way as described in Eickler et al (1997).

2.2.5.5 Alternative Approaches Commonly used in Existing Systems Today

In this section, the tradeoffs between alternative approaches which are commonly used in existing systems today would be presented and discussed.

i. **Query Shipping:** The first approach is called query shipping. Query shipping is used in many relational and objects –relational database systems today (e.g IBM DB2, oracle 8, and Microsoft SQL server). The principle of query shipping into executes queries at servers (ie at the lowest level possible in a hierarchy of sites). Figure 2.2 below illustrates query shipping in to a system with one server. A client ships the SQL (or OQL) code of a query to the server; the server evaluates the query and ships the results back to the client. In systems with several servers, query shipping works only if there is a middle-tier site that carries out joins between tables stored at different servers or if there are gateways between the servers so that inter site joints can be carried out at one of the servers (Kossmann, 2000).

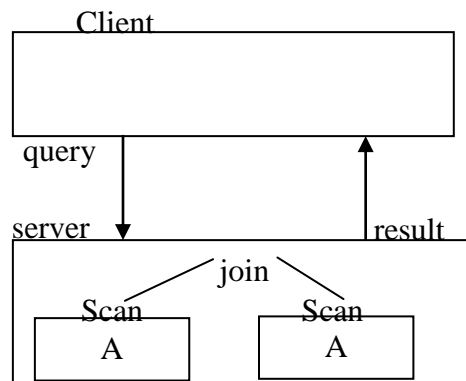


Figure 2.2: Query shipping (Adapted from Kossmann, 2000)

ii. **Data Shipping:** There exact opposite of query shipping is data shipping, which is used in many object-oriented database systems (e.g object store and O₂). In this approach, queries are executed at the client machine at which the query was initiated and data is rigorously cached at client machines in main memory or on disk (Franklin et al 1993). That is, copies of the data used in a query are kept at a client so that these copies can be used to execute subsequent queries at that client. Caching is typically carried out in the granularity of pages (ie 4k or 8k blocks of tuples) (DeWitt et al 1990), and it is possible to cache

individual pages of base tables and indices (Lomet 1996; Zaharioudakis and Carey 1997). To illustrate data shipping, consider the example shown in figure 2.2 below, where some pages of Tables A and B are already cached at the client (represented by the dashed boxes in the figure). The scan operators at the client use these cached copies of pages and fault in all the pages of A and B that are not cached (Kossmann 2000).

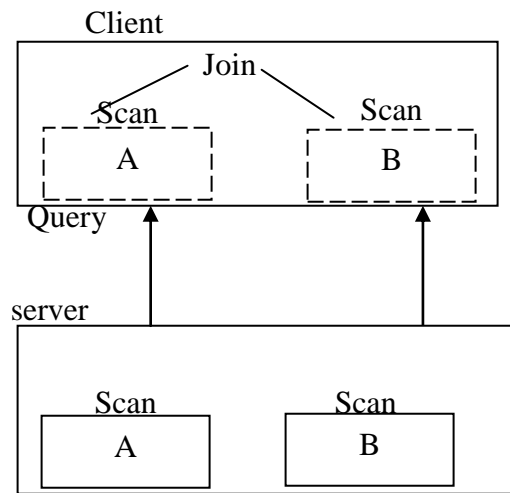


Figure 2.3: Data Shipping (Adapted from Kossmann, 2000)

iii. **Hybrid Shipping:** Neither data shipping or query shipping is the best policy for query processing in all situations. The advantage of both approaches can be combined in a hybrid shipping architecture (Franklin et al 1996). Hybrid shipping provides the flexibility to execute query operators on client and server machines, and it allows the caching of data by clients. The approach is illustrated in figure 2.3 below, where the scan (A) and join operators are carried out at the client whereas the scan (B) operator is carried out at the Server. The Scan (A) operator uses the Clients cache as much as possible and ships to the Client only those parts of A that are not in the cache. In contrast, the Scan (B) operator neither uses nor changes the state or the client's cache (Kossmann, 2000).

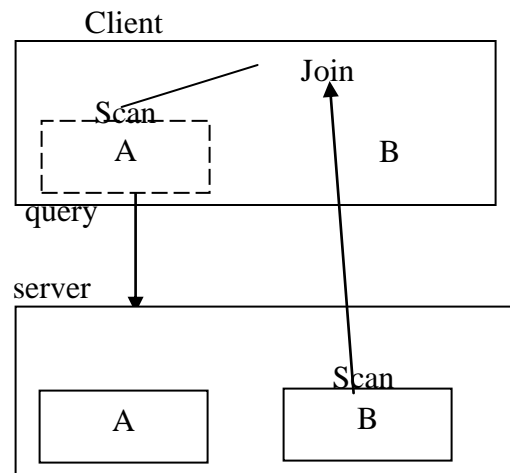


Figure 2.4: Hybrid Shipping (Adapted from Kossmann, 2000)

iv. Other Hybrid Shipping Variants:

For application programs that carry out SQL- style queries and C++ style methods, one special and restricted variant of hybrid shipping is to execute the SQL – style queries at the servers, without caching, and the C++ style methods at the Clients, using caching. Such an approach has been proposed, for example, as part of the KRISYS Project (Härder et al 1998). Persistence is a product that supports this is approach (Keller et al, 1993). This approach is reasonable because caching and Client-side execution are particularly effective for methods that repeatedly access the same objects in order to carry out complex computation. Queries that involve a great deal of data, on the other hand, can often be executed more efficiently at Server machines without making use of Client-side caching (Kossmann, 2000).

Another variant of hybrid shipping is used by certain decision support products (e.g., products by Micro Strategy). These products have threetier architecture. The bottom tier is a standard relational database system that stores the database and carries out join processing and other standard relational operations. The middle tier then carries out nonstandard operations for decision support like (moving averages, roll-up, drill-down, etc) (Gray et al, 1996; Kimball and Strehlo, 1995). Again, such an architecture is a special hybrid shipping variant because

query processing is carried out at Servers and at middle-tier machines, and the difference from the full-fledged hybrid shipping is that not all operations can be carried out at all the machines tiers (Kossmann, 2000).

2.2.6 Query Performance

In general, the more nodes and resources that are available, the better the potential query performance. Query processing uses the available memory and CPU resources of all nodes of the logical Server. The amount of improvement benefits depends on the type of query, the size of the query, and the current workload of the nodes in the logical server: It is unlikely that any two runs of the same query result in exactly the same work distribution as load levels change in the cluster, so does the load distribution. Query performance is determined by the overall workload of the logical Server at any given time. Similarly, in a single run of a query in a long processing time, the work distribution changes over the course of query execution as the load balance charges across worker nodes.

2.2.6.1 Query Execution

Query execution is the process of executing the plan chosen during query optimization. The objective is to execute the plan quickly by returning the answer to the user (or more often, the program run by the user) in the least amount of time. This is not the same as executing the plan with the lowest resources (CPU, I/O, and memory). For example, a parallel query almost always uses some more resources than a non-parallel query but it is often desirable because it returns the result more quickly.

Query execution is presented before query optimization because the set of available execution techniques determines the set of choices available to the optimizer. The techniques include disk I/O, sorting, join and hash operations, index intersections, index join, and parallelism.

2.2.6.2 Alternative Ways to Execute Queries in a Distributed Database System

This subsection describes alternative ways to execute queries in a distributed database system. In particular, how data can be shipped and how joins between tables stored at different sites can be computed would be described.

i. Row Blocking: Communication is typically implemented by send and receives operators. Naturally, the implementation of these operators is based on TCP/IP, UDP, or some other network protocol (Tanenbaum, et al.,1989). To reduce the overhead, almost all database systems employ a technique called row blocking. The idea is to ship tuples in a blockwise fashion, rather than every tuple individually. In other words, a send operator consumes several tuples of its child operator and sends these tuples as a batch. This approach is obviously much cheaper than the naïve approach of sending one tuple at a time because the data is packed into fewer messages. The size of the blocks is a parameter of the send and receive operators; this parameter is set taking into account the characteristics of the network (ie; the message size of the network) (Kossmann, 2000).

One particular advantage of row blocking is that it compensates for burstiness in the arrival of data up to a certain point if tuples are shipped one by one through the network, any short delay in the network would immediately stop the execution of the query at the receiving site because of a shortage of tuples to consume. Due to row blocking, the receive operator has a reservoir of tuples and can feed its parent operator even if the next block or tuples is delayed. As a result, it is often better to choose a block size used by the network (Kossmann, 2000).

ii. Optimization of Multicasts: In most environments, networks are organized in a hierarchical way so that communication costs vary significantly depending on the locations of the sending and receiving sites. It is, for instance, cheaper to send data from Munich to Passau, which are both in Germany, than from Washington, across the Atlantic, to Passau.

Sometimes, a site needs to send the same data to several sites to execute a query; it is, for instance, possible that the same data must be sent from Washington to Munich and Passau. If the network does not provide cheap ways to implement such multicasts, it is preferable to send the data from Washington to Munich and then forward it from Munich to Passau, rather than sending the data from Washington across the Atlantic twice (Kossmann, 2000).

Sometimes, this technique is useful even in a homogenous and fast network. Let us assume that the time on the wire to send messages between Washington, Munich, and Passau is negligible ; in this case, CPU costs to send (ie. Pack) and receive (unpack) messages dominate communication costs. If Washington is heavily loaded or has a slow CPU, then it might again be better if Passau receives the data from Munich rather than from Washington. Obviously, another option would be for Passau to receive the data from Washington and for Munich to receive the data from Passau. The best choice must be made by the query optimizer (Kossmann, 2000).

iii. Multithreaded Query Execution: To take the best advantage of intra-query parallelism, it is sometimes advantageous to establish several threads at a site (Graefe, 1990). As an example, consider the plan of figure 2.4 below, which implements the query $A_1 \cup A_2 \cup A_3$; A_1 is stored at Site 1, A_2 at Site 2, and A_3 at Site 3. If the union and receive operators at site O are executed within a single thread, then site O only requests one block, at a time (e.g. , in a round- robin way) and the opportunity to read and send the three partitions from Sites 1,2 and 3 to Site O in parallel is wasted. Only if the union and receive operators at Site O run in different threads can the three receive operators continuously ask for tuples from the send operators at Sites 1, 2 and 3 so that all three send operators run and produce tuples in parallel (Kossmann, 2000).

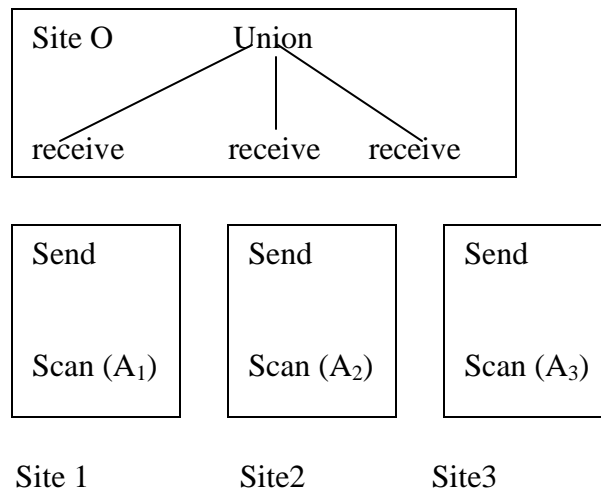


Figure 2.5: Example Union plans (Adapted from Kossmann, 2000)

Establishing a separate thread for every query operator, however, is not always the best thing to do. First, shared – memory communication between threads needs to be synchronized, resulting in additional cost. Second, it is not always advantageous to parallelize all operations. Consider, for example the plan of figure 2.5 below which carries out a sort-merge join of Tables A and B. Depending on the available main memory Site O, it might or might not be advantageous to receive and Sort Tables A and B in Parallel at Site O. If there is plenty of main memory to store large fractions of both A and B at Site O, then the two pairs of receive and sort operators should be carried out in parallel in order to parallelize the Send and Scan of A and B. Otherwise, the two receive – sort branches should

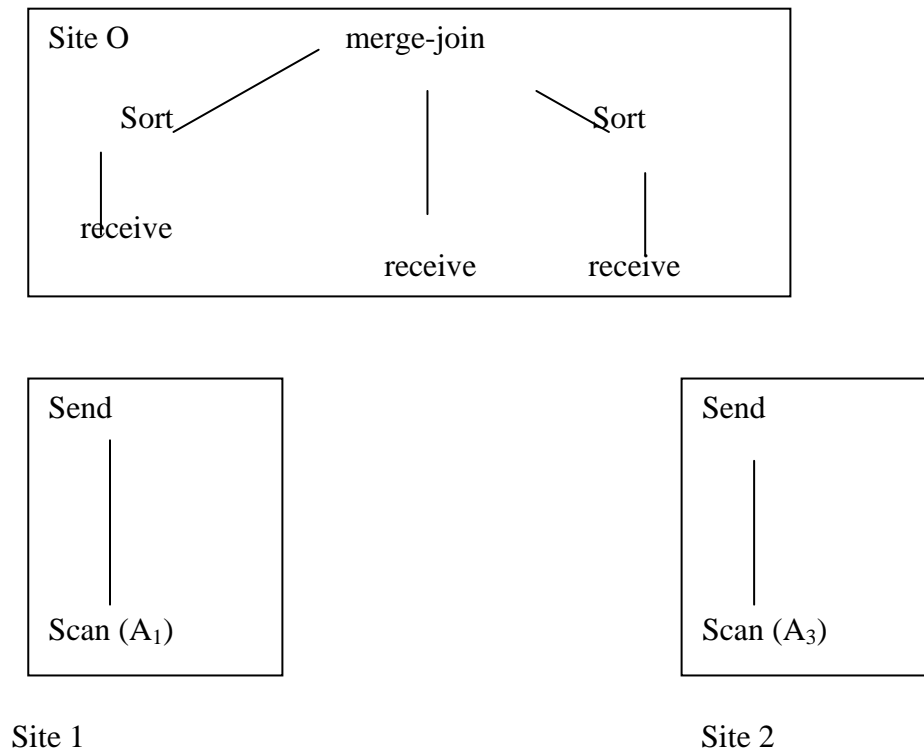


Figure 2.6: An Example of Join Plan (Adapted from Kossmann, 2000)

Be carried out one at a time in order to avoid resource contention at Site O (ie. thrashing if both Sorts write concurrently to the same disk). The query optimizer and/or a scheduler at run time must decide which parts of a query should run in parallel and, thus, which operators should run in the same thread (Kossmann, 2000).

iv. Joins With Horizontally Partitioned Data

The logical properties of the joins and union operators make it possible to process joins in a number of different ways if the tables are horizontally partitioned. If, for example, Table A is horizontally partitioned in such a way that $A = A_1 \cup A_2$, then $A \bowtie B$ can be computed in the following two ways (Vogels, 2003):

$$(A_1 \cup A_2) \bowtie B \text{ or } (A_1 \bowtie B) \cup (A_2 \bowtie B)$$

If A is partitioned into more than two partitions or if B is also partitioned, then even more variants are possible: for example,

$(A_1 \cup A_2) \bowtie B) \cup (A_3 \bowtie B)$ might be an attractive plan if B is replicated and one copy of B is located at a Site near the Sites that Store A_1 and A_2 and another Copy of B is located near the Site that stores A_3 . The optimizer ought to consider all these options (Kossmann, 2000).

In some situations, A and B are partitioned in such a way that it is possible to deduce that some of the $A_i \bowtie B_j$ are empty. The optimizer should, of course take advantage of such knowledge and eliminate such “empty” expressions in order to reduce the cost of join processing. One very common situation is that A and B are partitioned in such a way that $A_i \bowtie B_j$ is empty if $i \neq j$. Consider, for example, a company that has Dept table that is partitioned by Dept. location in order to store all the Dept information at the site of the department. This company may also have an Emp table that is partitioned according to the location of the Dept in which the Emp works $Emp \bowtie Dept$ can be carried out for this company by joining the Emp and Dept partitions separately at every Site. In other words, the following equation holds if the company has n Sites:

$$\begin{aligned} & (Emp_1 \cup \dots \cup Emp_n) \bowtie (Dept_1 \cup \dots \cup Dept_n) \\ &= (Emp_1 \bowtie Dept_1) \cup \dots \cup (Emp_n \bowtie Dept_n) \end{aligned}$$

v. SEMI JOINS: Semi join programs were proposed as another technique to process joins between tables stored at different sites (Bernstein et al.1981). If table A is stored at Site 1 and table B is stored at Site 2, then the “conventional “way to execute $A \bowtie B$ is to ship A from site 1 to Site 2 and execute the join at Side 2(or the other way around). The idea of a semi join program is to send only the column(s) of A that are needed to evaluate the join predicates from Site1 to 2, find the tuples of B that qualify the join at Site 1 and then match A

with those B tuples at Site 1. Formally, this procedure can be described as follows (\bowtie is the semi join operation and $\pi(A)$ projects out the columns from A).

$$A \bowtie B = A \bowtie (B \bowtie \pi(A))$$

Variants of this approach are meant to eliminate duplicate tuples from $\pi(A)$ (traditional additional work at Site 1 for less communication) and sending a signature file for A called a bloom-hash filter, rather than $\pi(A)$ (Babb,1979; Valduriez and Gardarin,1984) Again, the optimizer must decide which variant to use, if any, and which direction to carry out the semi join program, from Site 1 to Site 2 or vice versa, based on the cardinalities of the tables, the selectivity of the join predicate(s), and the location of the data in the other operations of the query.

Experimental work indicates that semi join programs are typically not very attractive for join processing in standard (relational) distributed database systems because the additional computational overhead is usually higher than the savings in communication costs (Lu and Carey, 1985; Mackert and Lohman, 1986) .Today, however, several applications that involve tables with very large tuple can be given and semi join style techniques can indeed be very attractive for such applications. Consider, for example, a table that store employee information including a picture of every employee. In this case, it does make sense to find the target employees of a query using, say, the *age*, *dept.no*, and so on columns and then *fetch the picture* and other columns of the query result at the end. In a Client- Server System, for example, the following plan might be very useful

$$(A \bowtie S_1 C) \bowtie S_3 (B \bowtie S_2 C)$$

If A is stored at Server S_1 , B is stored at Server S_2 , C is replicated at both Servers and the result of the whole query must be displayed at Client S_3 (Braumand et al 2001c., Stocker et al, 2001).

vi. Double-Pipelined Hash Joins: Recently, double- pipelined(or non-blocking) hash-join algorithm were proposed(Ives et al, 1999: Urhan and Franklin 1999; Wilshut and Apers, 1991).The use of such join algorithms makes it possible to deliver the first result of a query as early as possible. In addition, such join algorithms makes it possible to fully exploit pipelined parallelism and thus reduce the overall response time of a query in a distributed system. As described in Urhan and Franklin (1999), Variants of such join methods can be particularly useful in a distributed system in which the delivery of tuples through the network is bursty because certain phases of the join processing can be carried out at a Site while the Site waits for the next, possibly delayed, batch of tuples. The basic idea on which all these algorithms are based is quite simple. To execute $A \bowtie B$, two main-memory hash tables are constructed: one for tuples of A and one for tuples of B . The two hash tables are initially empty. The tuples of A and B a way that the main memory is exhausted. To remedy such a situation, the algorithms in Ives et al (1999) and Urban and Franklin (1999) adopt a hybrid harsing are partitioning scheme.

i. Pointer-Based Joins and Distributed Object Assembly: One particular kind of query that can be found in object oriented and object-relational database systems are called pointer-based joins. Pointer-based joins occur because foreign keys are implemented in these systems by explicit references that contain the address of an object or the address a placeholder of an object (Eickler et al, 1995).

2.2.6.3 Query Optimization

In an attempt to define query optimization, it is pertinent for us to define the concept, optimization. Waas and Galindo- Legaria (2000) defined optimization as a mandatory exercise since the difference between the cost of the best plan and a random choice could be in orders of magnitude. Gupta (2009) stated that optimization in non-procedural language therefore is much more complex task since it not only involves code optimization (how to carry out the operations that need to be carried out) but also selecting the best plan as well as selecting the best access path.

Having stated above, we can say that Query optimization according to Aljanaby, Abuelrub and Odeh (2005), refers to the process by which the best execution strategy for a given query is found from a set of alternatives. Kossmann and Stocker (2000) had earlier stated that the great commercial success of database systems is partly due to the development of sophisticated query optimization where users pose queries in a declarative way using SQL or OQL.

2.2.6.4 Types of Query Optimizers

There are two major types of query optimizers in relational databases. Syntax-based and cost-based.

i. Syntax-Based Query Optimizers

A syntax-based query optimizer creates a procedural plan for obtaining the answer to an SQL query, but the particular plan it chooses is dependent on the exact syntax of the query and the order of the clauses within the query. A syntax-based query optimizer executes the same plan every time, regardless of whether the number or composition of records in the database changes over time. Unlike a cost-based query optimizer, It neither maintains nor considers statistics about the database.

ii. Cost-Based Query Optimizers

A cost-based query optimizer chooses among alternative plans to answer an SQL query. Selection is based on estimates for different plans. The factors in making cost estimates include the number of I/O operators, the amount of CPU time, and so on. A cost-based query optimizer estimates these costs by keeping statistics about the number and composition of records in a table or index and is not dependent on the exact syntax of the query or the order of clauses within the query (unlike a syntax-based query optimizer).

2.2.7 Distributed Query Optimization

Distributed query optimization is a database feature that reduces the amount of data transfer required between Sites when a transaction retrieves data from remote tables referenced in a distributed SQL statement (Gupta, 2009).

Distributed query optimization uses cost-based optimization to find or generate SQL expressions that extract only the necessary data from remote tables, process that data at a remote site or sometimes at the local site, and send the results to the local site for final processing. This operation reduces the amount of required data transfer when compared to the time it takes to transfer all the table data to the local site for processing (Gupta, 2009). In distributed query optimization two more steps are involved between query decomposition and query optimization: Data localization and global query optimization: Data input to data localization is the initial algebraic query generated by the query decomposition step. The initial algebraic query is specified on global relations irrespective of their fragmentation or distribution (Aljanaby et al, 2005).

The main role of data localizations is to localize the query using rate distributed information. In this step, the fragments that are involved in the query are determined and the query is transformed into one that operates on fragments rather than global relation. This during the data localization step, each global relation is first replaced by its localization program, which is union of the fragment of a horizontally or vertically fragment query, and then the resulting fragment query is simplified and restructured to produce another good query. Simplification and restructuring may be done according to the same rules used in the decomposition step. The final fragment query is generally far from optimal; this process only eliminates bad queries (Aljanaby et al, 2005).

The input to the third step is a fragment query that is an algebraic query on fragments. By permuting the ordering of operations within one fragment query, many equivalent query execution plans may be found. The goal of query optimization is to find an execution strategy for the query that is close optimal. An execution strategy for a distributed query can be described with relational algebra operations and communication primitives (send/receive operations) for transferring data between Sites (Oszu and Valduriez, 1991).

The query optimizer that follows this approach is seen as three components: A search space, a search strategy and a cost model. The search space is the set of alternative execution to represent the input query. These strategies are equivalent, in the sense that they yield the same result but they differ on the execution order of operations and the way these operations are implemented. The search strategy explores the search space and selects the best plan. It defines which plans are examined and in which order. The cost model predicts the cost of a given execution plan which may consist of the following components (Elmasri and Navathe, 2000):

- i. Secondary storage cost: This is the cost of searching for reading and writing data blocks on secondary storage.
- ii. Memory storage cost: This is the cost pertaining to the number of memory buffers need during query execution
- iii. Computation cost: This is the act of performing in memory operations on the data buffers during query optimization.
- iv. Communication cost: This is the cost of shipping the query and its results from the database Site to the Site or terminal where the query originated.

Object databases development have created the storage of large amounts of complex data and integrated complex data with simple data which are common in many real life application development demands today (Hibatullah, 2016). The ability to simultaneously retrieve this complex and simple data is increasingly becoming very important in many real life applications. For instance, to review a patient's condition a physician may request the scanned image of the organ along with vital statistics of the patient. This complex objects stored in the database requires certain details which can only be provided during retrieval by object queries.

2.2.8 Impedance mismatch

Impedance mismatch is perceived by many professionals as a negative phenomenon in the software construction that arises from an eclectic mix of two incompatible languages: a *query language* that is used to access and update of a database and a *programming language* that is used for making client applications acting on the database. A bit more careful look shows that this negative connotation is perhaps inadequate. Impedance mismatch is an inevitable consequence of a (quite reasonable) principle known as data independence. Misunderstanding of the relationships between impedance mismatch and data independence is the reason of

some creativity within modern object-oriented programming languages (mostly Java), which some professionals could perceive as a medicine that is worse than the illness. Impedance mismatch has also less discussed aspect concerning database models and transformations between database schemas.

2.2.8.1 Impedance mismatch between query and programming languages

The concept of query languages developed in 1970-ties assumed no pragmatic universality. However, because eventually such universality is inevitable in real applications, there was an assumption that a query language is a “sublanguage” that is to be embedded in a universal programming language. A “sub-language” determines the access to a database only. The rest of the entire application has to be programmed in a typical programming language. This assumption requires joining a query language with a programming language in such a way that:

- i) Queries can be used inside programs;
- ii) Queries can be parameterized through values calculated within programs;
- iii) Results of queries are to be passed to programs.

Difference between language concepts cause significant technical difficulties in accomplishing this kind of connection. A lot of programmers and computer professionals were also disappointed by the technical, aesthetic and conceptual degradation of the programming environment. This degradation is commonly referred to as *impedance mismatch*. This term denotes a bunch of disadvantageous features that are implied by the eclectic mix of a query language (in particular SQL) with a programming language (such as C, C++ or Java) (Tanenbaum et al, 1989). Below we list and comment these features.

- **Syntax:** In the same code the programmer must use two programming styles and must follow two different grammars. Similar concepts are denoted differently (for instance,

strings in C are written within "...", in SQL – '...') and different concepts are denoted similarly (for instance, in C = denotes an assignment, in SQL – a comparison).

- **Typing:** Types and denotations of types assumed by query and programming languages differ, as a rule. This concerns atomic types such as integer, real, boolean, etc. Representation of atomic types in programming languages and in databases can be significantly different, even if the types are denoted by the same keyword, e.g. *integer*. A lossless conversion between such types could be impossible and might imply some performance overhead. This also concerns complex types, such as tables (a basic data type constructor in SQL, absent in programming languages). Popular programming languages introduce static (compile time) type checking, which is impossible e.g. in SQL (because query languages are based on dynamic rather than static binding).
- **Semantics and language paradigms:** The concept of semantics of both languages is totally different. Query languages are based on the declarative style (*what* is to be retrieved rather than *how*), while programming languages are based on the imperative style (a sequence of commands to a machine, which accomplishes *what*).
- **Abstraction level:** Query languages free the programmers from majority of the details concerning data organization and implementation, for instance, organization of collections, presence or absence indices, etc. In programming languages these details usually are explicit (although may be covered by some libraries).
- **Binding phases and mechanisms:** Query languages are based on late (run-time) binding of all the names that occur in queries, while programming languages are based on early (compile and linking time) binding. Thus, from the point of view of a program, queries are simply strings of characters.

- Name spaces and scope rules:** Queries do not see names occurring in programs and v/v. Because eventually there must be some intersection of these name spaces (e.g. program variables must parameterize queries) additional facilities, with own syntax, semantics and pragmatics, are required. These facilities are the burden for the size and legibility of the program code. Moreover, in programming languages name scopes are organized hierarchically and processed by special rules based on stacks. These rules are ignored by a query language. This leads e.g. to problems with recursive procedure calls (a well-known example concerns SQL cursors that severely reduce the possibility of recursion). Another disadvantage of separated name spaces concerns automatic refactoring of programs, which cannot be performed on queries.
- Collections:** Databases store collections (e.g. tables) which are processed by queries. In programming languages collections are absent or severely limited. Hence collections returned by queries have no direct counterparts in a programming language and must be processed by special constructs with own syntax and semantics.
- Null values:** Databases and their query languages have specialized features for storing and processing null values. Such features are absent in programming languages, thus some substitutes must be introduced. For instance, if some value in a relational database can be null, mapping it to a programming language requires two variables: one for storing information about null and another one for storing the value.
- Iteration facilities:** In query languages iterations are accomplished as macroscopic query operators, such as selection, projection, join, product, sum, intersect, etc. In programming languages iterations are coded explicitly as loops (*for*, *while*, *repeat*, etc.). Processing results of queries in a programming language requires special mechanisms, such as cursors and iterators.

- **Persistence:** Query languages address persistent data (stored on a disc or another long-term memory), while programming languages process only data stored in a volatile operating memory. Joining query and programming languages requires special facilities for parameterizing queries by volatile variables and transmission of persistent data to volatile memory and v/v.
- **Queries and expressions:** There is some competence mismatch between queries and programming language expressions. Some queries look as expressions and v/v, but there is strong syntactic subdivision of them, which can be poorly understood by the programmer. For instance, in some query languages $2+2$ is a query, but it is also an expression of a programming language. A query cannot be a parameter to a procedure, but an expression can. There could be other syntactic constraints, which cause a lot of chaos in the entire programming environment.
- **References:** If a query is to be used for updating, inserting or deleting, it must return references to stored data (i.e. data identifiers rather than values). According to the official semantics of the relational model, queries return tables of values, with no references. For updating constructs defined in a programming language such semantics is inconsistent; actually, it means that queries cannot be used for updating or require a special mode of execution and/or a special constructs in a programming language.
- **Refactoring:** decisions concerning new names used for data structures cannot be automatically propagated to queries, because – from the point of view of a programming language – queries are strings, sometimes not explicitly seen from the source program. Hence refactoring of queries should be done manually, with a lot of effort and possibilities of inconsistencies.

The consequences of impedance mismatch concerns not only aesthetics and ergonomics of the programming environment. Impedance mismatch implies an additional programming layer, with own syntax, semantics and pragmatics. This layer causes that learning of the language takes more time, programming is more error prone, programs are unnecessarily longer and less legible. This layer may also cause worse performance and maintainability of applications. If queries are strings then there is no explicit support for creating reusable query components. None of the reusability features of the programming language (functions, methods, and polymorphism) are available to support reuse. Passing parameters to queries written as strings (c.f. the ODMG standard) is awkward and error-prone. Queries, as strings embedded in a program, are also more prone to injection attack (Tanenbaum et al, 1989).

Some authors suggest that the source of impedance mismatch is in incompatibility of data models, in particular, access to relational databases is accomplished from an object-oriented programming environment (such as object-oriented DBMS). Such a suggestion presents the ODMG standard, with the conclusion that in this standard impedance mismatch no more holds. Unfortunately it is only partly true. Indeed, the mismatch is inevitable in situation of big differences between data models, in particular, between a relational system and an object-oriented programming language. However, even if both models are claimed to be “object-oriented”, the impedance mismatch still persists. There are significant differences between various object-oriented models. Actually, there are as many object models as different object-oriented artifacts and proposals; no standard object model exists. Differences between the object models of Smalltalk, CORBA, UML, C++, SQL-99, ODMG, Java, C# etc. are fundamental. Moreover, even if the model is exactly the same and would be the subject of some precise standard, some impedance mismatch can persist due to e.g. differences in binding

phases. These issues we discuss in the next subsection devoted to the relationship between impedance mismatch and data independence.

To avoid the impedance mismatch, a language should be *integrated*, where queries are smoothly connected with programming constructs and abstractions. This tendency is seen in such products as PL/SQL, T-SQL and standards SQL-99 and SQL-2003. The SBQL language is designed according to this tendency, where SBQL queries are integrated with updating constructs and programming abstractions. The typing system and a strong type checker is the same for the queries, updating constructs, procedures, functions, methods, parameters, views, transactions, etc. This significantly distinguishes SBQL from other query languages. In the following we discuss some pros and cons of the idea of integrated database programming languages.

2.2.9 Progressive development of Database Models

The data models are analyzed from the classical models to the proposed data model before designing them in the next chapter. First, the evolution of data models from the earliest time of computation to today is evaluated. In figure 2.7 the database model started from file systems and moved to Bachman diagrams.

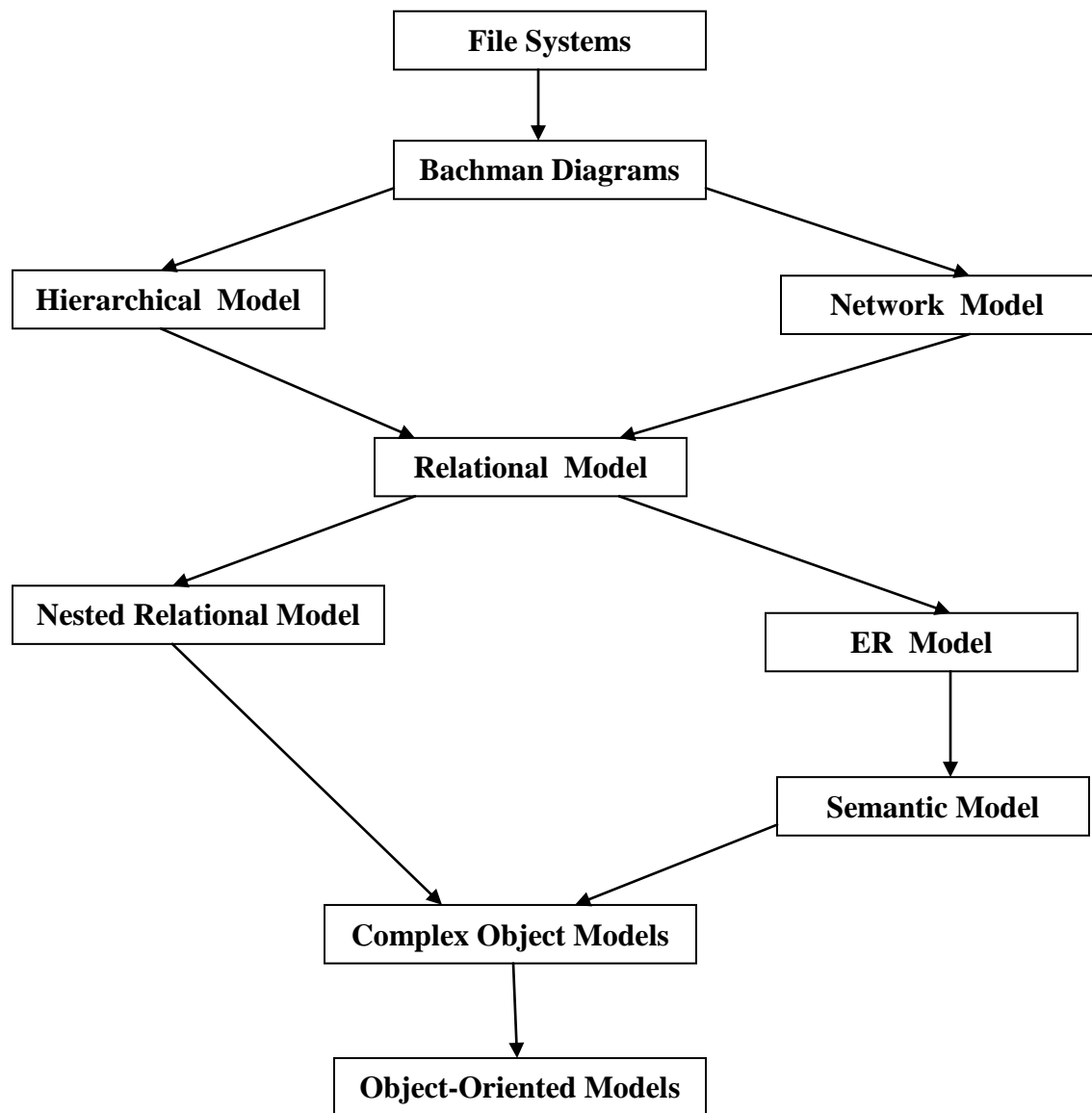


Figure. 2.7: Evolution of Data Models (Adopted from Gottfried, 1992)

The Bachman diagram then evolved to hierarchical model and network models at the same period when certain challenges were identified in their use and operational method of manipulating and storing databases. The two models then converged to form the relational database model in an effort to solve the challenges. The Entity Relational (ER) model and Nested relational model both were then developed from the classical relational model. The ER

model then produced the semantic models. The semantic models combined with the nested relational model gave birth to complex object models. In an effort at solving the complexity inherent in the complex object model the concept of object-oriented paradigm was introduced resulting in evolution of the object-oriented models in databases. The concept we have put together here is illustrated in Figure 2.7 which is an enhancement on a previous figure drawn by Gottfried Vossen (1992).

2.3 Summary of Review and Knowledge Gap

In this section, different related literature have been reviewed on databases and object-oriented process and data communication as it relates to simple and complex data storage and retrieval. It is clear from the review that Edgar (1960) worked on relational data model and design which was the foundation of relational database management system. RDBMS needed external language-SQL for query handling and have problem of Scalability. There is also the issue of impedance mismatch with the model. They model is only capable of handling only Structured data. Other models include the one worked by Stonebreaker (1994) on Object-relational databases model design which was the foundation of Object-Relational Databases a hybrid of object database and relational database. This model introduced complex data types and object features but to make it relational the complex data type was separated from the simple data type, these created Scaling problem and schema definition problem. In other to solve the problem of relational database structured type only and scalability challenge Adam (2014) introduced the document-oriented databases in form of NoSQL with the most popular implementation being MongoDB, this model actually allowed unstructured data to be handled in the database it was also scalable compared to relational databases but it does not follow a predefined schema and it also separates Complex data from simple data and uses hierarchical storage which occupies large data space. The Object Database management system group, an

organization formed to further develop object databases, ODBMS.org (2013) worked on Object-oriented databases model design with the benefits of easy query, less specification errors and relative objects independents. However their model proposal does not have the data properties bundled with the data methods and there is no communication between data and program process.

CHAPTER THREE

METHODOLOGY AND SYSTEMS ANALYSIS

3.1 Methodology Adapted

In the development of the system the methodology that will be used is the Object-Oriented System Analysis and Design methodology (OOADM). The choice is obvious due to the fact that the system developed is designed to handle databases and program development using the principles of Object orientation such as inheritance, polymorphism, encapsulation and data abstraction using class and object interfaces.

The analysis of the system in this chapter will be carried out using the Object-oriented Analysis and Design methodology (OOADM). This methodology provides a detail insight on the basic foundation of the process and link to the system being analyzed as well as on the necessary guiding tools and methods that will be adapted in expressing the logic of the processes that is used in the developing the system components required to design the system in this research.

Object-oriented system analysis and design methodology (OOADM) was selected due largely to the nature of the research work which is object-oriented based both in the database manipulation and in the class object communication which is the core of our reserch. The object-oriented nature of the system demands for an object methodology which Object-oriented Analysis and Design Methodology (OOADM) offers. The class nature of the system, studied in the analysis and design, which is made up of properties (data) and the methods that manipulate them makes the OOADM to be inevitable.

3.1.1 Object-Oriented Analysis and Design Methodology (OOADM)

Object Oriented Paradigm (OOP) has been touted as the next great advance in software engineering. It promises to reduce development time, reduce the time and resources required

to maintain existing applications, increase code reuse, and provide a competitive advantage to organizations that use it. While the potential benefits and advantages of OOP are real, excessive hype has led to unrealistic expectations among executives and managers. Even software developers often miss the subtle but profound differences between OOP and classic software development.

3.1.2 Expected Benefits of OOP

Many benefits are cited for OOP, often to an unrealistic degree. Some of these potential benefits are (Adhikari, 1995; Taylor, 1995).

Faster Development: OOP has long been flaunted as leading to faster development. Many of the claims of potentially reduced development time are correct in principle, if a bit overstated.

Reuse of Previous work: This is the benefit cited most commonly in literature, particularly in business periodicals. OOP produces software modules that can be plugged into one another, which allows creation of new programs. However, such reuse does not come easily. It takes planning and investment.

Increased Quality: Increases in quality are largely a by-product of this program reuse. If 90% of a new application consists of proven, existing components, then only the remaining 10% of the code has to be tested from scratch. That observation implies an order-of-magnitude reduction in defects.

Modular Architecture: Object-oriented systems have a natural structure for modular design: objects, subsystems, framework, and so on. Thus, OOP systems are easier to modify. OOP systems can be altered in fundamental ways without ever breaking up since changes are neatly encapsulated. However, nothing in OOP guarantees or requires that the code produced will be modular. The same level of care in design and implementation is required to produce a modular structure in OOP, as it is for any form of software development.

Client/Server Applications: By their very nature, client/server applications involve transmission of messages back and forth over a network, and the object-message paradigm of OOP meshes well with the physical and conceptual architecture of client/server applications.

Better Mapping to the Problem Domain: This is a clear winner for OOP, particularly when the project maps to the real world. Whether objects represent customers, machinery, banks, sensors or pieces of paper, they can provide a clean, self-contained implication which fits naturally into human thought processes.

Therefore, OOP offers significant benefits in many domains, but those benefits must be considered realistically. There are many pitfalls that await those who venture into OOP development. These pitfalls threaten to undermine the acceptance and use of object-oriented development before its promise can be achieved. Due to the excitement surrounding OOP, expectations are high and delays and failures, when they come, will have a greater negative impact.

3.2 System Analysis

Analysis is the decomposition of the activities that is required in the development of a system in order to discover the component part required for the design of the system. This will involve the dissection of the component parts necessary for an object in database to interact with object in code that is used in manipulating it. We also need to find out why a particular ordering, need to be followed in the process of building the application needed for the testing of the system developed in the research. There is also the need to specify certain steps and processes required to build the abstractions needed in the proposed OODBMS that is used in the implementation of the system being developed in this research. This processes could be represented graphically or using a sound mathematical expression in the next chapter.

3.2.1 Analysis of the Present System

In the present system most of the data used in handling application are stored in structured format in the database management system (DBMS) as well as in the data warehouse for structured databases. The present system also use the object-relational and document-oriented databases in handling the data but often the format of the data is different from the application when object-oriented systems whose data format are not predefined are to be used in the development of the system. The present system format had never generated concern all this while due to the nature of the most common programming paradigm since the early eighties which was largely structured in nature. Structured programming was very successful in the development of systems and in the process of communicating between system code and data used in manipulating the system usually stored in the databases as well as in data warehouses. This trend continues to make Structured Query Languages (SQL) the defacto standard in the development of systems.

However, continuous growth in data content and associated changes in modern programming language from structured (procedural) to object-oriented has introduced new problems. In addition, modern use and heavy reliance on object data such as video, audio, images and other serializable object data on the internet and in data warehouses leaves developers with much challenge on how to develop their systems to be able to communicate with the stored data easily. The use of other complex data that can only be defined by the user application also introduces the need to empower developers to be able to define, use and store their data via an object-oriented database system.

3.2.1.1 Problems of the Present System

It is important for us to note that within the recent past the present system is efficient and for technology experts and academics still thinking along the recent past there may be no need to condemn the present system. But that does not remove the challenge of complexity of information representation which is the major challenge of the present system. Simple data types are largely “primitive” (singly definable types) such as whole numbers (integer types), characters, dates, strings and even a combination of the simple types in records and lists. But the reality today is that three dimensional representations such as spatial data which are common today due to space and satellite technology improvements are increasingly being used in organizational databases. Voice data, video and even pictures are also increasingly used and they found application in many databases. Moreover there are other *data* and their associated *operations* that users need to store in databases but presently data bundling with operation are not yet used in present databases. These complex data sets which often are user defined are represented in form of classes which has proven successful in programming languages such as Java, C++ and other languages. Today any programming language that is not object-oriented seems not to be accepted for serious software engineering development.

Apart from data representation and storage, communication of databases with object-oriented program code leaves lots of operations implementations to the programmer, allowing for ineffective communication between a procedural database and object-oriented programming code. It is strongly believed that with transformed object-oriented databases communication between objects in code and objects in databases will improve communication tremendously.

It is also clear that data warehouses are even worse since some of them are still storing data in files while others have barely embraced structured storage. This is understandable since most data warehouses have been in existence for long time and updating them seem very expensive since it is often a repository of data where all the data are never in use at the same time. But we look at improvement so that new data warehouses can benefit from the storage improvement and data mining benefit that object-orientation of data storages will offer.

The resultant challenge of the present system could still be presented in more detail as:

- i) **Lack of Component Reuse:** Structured systems are difficult if not impossible to reuse. Reusability on the other hand is established features in object-oriented system. This makes modern system to have reusable code and data components that are not reusable without direct replication.
- ii) **Problem of Inheritance:** Structured system does not allow the sharing of object data component by several users of sub-data components as it is in object-oriented classes. This makes communication to be slow between the system and its data components. When the data warehouse is large the system either crashes or is extremely slow. This causes inefficiency in modern system development.
- iii) **Openness Challenges:** The systems are normally closed systems, which mean they are designed around proprietary concerns instead of standard protocols that allow data warehouse and software from different systems to be combined and communicate freely.
- iv) **Low Scalability:** In this system, several processes and actions that are required by users for unique circumstances can be easily developed as classes or new data types and incorporated in the system making the system scalable. But when the system is

structured classes cannot be built losing the benefit of scalability. If the system is object-oriented, the new classes developed can have methods that can override previously defined methods whose implementation need adjustment to meet new user's specific requirement. The tendency for increased resources and new data types and classes are high in computing and new demands on more complex data. With the new system complex data can be accommodated and incorporated by developers instead of the DBMS vendors.

- v) **Problem of Lack of Encapsulation:** The data in the ware house can properly be hidden from the communicating classes and processes within the database. This have a way of providing extra data security within the object database which combines with that of the object-oriented code to provide better secured system. In the present system this is lacking, system data is structured and relational and are not bundled with the processes that manipulate them.
- vi) **Future Explosion:** Data warehouses are not going to reduce rather we expect that data will increase. It might increase even in a rate that explosion will occur. Structure databases and data communication systems may find it difficult if not impossible to handle such challenges. Hence the need to develop a system that can handle the communication in an object-to-object manner.

3.2.1.2 Architecture of the Present System

In the present system, establishment of communication is expected between the programming language used in the manipulation of the OQL and the DBMS that uses the OQL. Certain principles need to be applied to both object-oriented programming languages and object DBMSs. However, the application or the principle varies somewhat between programming languages and DBMSs.

Programming languages have used object-oriented principles for many years now. Simula, a language developed in the 1960s, is the first reported object-oriented Programming language. As its name implies, SIMULA was originally developed as simulation Programming language. Objects and messages are natural in modeling simulations. Smalltalk, a language developed during the 1970s at the Xerox Palo Alto Research Center, was the first popular object-oriented Programming language. Smalltalk originally emphasized object for graphical users interface, inheritance among classes for windows and controls is a natural fit. Since the development of Smalltalk, there have been many other object-oriented programming language, Java and C++ are the two most widely used object-oriented Programming language today.

Object-oriented Programming languages emphasize software maintenance and code reusability to support software maintenance, encapsulation is strictly enforced. The internal details (variables and method implementation) cannot be referred. In addition, some languages support additional kinds of inheritance to share code. Reused of code can extend to collections of classes, classes with data type parameter and redefined code in a subclass.

Objects DBMSs are more recent than object-oriented Programming languages. Researched and development of object DBMS began in the years 1980s, by the early 1990s, there were number of commercial object DBMSs. In addition, there was consideration works on extending relational DBMSs with new object features. Because early object DBMS began as outgrowth of object-oriented Programming languages, query language support did not exist. Instead early object DBMS provided support for persistent data that last longer than the execution of the program. Languages object DBMSs were designed to support application with large amount of complex data. Most object DBMSs now support nonprocedural data access. Some of the object-oriented Programming languages principle are relaxed to

accommodate this emphasize. Encapsulation usually is relaxed so that an objects data can be referred in a query. Inheritance mechanisms usually are simpler because it is assumed that most coding is through a query language not a procedural language. In addition, object DBMSs need provide query optimization and transaction processing capabilities.

3.2.1.3 Architectures for Object Database Management

Although adding object-oriented Programming language features to a DBMS is a good ideal, there is no wide-spread agreement about what features to add and how features should be added. Some approached provide small extension that leaves most of the object-oriented Programming languages outside the DBMSs. Other approached involved a complete rewrite of the DBMSs to accommodate objects. The marketplace probably will support a variety of approaches because of the different requirement among customers. This section describes several object based management architectures along with their strengths and weakness.

3.2.1.4 Large objects and external Software Architecture

This earliest approach to add object to relational databases was to use large object with external software. Complex data are stored in field as a binary or text large object. For example, an image is stored in a field using the BLOB (binary large object) data type.

As depicted in Figure 3.1 large object usually are stored separately from other data in a table. A query can be retrieved but not display large objects. Software external to the DBMS display and manipulates large objects which include java applets and web browsers plug-ins. The large objects approach is simple to implement and universal. Only small changes to DBMSs are required. All kind of complex data can be stored; in addition, a large market for third-party software may be available for prominent kinds of complex data.

For example, many third-party tools have been implemented for popular image formats.

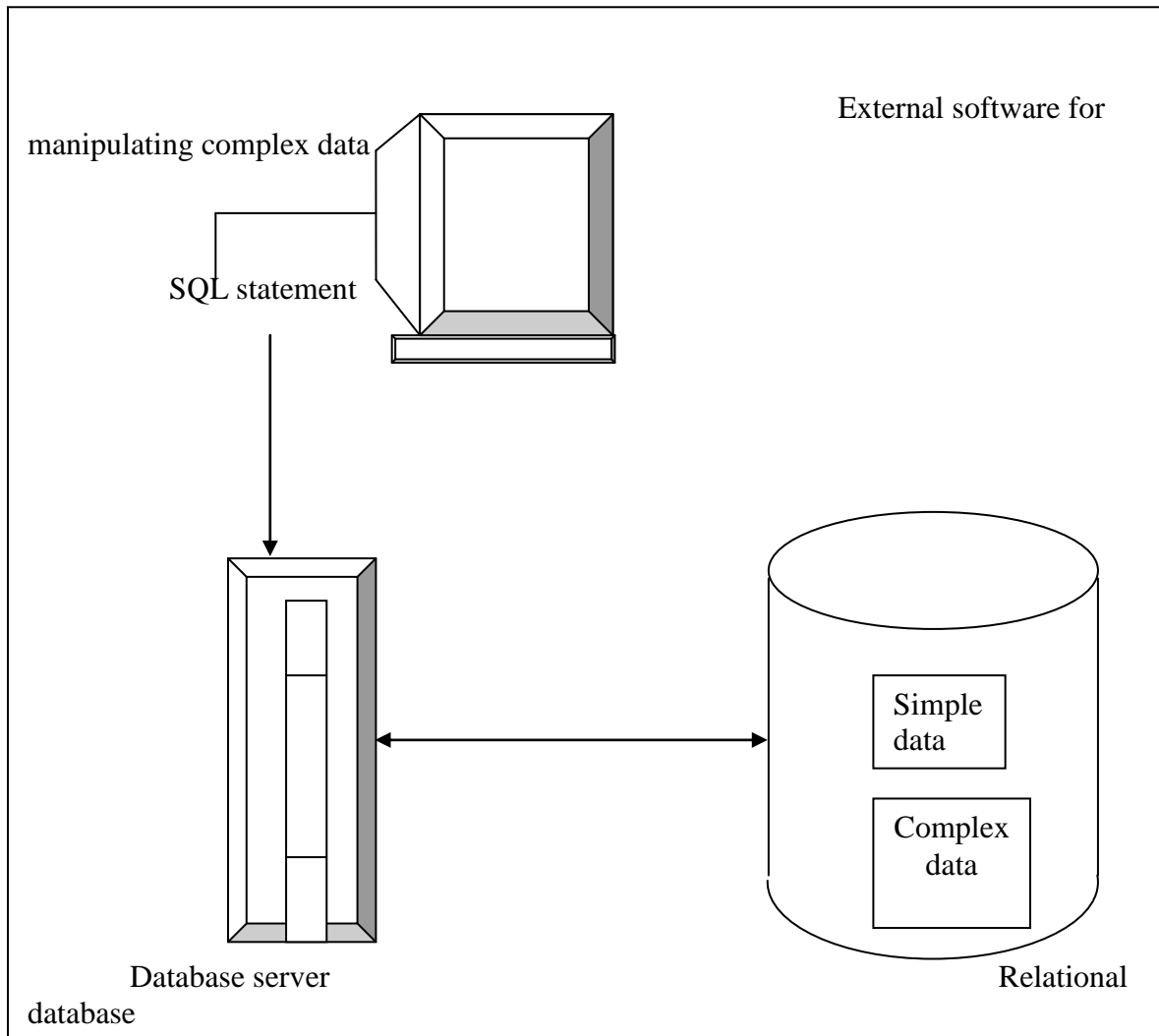


Figure 3.1: Separated Large Object data Architecture
(Adapted from Michael, 2006)

Despite some advantages, the large object approach suffers from serious performance drawbacks. Because the DBMSs do not understand the structure and the operations of the complex database, no indexes can be used to select records using characteristics of large objects. Because large objects are stores separately from other data, additional disk accesses may be necessary in other for the large objects not coincide with the order of other table data.

3.2.1.5 Specialized Media Server Architecture

In the specialized media server approach, complex data resided outside of the databases, as depicted in figure 3.2 a separate server may be dedicated to manipulating a single kind of complex data such as video and images. Programmers used an application programming interface (API) to access complex data through a media server. Specialized media servers provide better performance than the large object approach but performance for specific kinds of complex data. Because an API is provided rather than a query language, the range of operations may be limited through for example, a video server may support fast streaming of video but not searching by content.

When combining simple and complex data, the specialized server approach may perform poorly. A query optimizer cannot jointly optimize retrieval of simple and complex data because the DBMS is not aware of complex data. In addition, a media server may not provide indexing techniques for search conditions. Transaction processing is limited to simple data because specialized media servers do not typically support transaction processing.

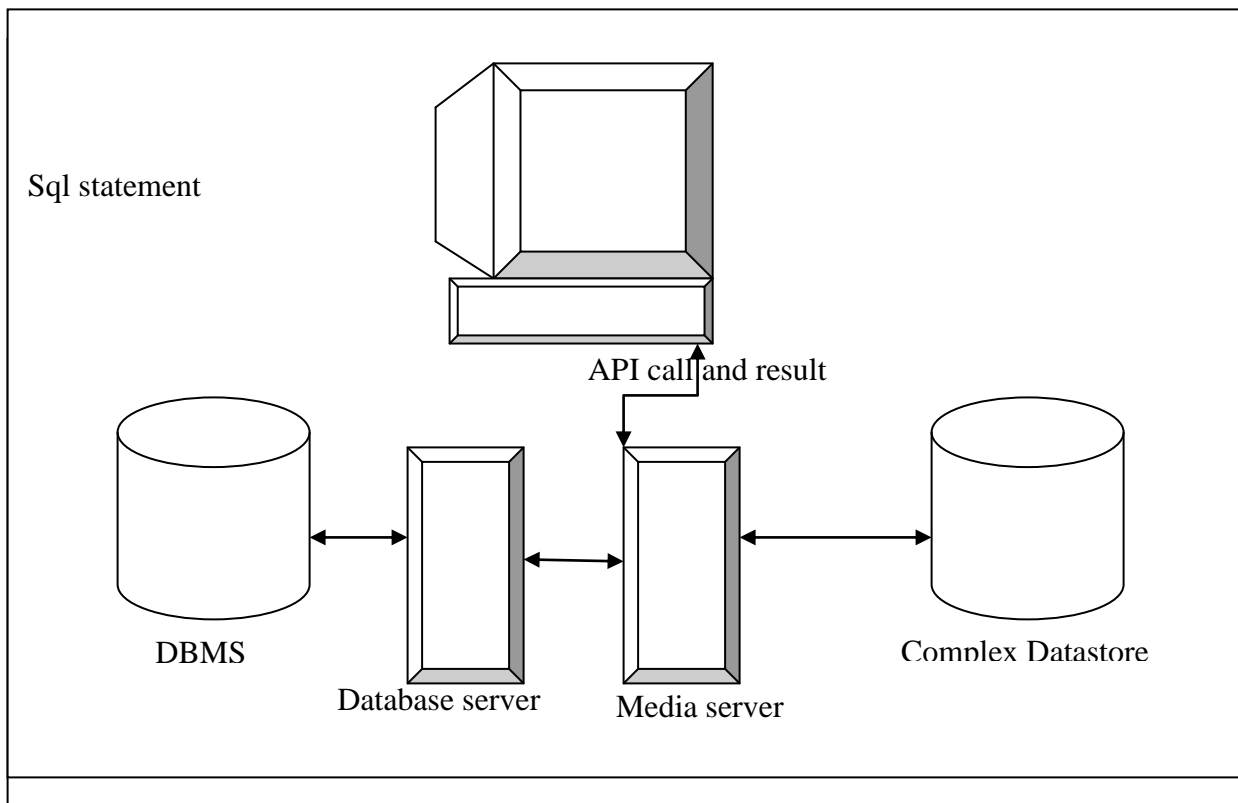


Figure 3.2: Special Media Server Architecture (Adapted from Michael, 2006)

This Architecture seem noble in approach but it carefully avoided the DBMS and the structured query language (SQL) manipulations. With the great increase in some systems such as YouTube and other multimedia news system online there is increased need to offer direct communication in form of query to the objects. This architecture is speed efficient but seriously lacking in transaction handling. The API manipulating the object have no direct communication ability and hence cannot be able to handle transactions which are purely based on query emanating from the user and processed by the DBMS. Since the DBMS does not communicate directly with the objects the transaction cannot be possible.

3.2.1.6 Object Relational DBMS for User Defined Types

The first three approaches involve little or no change to the DBMS but provide only limited query and optimization capabilities. With large changes to the DBMS, more query and optimization capabilities are possible. To provide additional query capabilities; object relational DBMSs support user-defined type and functions. Almost any kind of complex data can be added as a user-defined type. Image data, spatial data, time series, and video are just a few of the possible data types. For each user-defined type, a collection of functions can be defined. These functions can be used in SQL statements, not just in programming language code. Inheritance and polymorphism apply to the user-defined data types. To improve performance, multidimensional B-trees can be provided for accessing spatial data. User-defined types and functions are implemented in SQL3 and Odb2 some of the emerging standard for object relational DBMSs. These are keys for improvement needed in the development of classes and methods in the proposed system in this research. The architecture of object relational databases is therefore adapted as the existing system from which improvement can be done to fixed the challenges due to lack of action integration with the uch properties of the object. This architecture is illustrated in Figure 3.3. The table query processor uses table-driven architecture code for a user-defined type. The parser decomposes references to expressions involving a user-defined type and functions. The optimizer searches for storages structures for expressions involving a user-defined types and function. The relational kernel comprises transaction processing, storing management and buffer management. It provides the engines used by the object query processor. Little or no changes are necessary to the relational kernel. Object relational DBMS provide good integration of complex data but related. The integration of simple and complex data involves considerable

changes to the display manager, and optimizer. However, the base of code in the kernel remains unchanged.

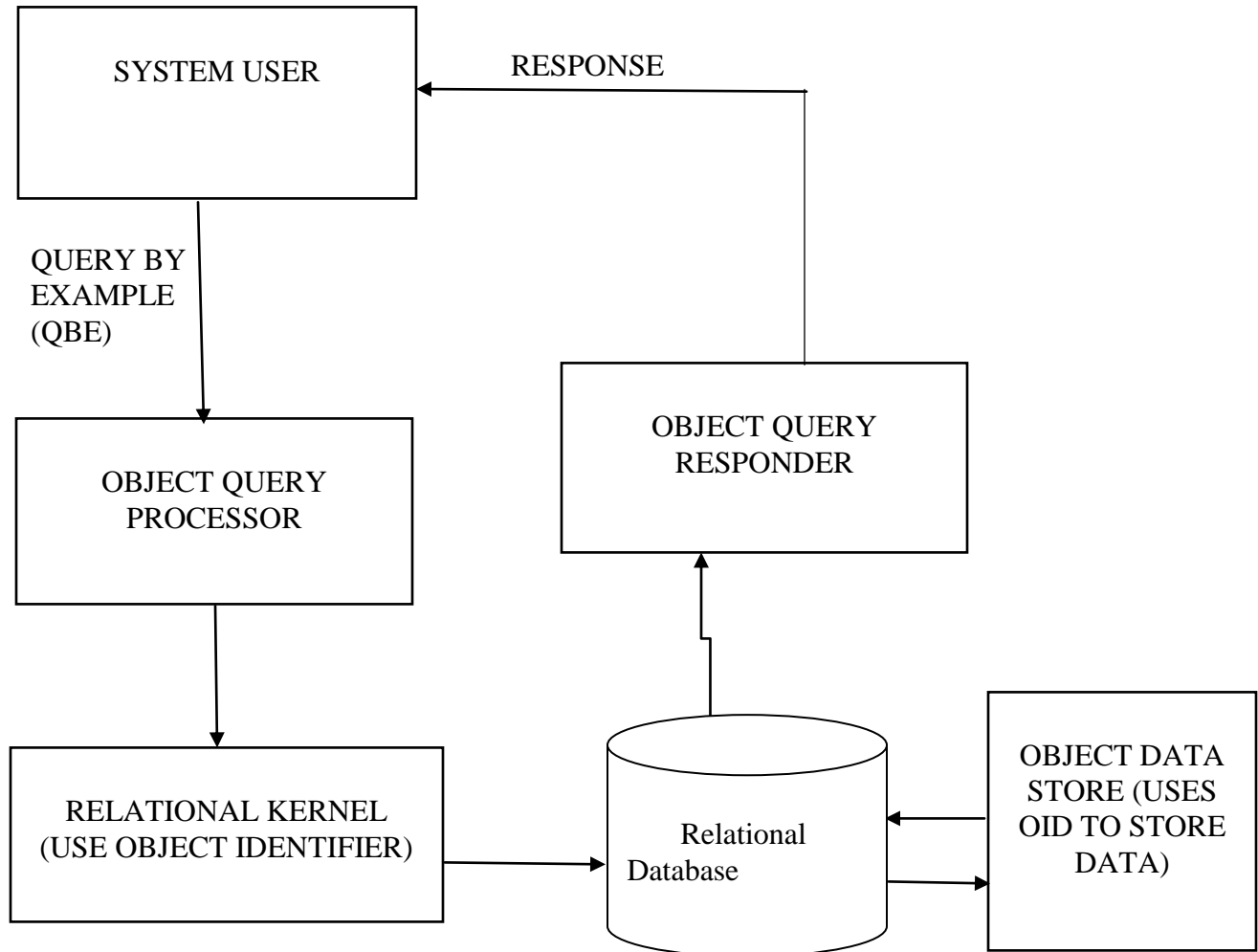


Figure 3.3 Architecture for the Existing System (Object relational DBMSs)

As illustrated in figure 3.3 the database communicates with the relational kernel which in turn communicates with the object query processor in order to handle query processing and result handling. Reliability can be compromised in the implementations of user-defined types, function and storage structure that are not direct relations. DBMSs vendors, third-party vendors, and in house developers can provide user-defined types and function, which can be complex and difficult to implement. Implementation errors can affect the integrity of both simple and complex data. In addition third-party data types can be subject to viruses or spywares making the Object-relational database unsafe for use in financial, governmental and critical database domains like medical imaging where the objects are in abundance.

3.3 Analysis of the Proposed System

Object-oriented database and class object communication System consists of multiple complex software components that interact based on certain object-oriented paradigm standards.

The new system intends to solve the problems of the old system by providing improvement on the architecture of object database models and class object communication to make it possible for the system to handle object databases as well as interact directly with programming language class objects in a manner that will simplify the user experience in using object-oriented database management system.

In the new system, the Object database need to be created in such a way that it directly communicates with object-oriented classes irrespective of the type of query or request made on the system whether they are related or not. This needs to be done without translating from higher-level language such as SQL, in a compact way for data manipulation operations. The

optimizer through the process of optimization determines which indices should be used to execute a query and in which order the operations of a query should be executed. To this end, the optimizer enumerates alternative plan to achieve the task once the query has been presented to the system for analysis and processing.

This will result to a possible rapprochement of SQL3, ODMG/OQL and other object database proposals. The idea is that DBMS products continue to maintain their relational roots for managing traditional data, while at the same time addressing some of the shortcomings of the relational model by providing some of the benefits offered by the object-oriented paradigm (and its facilities for handling non-traditional and complex data).

Proposed features: In the proposed system the features proposed are the combination of complex and simple data. This will result to new data model which include new and extended data types including abstract data types, and classes as well as multiple null states, support for objects and object identity, encapsulation, inheritance and triggers. They also include primitive types like INTEGER, DECIMAL, CHARACTER, BIT, TIME, DATE and many other variations of these. All of the data types specified in the existing SQL-92 standard could be maintained, though some may be modified and extended. For instance, both the CHARACTER and BIT data types could be extended to include the concept of a large object. A large object is used to store an entire object or piece of data in a single large field. A CHARACTER LARGE OBJECT is used to store a large piece of character data without the use of fields while a BINARY LARGE OBJECT can be used to facilitate the use of multimedia objects within the database. The key to a large object data type is that the database is not concerned with the data items internal structure. Applications or implementations that support these data types impose severe restrictions on how these large objects may be

referenced in SQL definitions and statements such as not allowing them to participate as a primary key.

Two additional data types not previously facilitated by the standard have also been proposed. These two new data types are BOOLEAN and ENUMERATED. BOOLEAN data types can only take on the values of 'True', 'False' or 'Unknown'. 'Unknown' is proposed as an extension due to the roll of fuzzy logic in the development of intelligent systems that can control the objects behaviour. ENUMERATED data types permit you to "define a domain whose values are restricted to a small set of values". The command CREATE DOMAIN colors (blue, red, yellow) defines a domain called colors and restricts values inserted into the column accepting the ENUMERATED data type to one of the three values specified by the domain.

The second group of data types that is important is the abstract data type, a data type not previously supported. As mentioned earlier, abstract data types are a feature of the object-oriented paradigm. The ADT data type is used by to facilitate the incorporation of objects within the SQL structure. "Abstract data types allow users to define new structures for their own data (Garvey,2010). New data types may be constructed by defining abstract data types in terms of existing predefined data types or previously generated abstract data types. These data types therefore have no logical representation. Abstract data types also support the object-oriented concepts of encapsulation and subtyping (inheritance). In the new proposal a structure for direct communication with classes and objects inside the manipulating code of the programming language used in the implementation of the databases is required. This will allow request and actions to response from the object-database to be built in code as a class that can be instantiated, and inherited by other classes for efficient implementation of system. This will form a meta type that have direct communication link to object at the two ends of

the system both at the object-database end and the objects on manipulating code end. This will surely lead to an improvement to Object-Query Language used in manipulating the database in the system.

3.3.1 Advantages of the Proposed System

The following advantages would be derived from the proposed system:

1. **Inheritance:** The system allows the sharing of object data component by several users of sub-data components as it is in object-oriented classes.
2. **Openness:** The systems are normally open systems, which mean they are designed around standard protocols that allow data warehouse and software from different systems to be combined and communicate freely.
3. **Scalability:** In this system, several processes and actions that are required by users for unique circumstances can be easily developed as classes or new data types and incorporated in the system making the system scalable. The new classes developed can also have methods that can override previously defined methods whose implementation need adjustment to meet new users' specific requirement. The tendency for increased resources and new data types and classes are high in computing and new demands on more complex data in the system can be accommodated and incorporated by developers instead of the DBMS vendors.
4. **Encapsulation:** The data in the ware house can properly be hidden from the communicating classes and processes within the database. This have a way of providing extra data security within the object database which combines with that of the object-oriented code to provide better secured system.
5. **Polymorphism:** Polymorphism supports fewer, more reusable methods. Because a method can have multiple implementations, the number of method names is reduced. A user

needs to know only the method name and interface, not the appropriate implementation to use the system. For instance, a user needs to know only that the area method applies to polygons, not the appropriate implementation for a rectangle. The DBMS assumes responsibility for finding the appropriate implementation. Polymorphism also supports incremental modification of code. Incoming another implementation of a method for a subclass much of the code for the method's implementation in the parent class often can be used. For example, to code the redefinition of the equals method in the color point class, another equality condition [for the color of a point] should be added to the conditions to test the x and y coordinates, the information required for that can be stored in the object-database for easy communication with the object code.

3.3.2 Possible Difficulties in achieving the Proposed System

1. **Complexity:** The systems are more complex than structured Systems. This makes it more difficult to understand their emergent properties and to handle their communication challenges by non-expert systems users.
2. **Manageability:** More effort is required to manage and maintain the System in Operation in other to gain all the benefits.
3. **Unpredictability:** The object based systems are unpredictable in their response to classical systems, as many data are already store in the relational format within many data warehouses.
4. **High Operational Cost:** In other to make the system effective there may be the need to convert relational data to the object-oriented structures which could be expensive based on the volume of stored data to be converted.
5. **Type checking challenge:** Along with binding, the DBMS ensures that objects and methods are compatible. This function is known as type checking. Strong type checking

ensures that there are no incompatibility errors in code. For example; strong type checking prevents computing an area for a point object because the area method applies to polygons not points. Because complex expressions can involve many methods and objects, strong type checking is an important kind of consistency check in object-oriented computing.

3.3.3 Architecture of the Proposed System

In the proposed architecture the object-oriented database combines the storage of both the simple data and the complex data and therefore does not need a middleware to circumvent problems with media servers. The functionality of the abstraction created by the media server is integrated into the object-oriented database server. This is done to avoid storage of object data in different locations as is done in object-relational databases. In this case clients send their query to the object-oriented database kernel through the Object-Oriented Query Processor (OOQP). This avoids the use of SQL statements and the preprocessing of SQL statement by the database server. The object-oriented query processed by the OOQP passes through the Object-Oriented Data Kernel that interacts directly with the Object-Oriented Database Store (OODS) where both the simple data and the complex data are stored. This enables the data to communicate with the system process on Object-to-Object level. Object-oriented database eliminates the need for the middleware and the media server. This architecture of the functionality of the Proposed Object-Oriented Database is illustrated in Figure3.4.

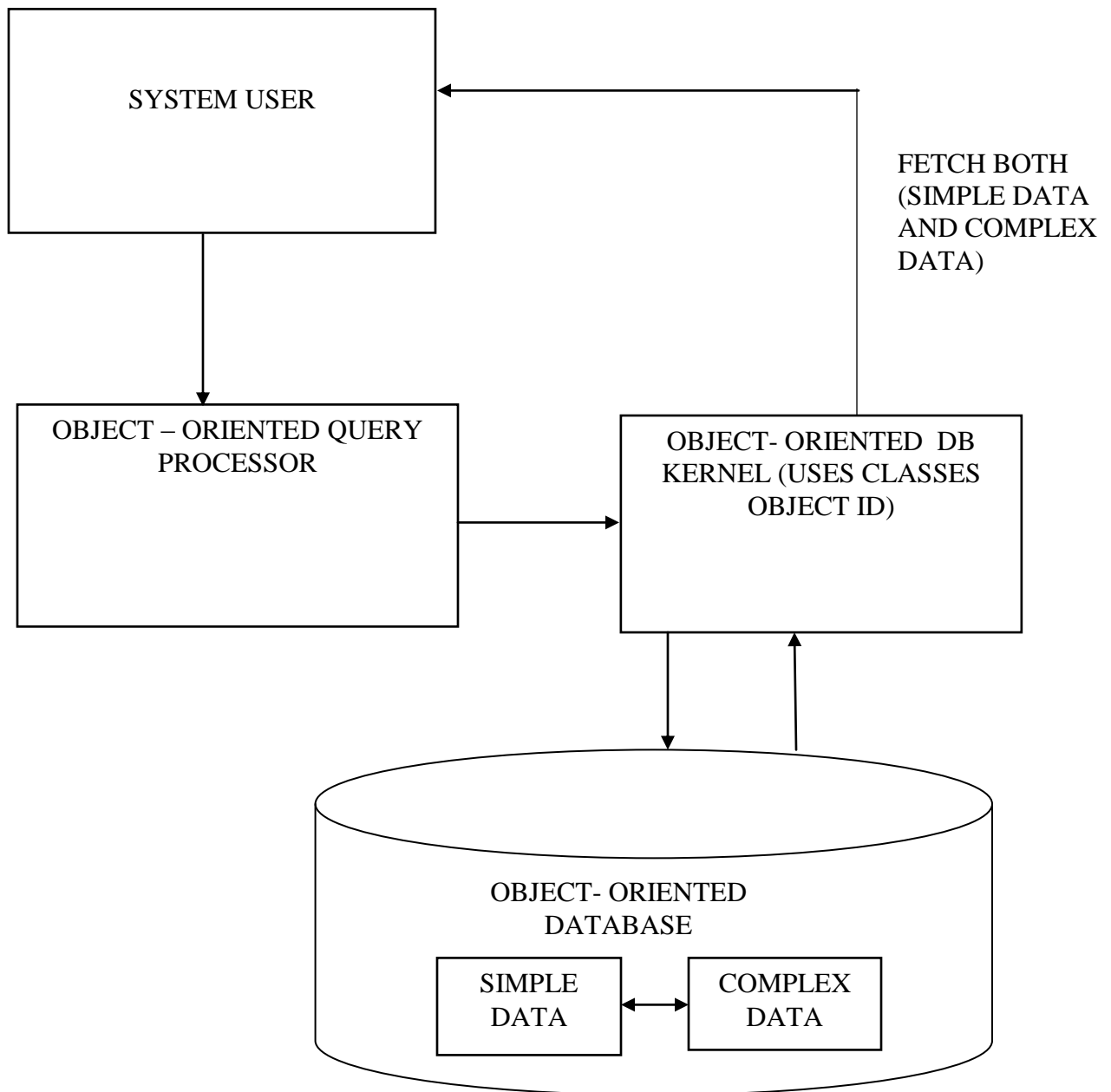


Figure 3.4: Architecture of the Proposed System

3.4.1 Object-oriented Database Management Systems (ODBMS)

Some experts have argued that more fundamental changes to a DBMS are needed to support object. Both the data model and the kernel must be changed to accommodate objects. This conviction has driven a number of start-up software companies to implement a new class of object DBMSs. The software companies have banded together to form the Object Database Management Group (ODMG). Despite some important advantages of object-oriented DBMS, they continue to occupy a market niche. Most of the products are used in application where ad hoc query occupy complex data in large software systems. Most of the object-oriented DBMSs began as extended programming languages with support for persistent object (i.e. objects that exist after a program terminates). Gradually, the object-oriented DBMSs have provided ad hoc query and efficient transaction support.

THE ODMG AND SQL standards groups have recognized the need for a unified standard. In the nearest future, unified standard may emerge that allows portability between SQL3 and ODMG-compliant DBMSs. Even when this unification occurs, considerable time may elapse before most products support the unified standard. Both ODMG and SQL3 are very large standards. Most DBMS vendors will support only a subset of the standards.

3.4.2 Object Database Architectures

The architectures of object databases fulfill a certain market niche. The simpler architectures [large objects and media servers] should become less popular over time. The struggle for dominance among the other three architectures may not be decided for some time. The object database middleware architecture will likely co-exist with the other architectures to handle complex data stored outside of a database.

3.4.3 User- Defined Types and Object Classes

One of the most fundamental extensions in SQL3 is the user –defined type for bundling data and procedures together. User-defined types support definition of new structured data types rather than refinement of the standard data types. It implements the user defined types in a way of specifying object classes in the system. A data type has a collection of properties and methods. The CREATE DOMAIN statement support refinements to standard data types. Example 3.1 shows the point type some DIFFERENCES are apparent, such as keywords [type versus CLASS] AND the order of specification [the field name before the data type]. The first path of a user-defined type contains the attributes definitions. The double hyphen denotes a comment. The keywords NOT FINAL mean that subtypes can be defined. For methods, the first parameter is implicit like the ODMG notation. For example, the distance method lists only one point parameter because the other point parameter is implicit. The body of methods is not shown in the CREATE TYPE statement but rather in the CREATE METHOD statement.

Example 3.1

Point type

```
CREATE TYPES point AS
```

```
    X FLOAT (15),          X coordinate
```

```
    Y FLOAT (15),          Y coordinates
```

```
    NOT FINAL
```

```
    METHOD Distance (P2 point) RETURN FLOAT (15),
```

```
        Compute the distance between 2 points
```

```
    METHOD Distance (P2 point) RETURN BOOLEAN
```

```
        Determines if 2point are equivalent;
```

Example 3.2

Color point type

```
CREATE TYPE color point UNDER Point AS  
    Color INTEGER  
  
    FINAL  
  
    METHOD Brighten (intensity INTEGER) RETURNS INTEGER,  
        Increase color intensity  
  
    OVERRIDING METHOD Equals (CP2 Color Point)  
        RETURNS BOOLEAN;
```

Determines if 2point are equivalent;

Besides the explicit methods listed in the CREATE TYPE statement, user-defined types have implicit methods that can be used in stored procedures as shown below:

1. Constructor method: creates an empty instance of the type. The constructor method has the same name as the data type. For example, the point () is the constructor method for the point type.
2. Observer methods: retrieve value from attributes, each observer method used the same name as its associated attributes. For example, X () is the observer method for the x attributes of the point types.
3. Mutator method: change value stored in attributes. Each mutator method uses the same name as its associated attributes with one parameter for the value.

For example

X (45.0) Changes the value of the X attribute.

SQL3 features the ARRAY collection type to support types with more than one value such as time series and geometric shapes. Examples 3.3 use an array to determine a polygon type with a maximum of 10 corners. The number following the array keyword indicates the maximum size of the array. Array can be storing any data types except others arrays and reference types presented above.

Example 3.3

Polygon types using an ARRAY

Creates types polygon AS

Corners point ARRAY (10), Array of corner points

Color INTEGER

NOT FINAL

METHOD Area () RETURNS FLOAT (15),

Compute the Area

METHOD Scales (Factor FLOAT (15)) RETURNS Polygon:

Computes new polygon scaled by factor

User-defined types support definition of new structured data types rather than refinement of the standard data types. User-defined functions can be used in expression in the SELECT, the WHERE, and the HAVING clauses.

Object database technology is driven by demands to integrate complex and simple data and software productivity problems due to type mismatches btw DBMSs and programming languages. Three principle of object-oriented computing encapsulation, inheritance, and polymorphism guide the development of object DBMSs. Encapsulations the hiding of implementation details, supports data independence

Because of the variety of ways to implement the object-oriented principles and the difficulty of implementation, a manner of object-oriented DBMSs architecture are postulated and commercially available. This chapter described some of this object-oriented architecture with their advantages and disadvantages with the hope of offering improved design. The first two doesnot fully support object-oriented principles as they involves simples schemes to stored large objects and invokes specialized server outside of a DBMSs. The last three architectures provide different paths to implement the object-oriented principles.

To provide a more concrete view of object database, this chapter presented object-oriented database definition and manipulation using SQL3, the emerging revision of SQL2, SQL3 support user-defined types to accommodate new kinds of complex data. Expression in query can reference columns based on user-defined types as well as the methods of the user-defined types. SQL3 support encapsulation, inheritance, and polymorphism guide the development of user-defined types. A separate inheritance capability for suitable families is also provided. Inheritance for sub-table families involve set inclusion relationships.

In the next chapter we will present an analysis and design for an improvement of the architecture to make it possible for the system to handle object databases as well as interact directly with programming language class objects in a manner that will simplify the user experience in using object database management system. This will enhanced the benefits of Object-oriented systems and reduce the challenges of usage.

3.4.4 Analysis of the Proposed Data Model

In object-oriented databases some of the core concepts include:

1. **Object modeling** of real world entity.
2. Database Structures and their behaviors are **Encapsulated**.
3. Manipulation of object states is done by **Messaging**.

4. Objects sharing the same structure and behavior are grouped into **Classes**.

5. A subclass can **inherit** both structure and behavior from its superclasses

The principles of object-orientation specified need to be augmented in a real life object-oriented databases with more requirements. These requirements include:

- a. Support of Complex Objects such as highly structured information, type-system orthogonality and extensibility.
- b. New types should be possible to be defined at any time by applying given constructors to already defined types in a vastly arbitrary fashion.

If an object-oriented data model captures this entire list of requirements it will be intuitively more complex than any previously developed model.

3.4.5 Detail Model Analysis

We need to emphasize the various types of the database models as they evolved over the years to provide the breakdown understanding on how the migration has been and where we are moving to today. This will clearly show the edge of database model research which is on the Object-oriented model level. Many database authors present these models as if they were optional models developed basically at the same time and as if one can simply choose one and use. This is far from been correct. The truth is that each of the models was an attempt at improving an older model, any competing models where at the same level like Hierarchical and Network database models which evolved at the same period and where computing options before they were merged to give rise to the relational model. The models according to their sequence of development include:

File System: File system (or filesystem) is data storage and retrieval technology used to control how data is stored and retrieved. Without a file system, information placed in a storage area would be one large body of data with no way to tell where one piece of

information stops and the next begins. By separating the data into individual pieces, and giving each piece a name, the information is easily separated and identified. Taking its name from the way paper-based information systems are named, each piece of data is called a file. The structure and logic rules used to manage the groups of information and their names are called a "file system". Files are identified by their characteristics, like type of file, topic, author etc.

A database File system is developed in late 1960s and was designed and deployed in real life databases by Frank G. Soltis IBM's former chief scientist for IBM (Pirkola, 1975). Around 1978 to 1988 Frank G. Soltis and his team at IBM Rochester had successfully designed and applied technologies like the database file system. IBM DB2 for (formerly known as DB2/400 and DB2 for i5/OS) was a database file system running on IBM Power Systems (formerly known as AS/400 and iSeries) These technologies are informally known as 'Fortress Rochester' and were in few basic aspects extended from early Mainframe technologies but in many ways more advanced from a technology perspective (Selzter, 1993).

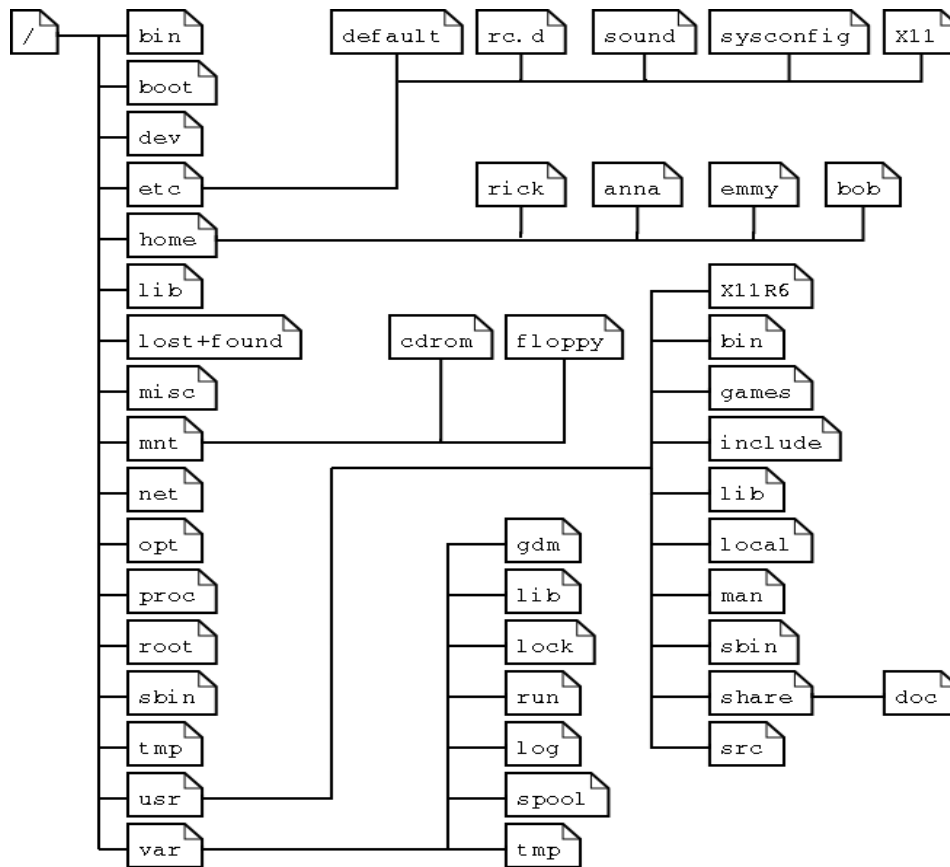


Figure 3.5: A File System Data Model (Adapted from Selzter, 1993)

Bachman diagrams: A Bachman diagram is a certain type of data structure diagram, and is used to design the data with a network or relational "logical" model, separating the data model from the way the data is stored in the system. The model is named after database pioneer Charles Bachman and mostly used in computer software design. The coupling between the relations is based on accordant attributes. For every relation, a rectangle has to be drawn and every coupling is illustrated by a line that connects the relations. On the edge of each line, arrows indicate the cardinality. We have 1-to-n, 1-to-1 and n-to-n. The latter has to be avoided and must be replaced by two 1-to-n couplings. Bachman model was used in the creation of powerful database like the General Electric Data store which gave rise to COBOL (Bachmann, 1969).

In Figure 3.3a, the Bechman database model shows the author association between books and the people who wrote them.

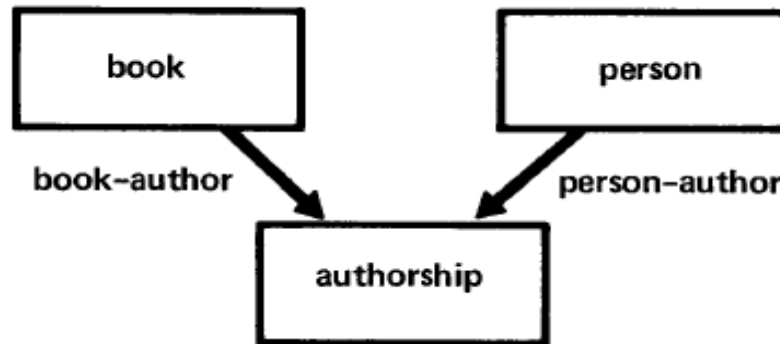


Figure 3.6a: Bechman Database Model of Authorship (Adapted from Bechman, 1970)

If one were to examine the books in any library, it would find that some books had one author while other books, especially in the technical and educational fields, had two and maybe three or four authors. If one considered the people who were authors of these books, it would be found that some were authors of more than one book. Certain prolific authors would have many books to their credit.

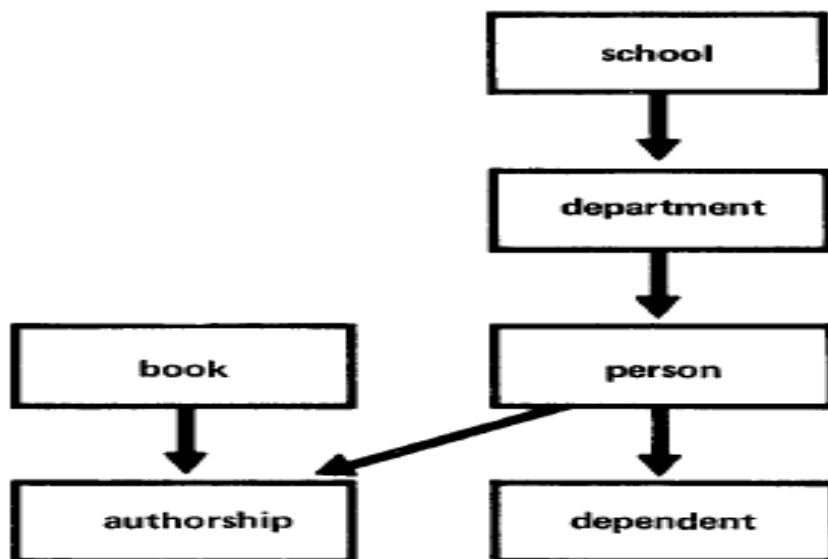


Figure 3.6b: Bechman Database Model of Authorship (Adapted from Bechman, 1970)

Therefore, the network created by the book/people/author relationship would consist of nodes for books (book entities), nodes for people (people entities), and a relationship between a book entity and a person entity that records authorship as shown in figure 3.3b using Bechman Data Structure Diagram (Bechman 1970).

3.5 The Universe of Discourse (*UoD*)

In this research, we focus on the data perspective, and assume the design involves building a formal model of the application area or *universe of discourse (UoD)*. To do this properly requires a good understanding of the UoD and a means of specifying this understanding in a clear, unambiguous way. Object-Role Modeling (ORM) simplifies the design process by using natural language - as well as intuitive diagrams that can be populated with examples- and by examining the information in terms of simple or *elementary facts*. By expressing the model in terms of natural concepts, like *objects* and *roles*, it provides a *conceptual* approach to modeling. The application area or universe of discourse selected in this research is a Conference management system. This is selected due to the multi-tasking involved during paper reviews on a distributed environment.

3.5.1 Analysis of UoD -Conference Management System

Conference is meeting of people to present and discuss their work in their field of endeavor. The field can be academic or other areas of life. In academic conferences research work are discussed as a channel for researchers. Generally, work is presented in form of short, concise presentation lasting about 10 to 30 minutes, usually including discussion. The work may be bundled in written form as academic papers reviewed by selected academics across large geographic area and published as the conference proceedings.

A conference management system is web-based software that supports the organization of academic conferences. It helps the program chair(s), the conference organizers, the authors

and the reviewers in their respective activities. A conference management system can be regarded as a domain-specific content management system. Similar systems are used today by editors of various journals (Barrett, 2000).

In this project we will study conference activities and use its domain components in the development of a conference management system that will be able to automate the activities that constitute the domain content of a conference system.

In conferences there are one or more keynote speakers, usually scholars of some standing. Panel discussions, round tables on various issues, workshop may be part of the conference, the later ones particularly if the conference is related to the performing arts. Prospective presenters are usually asked to submit a short abstract of their presentation which will be reviewed before the presentation is accepted for the meeting. Some disciplines require presenters to submit a paper of about 6-5 pages which is peer reviewed by members of the program committee or referee chosen by them. In some disciplines like sciences, presenters usually base their talk around a visual presentation that displays key figures and research results. A meeting might be single track or multiple tracks, where the former has only one session at a time, while a multiple track meeting has several parallel sessions with speakers in separate rooms speaking at the same time. Depending on the theme of the conference, social or entertainment activities may also be offered; if it's a large enough conference, academic publishing houses may set up displays, offering books at a discount. At a larger conference, business meetings for learned societies or group might also take place. Academic conferences fall into three categories

1. The themed conferences, small conferences organized around a particular topic',

2. The general conference, a conference with a wider focus, with sessions on a wide variety of topics. These conferences are often organized by regional, national, or international learned societies and held annually or on some other regular basis.
3. The professional conference, large conference not limited to academics, but with academically related issues. In organizing academic conference, the meeting is announced by away ‘’ call for papers’’ or call for abstracts, which lists the meeting’s topics and tells prospective presenters how to submit their abstract or papers. Increasingly, submissions take place online.
4. In this work we will study all the activities associated to conferences and extract key activities as an ‘’ information system’’ for computerization and automation. In this light, the work involve the management of the conference ‘’information system’’ in a way as to reduce and eliminate known bottlenecks associated with the local and manual handling of such information. Some of these conference functions and workflows supported by conference management systems include:
 - i) Receiving paper submissions (PDF upload, collection of bibliographic metadata)
 - ii) Anonymizing submissions
 - iii) Collecting reviewers' topic preferences
 - iv) Collecting conflicts of interest
 - v) Assigning reviewers to papers
 - vi) Disseminating submissions to reviewers
 - vii) Collecting reviews
 - viii) Monitoring review coverage
 - ix) Sharing reviews among the program committee

- x) Ensuring independence of reviews
(reviewers cannot see other reviews for a submission before they have submitted their own)
- xi) Providing a per-submission discussion forum for the reviewers
- xii) Ranking reviews and setting acceptance threshold
- xiii) Anonymizing reviews
- xiv) Reporting reviewers' comments and program committee decision to authors
- xv) Collecting final accepted versions

Some systems offer additional functions that go beyond supporting only the peer-review process:

- a) Creating a conference website and program
- b) Registering attendees
- c) Publishing proceedings

It is clear therefore, that conference management system is a branch of information management system and also related to management information system when examined from the point of the conference organizers and managers.

3.6 The Database Infrastructure.

The database forms a data infrastructure. They provide for storage of data needed for one or more organizational functions and one or more activities. There will be a number of databases based on organizational activities. Planning of the database infrastructures involves determining what should be stored, what relationships should be maintained among stored data, and what restrictions should be placed on access. The result of database planning and implementation with database management system is a capacity to provide data both for application and *ad hoc* needs. Comprehensive databases designed for *ad hoc* use may be termed data warehouses.

3.7 Development of Conference Strategies

Strategic planning like any other business activity planning begins with deciding the social responsibility and proceeds to spell out the business mission and goals and the strategies to achieve them. In the very beginning of the planning process it is necessary to establish and communicate to all concerned the social and economic responsibilities of the organization, in order to discharge these responsibilities. It is necessary to decide the purpose of the organization for which it works. Many organizations call it a mission. The task after deciding the mission or the aim is to set the goal (s) for the organization. The goal is more specific and has a time scale. The goals become a reference for the top management in planning the business activities. After determining the mission and the goals the next task is to set various objectives for the organization. The goal are described in terms of business results to be achieved in a short duration of a year or two the objectives are measurable and can be monitored with the help of business tools and technologies. When achieved, the objectives will contribute to the accomplishment of the goals and subsequently the mission. The next step in the planning process is to set targets for more detailed working and reference the objective of the business is to be translated in terms of functional and operational units for easy communication and decision making. The targets will be the direct descendants of the objective(s) (Doyle, 2000). The success in achieving the goals and objective is directly dependent on the managements' strategies. The resources of an organization being faced by it the game is of evolving strategies and counter strategies to win.

3.7.1 Types of Strategies

A strategy means a specific decision(s) usually but not always regarding the deployment of the resources to achieve the mission or goals of the organization the right strategy beats competition and ensures the attainment of goals while a wrong strategy fails to achieve the

goals. Correction and improvement in case of a wrong strategy is possible at a very high cost, such a situation is described as a strategic failure. If a strategy considers a single point of attack by a specific method it is a fixed strategy. If a strategy acts on many fronts by different means then it is a mixed strategy. The business strategy could be series of pure strategies handling several external forces simultaneously. Hence the strategy may fall in any area of the business and may deal with any aspects of the business.

3.7.2 Tools Of Planning

Planning, long-range or short-range, strategic or tactical, involves a series of decisions to be taken by the managers in the organization. So when we talk about the tools of planning, we are talking about the tools of decision-making with reference to planning. Decisions relate to several aspects of corporate business planning. There are numbers of alternatives, choices and options available while planning the business. Further, there is selection of resources and their allocation in an optimum manner to maximize the gains. Then there is selection of method whereby the efforts at all the levels are coordinated towards a common goal and direction. The planning, therefore, involves decision-making with the help of tools. These tools are based on one or more factors. These factors are: Creativity: Systems approach: Sensitivity analysis: and Modeling.

3.7.3 Creativity

Creativity comes out of an experience, a judgment, an intuition of an individual or a group of individuals. When decision making is called for a situation which has no precedent then creativity is the only tool to resolve the problem of decision making. Creativity is the result of the conceptual skills of an individual. The conceptional skills comprise the following skills.

1. The ability to generate a number of ideas rapidly.
2. The ability to change quickly from one frame of reference to another.

3. Originality in interpreting an event and generating different views on the situation.
4. The ability to handle with clarity and ease a complex relationship of various factors in a given situation.

A person who possesses these skills is said to have a conceptual fluency. If an organization has a number of people, at least at key positions, with conceptual fluency, then it becomes a creative organization. Such an organization creates new ideas and new strategies for development of business. The plans are made on the strength of experience and conceptual fluency.

3.7.4 Systems Approach

Systems approach to planning considers all the factors and their inter-relationship relevant to the subject. It takes a course to an analytical study of the total system, generates alternative courses of action and helps to select the best in the given circumstances. It is used in situation of risk or uncertainty, and examines the various alternative courses of action. It helps to find solutions to problems.

The systems approach helps to understand the situation with clarity. It helps to sort out the factors on the principles of critical and non-critical, significant and insignificant, relevant and irrelevant, and finally controllable and uncontrollable. It tests the solutions for feasibility-technical, operational and economic. It further studies the problems of implementation of the solution. Broadly, the systems approach has the following characteristic:

1. It uses all the areas and the branches of knowledge.
2. It follows a scientific analysis to identify the problem.
3. It uses a model of a complex situation to handle the problem.
4. It weighs cost against benefit for assessment of the alternatives.
5. It deals with the problems where time context is futuristic.

6. It considers the environment and its impact on the problem situation.
7. Every solution is tested on the grounds of rationality and feasibility, and accepts a given criterion for selection of the most preferred alternative.
8. It uses operations research models if the problem is well defined.

The systems approach is a way of looking at a problem in a systematic manner using the scientific methods and applying the principles of a rational decision making to solve the problem.

3.7.5 Modeling

A model is a meaningful representation of a real situation on a mini scale, where only the significant factors of the situation are highlighted. The purpose of a model is to understand the complex situation based on only the significant factors. There are several types of models. The model could be a physical model, like a model of a house, a park, a sports complex, etc. The model could be a scale model reducing a large body to a small one. The model could be mathematical model like break even analysis model, linear programming model, queuing model, network model, etc.

3.8 Conference Management System

A web application need to support every aspect of the conference organization process. This includes paper submission, reviewer assignments, revised and camera-ready paper submission, registration handling of the conference participants.

Software interfaces

One external system shall be the DBMS. There shall be an abstract layer between the system and the DBMS to provide database calls independent from the DBMS vendors. JDBC 3.x driver type 4 serves that purpose. Such drivers are usually developed by the corresponding database vendors (e.g. MySQL DBMS provides such a driver) (Plasmeijer, 2005).

3.8.1 Product functions and features

The system shall be able to handle multiple sub-conferences.

The system shall support special sessions.

The system shall support tutorials.

The system shall support invited or plenary talks.

The system shall support paper submission.

The system shall support review process.

The system shall support conference program creation.

The system shall support registration.

The system shall support creation of proceedings.

The system shall support on-site participants.

3.8.2 Security requirements

All, user provided data, which will be part of the SQL query must be preprocessed to put escape character (backslash) in front of a single or double quote or backslash. Otherwise misinterpretation of the SQL query is possible because these characters are part of the SQL syntax.

General role consideration

There are some information and principles which are common to all roles. These are mentioned in this part, and later specific roles are explained.

Personal data

Personal data includes:

- a) (Mandatory): title, name, last name, institution/affiliation, country, address, e-mail, areas of interest (multiple select from a list of areas of conference's interest).
- b) (Optional): department, phone, fax, home page URL

c) Username and password. Username must be unique, and username and password must be unique. They determine roles which person has in the System so appropriate modules can be shown.

Personal data is presented to all roles. Some roles can have additional items related to their functionality. Users can change their personal data later after login (Plasmeijer, 2006).

Activating accounts and invitations new roles can be assigned in two ways:

- 1 User with authority sends invitation with email which has active link to account activation page. That way invited person can fill all personal information by himself. If person is already registered in system he fills only his username and password so he could be assigned new role. Before sending this mail user should give the following information about invited person:

- i) Person's name
- ii) Person's last name
- iii) Person's e-mail
- iv) Title (optional)

Note: That way of registering is present at all later roles.

- 2 User with authority gives account by filling the form and selecting appropriate role. Form includes:

- i) Person's name
- ii) Person's last name
- iii) Person's e-mail
- iv) Persons username
- v) Persons password
- vi) Title (optional)

After filling the form and activating account user sends an email which informs an invited person about filled data and given account. Note: That way of registering is only available to Conference administrator. User gets account when visiting web site.

Administrator

Personal data

System administrator has username and password. These values are initially predefined by SystemAdministrator himself. His e-mail can be used as a contact for any problems related to system.

3.8.3 Conference creation

System administrator is role with the highest rank and is mandated that system functions properly. System administrator shall create new conference. Conference creation consists of filling in the conference data (only the first tag from) and answering a questionnaire. Questionnaire has the following questions that can be answered by yes or no. Questions are:

- a) Does the conference requires sub-conferences
- b) Does the conference requires tutorials
- c) Does the conference requires special sessions
- d) Does the conference requires invited/plenary talks

Answers to these questions enable/disable these functionalities.

System administrator shall also create a conference administrator. If there are more than one conference in database conference administrator will be able to browse conference.

Browsing conferences

System administrator shall be able to browse existing conferences in database. By clicking on the desired conference he becomes conference administrator for that conference and has ability to change all data.

Conference Administrator

Conference management

Only conference administrator is able to change all users and conference data. He has following permissions: He is able to list all users. He can change user's data and give new username or password.

Conference data

Conference parameters, which administrator defines or changes through a form are:

- i) Conference title, year, place and date of starting and ending of conference.
- ii) Deadline dates:
- iii) paper submission
- iv) notification of authors
- v) final paper submission
- vi) author registration
- vii) tutorial proposal (*if enabled*)

Paper management

Program Chair (PC) collects all papers sent by corresponding authors. After viewing papers metadata's he gives papers to sub-conference ProgramChairs (to certain sub conference)

3.8.4 Creating Conference Program

In this part Paper chair is dealing with building the program from sessions. Each of sub-conference PC gives ready session's plans to PC. PC chair collects all these sessions and makes daily schedules of conference. He does these by hand but all data related to where and when sessions are held is put into database. PC chair shall be able to add new events to a list of sessions. Such as coffee break, lunch break, excursions... These events are only denoted by a name. Invited talks and tutorials are special categories.

PC chair shall be able to define rooms/space for the presentations. These rooms are denoted only by their name.

Program is being built from sessions and events. Each session/event requires:

- i) Date/day
- ii) beginning time (precision to 5 minutes)
- iii) ending time (precision to 5 minutes)
- iv) room allocated

The system should be able to recognize and warn if any conflicts in the schedule occurred.

Notification system

All notifications are assumed to be made by email directly to PC Chair. This includes notification about paper withdraw, request invitation letter indicating his paper is accepted, author may notify organizer that another person will present his paper (if he will not attend) or author may notify about special presentation equipment requests.

3.9 High Level Model of the Conference Management System

The model shown in the Figure 3.7 shows the high level model of the conference management system. The system has HOME which is represented by the CMS system page followed by Conference, Papers, User Admin, Program, General Admin, and LogOut. The menus each have submenus that are carrying out different activities as listed in the figures.

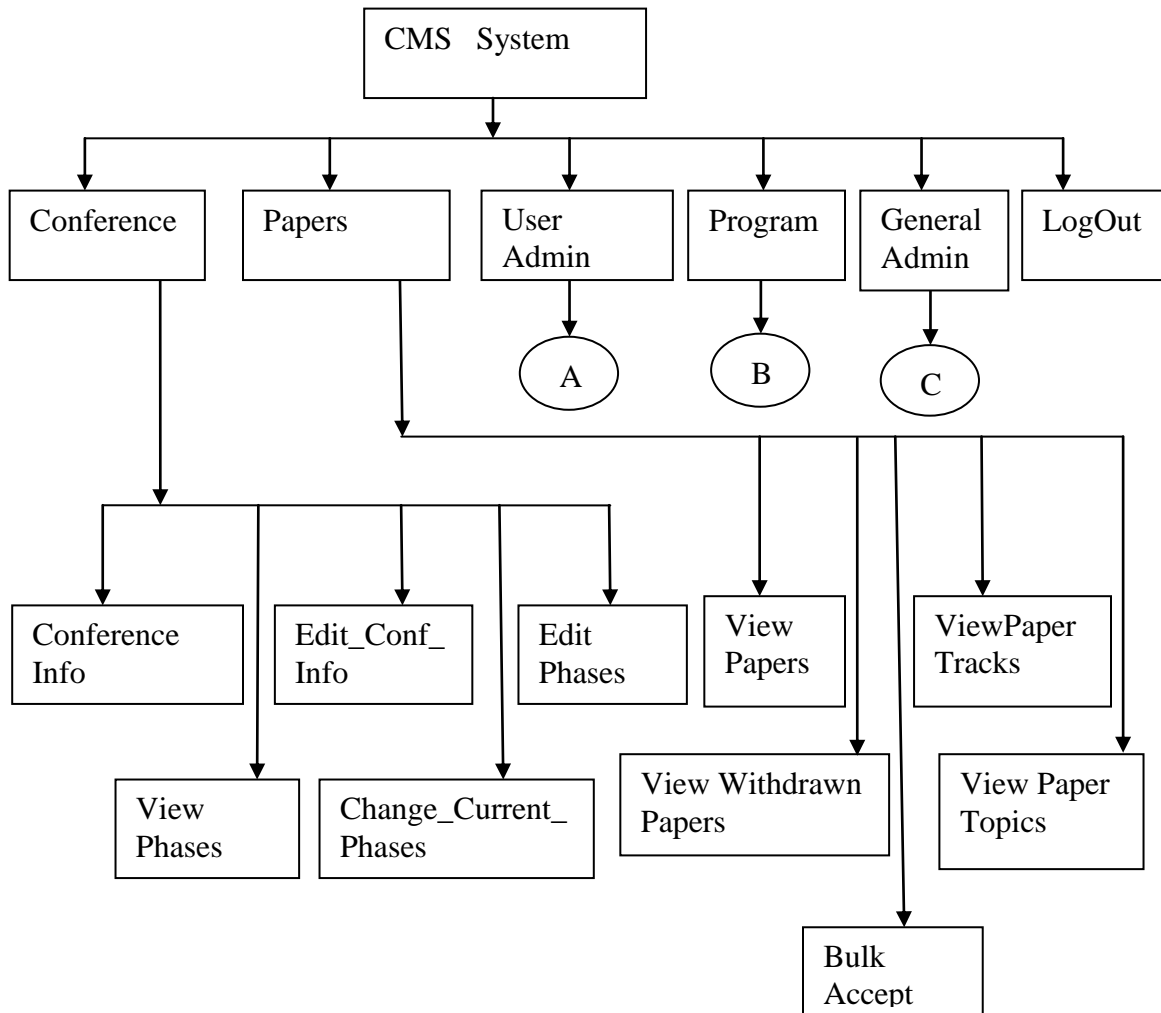


Figure 3.7: A High level Model of the Proposed CMS

The Conference menu has Conference Info which provides conference information to the public. This is followed by View Phases which provide conferees to know the various activities for the conference. The next is Edit_Conf_Info used for editing the conference information in case of changes. If the phases change the Change_current_phases submenu is used to handle it. But if the phases are to be edited, Edit Phases submenu is used. The papers menu equally contain submenus such as View Papers which allow reviewers to view different

papers, View Withdrawn Papers, a submenu allowing the papers withdrawn for the conference to be viewed. Accepted papers are handled by Bulk Accept and one can view the various conference tracks by using the View Paper Tracks. The View paper Topics submenu can be used to check all the topics of the conference. The remaining submenu items are described in A, B and C in Figures 3.8 and the others below.

In Figure 3.8 the admin menu offers services allowing the admin to view all users, view all reviewers and setup reviewers' account.

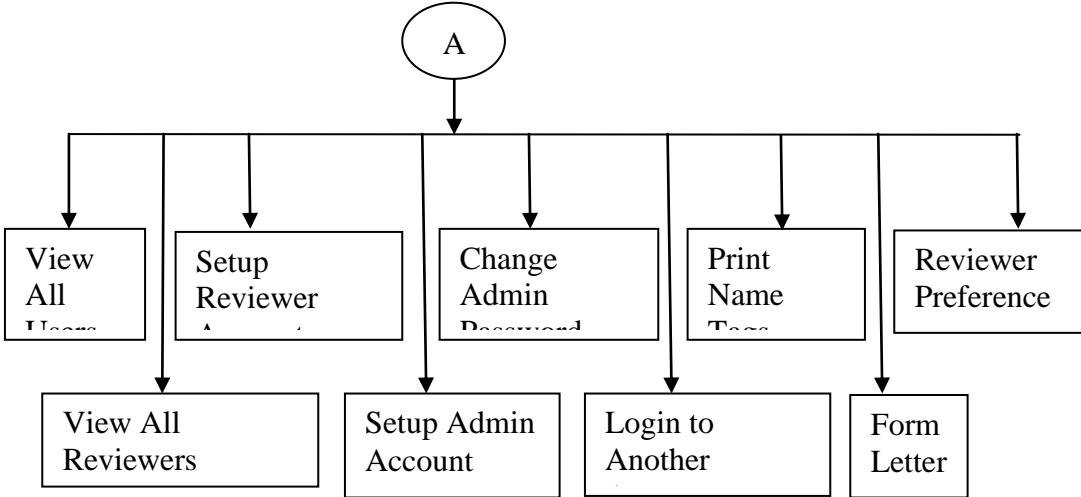


Figure 3.8: A High level Model of User Admin Menu

The model also allow the admin to Setup Admin Account, change Admin Password and login to another account within the CMS system. The admin submenu also allows the admin to be able to print names using the Print Name Tags submenu and gain access to Form letters. The admin can also check for reviewer preferences using Reviewer Preferences submenu.

In Figure 3.9, the program menu has six submenu Rooms, Presentation Types that take care of the various presentations, and Session Tracks that handles the various tracks in the conference. It also includes the sessions and the various programs by track. The program by

Track is also followed by program by Room which offers conferee various program options at the middle of the conference.

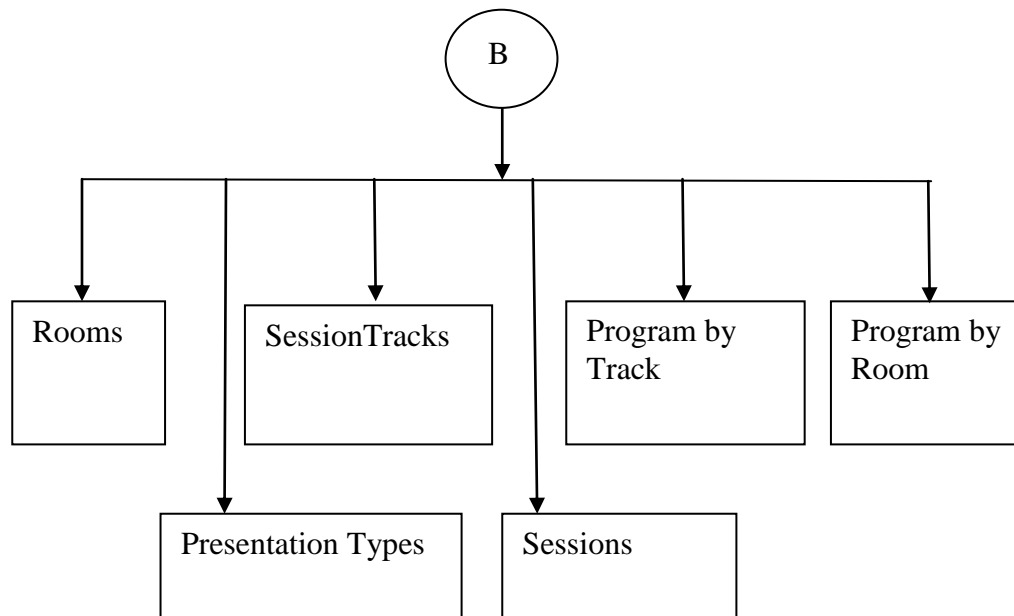


Figure 3.9: A High level Model of the Program Menu

In Figure 3.10 the general Admin menu model illustrates the various submenus which include Change settings used in adjusting the system settings. Import and Export of files and data info are also handled using Import and Export submenu. Settings, a submenu used in handling general administrations of the systems. Others include the Extract All Papers submenu used in getting the accepted papers for proceeding publication, the BuildCD structure submenu can be used in preparing the index for the publications article listing.

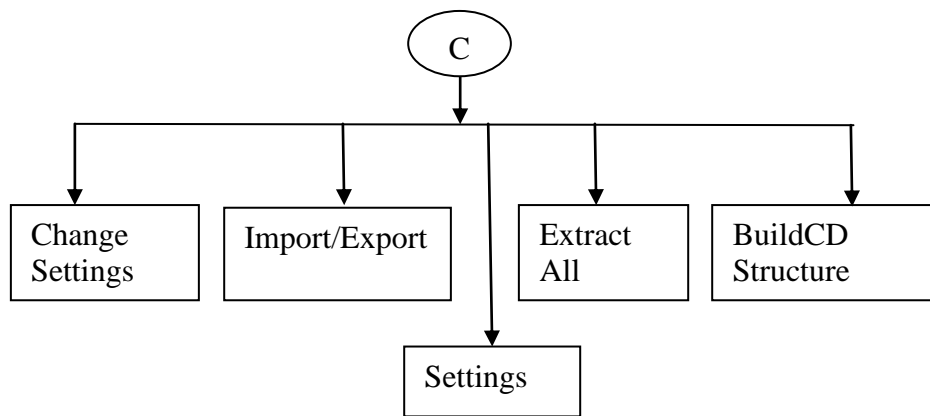


Figure 3.10: A High level Model of General Admin Menu

The last menu in the system is the LogOut menu which is used in logging out of the system.

CHAPTER FOUR

SYSTEM DESIGN AND IMPLEMENTATION

4.1 Design

Design involves the synthesis of the component part of a system in order to generate a new system that proffers solution to a specified problem which has previously been analyzed. The quality of a solution proffered for a problem depends critically on the design and the tools deployed. To help ensure correctness, clarity, adaptability and productivity, the design of object-database architecture for the development of an object database system communication are best specified first at the conceptual level, using concepts and language that people can readily understand. The conceptual design may include data, process and behavioral perspectives, and the actual database management system (DBMS) used to implement the design can be based on object-oriented logical data model.

In this research, we focus on the data perspective, and assume the design involves building a formal model of the application area or *universe of discourse (UoD)*. To do this properly requires a good understanding of the UoD and a means of specifying this understanding in a clear, unambiguous way. Object-Role Modeling (ORM) simplifies the design process by using natural language - as well as intuitive diagrams that can be populated with examples- and by examining the information in terms of simple or *elementary facts*. By expressing the model in terms of natural concepts, like *objects* and *roles*, it provides a *conceptual* approach to modeling.

The application area or universe of discourse selected in this research is a Conference management system. This is selected due to the multi-tasking involved during paper reviews on a distributed environment.

Design is an inevitable process of change when an old system becomes obsolete; a new system development is necessitated and provided. In this chapter, the system design process will be carried out from two different angles; the existing manual system flaws and limitations already analysed will be used to design the new system so that it will proffer solution to the limitation and faults of the old. We implement the designed system in PHP. An alternative is to use an inspiration from an existing package.

4.1.1 UoD Design – Conference Management System Design

Conference is meeting of people to present and discuss their work in their field of endeavor. The field can be academic or other areas of life. A conference presentation may be bundled in written form as academic papers reviewed by selected academics across geographic area and published as the conference proceedings. A conference management system is web-based software that supports the organization of academic conferences. This system is domain specific – based on conferences and publications. The idea of a conference management system includes the submission of conference papers and their review as well as communication with authors on the details of the conference. Each topic of submission is handled as a thread.

We will implement a web conference management system called E-conf with the following functionality. Users will be able to

- i) Select a thread of submission where they intend to submit.
- ii) Submit papers in reply to call for papers deadlines.
- iii) View call for papers that have been posted.
- iv) Download papers for review and upload reviewed comments
- v) Mail comments and corrections to the author for necessary correction
- vi) Uploading corrected versions
- vii) Mailing author on acceptance

4.1.2 Design Consideration of the CMS

The conference Management System domain is modeled in terms of its main stakeholders (users), namely papers' authors, editors and reviewers, the conference's program committee and its chair. Program Chair user is represented using (PC), paper reviewers by the actor Reviewer and the proceedings publisher by the actor Publisher. Stake holders' goals are then identified and, for every goal, the analyst can decide, on the basis of the domain documentation, if the goal is achievable by the actor itself or if the actor has to delegate it to another actors revealing a dependency relationship between the two actors, such as in the case of the dependency between Author and PC for the achievement of the goal publish proceedings

As illustrated in Figure 4.1 there are three main steps to be taken before conference can be developed and steps are as follows:

- i. **Survey Problems(s)** and possible solutions before conference management system can be developed there are certain requirements to be met; modeling can be further pursued by decomposing a goal into sub-goals and by employing the possible alternatives to achieve a goal. Alternatives are represented by OR- decomposition and characterized by multiple contributions. At this stage, also non-functional

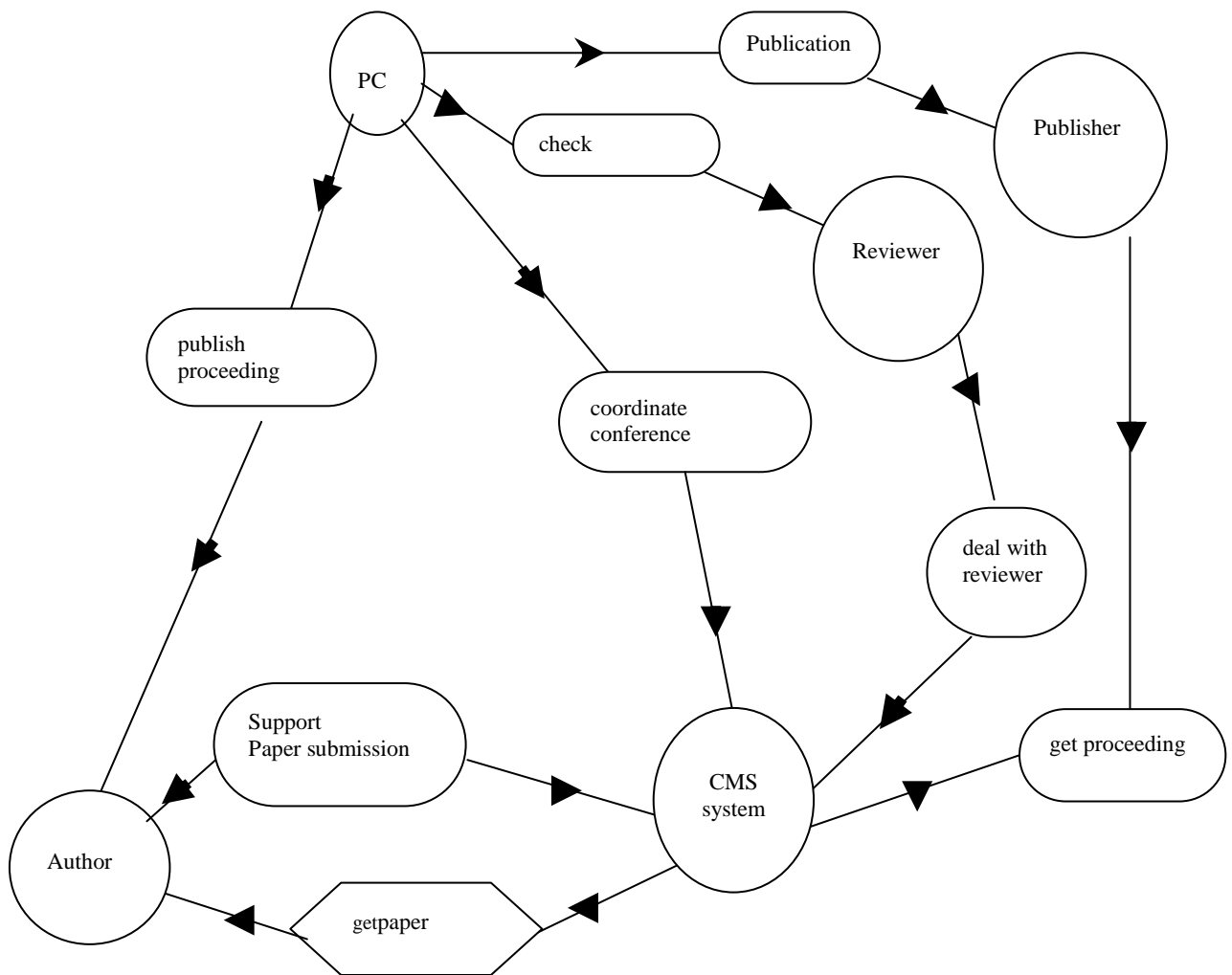


Figure 4.1: Early Requirements of CMS: Goal Dependency Design Diagram requirements can be represented as soft-goals. Choosing one alternative with respect to another, leads to different soft-goals achievement. By this way, it is possible to compare different alternatives and select the most appropriate one. Goal dependency diagrams can be dynamically created. The tool allows every actor in the model, to open (Close) their goal diagrams, which appear as

balloons attached to the relative actors. This will enable the tool to support the analyst in identifying the elements to be analyzed.

ii Early Requirements goals

This involves only two actors of the model, PC represented with two goal dependencies, that is Manage conference and decide deadlines. The goal manage conference is analyzed from the point of view of its responsible actor, PC chair, through an AND decomposition into several goals.

iii The late Requirement Phase

This is intended to capture the changes in the domain caused by the introduction of the system to be and the actual properties of the system. The phase starts by introducing in the domain model a new actor representing the system-to-be.

A partial view of the resulting model is shown in the Figure4.2; the system clearly shows management of submission, reviews, decision, and proceedings aggregate to conference coordination. In turn assign papers to reviewers and collect reviews aggregate to manage reviews.

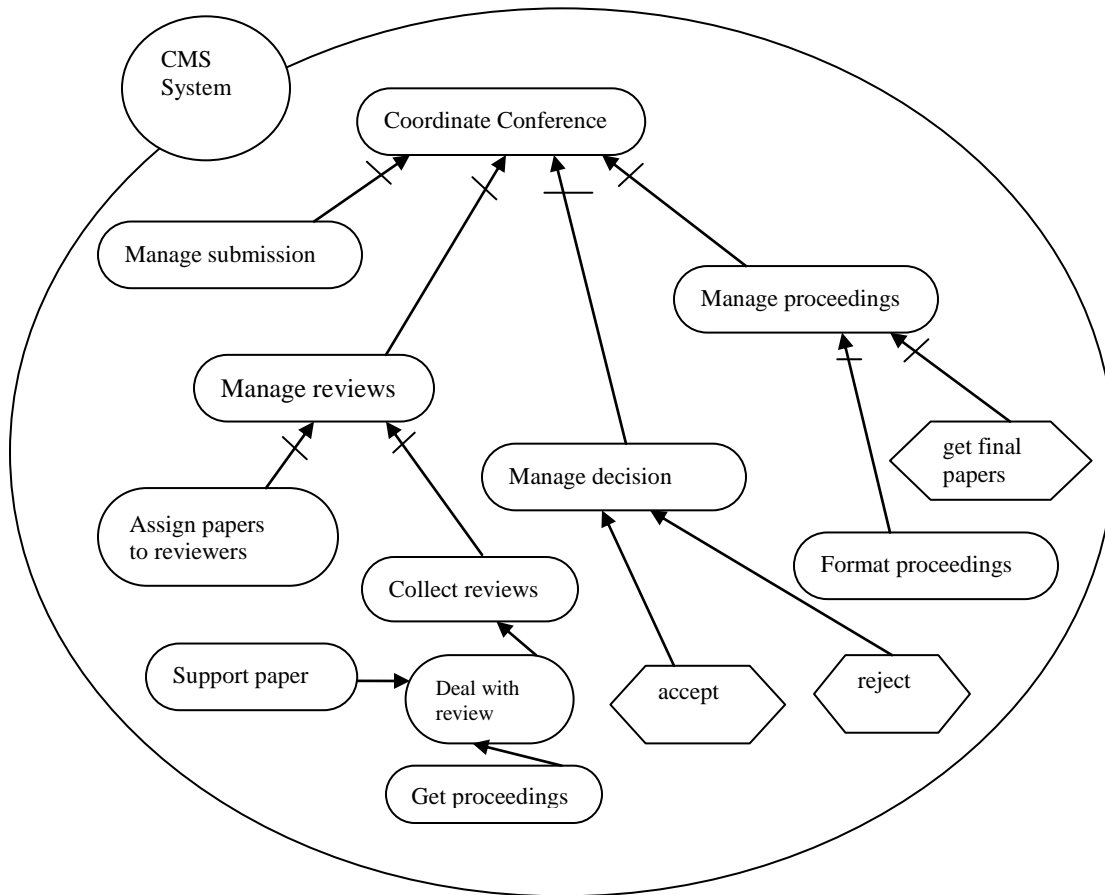


Figure 4.2: Aggregate Design of Conference Coordination

Proceedings format, final papers and accept or reject decisions also are aggregation contributors. When these and other components within the system aggregate, they form the Conference management system making the development of the system move from the analyzed requirement system components to the actual system.

4.2 Design Details of the CMS

The new System is the designed that follow the conference components identified in the design aggregation system and the way they interact. A database is also used as a resource base for storing all the information the system will work with.

4.2.1 Existing System Process Design

The CMS used as a use case in the design of the system is first presented using the existing object relational database to illustrate the possible variation with the proposed design system that is deployed on web server so that authors and reviewer can have access all over the globe. The key variation is clear from storage of data to the way the data is manipulated in the system.

The design using the existing system is illustrated in Figure 4.3. In the proposed system design the database stores the simple data using relational model and allow the media server to process the object stored in different location via the middleware. When the Author, Reviewer or editor interacts with the system the Controller clearly handles the simple data allowing the objects to be stored separately in the Object data store from where it can be conceptually linked by the middleware in the system controller.

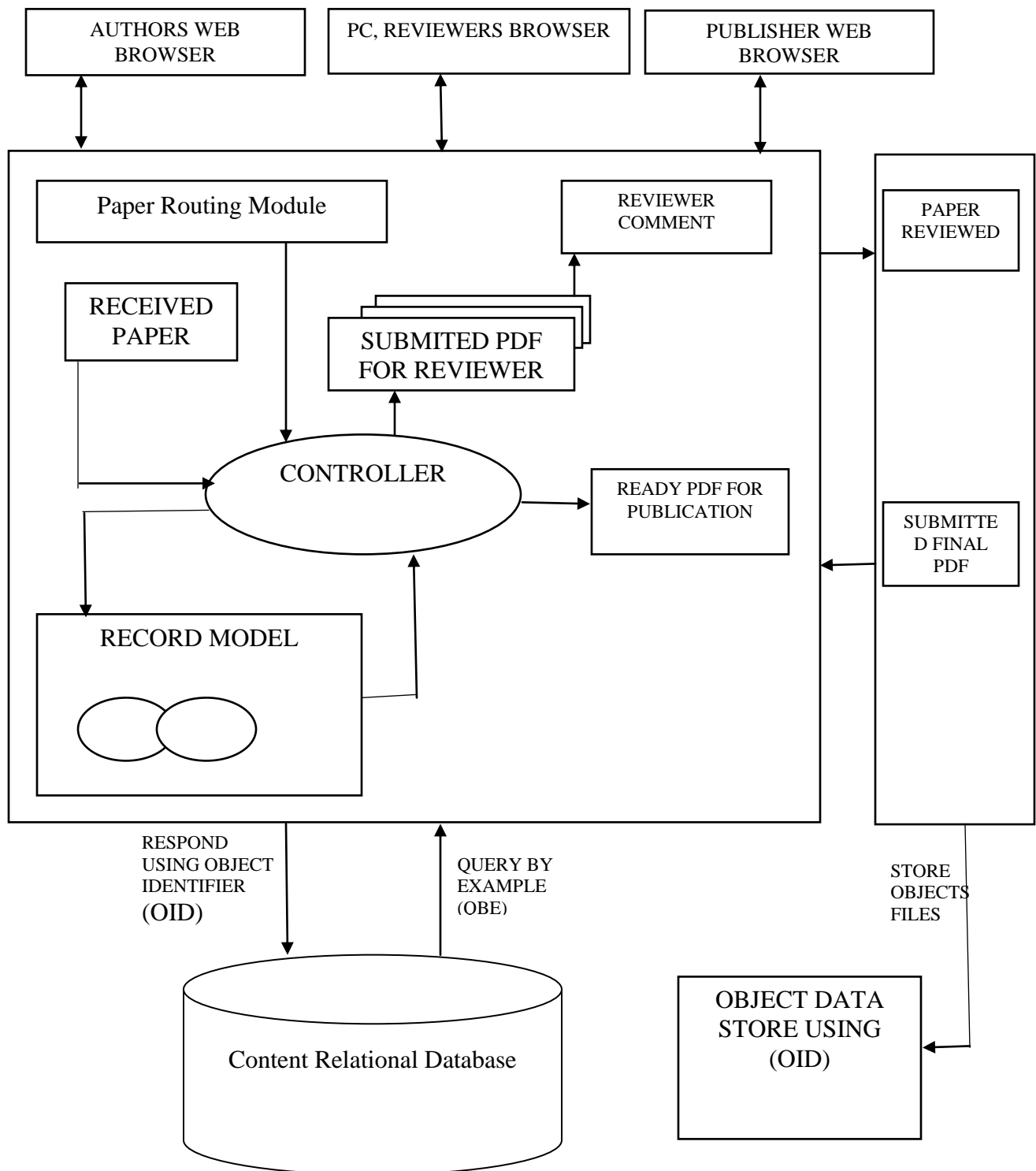


Figure 4.3: Process Design of the System Using Existing Object Relational Database

4.2.1.1 Proposed System Process Design

This proposed system is a web application that is deployed on web server so that authors and reviewer can have access all over the globe. It has a client interface where the authors use to interact with the PC and other coordinators of the conferences. The client will also be used for call_for_paper adverts which are usually preconference adverts. In Figure 4.4 the web browser is used by authors to submit their papers into the conference management system, the routing application inside the system will store the Pdf or other file formats like pictures as files and the system information associated to the file submission such as author, data of submission and other associated information are sent into the database for storage. Other users like PC, reviewers and conference coordinators can assess the content from their own browsers. Once the reviewers are done and decisions are made on the selected papers the publisher gets the accepted papers to press electronically or on print. When the system is implemented it will help in changing organizational and academic conferences to show case all the conferences in their store on an online catalogue accessible by a web browser worldwide. It will also increase the quality of papers by allowing reviewers who are experts to review the papers since they can be extracted from all over the globe. Paper submission will also be widely received from various parts of the globe since authors will be able to access and submit their papers directly online and the data and paper handled collectively as both objects and database. The database management system needs to handle the data and their associated object in a way to encourage easy access, storage and querying of the database. It is clear from the design that objects are not separated from simple data as in the present system design.

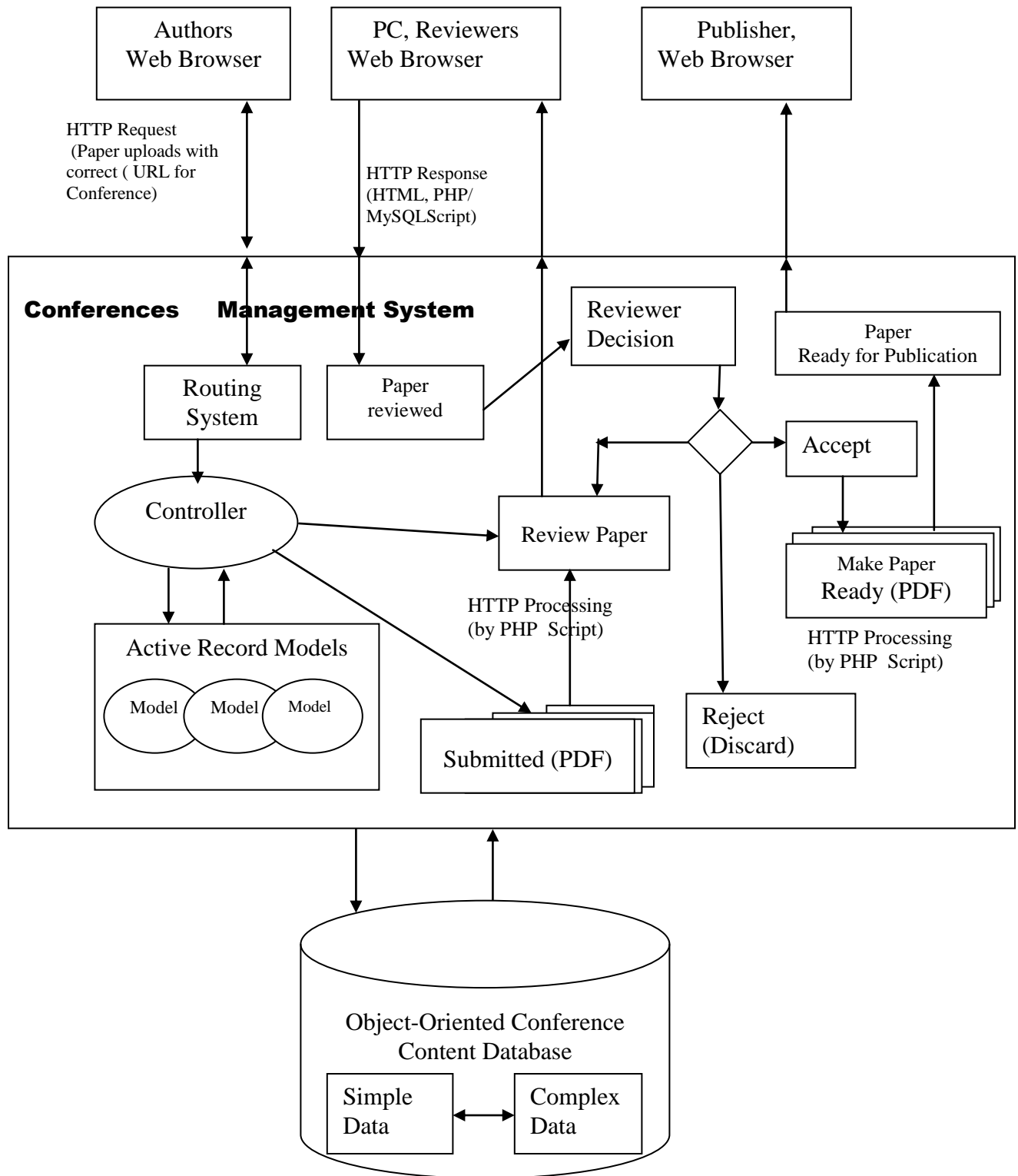


Figure 4.4: Design of the Conference System using the Proposed System

4.2.2 I/O Specification and Design

Every system is designed to produce specific output for its users; the output of our system is a browser data and activities that facilitate conference content storage and retrieval. Our design of a conference management system output for the academics and professionals should handle:

1. Registration of Participants
2. Submission of papers and other session materials
3. Creation of session plan, relevant to the website

In particulars it should be easy to set up a new conference by use of configuration of data rather than by reprogramming.

4.2.3 Input Specification and Design

The major means by which this system captures data is through the use of input forms, these forms are graphical user interfaces provided for the user to input information that will be stored for future referencing. For any system to work there has to be a combination of input and a corresponding output. Our proposed conference system for academic and industrialist is not lacking in this aspect, because there are avenues created for the system to receive the information that it will work with. The input specifications are as follows:

Postid: A unique ID for each article

Poster: The author of this article

Title: The title of this article

Posted: The date and time the article was posted.

Message: The body of the article:

All the input forms of the system are designed using conference HTML /PHP forms that provide dynamic ability. The input forms for user account setup is illustrated in figure 4.5. Hint: Use your email address as your user name – it is unique and easy to remember when you long in. Field that have a satiric (*) are mandatory

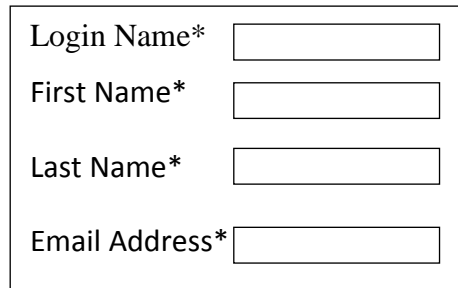
User Name*	<input type="text"/>
(or email)	
	<input type="text"/>
First Name*	<input type="text"/>
	<input type="text"/>
Middle Name	<input type="text"/>
	<input type="text"/>
Last Name*	<input type="text"/>
	<input type="text"/>
Organization*	<input type="text"/>
	<input type="text"/>
Address 1*	<input type="text"/>
	<input type="text"/>
Address 2	<input type="text"/>
	<input type="text"/>
City*	<input type="text"/>
State / Province*	<input type="text"/>
	<input type="text"/>
Postal Code*	<input type="text"/>
	<input type="text"/>
Country*	<input type="text"/>
	<input type="text"/>
Email (Primary) *	<input type="text"/>
Email (Secondary)	<input type="text"/>
Phone (Primary) *	<input type="text"/>
Phone (Secondary)	<input type="text"/>

Figure 4.5: User Account SetUp Input Form

After account setup, the user name and password will be email to the primary email address of the owner of the information. The user account set up enable new users of the system to register in the system and use their registration information in gaining access to the system.

The input forms for Reviewer Setup information is illustrated in Figure 4.6. The Conference Reviewer is expected to fill in the Login Name, First Name, Last Name and the Email Address and the organization he is affiliated. When done he click Submit.

Hint: Field that have a satiric (*) are mandatory.



A form for Reviewer Setup information. It contains four text input fields, each preceded by a label and an asterisk indicating it is mandatory: 'Login Name*', 'First Name*', 'Last Name*', and 'Email Address*'. The fields are stacked vertically within a rectangular border.

If you fill in the organization field, the new reviewer will not have to register himself.

Organization*

☐ Inform the user now.

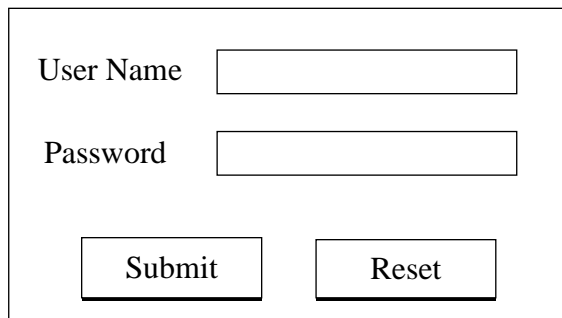
Submit

Cancel

Figure 4.6: A Review Set up Form for Reviewers to register for conference reviews in the System.

The form for login information is illustrated in Figure 4.7. The users are expected to fill in the User Name and the Password. When done he clicks Submit and gets logged in if correct.

Hint: Use your email address as your user name



A login form with two text input fields: 'User Name' and 'Password'. Below the fields are two buttons: 'Submit' and 'Reset'. The entire form is enclosed in a rectangular border.

Figure 4.7: A Login Form for conference System.

The Edit form for users to edit the information already supplied in the system is illustrated in Figure 4.8. The user is expected to fill in the Form if they needed to adjust the data previously submitted in the system. When done he click Submit.

Hint: Field that have a satiric (*) are mandatory.

Edit Conference Information

Name	<input type="text"/>
Code Name	<input type="text"/>
Start Date	<input type="text"/>
End Date	<input type="text"/>
Location	<input type="text"/>
Host Society Name	<input type="text"/>
Contact Email	<input type="text"/>
Logo for Conference	<input type="text"/>

Figure 4.8: A Form for Editting of the conference information supplied in the System.

4.3 The System Architecture Design

This consists of the system's overall structure represents inters of its sub-systems and their inter-dependencies.

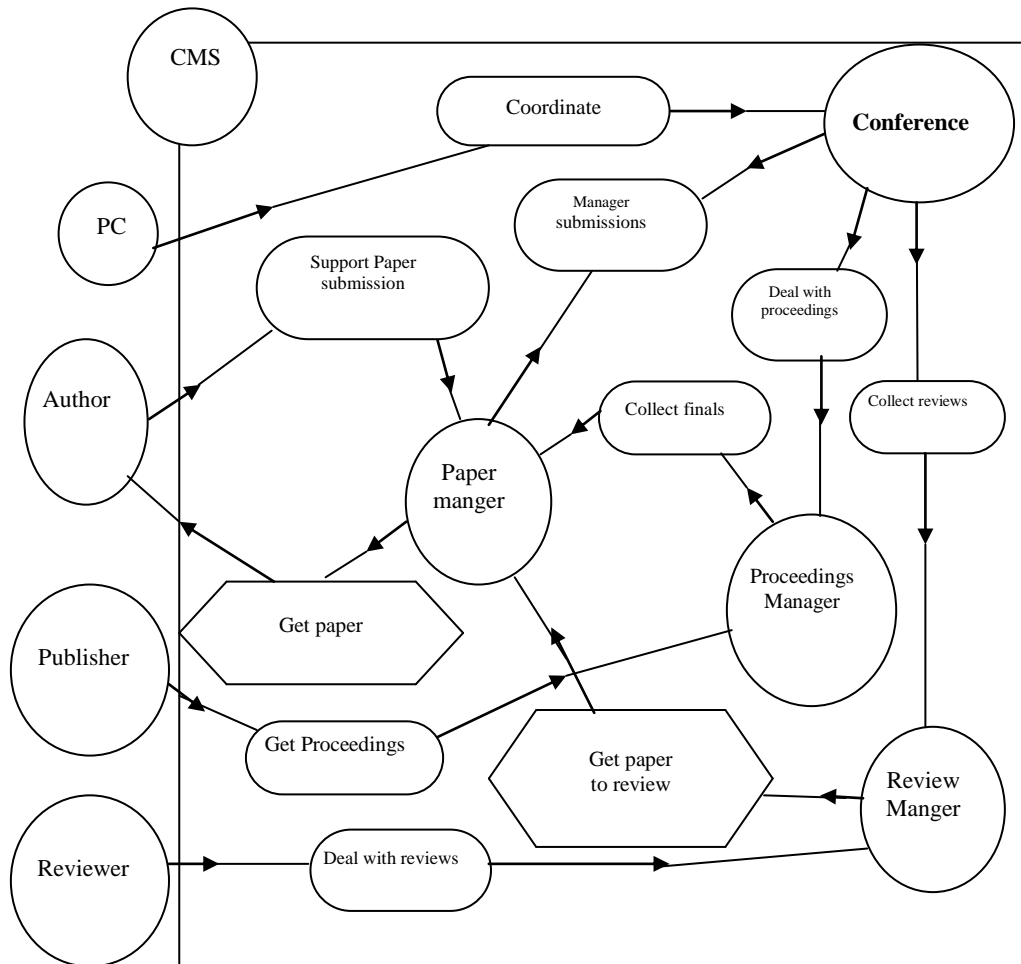


Figure 4.9: The database architecture of the conference management system.

MySQL stores information in a number of tables related by a common field known as the primary key. For instance we have academics information table that contains specific information about the academics, another table called industrials that contain information about outstanding industrials or certain organization. Both table contain a common filed (identity number). In a relational database, by keying on the identity Number, a third table is

made of data from each of the other tables. To design academic and industrial or organizational conference, the graphical representation is shown in Figure 4.9. The design clearly shows how the various components interact with the actors or users of the system.

Author interacts with the paper manager through the support paper submission and through the Get_paper action. Reviewer interacts with review manager by get_paper_to_review action and Deal_with reviews. Publisher on the other hand interacts with the Proceedings Manager through Get_Proceedings process. The PC interacts with all through coordinate process. Adopting the multi-agent system paradigm, subsystems are agents that can act independently and communicate with others through message passing.

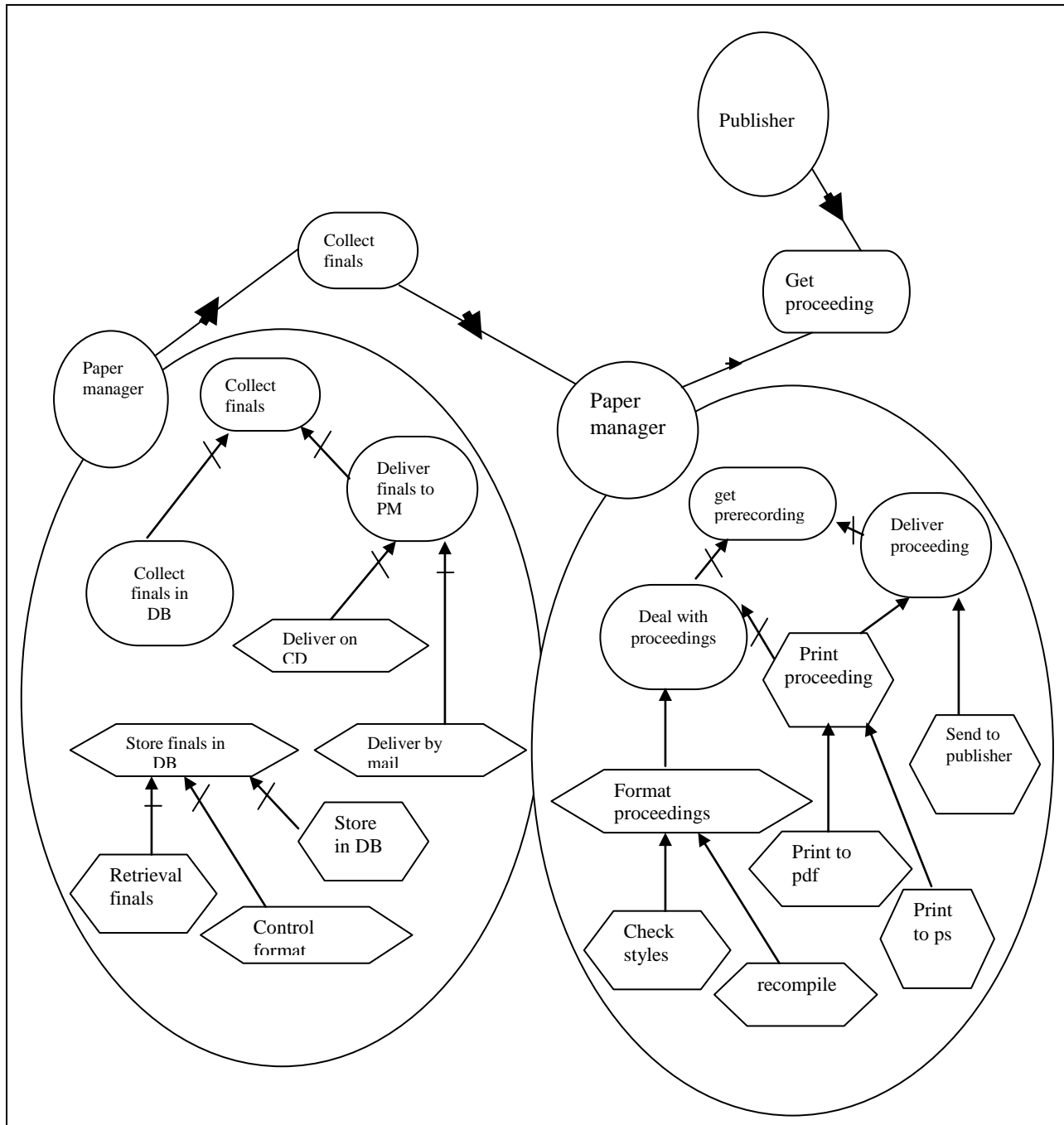


Figure 4.10: The Combine goal dependency architecture of the conference management system.

In order to build the architectural design, the engineer will refine the system actor by introducing sub-actors, which are responsible for actually carrying out the system's top goals.

Figure 4.10 shows an excerpt of the goal dependency model for two of the sub-actors- paper Manager and proceeding manager.

The conference system application used in the design of the proposed system for organization or academics conference online portal utilizes the frame work for software design. The model clearly shows the dependency between the conference manager and conference publisher within the framework of the content management system. The dependency will definitely aid the process of designing the object-role model components required in the implementation of the system and the relationship between the components as documented in the design on the next section.

4.4 Object Role Model Design

Object Role Model (ORM) is a method for designing a system model at the conceptual level and mapping between conceptual and logical (e.g. relational) levels, where the application is described in terms readily understood by users, rather than being recast in terms of implementation data structures. ORM includes a formal, textual specification language for both models and queries, as well as a formal, graphical modeling language (Terry, 2012).

Object Role Modeling got its name because it views the application world as a set of objects (entities or values) that plays roles (parts in relationships). It is sometimes called fact-based modeling because ORM verbalizes the relevant data as elementary facts. These facts cannot be split into smaller facts without losing information.

Object role modeling recognizes four basic kinds of data objects: simple, value, composite, and nested. A **simple object** is one in which real world instances are designated -- uniquely identified -- by a single data element; i.e., a single data element comprises the primary key (Stanley, 2012). Figure 4.11 shows how people would be represented in an object role model

when real-world people are to be designated in the database by an identifying number, Person_id, rather than by their names. Here, Person is an example of a simple object.

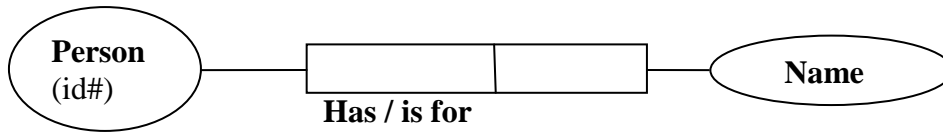


Figure 4.11: Simple and Value Object

A **value object** is something like a name (label), number, or date, which represents a simple scalar attribute. Name, in Figure 4.11, is a value object. It is clear from the object-oriented perspective that an object-person is distinctive from his name yet there is a relationship. When simple and value objects combined using a key, **composites** are formed. An object whose existence and identity stems from the relationship between two other objects is called a **nested object**. A “circle” (a rectangle with rounded corners) is drawn around the relationship to “objectify” it. A nested object can then participate in relationships with other objects.

4.4.1 ORM of UoD Design-Conference Management system Design

We have said that the universe of discourse (UoD) in this research is the conference management system. In this section, we will apply the ORM in the presentation of the conference management system from the conceptual level and mapping between conceptual and logical (e.g. relational) levels. We will also present the textual specification for models and queries, as well as a formal and graphical model of the CMS system. The CMS has many entities each relating to other entities within the system in a form that they cannot be separated without losing their collective perspective. In the system all conceptual factors viewed individually are elementary but they make a compound meaning when viewed as a whole. The whole presentation can be realistically understood only when the various entities

are related and linked using the object role model of the system. As shown in Figure 4.12, a role is a part played by an object in a relationship and is shown as a box connected to its object type. In the simple figure the Reviewer (a person object) reviews the Conference Papers (an electronic or physical document object) object. The dot is mandatory role constraint which is conceptual.

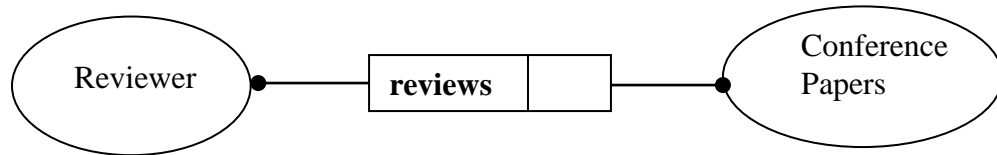


Figure 4.12: Illustration of Role between two CMS objects

The Reviewer plays the role of **reviewing** and the Conference paper plays the role of being reviewed. The keyword reviews appears on the part of the box connected to the Reviewer object indicating which object is playing the specified role.

4.4.2 The Object Role Model Conceptual Schema Design

In Figure 4.13, a conceptual design of the Conference Management System is presented using an Object Role Model. This design makes it easy to understand the movement of activity and interaction between objects without presenting the detail data components of the objects.

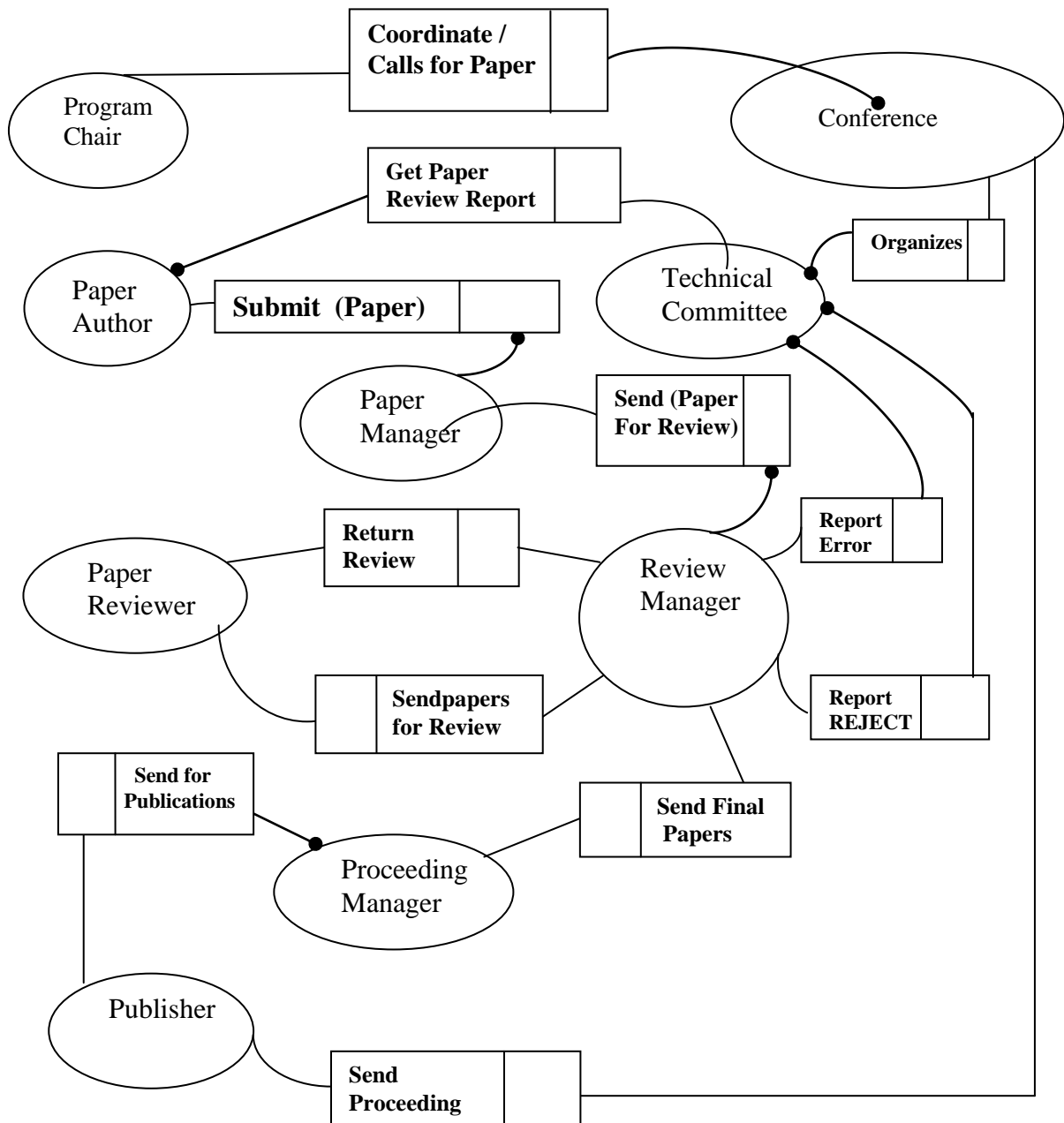


Figure 4.13: Conceptual Design of the Conference Management System using ORM

In other to explain in some detail the design developed in Figure 4.13 above, we will partition the design into four major sections and provide a more detailed description of the design so that when the system is examined as a whole it will become clearer.

4.4.3 Section I: Coordination of Conference and Call for Paper

When we examine the diagram in Coordination of conference and Call for Paper section of the Conceptual Conference Management System Design (CCMSD) in Figure 4.14, it is very clear that Program Chair **Coordinate and Calls for Paper** for the Conference. The Program Chair plays the role of **Coordinating** and the Conference plays the role of being coordinated. Similarly the Technical Committee **Organizes** the Conference and the Conference is organized by the Technical Committee. These interactions of objects have situations where each object is clearly assigned a role in the conference management system.

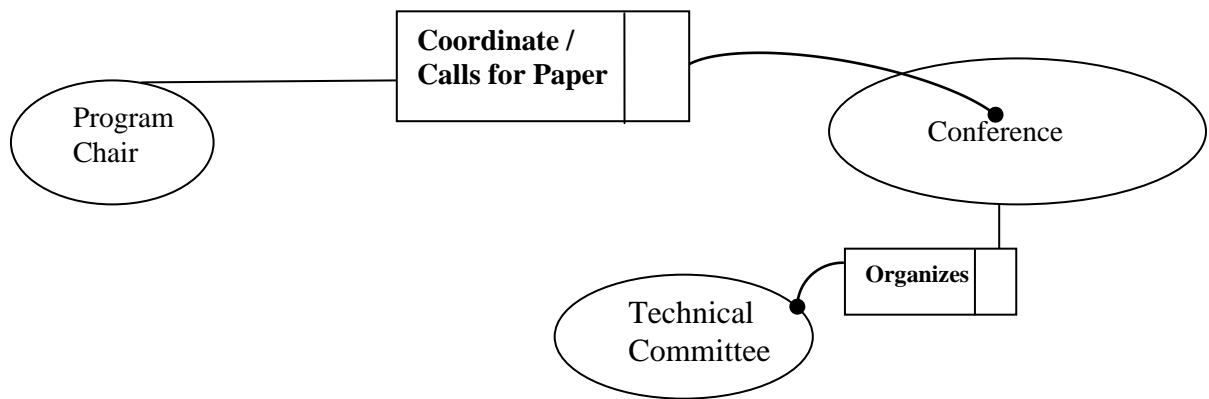


Figure 4.14 : Coordination and Call for Paper section of the CCMSD

This roles form the operational components required in the developmental and implementation stages of the system development. Since the system need the operation to be able to clearly define what each class does in the code developed for the system. It is also from the classes that the objects are extracted during the execution of the system. The three

objects Program Chair, Conference, and technical committee play the roles in the call for paper section of the design.

4.4.4 Section II: Paper Management Activities Section of the CCMSD

The diagram in Figure 4.15 clearly shows the objects that are components of the paper management activities of the Conference management system. These objects include the paper Author, the paper manager and the technical committee. Other objects include the paper reviewer and the review manager. These objects interact in a specific manner aimed at handling paper using the led down procedure. The Author object triggers the inter-communication by responding to call-for-paper and **submitting** a paper to the Paper Manager. The Paper Manager **sends** the paper to the Review Manager for allocation to the various paper reviewers.

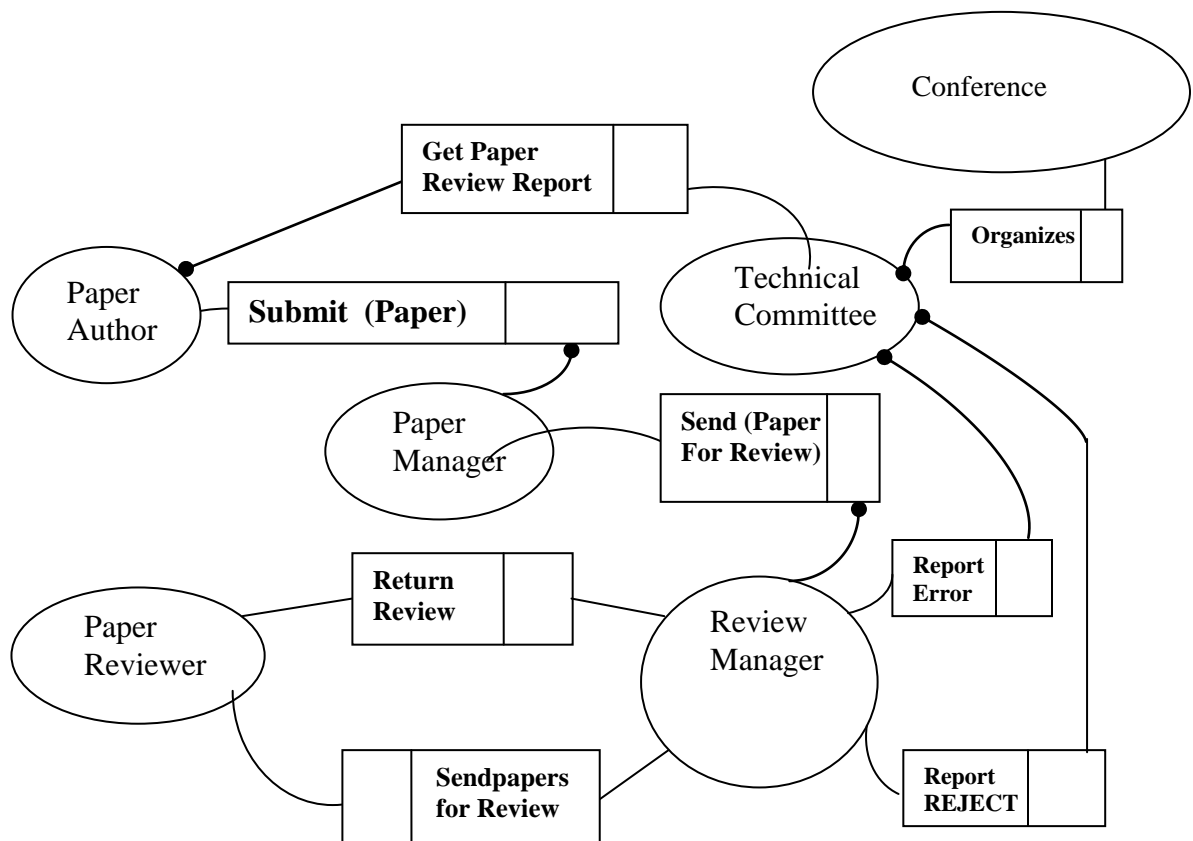


Figure 4.15: Paper Management Activities Section of the CCMSD

The Review Manager Sends papers for review to various Paper Reviewers and monitors the review process to make sure that the papers are reviewed at the appropriate time and that the Reviewers allocated papers are really experts in the area where they are saddled with the responsibility of reviews. The number of reviewer per paper is also determined by the review manager. Once the papers are reviewed they are returned to the Review Manager for the appropriate action. The Review manager will Report Error in paper and send to technical committee for onward transmission to the Authors for correction or report Rejection to the Author via the technical committee.

4.4.5 Section III: Paper publishing section of the CCMSD

The section on paper publishing still involves the Paper review manager who **sends** accepted papers to the Proceeding manager. The Proceeding manager in turn **sends** the papers in well-arranged proceeding format to the Publisher.

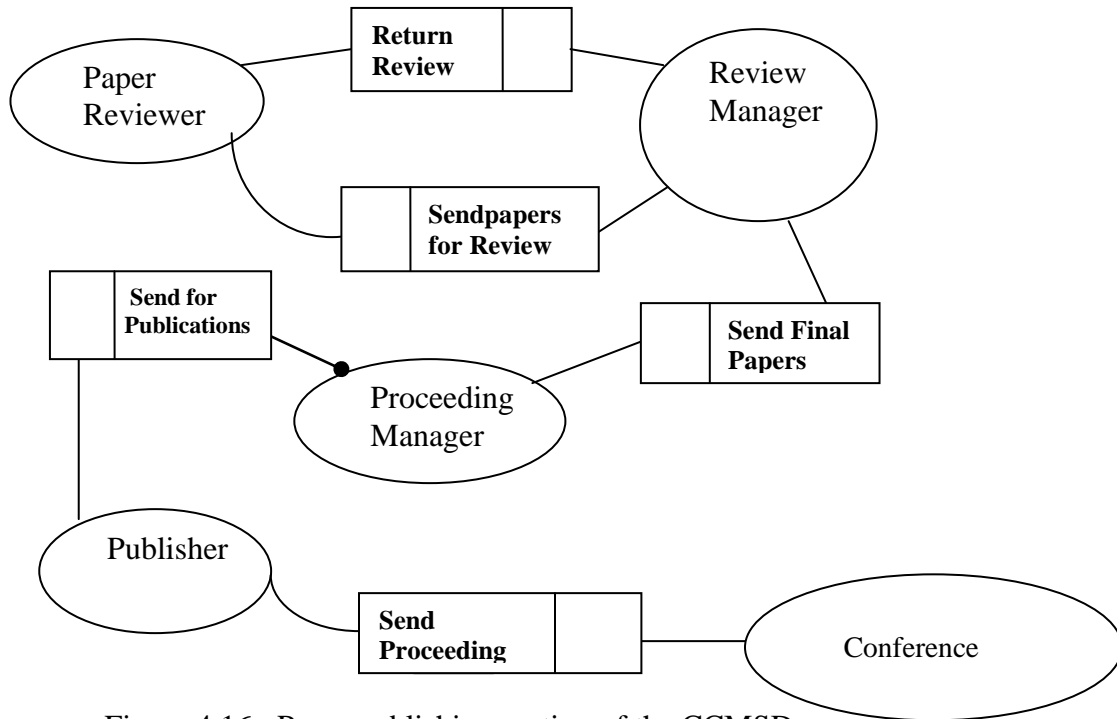
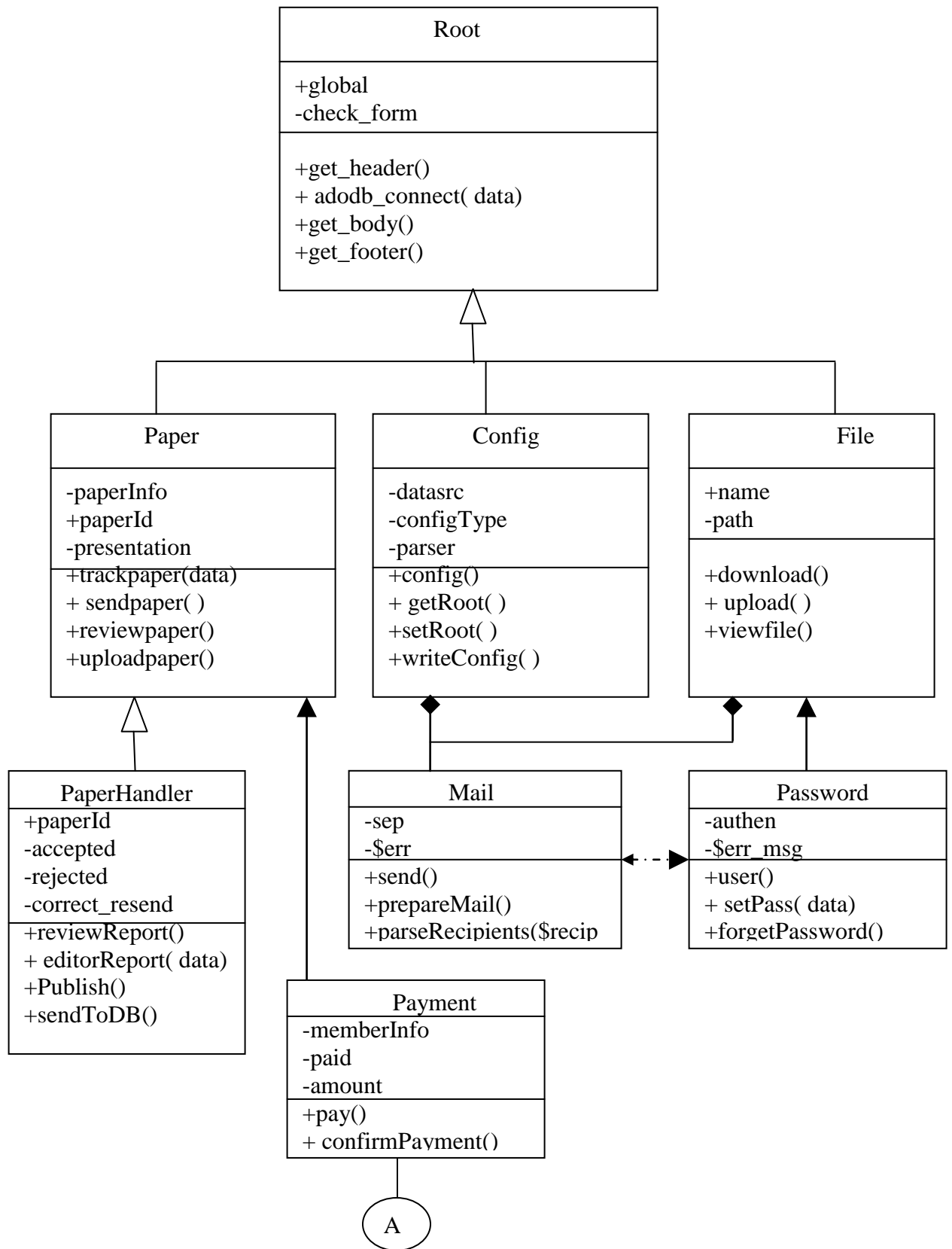


Figure 4.16: Paper publishing section of the CCMSD

Once the activity reaches the publisher, the proceeding will be prepared and published. The last phase involves **sending** of the proceeding to the conference.

4.5 UML Class Design of the System

UML is unified modeling language with different tools for use in designing systems. One of the major tools is the Class diagram which represents the classes used in the system and the relationships between the classes. In Figure 4.17, the UML design of the Conference Management System is presented. In the diagram there are major classes used by the system which include the File, the Config class and the Paper class. The File class makes sure that all the program modules that required file download, file upload and file view are provided with the core components to making the activities seamless. The major file activity is the conference and the paper that will be presented and this need to be well defined. The class Paper determines what the status of the paper is if it is to be uploaded, reviewed or tracked to see its progress by the authors in the system. The methods reviewpaper, trackpaper, uploadpaper and sendpaper make sure that these activities are well defined. The paperHandler inherits its functions from the paper class and extends its functionality with its own methods. The methods defined in the paperHandler includes reviewReport, editorReport, publish and sendtoDb. There is internal mailing system handled by the mail class as an aggregation of the file class and the Config class. There is a dependency relation between the File class and the password class. This indicate that before files are accessed users must use their password to gain authentication the activity is developed as a class to make it easy to setpass, retrieve forgotten pass via forgetpassword method and to identity user via the user method. The payment class enables the authors and conference attenders to pay fees and facilitate payment confirmation.



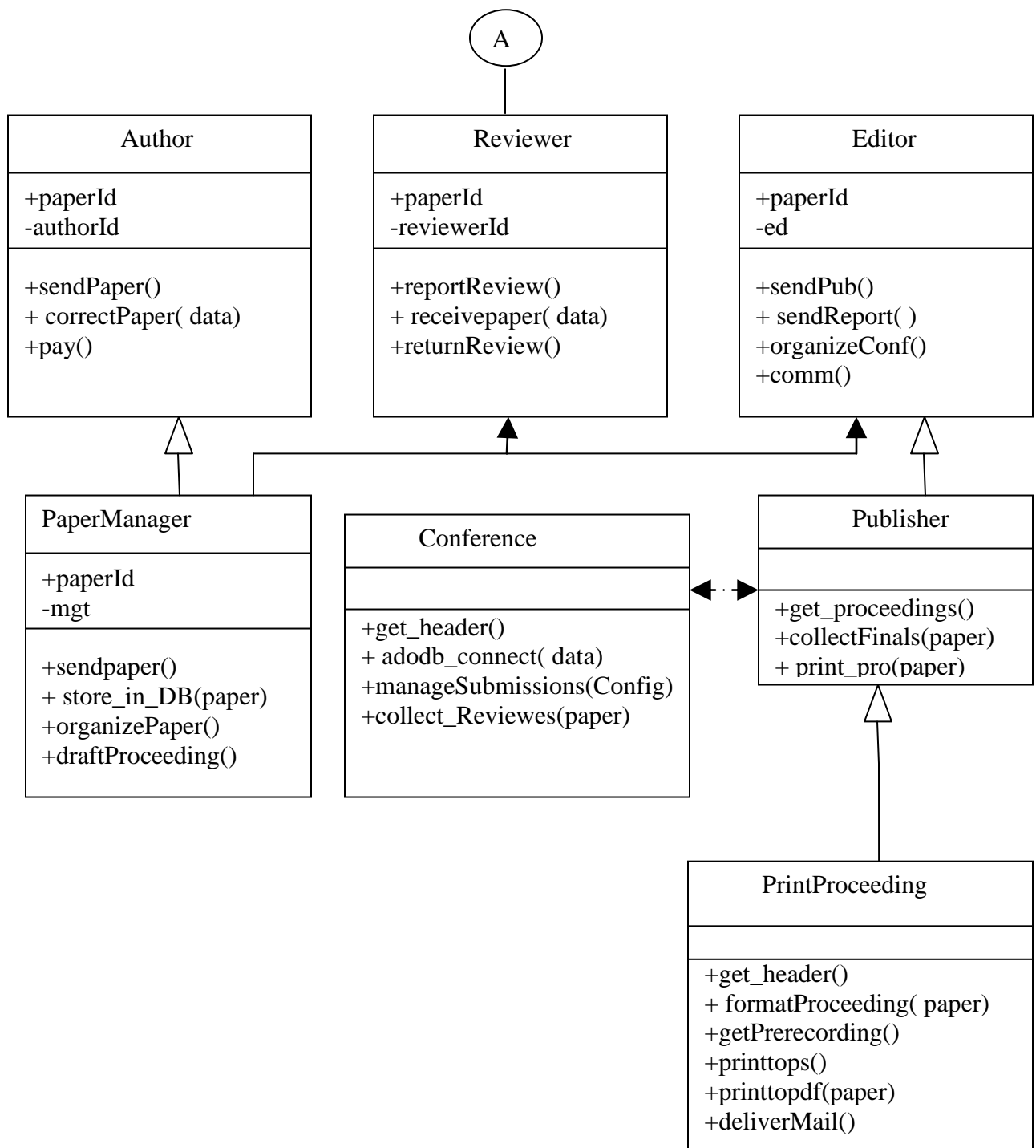


Figure 4.17 : UML Class Design of the Conference Management System

The Author class handles author pay, sendpaper and correctpaper involving the activity expected from the author. The reviewer class similarly handles reviewer report, receipt of paper for review and returning of reviewed report to the editor. The editor sends the report to author, send accepted papers for publication and communicate with other stakeholders via the editor class. The publisher class extends the Editor class and interfaces the conference class. The printProceeding class also extends the publisher class. The communication of these classes makes the system to be highly interactive and easy to implement using object-oriented implementation tools.

4.5.1 UML Interaction Design of the System

Interaction diagram is one of the fourteen types of diagrams of the Unified Modeling Language that describe how a group of objects collaborate in some behavior - typically a single use-case. In figure 4.18, the diagrams show a number of example objects, starting with the author use case and how it interacts with the system using the messages that are passed between these objects within the use-case. The model pictures a control flow with nodes that contain interaction with the CMS and the object-oriented database as shown in the diagrams. The Author log in to the system and his information is fetched from the database to authenticate the author. The Author also uploads paper which is saved as an object in the system. The paper is reviewed and the paper status on request can be displayed and the author can also make payment. The published paper can be sent to the Author's mail. When done the Author can also Logout from the system and get redirected to the Home page.

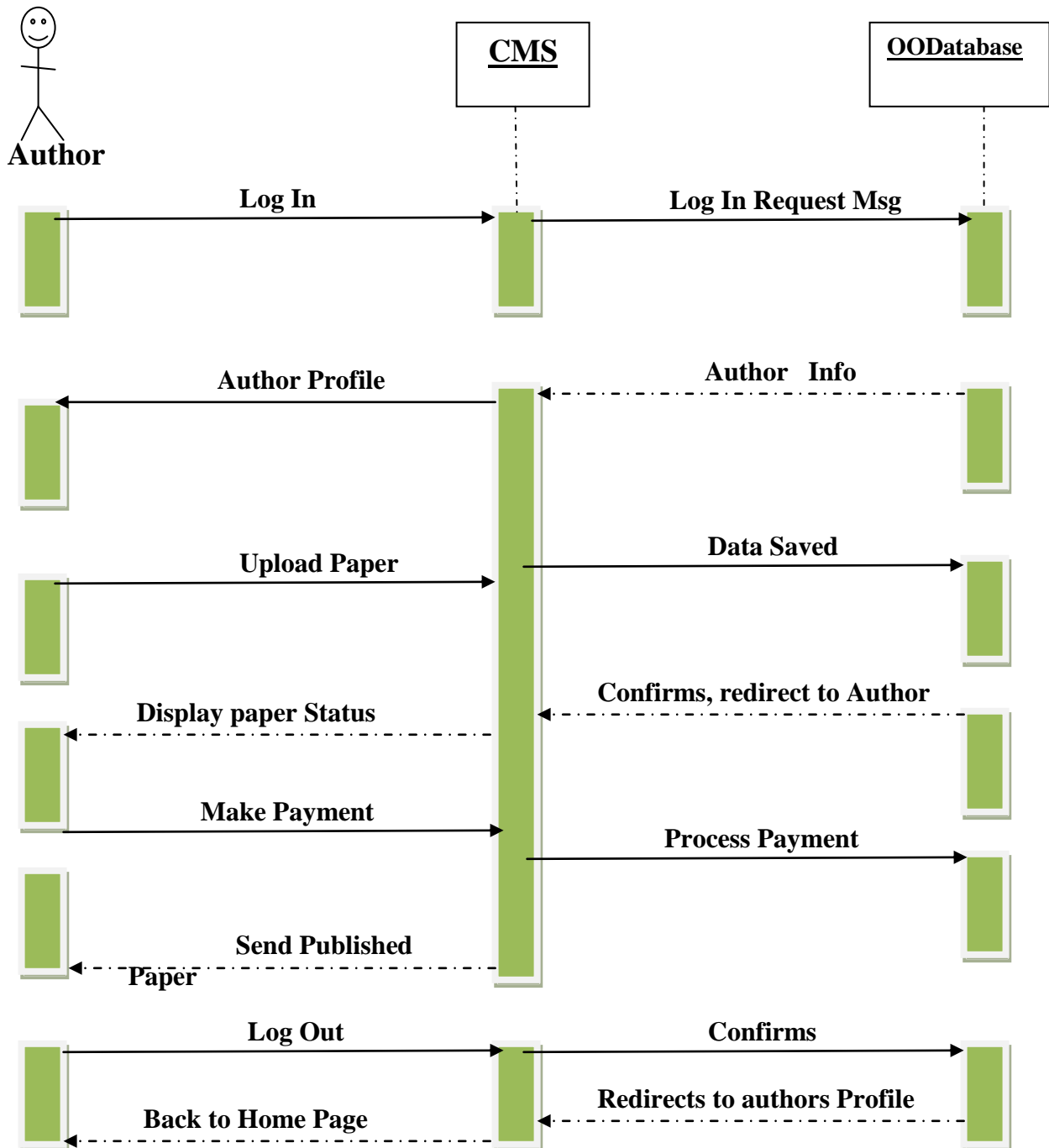


Figure 4.18 : Interaction Diagram showing Author interaction with the System

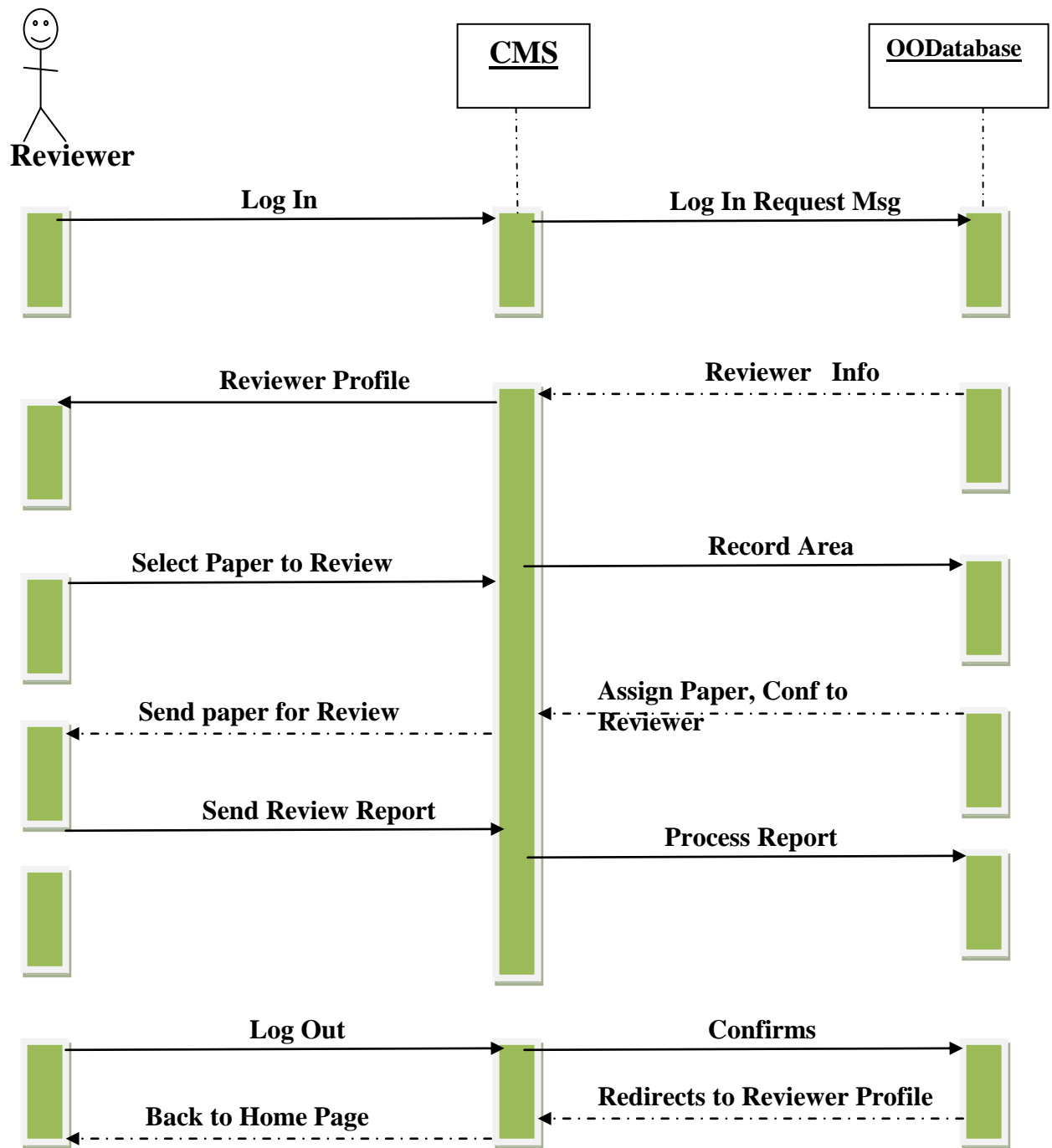


Figure 4.19 : Interaction Diagram showing Reviewer interaction with the System

The Interaction diagram in figure 4.19 shows the Reviewer actor and how it interacts with the system. The reviewer selects the area of specialization and the paper that he wants to review base on the reviewer menu button provided. The system uses the selections made and assigns paper to the reviewer and sends the paper to them for review. After review the reviewer send report into the provided area in the system for processing and for report of paper status to the authors. The reviewer must log in to the system and his information is fetched from the database for authentication before review will be carried out.

Interaction diagram in figure 4.20 typically use a single use-case the editor of the conference. In the diagram a log in action is used by the editor to access the system. The log in data is sent as messages that are passed between these objects within the use-case. The model illustrates how the editor interfaces between the author and the reviewer of the papers and also how final decision on the status of the paper is taken. The editor control flow defines the final outcome of the paper from the author whether it will even be submitted for review. When reviewed the outcome of the reviewers report is aggregated to enable the editor make final decision on the status of the paper. The status can be accept, reject or correct_and_resubmit based on the decision of the editor via aggregated report of the reviewers. The admin user or actor illustrated in figure 4.21 interact with the system by also logging in as an admin and providing system setting and configurations required for the smooth operation of the system. The configuration of the database and making sure that flow of operations is coordinated within the system are actions taken by the admin within the system.

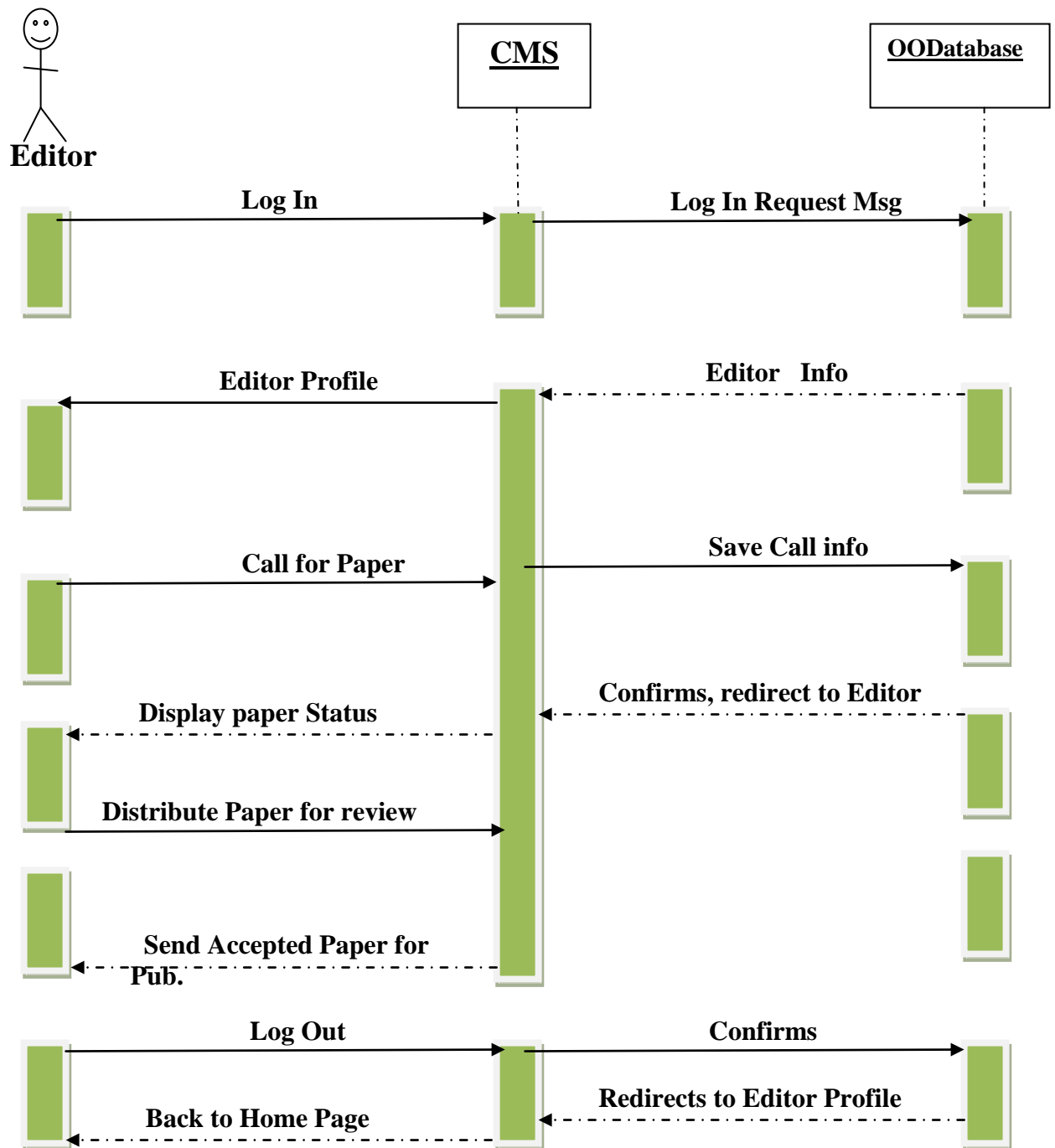


Figure 4.20 : Interaction Diagram showing Editor interaction with the System

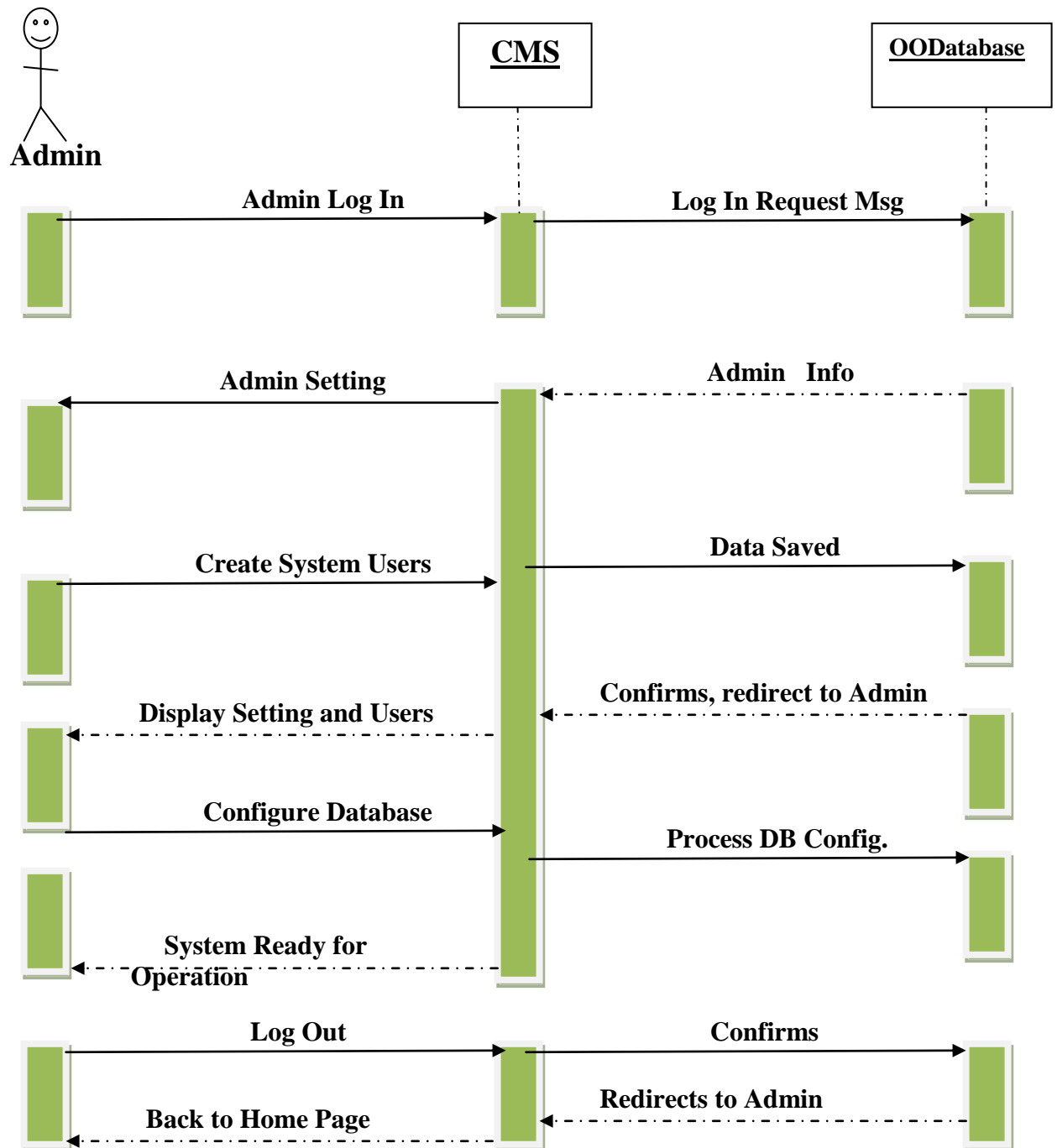


Figure 4.21 : Interaction Diagram showing Admin interaction with the System

In figure 4.22, the Interaction diagram shows the Publisher actor and who equally need to log into the system to extract papers which have already been approved for publication to prepare for printing of proceedings. The system by defaults already gets those papers ready for online viewing or publishing. Most conferences now still use printed proceeding so the publisher is still an important actor in the Conference Management System and its activity and interaction clearly defined. The system uses the list of papers selected for publication to assist the publisher to extract the papers. The downloaded paper can be printed as a proceeding and the publisher can logout of the system. The entire process work as a single entity even though each actor has a well specified duty within the system. The database aids the users and actors to ready find there activities and uses in object-oriented ability to assist in the navigation within the objects.

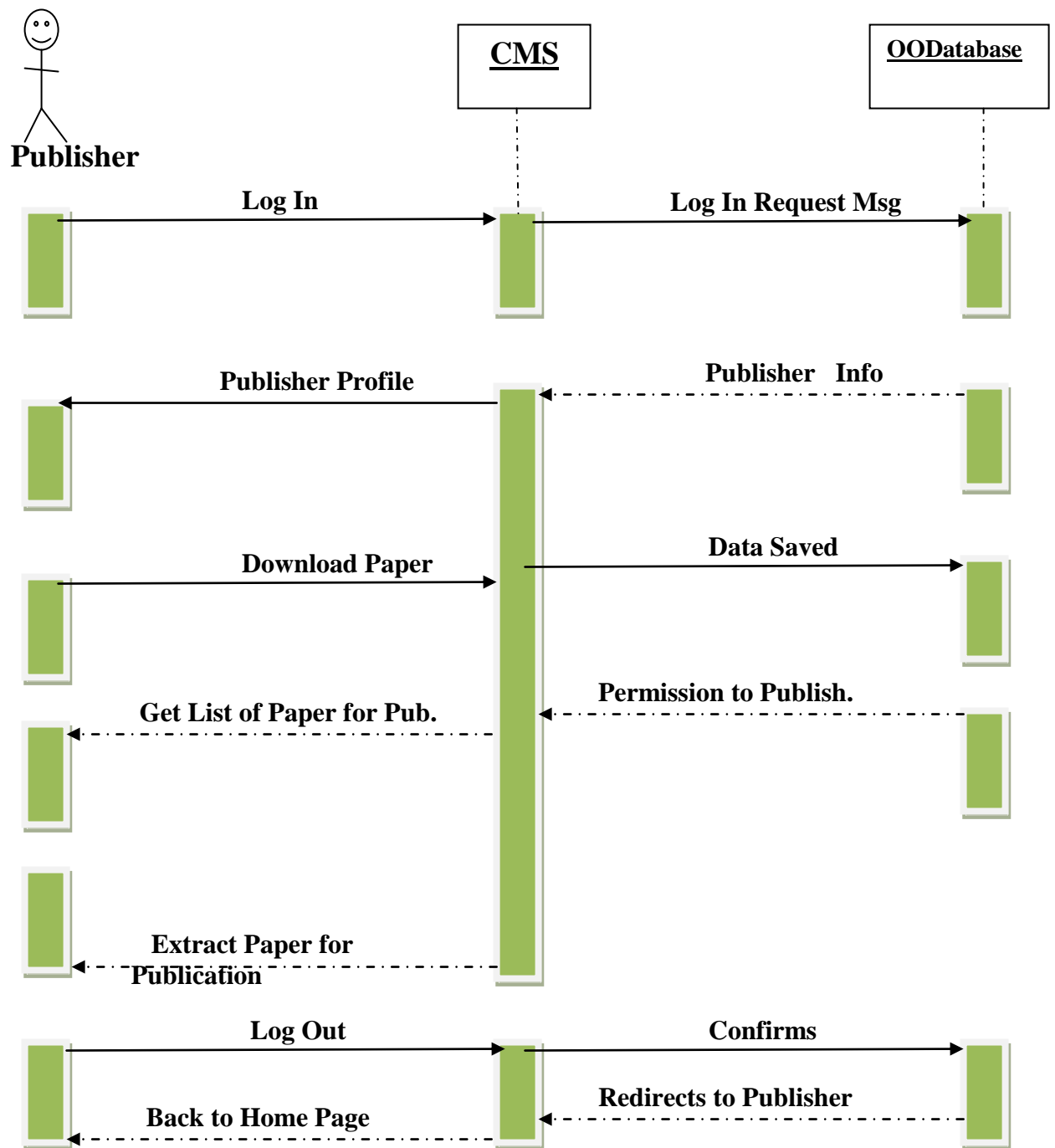


Figure 4.22 : Interaction Diagram showing Publisher interaction with the System

4.6 Formal Model Design of the Object-Oriented Database

The major form of storage in any application is usually implemented using a data base management system. A database management system (DBMS) consists of a software that organize the storage of data. A DBMS control the creation, maintenance, and use of the database storage structures of organizations and their users. It allows organizations to place control of organizational wide database Administrators (DBAS) and other specialists. In large systems, a DBMS allow users and other software to store and retrieve data in a structured way. Database management systems are usually categorized according to the database model that they support, such as the network, relational or object model. The model tends to determine the query languages that are available to access the database. In this conference system application being designed to solve the problem of the old obsolete one, the major storage unit is the MySQL database.

In this research we have pointed out the need to model databases technology in a way that encourages object-oriented communication with the application that drives the data. Schek (1990), at the early days of evolution of object-oriented databases believed that Object-oriented data models represents a current end-point in the evolution of data models, this is still true now. However developments in object-oriented data model formalism remain very slow due to the popularity of Unified Model Language (UML) which is more graphical. The slow pace at applying Object-oriented formal modeling in programming also affect the object-oriented database modeling. Object-orientation as a paradigm is based on the following five fundamental principles (Gottfried,1992):

- i) Each entity of real world like (Paper Author) is modeled as an **object** which has an existence of its own, manifested in terms of a unique **identifier** distinct from its value;

- ii) each object has **encapsulated** into it a **structure** and a **behavior**, the former is described in terms of attributes (instance variables), where attribute values, which together represent the state of the object, can be other objects so that complex objects can be defined via **aggregation**; the latter consists of a set of methods, i.e., procedures that can be executed on the objects;
- iii) the state of an object can be accessed or modified exclusively by sending **messages** to the object, which cause it to invoke corresponding methods;
- iv) objects sharing the same structure and behavior are grouped into **classes**; where a class represents a “template” for a set of similar objects; each object is an instance of some class;
- v) a class can be defined as a specialization of one or more other classes; a **class** defined as a specialization forms a subclass and inherits both structure and behavior (ie., attributes and methods) from its superclasses.

In object-oriented databases we need to augment these principles with additional requirements, namely the support of **complex objects**, i.e., highly structured information, and type-system orthogonality and extensibility; it should be possible to define new types at any time by applying given constructors to already defined types in a vastly arbitrary fashion.

The indifference of many system developers to formal model as a solid foundation of the system in which the listed requirements can be coded is clear in modern databases. We believe that formal model for OOD also need to capture the same intuition as models for other types of databases which include:

1. *Provision of adequate linguistic abstraction for certain database applications.* For instance the abstraction of the physical organization of the databases in order to support *physical data independence*.

2. *Provision of a precise semantics for a data definition language.* For instance the relational language SQL comprises a statement for creating tables in a relational database which the relational model underlying SQL provide semantics for such statement.
3. *The specification and operational part need to be inculcated.* Relational model specification refers to tables or relations as the structuring mechanism and relational model operational part as the operations of relational algebra.
4. *Computational paradigm is represented as a basis for formal investigations.* The relational model allows for in-depth investigation of database design or query languages. There is no reason why the same should not hold for object-oriented models.

4.6.1 Object-Oriented Formal Framework

OODBS is remarkably different from relational databases basically due to its strong theoretical foundation. In this research we discuss these theoretical foundations as they pertain to formal models for OODBS which is based on the notions of object, class, attributes, method (behavior) and inheritance. Deriving inspiration from Zhang et al., (2010) and (Gottfried, 1992) we give a formal definition of OODM to be a tuple S of the form

$S = (D_s, C_s, A_s, F_s),$ Where:

D_s , is a set of *type* names D (e.g., *string*, *integer*).

C_s , is a set of *class* names C .

A_s , is a set of *attribute* names $A_s = A_{ss} \cup A_{sc}$; an attribute can be either *simple attribute* A_{ss} when its domain is a type $D \in D_s$ or *complex attribute* A_{sc} when its domain is a class $C \in C_s$.

F_s is a finite set of *class declarations*; for each class $C \in C_s$.

F_s contains exactly one such declaration:

Class C is-a T_1, \dots, T_n type-is T ,

T denotes a *type expressive* built as follows:

$T \rightarrow D \mid C \mid \text{Union } T_1, \dots, T_k \text{End} \mid \text{Set-of } T \mid \text{Record } A_1:T_1, \dots, A_k:T_k \text{End} \mid f(): R$.

where:

The *is-a* part, which is optional, denotes inheritance relationships between Classes;

(ii) The *type-is* part specifies the structure of the class C through T ;

(iii) The *Union...End* part denotes a generalization relationship between a general class and several specific classes;

(iv) $f(): R$ represents a *method*, where f is the name of the method, and the type of parameter may be *null*, and R is the type of the result.

In listing 4.1, we show a simple *OODM S* modeling part of the Conference

Management System used in illustration of the concepts discussed in this research.

```

Fs={
Class Paper_Author type-is
    Record
        Name: String
        PaperTitle: String
        isInstitution(): String
    End
Class Technical_Committee type-is
    Union Program_Chair, Vice-Chair, Technicality, Protocol, Logistics
End
Class Program_Chair type-is
    Record
        Name: String
        Conference: String
        Coordinate: Set-of Papers
    End
Class Papers type-is
    Record
        Number: Integer
        isAuthor(): String
        Date_Submitted: Date
        Number-of-Pages: Integer
    End
Class Paper_Reviewer type-is
    Record
        Name: String
        PaperTitle:String
        Comment: String
    End
Class Publisher type-is
    Record
        Name: String
        isPublication():String
    End
}

```

Listing 4.1: A CMS OODM S_1

These databases have **schemas** which is the central notion of a conceptual database description. The Database schemas is pairs of the form

$$S = (S_{\text{attributes}}, S_{\text{method}})$$

We will examine each of the components of the pair in turn and how they interact to result to a vibrant OODB that will be relevant both in research and also in industrial application. The attributes of OODB schema can easily be captured from the relational model, and the methods or behavior generated from established classes of the databases.

4.6.2 Modeling the Attributes

Highly-structured information and complex data types are needed to model the attributes or structures of object-oriented databases since they serve as descriptions for domains of complex values. Let T be a type system then

- a) $\{I, S, F, B\} \subseteq T$ where I is integer type, S is String, F is float and B is Boolean type respectively.
- b) If A_i are distinct attributes and t_i tuple types such that $t_i \in T$, $1 \leq i \leq n$, then $[A_1 : t_1, \dots, A_n : t_n] \in T$ (“tuple type”);
- c) If $t \in T$, **then** $\{t\} \in T$ (“set type”)
- d) If $t \in T$, **then** $\langle t \rangle \in T$ (“list type”)

It is clear that the type system is made up of base types, from which complex types may be derived using attributes and *constructors*. The complex type requires nothing more but the attribute names. Other base types as well as additional or alternative constructors could directly be included. For instance a data type for persons participating in the conference management system (CMS) could be described as:

```
[ name: [first: string, last: string],
  age:integer,
  married: boolean,
  address: [street: string, city: string, zip: integer ],
  qualification: < string >,
  work: { [institution: [ name: string, location: string ],
            job_specification: string] }
  CMS-Paper: { paper-role: string, paper-no: integer, paper-status: string }
]
```

This example above models a data type for persons on the conference management system who have a name, an age, a marital status, an address, an academic qualification, a set of other work in a publishing company or academic institution and a (CMS Paper) role as author of a paper or paper reviewer, on a particular paper number which is on communication status, review status, accepted or rejected status.

4.6.3 Class Types (User Defined Types)

Beyond complex types is a possibility to share pieces of information between distinct types, or to aggregate objects from simpler ones. At the level of type declaration, an easy way to model this is the introduction of another reservoir of names called *class names*, which are additionally allowed as types or user defined object types. We can define this object types as complex types as above with the added condition:

(e) $C \subseteq T$, where C is a finite set of class names.

This implies that class names are a brand of user defined complex types since classes themselves have types. Hence objects from the underlying application all have identity

collected into classes and can reference other objects or share subobjects. Thus, classes are instantiated by objects by making class-name types to take object identifiers as value; in the same way as simple type like *int (integer)* type takes integer numbers as values.

For instance we can define the person data as a class $C = \{\text{Person, Address, Institute, and Paper}\}$. It is clear that both persons and companies of institute have address and children are in turn persons, types can be associated with these class names as:

1. Type of Class Person:

```
[ name: [ first: string, last: string],  
  Age: integer,  
  Married: Boolean,  
  Address: Address,  
  Qualification: <string>,  
  Work: { [institution : Institution, job_specification: string] }  
  cms-paper: CMS-Paper,  
]
```

2. Type of Class Address:

```
[ street: string, city: string, zip: integer]
```

3. Type of class Institution:

```
[ name: string, location: Address]
```

4. Type of Class CMS-Paper:

```
[paper-role: string, paper-no: integer, paper-status: string]
```

It is clear that the object-oriented paradigm uses subtyping in organizing information in such a way that a class is defined as a specialization of one or more other classes this is referred to as

inheritance. In this case a subclass **inherits** the database structure or attributes of a super class. Such a relation can be defined in various ways and one of such ways is the following:

Let T be a set of object types, a subtyping relation $t \subseteq T \times T$ is defined as

- i) $t \leq T$ for each $t \in T$,
- ii) $[A_1 : t_1, \dots, A_n : t_n] \leq [A'_1 : t'_1, \dots, A'_m : t'_m]$ if
 - (a) $(\forall A'_j, 1 \leq j \leq m) (\exists A'_i, 1 \leq i \leq n) A_i = A'_j \wedge t_i \leq t'_j$
 - (b) $n \geq m$,
- iii) $\{t\} \leq \{t'\}$ if $t \leq t'$,
- iv) $\langle t \rangle \leq \langle t' \rangle$ if $t \leq t'$

Using these specifications we can proceed to the definition for object-base schemas that can describe structure arbitrary complexity: A structural schema is a named quadruple of the form

$$S_{\text{struc}} = (C, T, \text{type}, \text{isa})$$

Where i) C is a finite set of class names,

ii) T is a finite set of types which uses as class names only elements from C .

iii) $\text{type}: C \rightarrow T$ is a total function associating a type with each class name,

iv) $\text{isa} \subseteq C \times C$ is a partial order on C which is consistent with respect to subtyping, i.e., $c \text{ isa } c' \Rightarrow \text{type}(c) \leq \text{type}(c') \quad \forall c, c' \in C$

The implementations typically add a number of additional features not included like single or multiple inheritance which are implemented differently on different languages for instance C++ implements both single and multiple inheritance while Java implements only single inheritance. Other concerns include

- i) implementation of attributes as functions
- ii) a distinction of class attributes from instance attributes (the former relevant to the class as a whole while the latter shared by all objects associated with a class)
- iii) a distinction between private and public attributes a different set of constructors
- iv) an explicit inclusion of distinct types of relationships between classes and their objects
- v) integrity constraints which represent semantic information on the set of valid databases instances.

4.6.4 Modeling the Method

The capturing of the method besides the structure is another important part of an object-oriented database. Classes have attached to them a set of messages, which are specified in the schema via signatures, and which are implemented as methods. Methods can also be inherited by subclasses, and message names can be overloaded, i.e., re-used in various contexts. A method schema can be defined as a named five-tuple of the form

$$S_{\text{behav}} = (C, M, P, \text{messg}, \text{impl})$$

Where

- (i) C is a finite set of class names
- (ii) M is a finite set of message names, where each $m \in M$ has associated with it a

nonempty set $\text{sign}(m) = \{s_1, \dots, s_t\}$, $t \geq 1$, of signatures; each s_h ,

$1 \leq h \leq t$, has the form $s_h: c \times t_1 \times \dots \times t_p \rightarrow t$

For $c \in C$, $t_1, \dots, t_p, t \in T$ (each signature has the receiver of the message as its first component),

- (iii) P is a finite set of methods or programs,

- (iv) $\text{Messg}: C \rightarrow 2^m$ such that

$$(\forall c \in C) (\forall m \in \text{messg}(c)) (\exists s \in \text{sign}(m)) \quad s[1] = c$$

(v) impl: $\{(m,n) \mid m \in \text{messg}(c)\} \rightarrow P$ is a partial function

In combining structural and behavioral schemas, we can finally obtain the object database schema has the form

$$S = (C, (T, \text{type}, \text{isa}), (M, P, \text{isa}, \text{messg}, \text{impl}))$$

S is called consistent if the following conditions are satisfied:

- i) $c \text{ isa } c' \Rightarrow \text{messg}(c') \subseteq \text{messg}(c) \quad \forall c, c' \in C,$
- ii) if $c \text{ isa } c'$ and $s, s' \in \text{sign}(m)$ for $m \in M$ such that $s: c \times t_1 \times \dots \times t_n \rightarrow t,$
 $s': c' \times t'_1 \times \dots \times t'_n \rightarrow t',$ then $t_i \leq t'_i$ for each $i, 1 \leq i \leq n,$ and $t \leq t',$
- iii) $(\forall m \in \text{messg}(c)) (\exists c' \in C) c \text{ isa } c' \wedge \text{impl}(m, c')$ is defined

Condition i) says that subclasses inherit the method of their superclasses. Condition ii) says that message-name overloading is done with compatible signatures. Lastly, condition iii) says that for each message associated with a class, its implementation must at least be available in some superclass. We will finalize the formal definitions by showing how object databases (sets of class instances or extensions, can be defined over a given schema: For a given object database schema S, an object database over S is a triple

$$d(S) = (O, \text{inst}, \text{val}) \text{ such that}$$

- i) O is a finite set of object identifiers,
- ii) $\text{inst}: C \rightarrow 2^O$ is a total function satisfying the following conditions:
 - a) if $c, c' \in C$ are not (direct or indirect) subclasses of each other,
then $\text{inst}(c) \cap \text{inst}(c') = \emptyset,$
 - b) if $c \text{ isa } c',$ then $\text{inst}(c) \subseteq \text{inst}(c'),$

- iii) $\text{Val}: O \rightarrow V$ is a function such that $(\forall c \in C) (\forall o \in \text{inst}(c)) \text{val}(o) \in \text{dom}(\text{type}(c))$.

In this chapter we have developed some designs for object-oriented databases and the formal model which can be easily understood and we have made case for the integration of the structural and the method design in such a way as to encourage free object-to-object communication between object-oriented programming language class objects and object-oriented database objects. This enhances interoperability in application execution when complex types are involved in modern system development.

4.7 Database Specification of the CMS

Database specification involve the clear description of the fields, entities and their types used in the development of the database management system for storing and retrieving data in the database. The database of the system is as illustrated in figure 4.17 in the PhpMySql environment. The database show thirty one tables inside the database and selected tables are explained in some detail in the following section. Some of the tables selected and explained are those once very close to the system functionalities as described in the design section of the system.

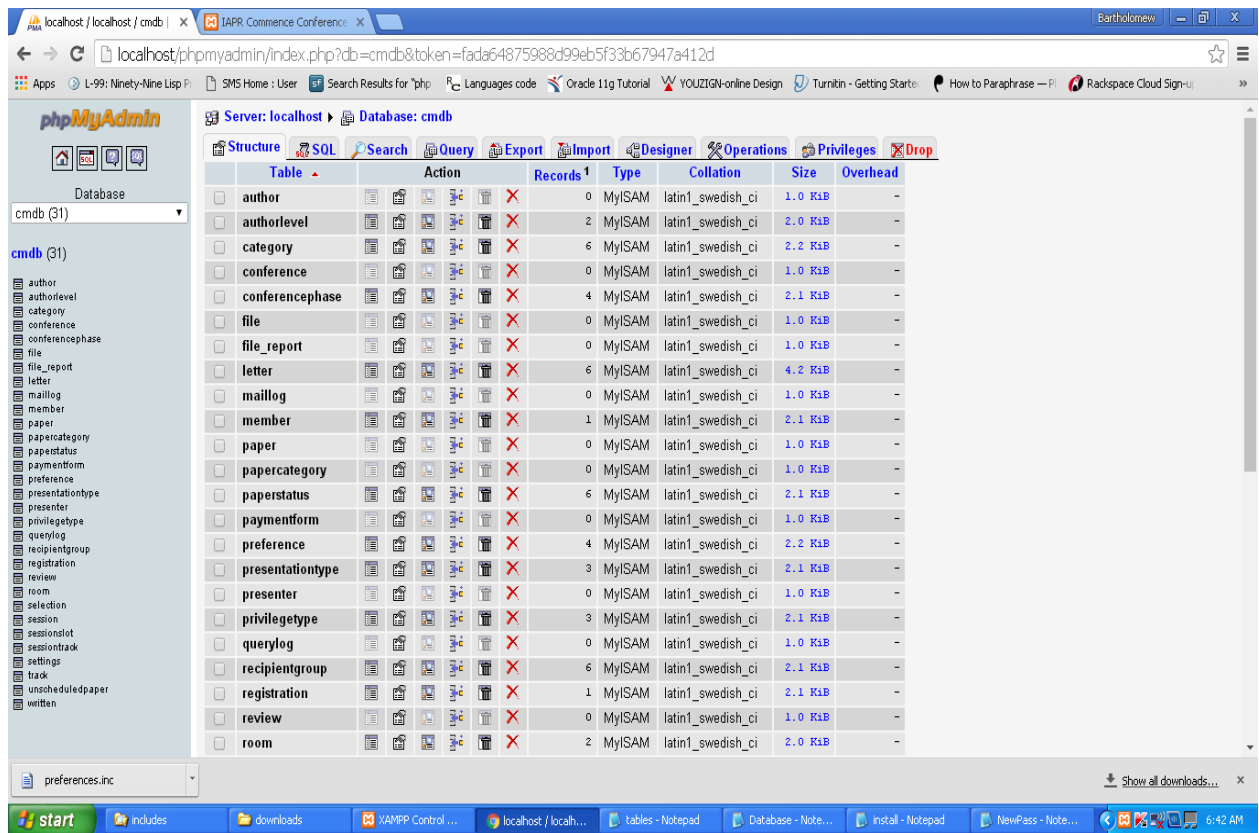


Fig. 4.23: Database Specification of the CMS system

The database contains 31 tables handling various data needed for the system as illustrated in figure 4.17, it is used in carrying out the various data storage and retrieval activities needed for the successful operation of the CMS system. They include *author*, *authorlevel*, *category*, *conference*, *conferencephase*, *file*, *file_report*, *letter*, *maillog*, *member*, *paper*, *papercategory*, *paperstatus*, *paymentform*, *preference*, *presentationtype*, *presenter*, *privilegetype*, *querylog*, *recipientgroup*, *registration*, *review*, *room*, *selection*, *session*, *sessionslot*, *sessiontrack*, *settings*, *track*, *unscheduledpaper* and *written*.

The databases each are specified based on their fieldnames, the data type specified for the data content of the field, the length specified by the developer of the databases and the action

expected to play by the field in the system-such as primary key, secondary key etc. Once the database is created the privileges needed to be specified.

4.7.1 Database Privilege Specification

The database privileges specified for the system can be grouped into four major types.

A) **Data:** These include privilege to Select Data from the database, Insert Data into the database, Update the database, Delete the database and Manipulate File-file import or export. The database user can be granted these privileges or denied the privileges.

B)**Structure:** These involve the privilege to change the structure of the database. These include *Create, Alter, Index, Drop, Create Temporary Tablr, Show View, Create Routine, Alter Routine, Execute, Create View, Event And Trigger* for any privilege granted in the structure group of privileges, the associating box is checked and registered.

C)**Administration:** Administration privileges are granted to users to allow them manage the databases effectively. They include : *Grant, Super, Process, Reload, Shut Down, Show Databases, Lock Table, References, Replication Client,Replication Slave, and Create User*. The user given any of these privileges is expected to manage the system with them. Other users are usually denied the privileges.

D) **Resource Limits:** These privileges allow the system user to be able to set maximum execution parameters for Queries, Updates and Connections per hour. It can also be used to set the maximum user connections to the database in other to stop the system from crashing. If the setting is left at zero then the limits are eliminated.

4.8 Database Dictionary

The data dictionary of few databases will be highlighted. These include:

- a) **Author:** This database table describes the person who prepares a paper for the conference. The author has five fields as illustrated in table 1.

Table 1 : A database description of the Author Table

FieldName	Type	Length	Action
AuthorID	Int	10	Primary Key
FirstName	Varchar	30	Not Null
MiddleName	Varchar	30	Not Null
LastName	Varchar	30	Not Null
Email	Varchar	50	Null

- b) **Conference:** The conference database table describes the detail about an upcoming conference so that authors who prepare a paper for the conference can send and prepare for the conferences. The conference table has thirteen fields as illustrated in table 2. It is important to note that the logfile is an object data feed into the database. In other to implement the object data, additional fields like filename, filesize and fileType are included.

Table 2: A database description of the Conference Table

FieldName	Type	Length	Action
ConferenceID	Int	10	Primary Key
ConferenceName	Varchar	100	Not Null
ConferenceCodeName	Varchar	20	Not Null
ConferenceStartDate	Date		Not Null
ConferenceEndDate	Date		Not Null
ConferenceLocation	Varchar	20	Not Null
ConferenceHostName	Varchar	100	Not Null
ConferenceContact	Varchar	50	Not Null
ConferenceContactName	Varchar	50	Not Null
LogoFile	longBlob		Null
Filename	Varchar	50	Null
FileSize	Varchar	50	Null
FileType	Varchar	50	Null

- c) **Paper:** The paper database table describes the detail about a sent paper for the conference so that each author's paper will be properly identified. The paper reviewers will need the paper data to handle their reviews and the editor needs it to manage its activities on the conference system. The paper table has fourteen fields as illustrated in table 3. The paperID, Title and PaperStatusID are important in identifying the state of a paper throughout the conference preparation period. The data type smallInt and Tinyint where used due the need for numbers expected to be below 4 or 6 digits.

Table 3: A Paper database description of the Conference Table

FieldName	Type	Length	Action
PaperID	Int	10	
Title	Varchar	255	
PaperAbstract	Text		
NumberOfPages	SmallInt	6	
PapersStatusID	TinyInt	4	
MemberName	Varchar	50	
Withdraw	Varchar	5	
Invited	Varchar	5	
Copyright	Varchar	5	
OverallRating	Float		
PresenterName	Varchar	50	
PresenterBio	Text		
TrackID	Int	10	
SessionTrackID	Int	10	

- d) **Review:** The review database table handles description of the detail about a paper review for the conference so that each reviewer (Member) will be properly identified and the paper being reviewed attached to the comments of the review. The reviewers will send their comments to the editor or comment administrator from where the overall evaluation will be presented. The paper author linked through the PaperID will also receive comments of the reviewer for final correction. The review table has nine fields as illustrated in table 4. The PaperID, MemberName, Comments and

OverallEvaluation are important in identifying the outcome of a paper review for the conference. The data type Tinyint where freely used except for the comments and member (reviewer name).

Table 4: A Review database description of the Conference Table

FieldName	Type	Length	Action
<u>PaperID</u>	Int	10	Primary Key
<u>MemberName</u>	Varchar	50	Secondary Key
AppropriatenessToConference	tinyInt	3	Null
Originality	tinyInt	3	Null
TechnicalStrength	tinyInt	3	Null
Presentation	tinyInt	3	Null
OverallEvaluation	tinyInt	3	Null
Comments	Text		Null
CommentsAdmin	Text		Null

- e) **PaymentForm:** The paymentForm database table describes the detail about the conference payment for the conference so that each registered person will be properly documented. The registered conferee will be identified and marked as paid having registered and paid for the conference. The formID connects the Form for payment and marks the conferee as registered and once payment is made the paid column will be marked. The PaymentForm table has four fields as illustrated in table 5. The FormID, RegisterID, Form and Paid fields are important in identifying the conferee and noting that he has paid for the conference. The object data type Blob was used for the form.

Table 5: A PaymentForm database description of the Conference Table

FieldName	Type	Length	Action
FormID	Int	10	Primary Key
RegisterID	Int	10	Not Null
Form	Blob		Not Null
Paid	tinyInt	1	Not Null

4.9 Program Development

In this section, the implementation of the conference management system is documented. The choice of the programming language, tools and technology that are useful in the development of the system are clearly addressed. The documentation of the process followed in the development and the process is also presented. The system requirement for the computer hardware and software for both the development and the deployment of the system and the testing process of the system were equally documentated and presented in this section.

4.9.1 Programming Language for New system Implementation

The programming languages used in the development of the new conference management system are PHP, HTML and SQL. These groups of languages are combined in the development of the system.

4.9.2 Reason for Using PHP-MySQL Technology in CMS

The main reason for using PHP-MySQL as the technology / language of implementation is that it is robust for various users. It is also scalable and capable of handling deadline rush in paper submission. The numbers of systems connected is handled with the extra performance that Apache provides. The technology also allows the application to use TCP/IP for packet sharing and file / information transfers across the system which is necessary for handling the

submitted papers in pdf or doc file format. The versatility of MySQL and PHP also offers lots of benefits that can help the user to customize tasking operations running on the server. This offers a great benefit above the languages that ran only on the CGI core in the server side and closed source languages that cannot be modified when the need arises.

PHP as programming language is increasingly being used in server technology. It is also a programming language model that is organized by objects and actions, data and logic. PHP allows the usage of different software methodology since it allows both procedural and Object-oriented programming. It can well identify objects sets of data and defines their relationship to each other. These objects are then generalized into a grouping called class. Actions into methods or sequences of logic are applied to classes of objects. Methods provide computational instructions and class attributes are the data members that is acted upon by the object. Objects interact in the system using specifically defined interfaces called messages.

These messages are sent across the network sockets which are distributed. Packets are also sent across the network using different system devices. A PHP interpreter executes the program on the source server and sends the result which is usually data or simple XHTML or other mark-up codes. The behavior of this kind of system offers portability and reusability that characterizes PHP and its related technology.

4.10 Installing PHP-MySQL

In the installation process, a user can download MySQL5.0 to download page, under the heading window downloads, select and download the release that includes the installer, download the file and unzip it and run the set.up.exe contained. Hence if a user decides to run the technology on window NT, 2000, XP, or Windows 7 or server 2003 and choose to use operating system, the user must create a file called (my.cnf in the root of c: drive to indicate where MYSQL is installed), then is open a notepad and type:

(Mysql)

Basedir =c: /mysql/

Datadir = c: /mysql/data/

WORKING WITH .cnf FILE ON WINDOW: The procedures are as follows:

- A) Open the registry editor
- B) Click start, Run and then type regedt32.exe to launch the editor
- C) Navigate to key-local-software-classes branch of register where a user will find a list of all the registered file type on the system
- D) Restart window for the change to take effect

4.10.1 PHP Installation

PHP is designed to run as a plug-in for existing web server software such as Internet information services, Apache. Testing dynamic web server requires equipping a computer with web server software so that PHP will have something to plug to.

Procedure:

- A) On start menu, point on Apache HTTP server
- B) Configure Apache server
- C) Edit the Apache httpd.configuration file to open the http.conf file
- D) Add load module php5-module cs/php/php5apache. dll
- E) Add module mod.php5.c
- F) Add type application/x-httpd-php.php
- G) Add type application/x-http-php-source.php
- H) In Apache2.0, the load module line must point to php5Apache2 dll and remove Add module line
- I) Point to directing Index, which tells Apache the name to use in terms of default page

J) Save the change and close Notepad

K) Restart Apache by restarting the Apache service in control panel, Alternatively a single bundle of Xampp, Wamp, reactor or other software bundles that have PHP,Apache, MySQL bundled together can be downloaded and installed and used for the same purpose.

4.11 Program Algorithm

The algorithm could be a pseudocode or a flowchart that represent the logic flow of the program operations. It shows the flow of control of the application for the implementation of the conference management system. The flow of logic makes it easy for the implementation since users can follow the steps to implement the system. This provides the developer a guide in the process of developing and integrating the system into design provided for the conference management system in the previous. The algorithm also helps in the process of providing the avenue for the system to process its major task using the steps outlined in the system.

4.11.1 CMS System Pseudocode

Step1. Start

Step2. Define all the server variables

Step3. Declare variable for the conference management components

Step4. Develop User interfaces

Step5. Place Call for Paper data Components in the XHTML document.

Step6. Create the Style Sheet Component for system Presentation

Step7. Create Database Components for CMS Content

Step 8 Assign fields' variable

8.1Assign content record parameters for the CMS

8.2 Create database components

8.3 Store components using necessary SQL query

Step9. Using DOM and ECMA Script facility

9.1 Perform User, Document, response form design

9.2 Assign form properties for client-server interaction

Step10. Create a Class of the objects and their attributes (properties)

Step11. Assign classes for property operations of events

Step12. Determine identifiers for server side PHP event properties

Step13. Determine MySQL-PHP Operations

Step14. Connect PHP logic with Queries and with GUI Browser forms and DOM components

Step15. Using MySQL

15.1 Link Input Object data to databases

15.2 Add content into databases

Step16. Using the forms object variable

16.1 Integrate form object with modules.

16.2 Test Interaction

Step17. Test Net Connection String

Step18. Initialize file Upload

Step19. Check file size

Step 19.1 if (size < max_size) then

Upload file

Else

Reject upload

Endif

Step20. If fileupload is successful then

Step 20.1 Create storage for uploaded papers

Step 20.2 Store paper files in approved other

Step 21. Deploy in the Web Server for Execution

Step 22. Stop

4.12 System Requirement

The running of the program and the implementation of the conference management system demand an effective computer system with the following minimum requirement:

4.12.1 Hard Ware Requirement:

- ♦ A processor machine of Pentium IV standard with 800MHz speed
- ♦ A memory (RAM) size of 1GB or above
- ♦ A Video Graphic Adapter (VGA) screen with 24bit high color or 286 color
- ♦ A disk space (Hard disk) of not less than 10GB
- ♦ A server Machine

4.12.2 Software Requirement:

- Windows or Linux Operating System
- A Javascript Enabled browser of Internet Explorer 5 and above or FireFox 2 and above, Opera 9 and above, Google Chrome or Hot Java.
- Apache 1.3 or 2.0 Server running on the local host or on the remote machine
- MySQL server running on the local host or on the remote machine
- PHP Nuke Engine Running on the Apache Web Server

4.13 System Testing and Result Discussion

This is the testing of the program to make sure that the software works when subjected to varying conditions. The software requirement and hardware requirement specify the environment where the testing is carried out.

4.13.1 Test Plan

Since the system is object-oriented the following where the plans specified, used and followed for the testing.

- i) Class Testing
- ii) Package Testing
- iii) Data Communication Testing
- iv) Integration Testing

i) Class Testing

When a class is created it is checked against the properties and methods and the methods are tested against the stub where the classes are instantiated and separately tested. The tested classes were also supplied with dummy data to see how the data communicates with the methods in the class. The classes developed in the dissertation were tested and each of the tested class worked as expected.

ii) Package Testing

Packages are special object-oriented units that make sure that all the classes that have similar functionality are kept together for easy operations. Packages like database communication, input, output and conference activities packages contain similar classes that are kept together for easy operations and clean code handling. The classes that were interacting within the packages were tested package by package. The packages tested show good performances and

were able to properly communicate with one another within the package where the testing was carried out.

iii) Data Communication Testing

This involves the testing of the object-oriented code with real data in the database. During the class and package level testing dummy data were used for testing but at the data communication testing the real life database was used. In carrying out this test, the class packages were separately tested to evaluate how they communicate with the data stored in the databases. It was observed that the data communicated very well with the packages. When insert operation was called it added data properly into the database and when the query to fetch data was called data was also properly extracted from the database.

iv) Integration Testing

In integration testing packages are combined the way it was presented in design so that they can be tested as a whole application. The integration testing in this dissertation as seamless due to the fact that a successful package testing and data communication testing facilitated execution. The classes and packages did not lose their reliability during the test rather they performed efficiently. In the subsequent test it was discovered that it is better to keep the admin login separate from the general user login due to conflict between these similar modules that carry out login but for separate target users in the system. With this experience the admin login was separated. The system was left to determine whether the user is Author, Reviewer, or ordinary visitor to the application. The system also segregated between the database admin and the editor of the conference.

4.13.2 Test Data

The system was tested using both real life data and prototype dataset. The first set of data was the data generated from past Nigerian Computer Society Conferences. The data comprises of

title of conference, detail location, and sub section of areas of discussions and call for papers for the conferences. Some of the data that was provided were used for the purpose of testing the system on the Conference Management System. However it was difficult to get the data that is expected to originate from the review process since the NCS conference was not reviewed using any application. On such cases where there are no direct data to use a prototype of simulated behavior of the reviewers during review was done. This was done by submitting documents to the system and filling of arbitrary areas of specialization for each selected reviewer. When the document was submitted and the title was specified the system was able to select the most appropriate reviewer that is not loaded to review the paper. This indicates that the program is working the way it was designed to work.

4.13.3 Actual Test Result versus Expected Test Result

In the testing of the system there are some expected results from the system which are presented based on the package of the system and where necessary the sub-packages. The actual test results was also presented which shows what the system was able to carryout when it was tested using the available data described in the previous section.

Table 6: Actual Test Result verses Expected Test Result

Package	Sub-Packages	Expected Test Result	Actual Test Results
Login	Admin	Expected to see the login form, enter the login data and successfully login or supply wrong data and be denialed Access.	When link was clicked, a form appeared and login in the user when correct data was supplied. When wrong data was tested it denialed access.
	User		
Author_Module		When clicked, the author page is expected to display where the author will supply his personal data and interact	By the time the button was clicked, the author page displayed where the author will supply his personal data and interact with the

		with the system.	system.
Reviewer-Module		It is expected that the reviewer will click the review button and see article ready for his review and do the review and submit review report.	When the reviewer button was clicked the reviewer page was displayed and a sample article ready for review and sample review report.
Editor_Module		It is expected that when this button was clicked the editor page will be displayed where the Editor activities are done.	When the editor button was clicked the editor page showed the page where the Editor activities are done.
Paper_assignment_and_view Module		In this module, the assigned papers are expected to be displayed with colors indicating the selection criteria.	The system automatically assigned papers to reviewers and displayed them with colors indicating the selection criteria.
Paper_Status Module		It is expected that when clicked by authors they will see the status of submitted paper whether reviewed and Accepted, Rejected or Accepted with Correction	When the Paper_status button was clicked it actually displayed the status of submitted paper whether reviewed and Accepted, Rejected or Accepted with Correction
Conference_Info Module		This is expected to show the public the Conference information and the various tracks available for the conference.	When clicked it showed the public the Conference information and the various tracks available for the conference.

The program starts with a file that is a start point known as index.php that directs the user on the processes involved in handling different system operations. The user follows the instruction of the file on the browser to navigate the system. Before commencing the navigation the user must register by supplying his/her personal information that is stored as

content in the system. After registration users can login using the log in form provided on the GUI on the browser. Once the user have logged in he can send his paper by uploading them into the system. He can also get information from the system such as submission deadlines and order conferencing information and react appropriately.

In figure 4.18, the illustration show a user registration window which the application provide for the users of the system needed to register before they can use the system either as an author or a conferee who intends to attend the conference. Other users such as the Reviewers do not need to use this user registration since it is the admin that has the responsibility of nominating and registering its reviewers. Sometimes the admin can use the general mail to call for CV of Reviewers before deciding whether they can be registered as reviewers in the conference. The user needed fill all the requirements provided on the form after which the user can submit. The system registers the user but only activates the user from the users email message link sent to the user when the user registered.

The screenshot shows a web browser window with the address bar displaying 'localhost/cms/user/registration.php'. The page has a yellow header with the text 'Okereke, Ngozi Caroline' and 'User Account Setup'. Below the header, there is a hint: 'Hint: Use your email address as your user name - it is unique and easy to remember when you log in. Fields that have an asterisk * are mandatory. After account setup, your username and password will be e-mailed to your primary email address.'

The registration form includes the following fields:

- User Name * (or Email): vera
- First Name *: Vera
- Middle Name: Akanwa
- Last Name *: Okorie
- Organisation *: Oko Polytechnic
- Address 1 *: 5 Awkuzu street Oko
- Address 2:
- City *: Oko
- State/Province *: Anambra
- Postal Code *: 500009
- Country *: Nigeria
- Email (Primary) *: vera@gmail.com
- Email (Secondary):
- Phone (Primary) *: 08033778910
- Phone (Secondary):
- Fax No:

At the bottom of the form are 'Submit' and 'Reset' buttons. The footer of the page reads: '(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE'. The Windows taskbar at the bottom shows the date and time as 7:38 PM on 5/25/2014.

Fig. 4.24: User Account Setup

In figure 4.19, the log in windows is shown the user, admin or reviewer needed to login before he can get into the application.

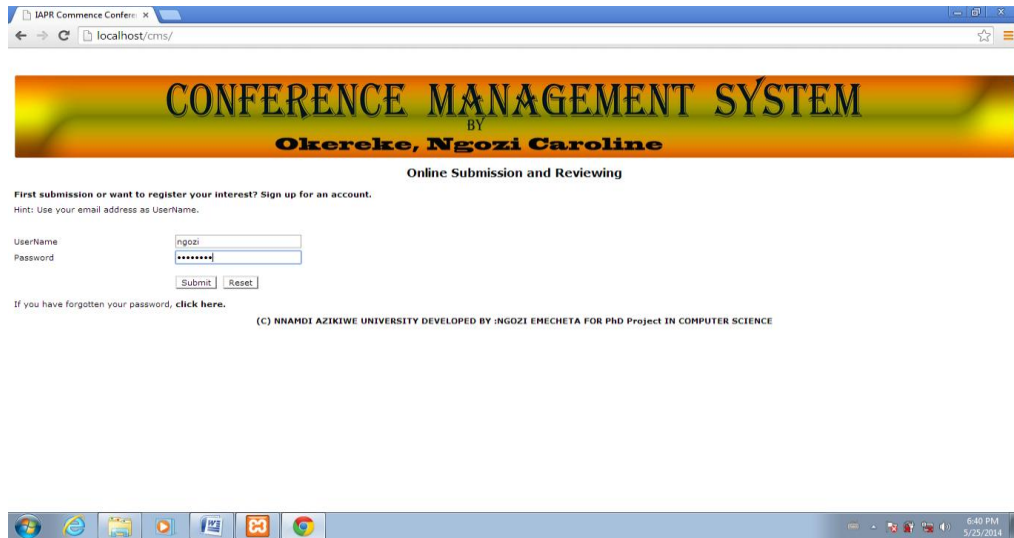


Fig. 4.25: Online Submission and Reviewing login

If the user needed to upload their papers on the application, the upload has to be based on the upload setting specified in figure 4.20. In this figure the paper size must be less than or equal to 10mb as specified in the upload setting. If the paper is larger in size it can be resaved in another format that will ensure better compression.

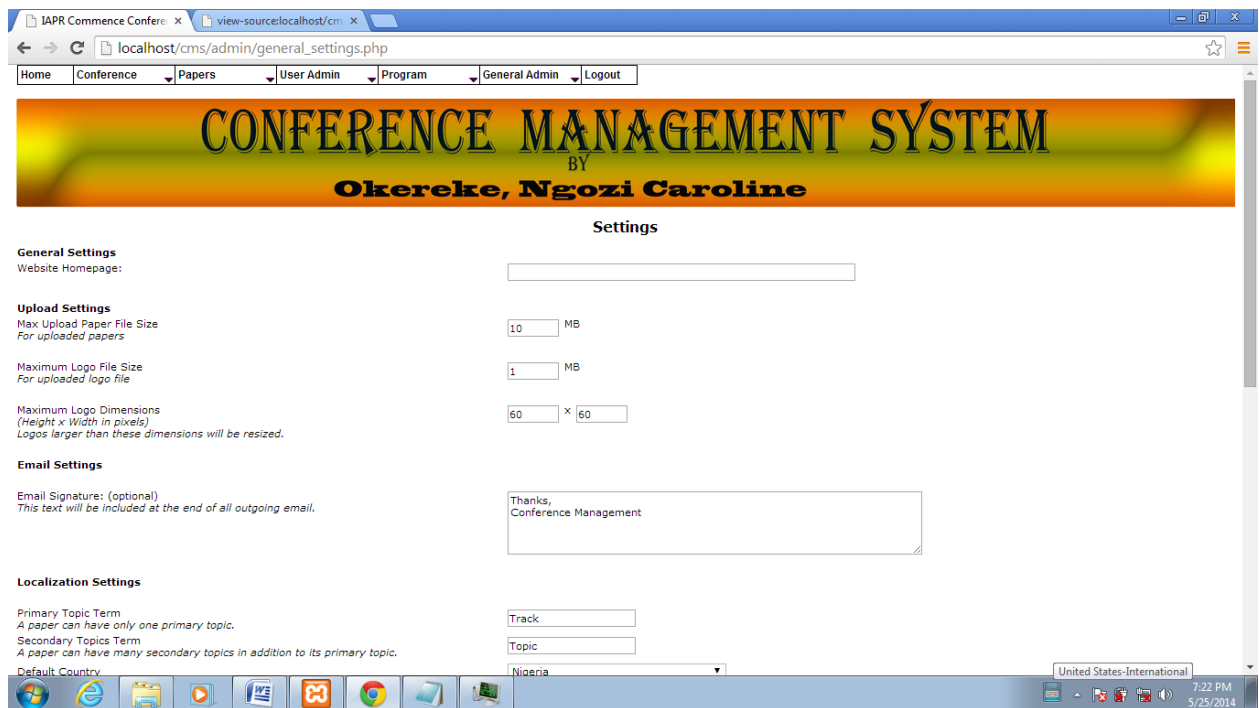


Fig 4.23 : Paper Upload Setting and Page

In figure 4.21, it is clear that when an Admin logs into the system he is provided with menu tools as shown on the top edge of the window. The admin uses this menus to perform different activities. One of the actions performed is the creation of accounts for the conference Reviewers. Figure 4.21 shows the page and the information needed to be fill for the reviewer which will enable him to login and have certain access and previlage to access papers assigned to him/her and to make the necessary reviews and report the observations and comments to the authors and the admin who may also double as the Editor.

CONFERENCE MANAGEMENT SYSTEM
BY
Okereke, Ngozi Caroline

Reviewer Account Setup

Reviewer Login Information

Fields that have an asterisk * are mandatory

Login Name *

First Name *

Middle Name

Last Name *

Email Address *

If you fill in the organization field, the new reviewer will not have to register himself.

Organization

☐ Inform the user now

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

Fig 4.26: Reviewer Account Setup

In figure 4.22 the reviewer preference selection is presented, each reviewer checks first, second and none check box to indicate the area the reviewer can be allocated review.

CONFERENCE MANAGEMENT SYSTEM
BY
Okereke, Ngozi Caroline

Reviewer Preferences

Total Reviewers: 10

Preferences for: **bartho** Batho Grandy Ekebon University of Leads

Track	1st	2nd	none
Pattern Recognition	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Computer Vision	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Medical Imaging	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Applications	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

Preferences for: **brandy** Brandy Demo Ebele University of Leads

Track	1st	2nd	none
Pattern Recognition	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Computer Vision	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Medical Imaging	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Applications	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

Preferences for: **emmanuel** Emmanuel Paul Gaibrel University of Poland

Track	1st	2nd	none
Pattern Recognition	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Computer Vision	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Medical Imaging	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Applications	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

Fig. 4.27: Reviewer Field of Review Preferences

In the appendix other screen shots of other operations are clearly show most of which are directly self explanatory. The screen shot contains mostly the core of the admin activities on the application. The conference managers and staffs also have a part of the system for their content information which includes hotel reservation on the city of the conference, public notices and other activities that revolve within the conference. Since the system is interactive and user friendly the paper review community can respond to all this conference demands by specifying their corrections and observations and opinion and the editor of the proceeding or conference coordinator can also make it available to authors. This enables the review activities to be closely monitored by the conferees creating transparency and confidence within the academic circles. In the system all documents submitted are stored in the system whether they are MS-Word document, picture files, PowerPoint presentation files or adobe PDF documents program files. The document window only provides an abstraction of the entire documents stored in the system. Reviewers report can also be made based on the report parameters provided. Other content include sundry information and content that the conference organizers want to add in the system.

4.13.4 Performance Evaluation

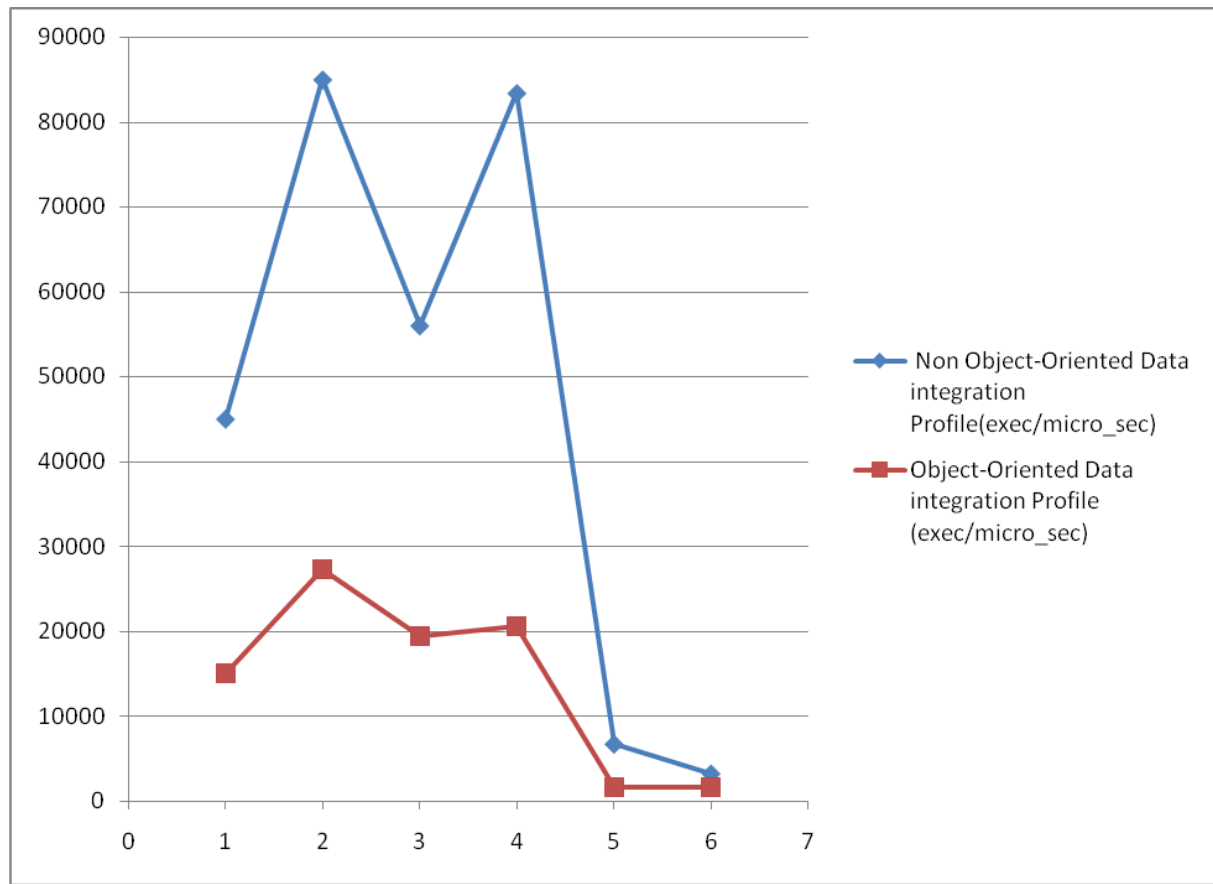
The existing object-relational database system separates the data stored in the databases and the objects created in the system process and in the file. However the system developed in this dissertation integrated the simple data stored in the database with the complex data defined in the classes and the objects. The issues here are how the variation affects the system, but this can be easily determined by carrying out some deep system test profiling. The test profiling carried out at the class and package level show major improvements in the integrated data system provided by the Object-Oriented database implementation.

Table 4.2: The Profiling of Class/Package Modules of Object-Oriented Integration Against Non-Object-Oriented Integration.

Package / Module Profiles	Non Object-Oriented Data integration Profile(exec/micro_sec)	Object-Oriented Data integration Profile (exec/micro_sec)
Author_Module	45000	1500
Reviewer-Module	85000	27300
Editor_Module	56000	1936
Paper_assignment_and_view Module	83405	20592
Paper_Status Module	6700	1560
Conference_Info Module	3200	5600

The result of the profiling was generated using PHP Profiler App and the generated result is as listed in the table and plotted in the graph in figure 4.26. In the figure, it is clear that the plot of Object-Oriented data integration showed lower time of execution compared with the Non Object-Oriented data integration. This is the same for class package/module 1, 2, 3 and 4 where higher data storage and retrieval is required. However, for module 5 and 6 where much data fetch were not needed or where only simple text is required the two seem to be similar.

Fig 4.26: A Plot of the performance of Object-Oriented Data Integration against Non Object-Oriented Data Integration Class/ Package Module Profile



4.13.5 Limitation of the System

The project really paid much attention on complex data storage and retrieval improvement with little attention on further improvement on the simple data storage. The assumption is that relational databases or other non Object-oriented databases have fully improved on the simple data types. But the reality is that there is still more need to further improve the simple data storage system especially the floating point text involving currency, double precision, and recent 8 to 32 digit cryptocurrency values. The limitation of the application is that if users are to pay for the conferency using blockchain tokens the system will find it difficult to categorise the value as either simple or complex value since currency is limited to 2 decimal point and double precision is not currency.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

In this research the need for object database and its ability to handle complex data has been presented. The ability of these object databases to communicate with object-oriented applications class-to-class, object-to-object and programming construct to construct are desired. The research looked at how the features could be incorporated into modern system development. This concept was tested in the research using a conference management system as a use case. The conference management system was analyzed based on the proposed concept, the design and the implementation was also carried out based on the design provided. Conference management system works to enhance the conference service delivery for enhancement and automation of conference processes and proceeding. It provides tools to enhance the activities carried out at pre-conference, during conference and at post conference. Conference services include the submission of papers, review of papers, enhancing access to conference information including information and communication technology in delivery of conference services. The conference management system serves as a conduit for provision of effective and affordable tools for bridging the gap between authors and conference organizers and reviewers of conference papers. This employs communication technology, which work to enable the involvement and empowerment of contributors to conference proceedings from all over the globe. It also encourages wider participation in the conference and the conference process. Hence Conference management system provides an easy access in the management of content on a site and as a simple tool in creating of conference content through submission and processing of pages, authoring based on the content creation, management of site and its page including the total maintenance of conference web application through the development

of CMS using AMP technology. The CMS system was basically selected as a use case due to its large usage of databases and the need for these databases to communicate with application objects and all the user objects within the application usage environment.

5.2 Conclusion

In this research, a formal model for the development of object database systems using Conference Management System (CMS) as a use case was developed. This model will serve as a guide for database management vendors and researchers in development of real life object database systems for use in the businesses and organization. We also designed object-database architecture for the CMS using Object-Role Modeling. Object-Role modeling is tailored towards developing object based representation of database entities.

It has been proven that it is easier for third parties to understand and represent entities attributes and their interactive behaviors when compared to entity relationship models (ER model). In the design ER models are less suitable for formulating, transforming, or evolving a design. ER diagrams are further removed from natural language, cannot be populated with fact instances, require complex design choices about attributes, lack the expressibility and simplicity of a role-based notation for constraints, hide information about the semantic domains that glue the model together, and lack adequate support for formal transformations (Terry, 2012). This is why we have used the Object Role Models in the design though it seems to be relatively new to academic designs.

The research work was finally implemented using the CMS design produced with object data and associated database. The implementation was done using PHP-MySQL programming language and technology due to the object-oriented nature of PHP and the object-orientation features currently imbedded in MySQL. Though we may not have all the features required in the DBMS but there are certain features that we have to code manually.

The implemented system is a tool of conference communication and publication information dissemination that empowers authors and publishers alike and bridge the gap between conference paper call and conference proceeding publication. This can boost quality of conferences by widening the submission scope and also the paper review scope for submitted papers. It also provides information, knowledge and conference quality as was never possible before. In this realm, conference organizers have a duty to provide services as efficiently as possible and to make the conference participants have value for their time and money in participating in conferences. Thus conference management system is not just about developing a web application; it is also the basic tool developed to control the function of pages with the transferring of information to enable proper delivery of conference services. This has also served as a tool for implementing the novel concepts developed in this research.

5.3 Contribution to Knowledge

This project has made some contributions which include:

- i) The design of formal schemes and models for object-oriented databases which we believe will form the foundation for development of complex applications that can easily communicate object entities, their attributes and behaviours with databases without any middleware or connectors. Messages can then be easily communicated between objects in application and database objects.
- ii) Object Role models of the system was equally developed which is modern tool for representing object database entity relationships in an easy to understand model
- iii) The high level model of the system was developed for representing complex concepts that are built into the development of the system in this project.

- iv) We have also developed a Conference management system in the thesis to illustrate the concepts proposed in the thesis. The application can be useful in handling conferences especially in Nigeria and developers can also get inspirations from the system in developing much more challenging applications.

5.4 Recommendation

The need for object-databases cannot be emphasized and the need to get them ready for communication with the object-oriented applications that drives them especially in contemporary system development where object-oriented application development remains the in thing in modern application development. Developers can benefit from these concepts and application quality assurance group can also use the concept in recommending database for use by organizations. We also recommend the research in this work to academics that have the need of simplifying their development process as well as implementation of systems that need complex data. The conference management system developed in the work can also benefit academic conferences and the development process can also be used in the development of more complex systems using AMP technology and other facilities used in this research. The system developed in this research can also enhance the efficient and effective delivery of conference services in the academics and other organizations that handle conferences that are based on paper submissions. We recommend this research to the academic conference organizers and professional organizations such as Computer Society of Nigeria (NCS) and other professional organizations that organize conferences across different issues for real life deployment so that the benefits inherent can be derived in our society. We also recommend the work to students who wish to carry out other research work related to conference management system development so that they can get inspiration from this work

in the process of developing and electronic article and journal publications at a more advanced level.

REFERENCE

- Acharya, S., Alonso, R., Franklin, M., and Zdonik, S. (1995). Broadcast disks: Data management for asymmetric communication environments. In Proceedings of the ACM SIGMOD Conference on Management of Data, 199-210
- Acharya S., Franklin, M., and Zdonik, S. (1996). Perfecting from a broadcast disk. In Proceedings IEEE Conference on Data Engineering, 276-285
- Acharya, S., Franklin, M., and Zdonik, S. (1997). Balancing push and pull for Data broadcast. In Proceedings of the ACM SIGMOD Conference on Management of Data, 183-194.
- Adam F. (2015) NoSQL For Dummies, Published by: John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030-5774.
- Aho, A., Sethi, R., and Ullman, J. (1987). Compilers: Principles, Techniques and Tools. AddisonWesley.
- Aksoy, D. and Franklin, M., (1998). Scheduling For large-scale on-demand data broadcasting. In Proceedings IEEE INFOCOM Conference (San Francisco, CA, March).
- Aljanaby, A., Abuelrub, E. and Odeh, M (2005). A survey of Distributed Query optimization. The International Arab Journal of Information Technology. 2(1), 48-57.
- Atkinson, Malcolm et al, (1989). The Object-Oriented Database Manifesto. In Proceeding of the First International Conference on Deductive and Object-Oriented Databases, 223-240.
- Atkinson, M., & Bancilhon, F., & DeWitt, D., & Dittrich, K., & Maier, D., & Zdonik, S. (1995). The Object-oriented Database System Manifesto, [On-line].
- Babb, E. (1979). Implementing a relational database by means of specialized hardware. ACM Transactions on Database Systems 4, (1), 1-29.
- Bachman, C. W. (2000) Integrated Data store Data Base study, second symposium on Computer-centered Data Base Systems, phoenix, Arizons
- Bancilhon, F; Delobel, I. and Kaneltakis, P. (1992). Building an object-oriented Database System: The story of O2 margan Kaufmann Publishers, ISBN 1-55860 – 169 – 4.
- Bernstein, P., Goodman, N., Wong, E., Reeve, C., and Rothnie, J.(1981).Query processing in a system for distributed database (SDD-1). ACM Transactions on Database System 6, 4, 602-625.

- Betts, B. (1997). Objects of desire? Computer Weekly, [On-line].
- Braumandl, R., Kemper, A., and Kossmann, D. (1999). Database patchwork on the internet (project demo description). In Proceedings of the ACM SIGMOD Conference on Management. of Data, 550-552.
- Braumand R., Konrad Stocker, Donald Kossmann, and Alfons Kemper,(2001) “Integrating Semi-Join-Reducers into State-of-the-Art Query Processors”, Proceedings of the 17th International Conference on Data Engineering, HYPERLINK IEEE Computer Society Washington, DC, USA.
- Burleson, D. (1994). OODBMSs gaining MIS ground but RDBMSs still own the road. Software Magazine. 14(11), 63.
- Ceri, S. and Pelagatti, G. (1984). Distributed Database- Principles and Systems. McGraw- Hill Inc., New York, San Francisco, Washington, D.C
- Checkland P. and Helwell S. (1998). Information, Systems, Information systems: Making Sense of the field. Chichester, West Sussex :John Wiley and Sons ISBN 0-471-95820-4, 86-89.
- Chen M. S. Yu p. S. and Wu K. L.(1996). Optimization of parallel Execution for multi-join Queries. IEEE Transaction on knowledge and Data Engineering, 6, 3.
- Codd E. F. (1970) Relational Model of Data for Large Shared Data Banks, IBM Research Laboratory, San Jose, Communications of the ACM 13(6) 377-387
- Data Integration Glossary (2001). US Department of Transportation.
- Dewitt, D., Fattersack, P., Maier, D., and Velez, F. (1990). A study of three alternatives workstation server architectures for object- oriented database systems. In Proceedings of the Conference on Very Large Data Base (VLDB), 107-121.
- DeWitt D. J. and Gray J. (1992). Parallel Database systems: The Future of High performance Database Systems. Communication of ACM, 35, (6) 85-98.
- Doyle, L. (2000) Content management system workshop Report, Fourth Institutional web management, University of Bath. UK,
- Distributed Query Performance (2012) .[http://infocenter.sybase.com/help/index.jsp?](http://infocenter.sybase.com/help/index.jsp?Distributed%20system%20(2012).http://www.ablongman.com/samplechapter/013095683x.pdf)
- Distributed system (2012).<http://www.ablongman.com/samplechapter/013095683x.pdf>,
- Eickler, A., Gerlhof, C., and Kossmann, D. (1995). A Performance evaluation of OID mapping techniques. In Proceedings of the Conference on Very Large Data Bases (VLDB), 18-29.

- Eickler, A., Kemper, A., and Kossmann, D. (1997) . Finding data in the neighborhood. In Proceedings Of The Conference On Very Large Data Bases (VLDB), 336-345.
- Elmasri, R and Navathe, S. B. (2000) Fundamentals of Database Systems. Reading. MA, Addison-Wesley.
- Eric P, Wenny R. J And David T. (2003). New SQL Standard For Object-Relational Database Applications, IEEE Computer Society SIIT2003
- Falkenberg, E.D. & Oei, J.L.H. 1994, 'Meta-model hierarchies from an Object-Role Modeling perspective', Proc. First Int. Conf. On Object-Role Modeling (ORM-1), eds T.A. Halpin & R.M.Meersman, Magnetic Island, Australia, 218-227.
- Ferguson, D., Nikolaou, C., Sairamesh, J., and Yemini, Y. (1996). Economic models for allocating resources in computer systems. In S. Clearwater (ed)., Market based Control of Distributed Systems. World Scientific Press.
- Franklin, M., Carey, M., and Livny, M. (1993).
Local disk caching for client-server database systems. In Proceedings of the Conference on Very Large Data Bases Systems (VLDB) (Dublin Ireland, Aug), 543-554.
- Franklin, M. and Jonsson, B., and Kossmann, D. (1996.) Performance tradeoffs for client-server query processing. In Proceedings of the ACM SIGMOD Conference on Management of Data (Montreal, Canada, June), 149-160
- Franklin, M and Zdonik, S. (1998.) Data in your face: Push technology in perspective. In Proceedings of the ACM SIGMOD Conference on Management of Data (Seattle, Wash, June), 516-519.
- Fuxman A., M. Pistore, J. Mylopoulos, and P. Traverso (2001). Model checking early requirements specifications in Tropos. In IEEE Int. Symposium on Requirements Engineering, 174–181.
- Garvey, M. A., and Jackson, M. S. (2010), " Object-Oriented Databases", Information and Software Technology, 31,(10), 524-525.
- Gottfried V. (1992) On Formal Model for Object-Oriented Databases, Journal of Fachbereich Mathematik, Arbeitsgruppe Informatik , 22-40
- Graefe, G. (1990.) Encapsulation of parallelism in the volcano Query processing system. In Proceedings of the ACM SIGMOD Conference on Management

- Data (Atlantic City, NJ, June), 102-111
- Graefe, G. (1993). Query evaluation techniques for large databases. *ACM Computing Surveys* 25 (2)73-170.
- Gray, J et al., (1996).Data cube: A relational aggregation operator generalizing group- by, cross –tab, and sub-total. In *Proceedings of the IEEE Conference on Data Engineering* (New Orleans, LA, Feb.), 152-159
- Gray, J., Bosworth, A., Layman, A., and Pirahesh,H. (1996).Data cube: A relational aggregation operator generalizing group- by, cross –tab, and sub-total. In *Proceedings of the IEEE Conference on Data Engineering* (New Orleans, LA, Feb.), 152-159
- Gupta, A. (2009). Database Management System in the Practical Approach to SQL & PL/SQL. Daryaganj Delhi: S. K. Kataria & Sons.
- Han, W. S, Lee K. H., and Lee B. S. (2003), “*An XML Storage System For Object-Orientedobject-Relational DBMSs*”,*Joumalof Object Technology* 2(1), 113-126
- Henderson-Sellers, B., and Edwards, J. (1994). Book two of object-oriented knowledge: the working object: object-oriented software engineering: methods and management. Sydney, Australia: Prentice Hall, 30-35.
- Hernandez, M. (1997). Database design for mere mortals. Reading, Massachusetts: Addison-Wesley.
- Hibatullah A. (2016). Evolution of Object-Oriented Database Systems, *Global Journal of Computer Science and Technology: Software & Data Engineering* 16(3)33-36
- Härder T., Stefan D., Nelson M., Bernhard M., Joachim T. (1998) Advanced Data Processing in KRISYS: Modeling Concepts, Implementation, Techniques, and Client/Server Issues, *The VLDB Journal* 7 (2), 79-95
- <http://download.oracle.com/docs/cd/b10500-01/server.920/a96520/concept,htm#49840>.
- Huh, S., Kim, H., and Chung, Q. (1999). Framework for change notification and view synchronization in distributed model management. *Omega*, August [On-line].
- Hunt A. and D. Thomas (2000). *Programming Ruby: The Pragmatic Programmer's Guide*.Addison Wesley Professional, 1st edition
- Ibraraki, T. and Kameda, T., (1984) Optimal Nesting for computing N-Relational joins. *ACM Transaction on Database Systems*, 9(3) 482-502.
- Inmon W. H. (1995). What is a Data Warehouse? Prism, 1
- Ioannidis Y. E. and Kang Y.C, (1990). Randomized Algorithms for

- Optimizing large join Queries. In Proceedings of the ACM SIGMOD Conference on management of Data, 312-321.
- Ives, Z.; Florescu, D.; Friedman, M.; Levy, A.; and Weld, D. (1999). An adaptive query execution system for data integration. In Proc. of ACM SIGMOD Conf. on Management.
- Jaspreet S., Kaur H. and Kaur K. (2013). A Review On Document Oriented And Column Oriented Databases, International Journal of Computer Trends and Technology- 4 (3) 338-344.
- Jenq B., Woelk D., Kim, W and Lee W., (1990). Query Processing in distribute ORION. In proceedings of the International Conference on Extending Database Technology (EDBT) 169-187.
- Keller, A, Jensen, R, and Agrawal, S (1993). Persistence Software: Bridging object-oriented programming and relational databases. In Proceedings of the ACM SIGMOD Conference on Management of Data, 523-528.
- Kim, Won (1990). Introduction to Object- Oriented Databases. The MIT Press, ISBN 0-262-11124-1.
- Kimball, R. and Strehlo, K. (1995). Why decision support fails and how to fix it. ACM SIGMOD Record 24 (3) 92-97.
- Klettke M. And Meyer H., (2000) XML And Object-Relational Database Systems – Enhancing Stluctural Mappings Based On Statistics”, *Webdb*, Springer-Verlag, 151-170
- Kossmann, D and Stocker, K (2000). Iterative Dynamic Programming: A new class of query optimization algorithms. ACM Transactions on Database Systems.
- Kossmann D., (2000).The State of Art in Distributed Query Processing. ACM Computing Surveys. Volume 32 (4) 422-469
- Krasner G. and S. Pope.(1988) A cookbook for using the Model-View-Controller, serinterface paradigm in Smalltalk-80. Journal of Object-Oriented Programming, 1(3) 26-49.
- Kremer M. and Gray J., (1999). A Survey of Query Optimizer in parallel Databases. echnical Report, CS-04-1999.
- Lancelot, R. S. G., Validities p and Zait M., and Zane M., (1994). Industrial-Strength Parallel Query Optimization: Issues and lesson. Information Systems, 19 (4) 311-330.
- Lema J. A. C., Forlizzi, L., G’uting, R. H., Nardelli, E., and Schneider, M.,(2003). Algorithms for Moving Objects Databases. Computer Journal, 46 (6) 680–712.

- Lomet, D. (1996). Replicated indexes for distributed data. In Proceedings of the International IEEE Conference on Parallel and Distributed Information Systems.
- Lorie, R and Wade, B. (1979). The Compilation of a High Level Data Language. Technical Report RJ 2598, IBM Research, San Jose, CA.
- Lu, H. and Carey, M. (1985). Some experimental results on distributed join algorithms In a local network. In Proceedings of the Conference on very large Data Base (VLDB), 229-304.
- Mackert, L and Lohman, G. (1986). R* optimizer validation and performance evaluation for distributed queries. In Proceedings of the Conference on Very Large Data Bases (VLDB), 149-159.
- Maier, D., Otis, A., and Purdy, A. (1985) "Some Aspects of Operations in an Object-Oriented Database," Database Engineering, 8 (4), IEEE Computer Society, December; "Object-Oriented Database Development at Servio Logic," Database Engineering, 18 (4)
- Mantelman, L.(1992), "Object-Oriented Databases Challenge Users to Change", Infoworld, 14 (11) 86.
- Martin, J. and Leben, J. (1995). Client-server databases: enterprise computing. Upper Saddle River, New Jersey: Prentice Hall.
- Martin, J. and Odell, J. (1992). Object-oriented analysis & design. Englewood Cliffs, New Jersey: Prentice Hall.
- Meijer E.(2000) Server Side Web Scripting in Haskell. Journal of Functional Programming, 10 (1) 1-18.
- Michael , V. M. (2006) Database Application Development and Design, McGraw-Hill New York
- Miller Freeman, Inc. (1999). Expert Opinion: short-term strategies for upgrading to Component-based, object-oriented technology. Insurance & Technology, [On-line].
- Morandini M., D. C. Nguyen, A. Perini, A. Siena, and A. Susi (2009) Tool-supported Development with Tropos: The Conference Management System Case Study. Fondazione Bruno Kessler – IRST Via Sommarive, 18 38050 Trento, Italy.

- Mullins, C. S.,(1994) "The Great Debate", Byte, 19 (4) 52.
- Niccu, T. M., Srivastava J., Himatstingka, B. and L.I J., (1993). A Tree-decomposition Approach to Parallel Query Optimization. Technical Report TR 93-016.
- O'Brien, J. A. & Marakas, G. M. (2009), Management information System. (9).
- ODBMS.ORG (2013). Object Database (OODBMS) / Free Resource Portal. ODBMS .
- Ogunlere S. O and Idowu S A. (2015). Comparison Analysis of Object-Based Databases, Object-Oriented Databases, and Object Relational Databases, Asian Journal of Computer and Information Systems 3(02) 52-57.
- Ono, K. and Lohman G. M. (1990). Measuring the complexity of Join Enumeration in Query optimization. In Proceedings of the conference on Very Large Database(VLDB).314-325.
- Oracle (2012). Retrived from <http://download.oracle.com/docs/cd/b10500-01/server.920/a96520/concept, htm#49840>.
- Oszu, M. T and Valduriez, P. (1999). Principles of Distributed Database Systems N. J, Prentice Hall International
- Oszu, M. T. and Valduriez, P. (1997). Distributed and parallel Database Systems. In Trucker A. (Ed). The Computer Science and Engineering Handbook. CRC press. 1093-1111.
- Oszu. M. T. and Valdurie z P., (1991). Distributed Database Systems: Where are We Now. IEEE Computer, 2 (8) 68-78.
- Oszu. M. T. and Valduriez P., (1999). Principles of Distributed Database Systems, Prentice Hall International, NJ.
- Pirahesh, H., Hellertein, J.,and Hasan, W, (1992). Extensible/rule based query rewrite optimization in starburst. In Proceeding of the ACM SIGMOD Conference on Management of Data, 39-48.
- Piotr Jaszczyk Tomasz Kolasa (2009), Developing Conference Management System with JBoss Seam, Department of Computer Science and ManagementJavaTech Research Group
- Plasmeijer R.and P. Achten (2005). Generic Editors for the World Wide Web. In Central-European Functional Programming School, EÄotvÄos Lorand University.
- Plasmeijer R. and P. Achten.(2006) iData For The World Wide Web –Programming Interconnected Web Forms. In Proceedings Eighth International Symposium on Functional and Logic Programming FLOPS 3945.

- Plasmeijer R. and P. Achten.(2006) The Implementation of iData - A Case Study in Generic Programming. In A. Butterfeld, editor, Proceedings Implementation and Application of Functional Languages - Revised Selected Papers, 17th International Workshop, IFL05, LNCS 4015.
- Pratt, P. J., and Adamski, J. J. (1991) Database Systems - Management and Design, 2nd Edition,
- Query Processor (2012), <http://technet.microsoft.com/en-us/library/cc966472.aspx#mainsection>.
- Radding, Alan (1995). So what the Hell is ODBMS? Computerworld. 29(45): 121-122, 129.
- Radding, A, (1993), "Not Quite Ready For Prime Time: The CW Guide To Object-Oriented Programming", Computerworld, 27(24).
- Rinus P. and Peter A.(2009) A Conference Management System based on the iData Toolkit, Software Technology, Nijmegen Institute for Computing and Information Sciences, Radboud University Nijmegen
- Robert, G. N., and David, L. D., (2007). IQ/OBJECTS- QUICK Query RPIS VERSION 3. Technical Report on IQ/Objects, Massachusetts, USA.
- Schek H. J., Scholl M. H. (1990) Evolution of Data Model models, Proc. Database Systems of the 90s, Springer LNCS 466, 135-153.
- Sellinger, P G., Astrahan, M. M., Chamberlin, D. D., Lorie ,R.A., and Price T. G. (1979). Access Path Selection in A Relational Database Management System. In Proceeding of the ACM SIGMOD Conference on Management of Data. Boston. USA. 23-34.
- Senn, J. A. (1990), Information Systems in Management, 4th Edition, Wadsworth,.
- Schach, S. (1996). Classical and object-oriented software engineering (3rd ed.) 140, 170.
- Sidell, J., Aoki, P., Barr, S., Sah, A., Staelin, C., Stonebraker, M., and Yu, A. (1996). Data replication in Mariposa. In Proceedings IEEE Conference on Data Engineering, 485-494.
- Siena A (2007). Engineering Normative Requirements. In Proceedings of the First International Conference on Research Challenges in Information Science, RCIS 439–444.
- Stanley D. B. (2012) A Primer on Object Role Modeling, University of California Press, Berkeley.

- Steinbrunn, M., Moerkotte, G., and Kemper, A. (1997). Heuristic and Randomize Optimization for the Join-Ordering Problem. *VLDB* 6 (3) 191-20.
- Stocker, K., Kossmann, D., Braumandi, R., and Kemper, A. (2001). Integrating semi join reducers into state-of-the-art query processors. In *Proceedings Of The IEEE Conference On Data*.
- Stonebraker, M (1985). *The design and implementation of distributed INGRES*. Reading, MA.
- Stonebraker, M. (1994). *Readings in Database Systems* (second ed.). Morgan Kaufmann Publishers, San Mateo, CA
- Stonebraker, M., Aoki, P., Litwin, W., Pfeffer, A., Sah, A., Sideli, J., Staelin, C., and Yu, A. (1996). Mariposa: a wide-area distributed database system. *The VLDB* 5 (1) 48-63.
- Tanenbaum, A. S. Bal, H. E., Steiner, J. G.; (1989). Programming languages for distributed computing systems. *ACM Computing Surveys*. 21 (3), 261.
- Tang W., Ho, S.-S. Ho, Liu, W. T., and Schneider, M., (2010). A Framework for Moving Sensor Data Query and Retrieval of Dynamic Atmospheric Events. In *22nd Int. Conf. on Scientific and Statistical Database Management (SSDBM)*, ser. *Lecture Notes in Computer Science*, 6187, 96–113.
- Terry Halpin (2012) *Object-Role Modeling (ORM/NIAM)*, Microsoft Corporation, USA reproduced by permission from *Handbook on Architectures of Information Systems*, eds P. Bernus, K. Mertins & G. Schmidt, Springer-Verlag, Berlin, 1998, www.springer.de/cgi-bin/search_book.pl?isbn=3-540-64453-9.
- Thiemann P. (2002) *WASH/CGI: Server-side Web Scripting with Sessions and Typed, Compositional Forms*. In S. Krishnamurthi and C. Ramakrishnan, editors, *Practical Aspects of Declarative Languages: 4th International Symposium, PADL 2257 of LNCS*, 192-208.
- Urhan, T. and Franklin, M., (1999). Xjoin: Getting Fast Answers from Slow and Busty Networks. Technical report CS-TR (Feb), University of Maryland, College Park.
- Vaduriez, P. and Gardarin, G. (1984). Join and semi-join algorithms for a multiprocessor database machine. *ACM Transactions on Database Systems* 9 (1) 133-161.
- Vogels W. 2003). *Scalable Cluster Technologies for Mission Critical Enterprise Computing* (PhD thesis). Vrije Universiteit. [1871-10357](https://doi.org/10.1871/10357).
- Waas, F, and Galindo-legaria, C. (2000). Counting, Enumerating, and sampling of execution plans in a cost-based query optimizer, *Proceedings of ACM SIGMOD International Conference on Management of Data*.

- Williams R, Daniels, D , Haas, L, Lapis, G, Lindsay, B., Ng ,P, Obermarck, R, Selinger, P, Walker,A., Wilms, P., and Yost, R. (1981).R.*: An Overview of the architecture. IBM Research, San Jose, CA, RJ3325. Reprinted in: M. Stonebreaker (ed.), (1994). Readings in Database systems, Morgan Kaufmann publishers, 515-536.
- Wilshut, A. and Apers, P. (1991). Dataflow query execution in a parallel Main memory. In Proceedings of the International IEEE Conference on Parallel and Distributed Information Systems, 68-77.
- Zaharioudakis, M. and Carey, M. (1997). Highly concurrent cache consistency for indices in client-server database systems. In Proceedings of the ACM SIGMOD Conference on Management of Data, 50-61.
- Ziane M., Ziat M., and Borle- Salanet P. (1993). Parallel Query Processing with Zigzag Trees. VLDB 2 (3) 277-301.
- Zhang F., Ma Z. M., Wang X., Wang Y. (2010) Formal Approach and Automated Tool for ConstructingOntology from Object-oriented Database Model, In proceeding of ACM CIKM'10 1329-1332 .

APPENDIX A: SOURCE CODE

```
<?php

////////////////////////////////////
/////
//
// THIS PROGRAM HANDLES THE CONFERENCE MANAGEMENT SYSTEM WHERE CALL FOR
PAPER
// IS MADE, PAPER SUBMISSION IS MADE AND REVIEWERS CAN LOGIN AND REVIEW
PAPERS
// AND ACCEPTANCE OF PAPER CAN BE COMMUNICATED TO AUTHORS.
// user:ngozi
// pass:ngozi123
//
// BY:NGOZI CAROLINE OKEREKE
// REG. NO: 2008517005P

//
////////////////////////////////////
/////
// THIS PROGRAM IS DEVELOPED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR
// THE AWARD OF A PhD IN COMPUTER SCIENCE OF NNAMDI AZIKIWE UNIVERSITY AWKA
////////////////////////////////////
/////

$php_root_path = ".";

require_once("$php_root_path/includes/include_all_fns.inc");

global $homepage ;
$error_array = array() ;
if (!file_exists("$php_root_path/includes/preferences.inc"))
{
    echo "<font color=\"#FF0000\"><strong>Warning</strong>:
preferences.inc file is not installed in \\includes directory. Please use
\\install\\install.php to create this file.</font><br><br>";
}
if ($_POST["submit"] == "Submit") //disable validation if resetting
    $exempt_array = array() ;
else
    $exempt_array = array("username", "logpassword");

$message = " Unable to process your request due to the following
problems: <br>\n" ;

check_form( $_POST , $error_array , &$exempt_array ) ;

if ( count ( $error_array ) == 0 && count ( $_POST ) > 0 &&
$_POST["submit"] == "Submit")
{
    // connect to db
```

```

$db = adodb_connect( &$err_message );
if (!$db)
{
    $homepage->showmenu = 0 ;
    do_html_header("Login Failed");
    $err_message .= " Unable to connect to database. <br>\n" ;
}
else if ( $PrivilegeTypeID = login( $_POST["username"],
$_POST["logpassword"] , &$err_message ) )
{
    session_start();
    // if they are in the database register the user id
    $_SESSION["valid_user"] = $_POST["username"] ;
//    $_SESSION["phase"] ;    // 4.0.6
    $_SESSION["phase"] ;
//    session_register("valid_user");
//    echo "<br>\nSession: " . $_SESSION["valid_user"] . "<br>\n" ;

    if ( !check_conference_phase( &$err_message, $PrivilegeTypeID )
)
    {
        $homepage->showmenu = 0 ;
        do_html_header("Login Failed");
//        session_unregister("valid_user");
        unset ( $_SESSION["valid_user"] ) ;
        $err_message .= " Unable to connect to conference
database. <br>\n" ;
    }
    else
    {
//        session_register("phase") ;
        switch ( $PrivilegeTypeID )
        {
            case 1:
            {
                $str = "Location:
$php_root_path/user/view_papers.php";
                header( $str ); // Redirect browser
                exit; // Make sure that code below does not
get executed when we redirect.
                break ;
            }
            case 2:
            {
//                if ( !session_is_registered ( "s_reviewer" )
)
//                {
//                    session_register ( "s_reviewer" ) ;
//                }
                $_SESSION["s_reviewer"] = array() ;
                $str = "Location:
$php_root_path/reviewer/reviewer_home.php";
                header( $str ); // Redirect browser
                exit; // Make sure that code below does not
get executed when we redirect.
                break ;
            }
        }
    }
}

```

```

        case 3:
        {
            $str = "Location:
$php_root_path/admin/admin_home.php";
            header( $str ); // Redirect browser
            exit; // Make sure that code below does not
get executed when we redirect.
            break ;
        }
        default :
        {
            $homepage->showmenu = 0 ;

            do_html_header("Login Failed" , &$err_message
);
            $err_message .= " Unknown User's
PrivilegeTypeID. <br>\n" ;
            break ;
        }
    }
}
////////// Debug ////////////
//      echo gettype( $_SESSION["phase1"] ) . "<br>\n" ;      // 4.1.1
//      echo gettype( $_SESSION["phase2"] ) . "<br>\n" ;      // 4.1.1
//      echo gettype( $_SESSION["phase"]3 ) . "<br>\n" ;      // 4.0.6
//      echo "Login phaseID 1: " . $_SESSION["phase1"]->phaseID .
"<BR>\n" ; // 4.1.1
//      echo "Login phaseID 2: " . $_SESSION["phase2"]->phaseID .
"<BR>\n" ; // 4.1.1
//      echo "Login phaseID: " . $_SESSION["phase"]->phaseID .
"<br>\n"; // 4.0.6
//////////
    }
    else
    {
        // unsuccessful login
        $homepage->showmenu = 0 ;
        do_html_header("Login Failed" , &$err_message );
        $err_message .= " Please re-enter your username and password.
<br>\n" ;
        //      echo $err_message . "<br><br> Try <a
href='index.php'>again</a>?" ;
        //      do_html_footer();
    }
}
else
{
    $homepage->showmenu = 0 ;
    //Call the function to get the conference information
    // $confName = $conferenceInfo -> ConferenceCodeName;
    $conferenceInfo = get_conference_info();
    do_html_header("Online Submission and Reviewing" , &$err_message );
}

?>

<form action="index.php" method="post" name="loginForm" id="loginForm">

```

```

<table width="80%" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td height="24" colspan="2"><a href="<?php echo $php_root_path ;
?>/user/registration.php">First submission or want to register your
interest? Sign
      up for an account.</a></td>
    </tr>
    <tr>
      <td height="20" colspan="2">Hint: Use your email address as
UserName.</td>
    </tr>
    <!--      <tr>
      <td height="20" colspan="2"><span style="color:red">Login temporarily
unavailable. sorry.</span></td>
    </tr-->
    <tr>
      <td width="20%">&nbsp;</td>
      <td width="80%" height="24">&nbsp;</td>
    </tr>
    <tr>
      <td>UserName</td>
      <td><input name="username" type="text" id="username" value="<?php if
($ _POST["submit"] == "Submit") echo $ _POST["username"] ; ?>" size="25"
maxlength="50">
        <?php echo "<font color=\"#FF0000\">" . $error_array["username"][0]
. "</font>" ?></td>
      </tr>
      <tr>
        <td>Password</td>
        <td><input name="logpassword" type="password" id="logpassword2"
value="<?php if ($ _POST["submit"] == "Submit") echo $ _POST["logpassword"] ;
?>" size="25" maxlength="50">
          <?php echo "<font color=\"#FF0000\">" .
$error_array["logpassword"][0] . "</font>" ?></td>
        </tr>
        <tr>
          <td>&nbsp;</td>
          <td>&nbsp;</td>
        </tr>
        <tr>
          <td>&nbsp;</td>
          <td><input type="submit" name="submit" value="Submit"> <input
type="submit" name="reset" value="Reset"></td>
        </tr>
        <tr>
          <td>&nbsp;</td>
          <td>&nbsp;</td>
        </tr>
        <tr>
          <td colspan="2">If you have forgotten your password,<a href="<?php
echo $php_root_path ; ?>/user/forget_pwd.php">
            click here.</a></td>
          </tr>
        </table>
      </form>
    <?php

```

```

do_html_footer( &$err_message );

?>
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
<?php

    $php_root_path = ".." ;
    $privilege_root_path = "/admin" ;
    require_once("includes/include_all_fns.inc");
    session_start();
    $err_message = " Unable to process your request due to the following
problems: <br>\n" ;
    // extract ( $_SESSION , EXTR_REFS ) ;
    $valid_user = $_SESSION["valid_user"] ;

    do_html_header("Admin Home" , &$err_message );

    //Call the function to get the conference information
    $conferenceInfo = get_conference_info();

    //Get Current Phase
    $currentPhaseInfo = getCurrentPhase();

    //Retrieve the setting information
    $settingInfo = get_Conference_Settings();

    //Connect to database
    $db = adodb_connect();

    if($currentPhaseInfo -> PhaseName == "Reviewing" || $currentPhaseInfo -
> PhaseName == "Final Paper Submission")
        $arrPaperReviewing = get_Paper_Reviewing_Statistic();

    //Get the total number of submitted papers
    $countPapersSQL = "SELECT COUNT(*) AS totalPapers FROM " .
$GLOBALS["DB_PREFIX"] . "Paper WHERE Withdraw = 'false'";
    $countPapersResult = $db -> Execute($countPapersSQL);
    $countPapersInfo = $countPapersResult -> FetchNextObj();

    //Get the total number of members
    $countUserSQL = "SELECT COUNT(*) AS totalUsers FROM " .
$GLOBALS["DB_PREFIX"] . "Member M," . $GLOBALS["DB_PREFIX"] .
"PrivilegeType P";
    $countUserSQL .= " WHERE M.PrivilegeTypeID = P.PrivilegeTypeID";
    $countUserSQL .= " AND PrivilegeTypeName = 'User'";
    $countUserResult = $db -> Execute($countUserSQL);
    $countUserInfo = $countUserResult -> FetchNextObj();

    //Get the total number of reviewers
    $countReviewerSQL = "SELECT COUNT(*) AS totalReviewers FROM " .
$GLOBALS["DB_PREFIX"] . "Member M," . $GLOBALS["DB_PREFIX"] .
"PrivilegeType P";
    $countReviewerSQL .= " WHERE M.PrivilegeTypeID = P.PrivilegeTypeID";
    $countReviewerSQL .= " AND PrivilegeTypeName = 'Reviewer'";
    $countReviewerResult = $db -> Execute($countReviewerSQL);
    $countReviewerInfo = $countReviewerResult -> FetchNextObj();

```

```

        //Get the total number of withdrawn papers
        $countWithdrawnSQL = "SELECT COUNT(*) AS totalWithdrawnPapers FROM "
    . $GLOBALS["DB_PREFIX"] . "Paper WHERE Withdraw = 'true'";
        $countWithdrawnResult = $db -> Execute($countWithdrawnSQL);
        $countWithdrawnInfo = $countWithdrawnResult -> FetchNextObj();

        //Get the total number of accepted papers
        $countAcceptedSQL = "SELECT COUNT(*) AS totalAcceptedPapers FROM "
    . $GLOBALS["DB_PREFIX"] . "Paper WHERE PaperStatusID= '3' AND Withdraw =
    'false'";
        $countAcceptedResult = $db -> Execute($countAcceptedSQL);
        $countAcceptedInfo = $countAcceptedResult -> FetchNextObj();

        //Get the total number of rejected papers
        $countRejectedSQL = "SELECT COUNT(*) AS totalRejectedPapers FROM "
    . $GLOBALS["DB_PREFIX"] . "Paper WHERE PaperStatusID= '4' AND Withdraw =
    'false'";
        $countRejectedResult = $db -> Execute($countRejectedSQL);
        $countRejectedInfo = $countRejectedResult -> FetchNextObj();

        //Get the total number of marginal
        $countMarginalSQL = "SELECT COUNT(*) AS totalMarginalPapers FROM "
    . $GLOBALS["DB_PREFIX"] . "Paper WHERE PaperStatusID= '6' AND Withdraw =
    'false'";
        $countMarginalResult = $db -> Execute($countMarginalSQL);
        $countMarginalInfo = $countMarginalResult -> FetchNextObj();

        //Get the total number of papers in review
        $countReviewingSQL = "SELECT COUNT(*) AS totalReviewingPapers FROM "
    . $GLOBALS["DB_PREFIX"] . "Paper WHERE PaperStatusID in (1,2,5) AND
    Withdraw = 'false'";
        $countReviewingResult = $db -> Execute($countReviewingSQL);
        $countReviewingInfo = $countReviewingResult -> FetchNextObj();

        //Get the total number of papers unscheduled
        $countUnscheduledSQL = "SELECT COUNT(*) AS totalUnscheduledPapers
    FROM " . $GLOBALS["DB_PREFIX"] . "UnscheduledPaper AS U, ";
        $countUnscheduledSQL .= $GLOBALS["DB_PREFIX"] . "Paper AS P WHERE
    P.PaperID = U.PaperID AND P.Withdraw = 'false'";
        $countUnscheduledResult = $db -> Execute($countUnscheduledSQL);
        $countUnscheduledInfo = $countUnscheduledResult -> FetchNextObj();

    ?>
    <table width="100%" border="0" cellspacing="0" cellpadding="1">
        <tr>
            <td width="10%">&nbsp; </td>
            <td width="90%" align="right"><strong><?php echo
    format_date($settingInfo->DateFormatLong); ?></strong></td>
        </tr>
        <tr>
            <td>&nbsp; </td>
            <td>
                <?php
                    if ($conferenceInfo -> FileName != "")
                        echo "<img src=\"view_logofile.php\"
    alt=\"Logo\">";
                ?>
                <h4><?php echo $conferenceInfo -> ConferenceName; ?></h4></td>
            </td>
        </tr>
    </table>

```

```

</tr>
<tr>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
</tr>
<tr>
  <td>&nbsp;</td>
  <td>Welcome <strong><?php echo $valid_user; ?></strong>!</td>
</tr>
<tr>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
</tr>
<tr>
  <td>&nbsp;</td>
  <td><table width="60%" border="1" cellpadding="3" cellspacing="0"
bordercolor="#999999">
    <tr>
      <td colspan="2">
        <?php

          if ($currentPhaseInfo != false){

            $result = is_date_expired( $currentPhaseInfo ->
EndDate , date ( "Y-m-d" , time() ) , &$err_message , "date" );

            if($result === true) {
              ?>
              <font color='#FF0000'>The current running phase is
expired.</font><br>
              <br><font color='#FF0000'>
              <?php
                } else {
              ?>

              <?php
            }

            ?>
            <strong>
            <?php echo $currentPhaseInfo -> PhaseName; ?>
            </strong><br>
            From <strong>
            <?php echo format_date($settingInfo->DateFormatShort,
$currentPhaseInfo -> StartDate); ?>
            </strong> To <strong>
            <?php echo format_date($settingInfo->DateFormatShort,
$currentPhaseInfo -> EndDate); ?>
            </strong>
            </font>
            <?php
              } else echo "Current phase is not activated yet";
            ?>
          </td>
        </tr>
      <tr>
        <td width="70%">Total Papers Submitted:</td>

```

```

        <td width="30%"><?php echo $countPapersInfo -> totalPapers;
?></td>
    </tr>
    <tr>
        <td>Total Number of Users:</td>
        <td><?php echo $countUserInfo -> totalUsers; ?></td>
    </tr>
    <tr>
        <td>Total Number of Reviewers:</td>
        <td><?php echo $countReviewerInfo -> totalReviewers; ?></td>
    </tr>
    <tr>
        <td>Number of Papers Withdrawn:</td>
        <td><?php echo $countWithdrawnInfo -> totalWithdrawnPapers;
?></td>
    </tr>
</table> </td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td>
        <?php
            if(($currentPhaseInfo -> PhaseName == "Reviewing") ||
$currentPhaseInfo -> PhaseName == "Final Paper Submission") &&
(count($arrPaperReviewing) > 0)){?>
                <strong>Scheduling Status</strong><br>
                <br>
                <table width="60%" border="1" cellpadding="3" cellspacing="0"
bordercolor="#999999">
                    <tr>
                        <td width="70%"> Scheduled </td><td> <?php echo
$countAcceptedInfo->totalAcceptedPapers - $countUnscheduledInfo-
>totalUnscheduledPapers ; ?> </td>
                    </tr>
                    <tr>
                        <td width="70%"> Unscheduled </td><td> <?php echo
$countUnscheduledInfo->totalUnscheduledPapers ; ?> </td>
                    </tr>
                </td></tr>
                </table>

                <tr>
                    <td>&nbsp;</td>
                    <td>&nbsp;</td>
                </tr>
                <tr>
                    <td>&nbsp;</td>
                    <td>
                        <strong>Decision Status</strong><br>
                        <br>
                        <table width="60%" border="1" cellpadding="3" cellspacing="0"
bordercolor="#999999">
                            <tr>

```



```

        <td width ="70%"> Accepted </td><td> <?php echo $countAcceptedInfo-
>totalAcceptedPapers ; ?> </td>
    </tr>
    <tr>
        <td width ="70%"> Rejected </td><td> <?php echo $countRejectedInfo-
>totalRejectedPapers ; ?> </td>
    </tr>
    <tr>
        <td width ="70%"> Marginal </td><td> <?php echo $countMarginalInfo-
>totalMarginalPapers ; ?> </td>
    </tr>
    <tr>
        <td width ="70%"> Pending </td><td> <?php echo $countReviewingInfo-
>totalReviewingPapers ; ?> </td>
    </tr>
</td></tr>
</table>

<tr>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td>
        <strong>Review Status</strong><br>
        <br>
        <table width="60%" border="1" cellpadding="3" cellspacing="0"
bordercolor="#999999">
            <?php while(list($numReviews,$count) = each($arrPaperReviewing)){?>
            <tr>
                <td width="70%">
                    <?php
                        if($numReviews == 0) echo "No reviews:";
                        else if($numReviews > 1) echo "$numReviews reviews:";
                        else echo "$numReviews review:"; ?>
                </td>
                <td width="60%"><?php echo $count; ?></td>
            </tr>
            <?php } ?>
        </table>
        <?php } ?>
    </td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
</tr>
</table>
<?php

do_html_footer( &$err_message );

```

```

?>
\/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

<?php

    $php_root_path = ".." ;
    $privilege_root_path = "/admin" ;
    require_once("includes/include_all_fns.inc");
    require_once("$php_root_path/includes/page_includes/page_fns.php");
    // only for numCategories()
    session_start();
    //extract ( $_SESSION , EXTR_REFS ) ;

    // Define a few page vars
    $settingInfo = get_Conference_Settings();
    $trackStr = $settingInfo->TrackName; //Name for Track
    $topicStr = $settingInfo->TopicName; //Name for Topic
    $levelStr = $settingInfo->LevelName; //Name for Level

    $err_message = " Unable to process your request due to the following
problems: <br>\n" ;

    $paperID = & $_GET["paperID"];
    $status = & $_GET["status"];
    $title = $status." Paper";
    do_html_header($title);

    //Get the paper information
    $paperInfo = get_paper_info($paperID);

    $type = get_presentation_info($_GET["type"]);
    $curtype = get_presentation_info(
get_presentation_type_for_paper($paperInfo -> PaperID) );
?>
<form action="process_accept_reject_paper.php" method="post" name="form1">
    <table width="100%" border="0" cellspacing="0" cellpadding="1">
        <tr>
            <td colspan="2"><?php echo stripslashes("<h3>#".$paperInfo->PaperID."
".$paperInfo -> Title."</h3>"); ?></td>
        </tr>
        <tr>
            <td colspan="2"><input type="hidden" name="paperID" value="<?php echo $paperInfo-
>PaperID; ?>">
        </tr>
        <tr>
            <td width="15%"><strong>Authors:</strong> </td>
            <td width="85%"><?php echo retrieve_authors($paperInfo ->
PaperID);?></td>
        </tr>
        <tr>
            <td><strong><?php echo "$trackStr:"?></strong> </td>
            <td><?php echo GetSelectedTrackText($paperInfo -> PaperID ,
&$err_message );?></td>
        </tr>

<?php

```

```

        if (numCategories( &$err_message ) > 0) // allow conferences with
only Tracks, but no Topics
        {
            echo '<tr><td><strong>';
            echo "$topicStr(s):";
            echo '</strong></td><td>';
            echo getSelectedCategoryCommaSeparated($paperInfo -> PaperID ,
&$err_message );
            echo '</td></tr>';
        }
?>

<tr>
    <td colspan="2">&nbsp;</td>
</tr>
<input type="hidden" name="status" value="<?php echo $status; ?>">
<input type="hidden" name="type" value="<?php echo $type ->
PresentationTypeID; ?>">
<tr>
    <td><strong>Current Status:</strong></td>
    <td><?php echo $paperInfo -> PaperStatusName; ?>
        <?php if ($paperInfo -> PaperStatusName == "Accepted") { ?>
            as <?php echo $curtype -> PresentationTypeName; ?>
            <?php } ?>
        </td>
</tr>
<?php
    if (array_key_exists( "SessionTrackID", $_GET )) {
        echo '<tr><td><strong>Current
SessionTrack:</strong></td>';
        echo '<td>' . getSelectedSessionTrackText($paperInfo-
>PaperID) . '</td></tr>' . "\n";
    }
?>
<tr>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
</tr>
<tr>
    <td><strong>Change to:</strong></td>
    <td>
        <?php echo $status; ?>
        <?php if ($status == "Accepted") { ?>
            as <?php echo $type -> PresentationTypeName; ?></td>
        <?php } ?>
    </td>
</tr>
<?php
    if (array_key_exists( "SessionTrackID", $_GET )) {
        echo '<tr><td><strong>SessionTrack:</strong></td>';
        $info = get_SessionTrack_info($_GET["SessionTrackID"]);
        echo '<td>' . $info->SessionTrackName . '</td></tr>' .
"\n";
        echo '<input type="hidden" name="SessionTrackID" value="' .
$_GET["SessionTrackID"] . '">';
    }
?>
<tr>

```

```

        <td colspan="2">&nbsp;  </td>
    </tr>
    <?php
        if($status == "Accepted")
            $result = check_Letter_Already_Sent($paperInfo ->
PaperID,"5");
        else if($status == "Rejected")
            $result = check_Letter_Already_Sent($paperInfo ->
PaperID,"6");

        if($result === true){
            echo "<tr>\n<td colspan=\"2\">\n";
            echo "<font color=FF0000> The notification letter has
already been sent to inform the paper owner.</font><br><br>If you wish to
inform the paper owner again, check the box below.";
            echo "</td>\n\n</tr>";
        }

    ?>
    <tr>
        <td colspan="2"><input name="informuser" type="checkbox"
id="informuser" value="yes" <?php if ($_GET["back"] == "true") echo
"checked"; ?>>
            Inform the user now <br><br>
            (Note: Tick this box if want to send one letter only. The
accept/reject form letter is used to send all pending letters as a batch on
the day of notification of acceptance.)</td>
        </tr>
        <tr>
            <td colspan="2">&nbsp;  </td>
        </tr>
        <tr>
            <td colspan="2">&nbsp;  </td>
        </tr>
        <tr>
            <td colspan="2"><input name="Submit" type="submit" id="Submit"
value="Submit">
                <input name="Submit" type="submit" value="Back">
                <input name="Submit" type="submit" id="Submit" value="Cancel">
            </td>
        </tr>
    </table>
</form>
<?php

    do_html_footer();

?>
////////////////////////////////////
// UPLOAD PAPER

//<?php
    $php_root_path = ".." ;
    require_once("$php_root_path/includes/include_all_fns.inc");
    require_once("$php_root_path/includes/page_includes/page_fns.php");

```

```

        session_start();
        $err_message = " Unable to process your request due to the following
problems: <br>\n" ;
        $header = "Upload Paper" ;
        $accepted_privilegeID_arr = array ( 1 => "" ) ;
        $accepted_phaseID_arr = array ( 1 => "" ) ;
        authentication( $header , $accepted_privilegeID_arr ,
        $accepted_phaseID_arr , $homepage , $php_root_path , $GLOBALS["DB_PREFIX"]
        , &$err_message ) ;

        $error_array = array() ;

        // Define a few page vars
        $settingInfo = get_Conference_Settings();
        $trackStr = $settingInfo->TrackName; //Name for Track
        $topicStr = $settingInfo->TopicName; //Name for Topic
        $levelStr = $settingInfo->LevelName; //Name for Level

        if ($settingInfo->AbstractOnlySubmissions || $settingInfo->SESUG)
        //Abstract submission only for SESUG
        {
            $exempt_array = array ( "email" , "middlename" , "presenterbio"
, "keyword1" , "keyword2" , "keyword3" , "userfile", "numpages" ) ;
            $fullPaper = false;
        }
        else{
            $exempt_array = array ( "email" , "middlename", "presenterbio",
"keyword1" , "keyword2" , "keyword3" ) ;
            $fullPaper = true;
        }

        if ( count ( $_POST ) > 0 )
        {
            if ( $_POST["Submit"] == "Update Authors" )
            {
                if ( isIntegerMoreThanZero ( $_POST["numauthors"] ,
&$error_array["numauthors"] ) || !empty ( $_POST["numauthors"] ) )
                {
                }
                else if ( trim ( $_POST["numauthors"] ) == "" )
                {
                    $error_array["numauthors"][0] = " This entry cannot
be empty. <br>\n" ;
                }
            }
            else
            {
                if ( $settingInfo->SESUG && !$_POST["level"] )
                {
                    $error_array["level"][0] = "You must choose at least one
$levelStr.<br>\n" ;
                }
                if ( !$_POST["track"] )
                {

```

```

        $error_array["track"][0] = "You must choose a
$trackStr.<br>\n" ;
    }
    if ( !$_POST["category"] )
    {
        if (numCategories( &$err_message ) > 0) // allow
conferences with only Tracks, but no Topics
            $error_array["category"][0] = "You must
choose at least one $topicStr.<br>\n" ;
    }

    $vars = array_merge ( $_POST , $_FILES ) ;
    //display( $vars ) ;

    check_form( $vars , $error_array , &$exempt_array ) ;
}

if ( count ( $error_array ) == 0 && count ( $_POST ) > 0 )
{
    if ( $_POST["Submit"] === "Submit" )
    {
        //Everything is fine, then upload the file
        if ( $fileID = upload_file( $_POST["title"] ,
$_POST["abstract"] , $_POST["presenterbio"] , $_POST["numpages"] ,
$_FILES["userfile"]["tmp_name"] , $_FILES["userfile"]["name"] ,
$_FILES["userfile"]["size"] , $_FILES["userfile"]["type"] ,
$_POST["firstname"] ,
$_POST["middlename"] , $_POST["lastname"] , $_POST["email"] ,
$_POST["attended"] , $_POST["presented"] , $_POST["keyword1"]
, $_POST["keyword2"] , $_POST["keyword3"] , $_POST["level"] , $_POST["track"]
, $_POST["category"] , &$err_message ) )
        {
            do_html_header("Successful Uploading..." ,
&$err_message );
            echo " The file is uploaded successfully to the
database. <br><br> View your new paper at <a
href='view_paper_details.php?fileid=" . $fileID . "'>View Paper
Details</a> page. <br>" ;
            do_html_footer( &$err_message );
            exit ;
        }
        else
        {
            do_html_header("Problem Uploading..." ,
&$err_message );
            $err_message .= "<br><br> Go to the <a
href='upload_paper.php'>Upload Paper</a> page. " ;
        }
    }
    else
    {
        do_html_header("Upload Paper" , &$err_message ) ;
    }
}
else
{

```

```

        if ( count ( $_POST ) == 0 )
        {
//            echo "<br>\n POST = 0 <br>\n" ;

        }
        do_html_header("Upload Paper" , &$err_message ) ;
    }

    $maxfilesize = $settingInfo->MaxUploadSize ;
?>

<form enctype="multipart/form-data" name="frmupload" method="post"
action="upload_paper.php">
<!-- <form enctype="multipart/form-data" name="frmupload" method="post"
action="phpinfo.php"> -->
(* indicates mandatory field)<br><br>

    <table width="100%" border="0" cellpadding="3" cellspacing="0">
        <tr>
            <td width="20%"><strong>Title *:</strong></td>
            <td width="80%"> </td>
        </tr>
        <tr>
            <td colspan="2"><input name="title" type="text" value="<?php echo
stripslashes($_POST["title"]) ?>" id="title" size="75" maxlength="255">
                <font color="#FF0000"><?php echo $error_array["title"][0] ?></font>
</td>
            </tr>
            <tr>
                <td><strong>Number of Pages <?php if ($fullPaper) echo " *";
?>:</strong></td>
                <td> <input name="numpages" type="text" value="<?php echo
$_POST["numpages"] ?>" id="numpages" size="3" maxlength="3">
                    <font color="#FF0000"><?php echo $error_array["numpages"][0]
?></font>
</td>
            </tr>
            <tr>
                <td>&nbsp;</td>
                <td> <font color="#FF0000">&nbsp;</font> </td>
            </tr>
            <tr>
                <td><strong>Number of Authors *:</strong></td>
                <td>
                    <?php // show at least one author field
                    if (isset( $_POST["numauthors"] )){
                        $numauthors = $_POST["numauthors"];
                    }
                    else{
                        $numauthors = 1 ;
                    }
                    ?>
                <td><input name="numauthors" type="text" value="<?php echo
$numauthors ?>" id="numauthors" size="3" maxlength="2">
                    <input type="submit" name="Submit" value="Update Authors"> <font
color="#FF0000"><?php echo $error_array["numauthors"][0] ?></font></td>
            </tr>
            <tr>

```



```

        echo "<strong>$trackStr *:</strong>\n " ;
        echo "<font color=\"#FF0000\">" . $error_array["track"][0] .
"</font>" ;

        if ( $result = GenerateSelectedCategoryInputTable(
$_POST["track"] , &$err_message , 0 , "Track" ) )
        {
            echo $result ;
        }
        else
        {
            $err_message .= "<br><br> Try <a
href='upload_paper_info.php?paperid=" . $_POST["paperid"] . "'>again</a>?"
;
        }

        if (numCategories( &$err_message ) > 0) // allow conferences with
only Tracks, but no Topics
        {
            echo "<br>" ;
            echo "<STRONG>$topicStr(s) *:</STRONG>\n" ;
            echo "<font color=\"#FF0000\">" . $error_array["category"][0] .
"</font>" ;
            if ( $result = GenerateSelectedCategoryInputTable (
$_POST["category"] , &$err_message ) )
            {
                echo $result ;
            }
            else
            {
                $err_message .= "<br><br> Try <a
href='upload_paper.php'>again</a>?" ;
            }
        }

?>
    </td>
</tr>
<?php if ($settingInfo->SESUG) { ?>
    <tr>
        <td><strong><?php echo $keyword ?> :</strong></td>
        <td> <input name="keyword1" type="text" value="<?php echo
$_POST["keyword1"] ?>" id="keyword1" size="20" maxlength="50">
            <font color="#FF0000"><?php echo $error_array["keyword1"][0]
?></font>
        </td>
    </tr>

    <tr>
        <td><strong><?php echo $keyword ?> :</strong></td>
        <td> <input name="keyword2" type="text" value="<?php echo
$_POST["keyword2"] ?>" id="keyword2" size="20" maxlength="50">
            <font color="#FF0000"><?php echo $error_array["keyword2"][0]
?></font>
        </td>
    </tr>

```

```

        <tr>
            <td><strong><?php echo $keyword ?> :</strong></td>
            <td> <input name="keyword3" type="text" value="<?php echo
$_POST["keyword3"] ?>" id="keyword3" size="20" maxlength="50">
                <font color="#FF0000"><?php echo $error_array["keyword3"][0]
?></font>
            </td>
        </tr>
        <?php } ?>
        <tr>
            <td>&nbsp;</td>
            <td>&nbsp;</td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><input type="submit" name="Submit" value="Submit">
        </td>
    </tr>
</table>
</form>

<?php

do_html_footer( &$err_message );

?>
////////////////////////////////////
// CONFERENCE PAYMENT FORM
<?php

$php_root_path = ".." ;
require_once("$php_root_path/includes/include_all_fns.inc");
session_start();

$err_message = " Unable to process your request due to the following
problems: <br>\n" ;
$header = "Registration Forms" ;
$accepted_privilegeID_arr = array ( 1 => "" ) ;
$accepted_phaseID_arr = array ( 1 => "" , 2 => "" , 3 => "" , 4 => "" ) ;
authentication( $header , $accepted_privilegeID_arr , $accepted_phaseID_arr
, $homepage , $php_root_path , $GLOBALS["DB_PREFIX"] , &$err_message ) ;

$memberInfo = getMemberInfo($_SESSION["valid_user"]);
if ($memberInfo == null) exit;

if (has_paid_registration($memberInfo->RegisterID))
{
    $paid = get_paid_registration($memberInfo->RegisterID);
    do_html_header("Paid Registration - #".$paid->FormID , &$err_message
);
?>
<div style="padding: 20">
    <?php echo get_registration_statement($paid->Form) ?>
</div>
<?php

```

```

} else {
    do_html_header($header , &$err_message );
    $forms = retrieve_selection_xml_for_registerid($memberInfo->RegisterID);
    if (count($forms) > 0)
    {
?>
        <p>
        Previously filled forms:
        <ul>
<?php
        foreach ($forms as $form)
        {
            ?>
            <li>
            <a href="printable_statement.php?formid=<?php echo $form->FormID?>">
            <?php echo $form->FormID?>
            </a>
            </li>
            <?php
        }
    ?>
        </ul>
        </p>
<?php
    }
?>
    <a href="payment_form.php">Fill out new form</a>
<?php
}
do_html_footer( &$err_message );
?>

```

////////////////////////////////////

// EDIT PAPER

```

<?php
$php_root_path = ".." ;
require_once("$php_root_path/includes/include_all_fns.inc");
require_once("$php_root_path/includes/page_includes/page_fns.php");
session_start();
$error_message = " Unable to process your request due to the following
problems: <br>\n" ;
$header = "Edit Paper Details" ;
$accepted_privilegeID_arr = array ( 1 => "" ) ;
$accepted_phaseID_arr = array ( 1 => "" , 2 => "" , 3 => "" , 4 => "" ) ;
authentication( $header , $accepted_privilegeID_arr , $accepted_phaseID_arr
, $homepage , $php_root_path , $GLOBALS["DB_PREFIX"] , &$err_message ) ;
$error_array = array() ;

//Retrieve the setting information
$settingInfo = get_Conference_Settings();
$trackStr = $settingInfo->TrackName; //Name for Track
$topicStr = $settingInfo->TopicName; //Name for Topic
$levelStr = $settingInfo->LevelName; //Name for Level

```

```

if ($settingInfo->AbstractOnlySubmissions || $settingInfo->SESUG)
//Abstract submission only for SESUG
{
    $exempt_array = array ( "email" , "middlename" , "presenterbio"
, "keyword1" , "keyword2" , "keyword3" , "userfile", "numpages" ) ;
    $fullPaper = false;
}
else{
    $exempt_array = array ( "email" , "middlename", "presenterbio",
"keyword1" , "keyword2" , "keyword3" , "userfile" ) ;
    $fullPaper = true;
}

if ( count ( $_POST ) > 0 )
{
    if ( $_POST["submit"] == "Update number of Authors" )
    {
        if ( isIntegerMoreThanZero ( $_POST["numauthors"] ,
&$error_array["numauthors"] ) || !empty ( $_POST["numauthors"] ) )
        {
        }
        else
        {
            if ( trim ( $_POST["numauthors"] ) == "" )
            {
                $error_array["numauthors"][0] = " This entry cannot
be empty. <br>\n" ;
            }
        }
    }
    //Can only change category in Phase 1
    else if ( ($_SESSION["phase"]->phaseID == 1) && ($_POST["submit"] ==
"Update") )
    {
        if ( $settingInfo -> SESUG ) {
            if ( !$_POST["level"] )
            {
                $error_array["level"][0] = "You must choose at least one
$levelStr.<br>\n" ;
            }
        }
        if ( !$_POST["track"] )
        {
            $error_array["track"][0] = "You must choose a
$trackStr.<br>\n" ;
        }
        if ( !$_POST["category"] )
        {
            if (numCategories( &$err_message ) > 0) // allow
conferences with only Tracks, but no Topics
                $error_array["category"][0] = "You must choose at
least one $topicStr.<br>\n" ;
        }

        $vars = array_merge ( $_POST , $_FILES ) ;
    }
}

```

```

        check_form ( $vars , $error_array , &$exempt_array ) ;

    }
}

if ( count ( $error_array ) == 0 && count ( $_POST ) > 0 )
{

    if ( $_POST["submit"] == "Update number of Authors" )
    {
        do_html_header("Edit Paper Details" , &$err_message );

    }
    else if ( $_POST["submit"] == "Update" )
    {
        //Submit to update the paper
        if ( $fileID = update_paper_details ( $_GET["paperid"] ,
            $_POST["title"] , $_POST["abstract"] , $_POST["presenterbio"] ,
            $_POST["numpages"] , $_FILES["userfile"]["tmp_name"] ,
            $_FILES["userfile"]["name"] , $_FILES["userfile"]["size"] ,
            $_FILES["userfile"]["type"] ,
            $_POST["firstname"] , $_POST["middlename"] ,
            $_POST["lastname"] , $_POST["email"] , $_POST["level"] ,
            $_POST["track"] , $_POST["category"] , $_POST["attended"]
            ,$_POST["presented"] ,$_POST["keyword1"] ,$_POST["keyword2"]
            ,$_POST["keyword3"] , &$err_message ) )
        {
            do_html_header("Paper Updating Successful..." ,
            &$err_message );
            echo " The paper information has been updated <br><br>
View your updated paper at <a href='view_paper_details.php?fileid=" .
$fileID . "'>View Papers Details</a> page." ;
            do_html_footer( &$err_message );
            exit ;
        }
        else
        {
            do_html_header("Paper Updating Failed..." , &$err_message
            );
            $err_message .= "<br><br> Try <a
href='edit_paper_info.php?paperid=" . $_GET["paperid"] . "'>again</a>?" ;
        }
    }
    else if($_POST["submit"] == "Withdraw")
    {
        //Withdraw the paper here
        if ( withdraw_paper( $_GET["paperid"] , &$err_message ) )
        {
            do_html_header("Withdrawing Paper Successful" ,
            &$err_message );
            echo " The paper has been withdrawn
successfully.<br><br>\n" ;
            do_html_footer( &$err_message );
            exit ;
        }
        else
        {

```

```

do_html_header("Withdrawing Paper Failed..." ,
&$err_message );
    $err_message .= "<br><br> <a
href='edit_paper_info.php?paperid=" . $_GET["paperid"] . "'>Reload</a> this
page.";
    }
}
else if($_POST["submit"] == "Undo Changes")
{
    //Refresh the same page
    $str = "Location: edit_paper_info.php?paperid=" .
$_GET["paperid"] ;
    header( $str ); /* Redirect browser */
    exit; /* Make sure that code below does not get executed when
we redirect. */
}
else
{
    if ( count ( $_POST ) == 0 )
    {
        $_SESSION["phase"]->set_edit_paper_info( $_GET["paperid"] ,
$_POST , &$err_message ) ;
    }
    do_html_header("Edit Paper Details" , &$err_message );
}

$maxfilesize = $settingInfo->MaxUploadSize ;
?>

<form enctype="multipart/form-data" name="frmupload" method="post"
action="edit_paper_info.php?paperid=<?php echo $_GET["paperid"] ?> ">
<!-- <form enctype="multipart/form-data" name="frmupload" method="post"
action="phpinfo.php"> -->
    <table width="100%" border="0" cellspacing="0" cellpadding="3">
        <tr>
            <td width="20%"><strong>Title:</strong></td>
            <td width="80%"> </td>
        </tr>
        <tr>
            <td colspan="2"><input name="title" type="text" value="<?php echo
stripslashes($_POST["title"]) ?>" id="title" size="75" maxlength="255">
                <font color="#FF0000"><?php echo $error_array["title"][0] ?></font>
            </td>
        </tr>
        <tr>
            <td><strong>Number of Pages:</strong></td>
            <td><input name="numpages" type="text" id="numpages" size="5"
maxlength="4" value="<?php echo $_POST["numpages"] ; ?>">
                <?php echo "<font color='\"#FF0000\"'>" . $error_array["numpages"][0]
. "</font>" ; ?>
            </td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td>&nbsp;</td>
        </tr>
    </table>

```



```

        if ($settingInfo -> SESUG) {
            echo "<strong>$levelStr(s):</strong><br>\n " ;
            echo "<font color=\"#FF0000\">" .
$error_array["level"][0] . "</font>" ;

            if ( $result = GenerateSelectedCategoryInputTable(
$_POST["level"] , &$err_message , 0 , "Level" ) )
            {
                echo $result ;
            }
            else
            {
                $err_message .= "<br><br> Try <a
href='edit_paper_info.php?paperid=" . $_POST["paperid"] . "'>again</a>?" ;
            }
            echo "<br>" ;
        }
        echo "<strong>".$settingInfo -> TrackName."</strong><br>\n " ;
        echo "<font color=\"#FF0000\">" . $error_array["track"][0] .
"</font>" ;

        if ( $result = GenerateSelectedCategoryInputTable(
$_POST["track"] , &$err_message , 0 , "Track" ) )
        {
            echo $result ;
        }
        else
        {
            $err_message .= "<br><br> Try <a
href='edit_paper_info.php?paperid=" . $_POST["paperid"] . "'>again</a>?" ;
        }

        echo "<br>" ;
        if (numCategories( &$err_message ) > 0) // allow conferences
with only Tracks, but no Topics
        {
            echo "<strong>".$settingInfo ->
TopicName."(s):</strong><br>\n " ;
            echo "<font color=\"#FF0000\">" .
$error_array["category"][0] . "</font>" ;
            if ( $result = GenerateSelectedCategoryInputTable(
$_POST["category"] , &$err_message ) )
            {
                echo $result ;
            }
            else
            {
                $err_message .= "<br><br> Try <a
href='edit_paper_info.php?paperid=" . $_POST["paperid"] . "'>again</a>?" ;
            }
        }
    }
    else
    {
        if ($settingInfo -> SESUG) {
            echo "<strong>$levelstr:</strong><br>\n " ;

```

```

        echo "<font color=\"#FF0000\">" . $error_array["level"][0] .
"</font>" ;

        if ( $result = GenerateSelectedCategoryInputTable(
$_POST["level"] , &$err_message , 1 , "Level" ) )
        {
            echo $result ;
        }
        else
        {
            $err_message .= "<br><br> Try <a
href='edit_paper_info.php?paperid=" . $_POST["paperid"] . "'>again</a>?" ;
        }
        echo "<br>" ;
    }
    echo "<strong>".$settingInfo -> TrackName."</strong><br>\n " ;
    echo "<font color=\"#FF0000\">" . $error_array["track"][0] .
"</font>" ;

    if ( $result = GenerateSelectedCategoryInputTable(
$_POST["track"] , &$err_message , 1 , "Track" ) )
    {
        echo $result ;
    }
    else
    {
        $err_message .= "<br><br> Try <a
href='edit_paper_info.php?paperid=" . $_POST["paperid"] . "'>again</a>?" ;
    }

    if (numCategories( &$err_message ) > 0) // allow conferences
with only Tracks, but no Topics
    {
        echo "<br>" ;
        echo "<strong>".$settingInfo ->
TopicName."(s):</strong><br>\n " ;
        echo "<font color=\"#FF0000\">" .
$error_array["category"][0] . "</font>" ;
        if ( $result = GenerateSelectedCategoryInputTable(
$_POST["category"] , &$err_message , 1) )
        {
            echo $result ;
        }
        else
        {
            $err_message .= "<br><br> Try <a
href='edit_paper_info.php?paperid=" . $_POST["paperid"] . "'>again</a>?" ;
        }
    }
}
?>
</td>
</tr>
<?php if ($settingInfo -> SESUG) { ?>
<tr>
<td><strong><?php echo $keyword ?> :</strong></td>

```


// CHANGE PASSWORD

```
// <?php
    $php_root_path = ".." ;
    require_once("$php_root_path/includes/include_all_fns.inc");
    session_start();
//    extract ( $_SESSION , EXTR_REFS ) ;

    $err_message = " Unable to process your request due to the following
problems: <br>\n" ;
    $header = "Change Password" ;
    $accepted_privilegeID_arr = array ( 1 => "" ) ;
    $accepted_phaseID_arr = array ( 1 => "" , 2 => "" , 3 => "" , 4 => ""
) ;
    authentication( $header , $accepted_privilegeID_arr ,
$accepted_phaseID_arr , $homepage , $php_root_path , $GLOBALS["DB_PREFIX"]
, &$err_message ) ;
/*
    if ( !check_valid_user( &$err_message ) )
    {
        //This user is not logged in
        do_html_header("Change Password Failed" , &$err_message ) ;

        $err_message .= " Sorry, You must login to change your
password. <br>\n";
        $err_message .= "<br><br> Go to <a
href='$php_root_path/index.php'>Login</a> page." ;
        do_html_footer( &$err_message ) ;
        exit;
    }
*/
    if ( $_POST["submit"] == "Cancel" )
    {
        header("Location: view_papers.php") ;
        exit ;
    }

    $error_array = array() ;
    $exempt_array = array() ;

    check_form ( $_POST , $error_array , &$exempt_array ) ;

    if ( count ( $error_array ) == 0 && count ( $_POST ) > 0 )
    {
        if ( change_password ( $_SESSION["valid_user"] , trim (
$_POST["oldpwd"] ) , trim ( $_POST["newpwd"] ) , trim (
$_POST["confirmpwd"] ) , &$err_message ) )
        {
            // provide link to members page
            do_html_header("Change Password Successful" ,
&$err_message );
            echo "The password has been changed.";
            do_html_footer( &$err_message );
            exit ;
        }
        else
        {

```

```

        // otherwise provide link back, tell them to try again
do_html_header("Change password failed" , &$err_message
);
        $err_message .= "<br><br> Try <a
href='change_pwd.php'>Again</a>? <br>\n" ;
    }
}
else
{
    if ( count ( $_POST ) == 0 )
    {
        do_html_header("Change Password" , &$err_message );

    }
    else
    {
        do_html_header("Change Password" , &$err_message );

    }
}
?>
<form action="change_pwd.php" name="frmReset" method="post">
    <table width="80%" border="0" cellspacing="0" cellpadding="0">
        <tr>
            <td colspan="2"><br>
                <strong> Note: </strong> You may leave this page without changing
your password by clicking
                on any other links in the menu above or by clicking the "Cancel"
button below.</p></td>
        </tr>
        <tr>
            <td width="24%">&nbsp;</td>
            <td width="80%">&nbsp;</td>
        </tr>
        <tr>
            <td>Old Password:</td>
            <td><input name="oldpwd" type="password" id="oldpwd" size="20"
maxlength="15">
                <?php echo "<font color=\"#FF0000\">" . $error_array["oldpwd"][0] .
"</font>" ; ?>
            </td>
        </tr>
        <tr>
            <td>New Password:</td>
            <td><input name="newpwd" type="password" id="newpwd" size="20"
maxlength="15">
                <?php echo "<font color=\"#FF0000\">" . $error_array["newpwd"][0] .
"</font>" ; ?>
            </td>
        </tr>
        <tr>
            <td>Confirm Password:</td>
            <td><input name="confirmpwd" type="password" id="confirmpwd"
size="20" maxlength="15">
                <?php echo "<font color=\"#FF0000\">" .
$error_array["confirmpwd"][0] . "</font>" ; ?>
            </td>
        </tr>
    </table>

```

```

        </tr>
        <tr>
            <td>&nbsp;</td>
            <td>&nbsp;</td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><input name="submit" type="submit" id="submit" value="Submit">
&nbsp;<input name="submit" type="submit" id="cancel" value="Cancel"></td>
        </tr>
    </table>

</form>

<?php
    do_html_footer(&$err_message);

?>

////////////////////////////////////

// FORGET PASSWORD

<?php
    $php_root_path = ".." ;
    $privilege_root_path = "/user" ;
    require_once("$php_root_path/includes/include_all_fns.inc");
    //require_once("$php_root_path."/admin/includes/libmail.php");
    $homepage->showmenu = 0 ;

    $err_message = " Unable to process your request due to the following
problems: <br>\n" ;

    $error_array = array() ;

    check_form ( $_POST , $error_array ) ;

    if ( count ( $error_array ) == 0 && count ( $_POST ) > 0 )
    {
        if ( forget_password( $_POST["username"] , &$err_message ) )
        {
            do_html_header("Reseting Password Successful" ,
&$err_message );
            echo "Your password has been reset and you will receive a
new password to your email address shortly.<br><br>Go to the <a
href='$php_root_path/index.php'>Login</a> page" ;
            do_html_footer( &$err_message );
            exit ;
        }
        else
        {
            do_html_header("Reseting Password Failed" , &$err_message
);

```

```

        $err_message .= " <br><br> Try <a
href='forget_pwd.php'>Again</a>? <br>Go to the <a
href='$php_root_path/index.php'>Login</a> page. <br>\n" ;
    }
}
else
{
    do_html_header("Reset Password" , &$err_message );
}

?>
<form action="forget_pwd.php" method="post" name="loginForm"
id="loginForm">
    <table width="80%" border="0" cellpadding="0" cellspacing="0">
        <tr>
            <td height="24" colspan="2">Not registered. <a
href="registration.php">Sign up for an account.</a></td>
        </tr>
        <tr>
            <td height="10" colspan="2">Hint: Use your email address as
UserName.</td>
        </tr>
        <tr>
            <td width="20%" height="24">&nbsp;</td>
            <td width="80%" height="24">&nbsp;</td>
        </tr>
        <tr>
            <td><strong>UserName</strong></td>
            <td><input name="username" type="text" id="username" size="25"
value="<?php echo $_POST["username"] ; ?>" maxlength="50">
                <?php echo "<font color=\"#FF0000\">" . $error_array["username"][0]
. "</font>" ; ?></td>
            </tr>
        <tr>
            <td>&nbsp;</td>
            <td>&nbsp;</td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><input type="submit" name="Submit" value="Submit"></td>
        </tr>
    </table>
</form>
<?php

do_html_footer( &$err_message );

?>
////////////////////////////////////

// DOWNLOAD FILE

<?php
    //Establish connection with database
    // require_once("includes/db_connect.inc");
    // require_once("includes/user_authen_fns.inc");
    $php_root_path = ".." ;

```



```

        require_once("$php_root_path/includes/include_all_fns.inc");
        session_start();
        // extract ( $_SESSION , EXTR_REFS ) ;
        // $fileid =& $_GET["fileid"] ;

        $err_message = " Unable to process your request due to the following
problems: <br>\n" ;
        $header = "Download File" ;
        $accepted_privilegeID_arr = array ( 1 => "" ) ;
        $accepted_phaseID_arr = array ( 1 => "" , 2 => "" , 3 => "" , 4 => ""
) ;
        authentication( $header , $accepted_privilegeID_arr ,
$accepted_phaseID_arr , $homepage , $php_root_path , $GLOBALS["DB_PREFIX"]
, &$err_message ) ;

        //Establish connection with database
        $db = adodb_connect( &$err_message );

        $sql = "SELECT File,FileName,FileSize,FileType FROM " .
$GLOBALS["DB_PREFIX"] . "File F , " . $GLOBALS["DB_PREFIX"] . "Paper P" ;
        $sql .= " WHERE F.FileID=" . $_GET["fileid"] . " AND
F.PaperID=P.PaperID AND P.MemberName='" . $_SESSION["valid_user"] . "' AND
Withdraw='false'" ;

        // echo "\$sql: " . $sql . "<br>\n";

        $result = $db -> Execute($sql);
        $rows = $result -> RecordCount() ;

        if ( !$result )
        {
            do_html_header("Download Paper Failed" , &$err_message ) ;
            $err_message .= " Could not connect to File database.<br>\n";
            $err_message .= "<br><br> Try <a
href='download_file.php?fileid=" . $_GET["fileid"] . "'>again</a>?" ;

            do_html_footer( &$err_message );
            exit;
        }
        else if ( !$rows )
        {
            do_html_header("Download Paper Failed" , &$err_message ) ;
            $err_message .= " The requested file is not available.<br>\n";

            $err_message .= "<br><br> Try <a
href='download_file.php?fileid=" . $_GET["fileid"] . "'>again</a>?" ;

            do_html_footer( &$err_message );
            exit;
        }

        $row = $result -> FetchNextObj();
        $data = $row -> File;
        $name = $row -> FileName;
        $size = $row -> FileSize;
        $type = $row -> FileType;

```

```

header("Cache-control: private");
header("Content-type: $type");
header("Content-length: $size");
header("Content-Disposition: attachment; filename=$name");
header("Content-Description: PHP Generated Data");
echo $data;

?>

////////////////////////////////////

//VIEW FILE

<?php
    //Establish connection with database
    // require_once("includes/db_connect.inc");
    $php_root_path = ".." ;
    require_once("$php_root_path/includes/include_all_fns.inc");

    session_start();
    // extract ( $_SESSION , EXTR_REFS ) ;
    // $fileid =& $_GET["fileid"] ;
    $err_message = " Unable to process your request due to the following
problems: <br>\n" ;

    $header = "View File" ;
    $accepted_privilegeID_arr = array ( 1 => "" ) ;
    $accepted_phaseID_arr = array ( 1 => "" , 2 => "" , 3 => "" , 4 => ""
) ;
    authentication( $header , $accepted_privilegeID_arr ,
$accepted_phaseID_arr , $homepage , $php_root_path , $GLOBALS["DB_PREFIX"]
, &$err_message ) ;

    //Establish connection with database
    $db = adodb_connect( &$err_message ) ;

    $sql = "SELECT File,FileName,FileSize,FileType FROM " .
$GLOBALS["DB_PREFIX"] . "File F , " . $GLOBALS["DB_PREFIX"] . "Paper P" ;
    $sql .= " WHERE F.FileID=" . $_GET["fileid"] . " AND
F.PaperID=P.PaperID AND P.MemberName='" . $_SESSION["valid_user"] . "' AND
Withdraw='false'" ;
    $result = $db -> Execute($sql);
    $rows = $result -> RecordCount() ;

    if (!$result )
    {
        do_html_header("View File Failed" , &$err_message ) ;
        $err_message .= " Could not connect to File database.<br>\n";

        $err_message .= "<br><br> Try <a href='view_file.php?fileid=" .
$_GET["fileid"] . "'>again</a>?" ;
        do_html_footer( &$err_message ) ;
        exit;
    }
    else if ( !$rows )
    {
        do_html_header("View File Failed" , &$err_message ) ;

```

```

        $err_message .= " The requested file is not available.<br>\n";

        $err_message .= "<br><br> Try <a href='view_file.php?fileid=" .
$_GET["fileid"] . "'>again</a>?" ;
        do_html_footer( &$err_message );
        exit;
    }

    $row = $result -> FetchNextObj();
    $data = $row -> File;
    $name = $row -> FileName;
    $size = $row -> FileSize;
    $type = $row -> FileType;

    // Check for Internet Explorer to avoid inline PDF viewing bug

    $browser = getBrowser( ) ;
    if ($browser == "IEMWin")
    {
        $method = "attachment" ;
    }
    else
    {
        $method = "inline" ;
    }

    header("Cache-control: private");
    header("Content-type: $type" );
    header("Content-length: $size");
    header("Content-Disposition: $method; filename=$name");
    header("Content-Description: PHP Generated Data");
    echo $data;
?>

////////////////////////////////////

//

<?php

require_once('PEAR.php');
require_once('Config/Container.php');

$GLOBALS['CONFIG_TYPES'] =
    array(
        'apache'          => array('Config/Container/Apache.php',
'Config_Container_Apache'),
        'genericconf'     => array('Config/Container/GenericConf.php',
'Config_Container_GenericConf'),
        'inifile'         => array('Config/Container/IniFile.php',
'Config_Container_IniFile'),
        'inicommented'    => array('Config/Container/IniCommented.php',
'Config_Container_IniCommented'),
        'phparray'        => array('Config/Container/PHPArray.php',
'Config_Container_PHPArray'),
    )

```

```

                                'phpconstants'    =>
array('Config/Container/PHPConstants.php',
'Config_Container_PHPConstants'),
                                'xml'            => array('Config/Container/XML.php',
'Config_Container_XML')
                                );

/**
 * Config
 *
 * This class allows for parsing and editing of configuration datasources.
 * Do not use this class only to read datasources because of the overhead
 * it creates to keep track of the configuration structure.
 *
 * @author   Bertrand Mansion <bmansion@mamasam.com>
 * @package  Config
 */
class Config {

    /**
     * Datasource
     * Can be a file url, a dsn, an object...
     * @var mixed
     */
    var $datasrc;

    /**
     * Type of datasource for config
     * Ex: IniCommented, Apache...
     * @var string
     */
    var $configType = '';

    /**
     * Options for parser
     * @var string
     */
    var $parserOptions = array();

    /**
     * Container object
     * @var object
     */
    var $container;

    /**
     * Constructor
     * Creates a root container
     *
     * @access public
     */
    function Config()
    {
        $this->container =& new Config_Container('section', 'root');
    } // end constructor

    /**

```

```

* Returns true if container is registered
*
* @param    string    $configType    Type of config
* @access public
* @return   bool
*/
function isConfigTypeRegistered($configType)
{
    return isset($GLOBALS['CONFIG_TYPES'][strtolower($configType)]);
} // end func isConfigTypeRegistered

/**
* Register a new container
*
* @param    string    $configType    Type of config
* @param    array|false $configInfo    Array of format:
*      array('path/to/Name.php',
*            'Config_Container_Class_Name').
*
*      If left false, defaults to:
*      array('Config/Container/$configType.php',
*            'Config_Container_$configType')
* @access   public
* @static
* @author   Greg Beaver <cellog@users.sourceforge.net>
* @return   true|PEAR_Error    true on success
*/
function registerConfigType($configType, $configInfo = false)
{
    if (Config::isConfigTypeRegistered($configType)) {
        $info = $GLOBALS['CONFIG_TYPES'][strtolower($configType)];
        if ($info[0] == $configInfo[0] &&
            $info[1] == $configInfo[1]) {
            return true;
        } else {
            return PEAR::raiseError("Config::registerConfigType
registration of existing $configType failed.", null, PEAR_ERROR_RETURN);
        }
    }
    if (!is_array($configInfo)) {
        // make the normal assumption, that this is a standard config
        container added in at runtime
        $configInfo = array('Config/Container/' . $configType . '.php',
                           'Config_Container_' . $configType);
    }
    $file_exists = @include_once($configInfo[0]);
    if ($file_exists) {
        if (!class_exists($configInfo[1])) {
            return PEAR::raiseError("Config::registerConfigType class
'$configInfo[1]' not found in $configInfo[0]", null, PEAR_ERROR_RETURN);
        }
    } else {
        return PEAR::raiseError("Config::registerConfigType file
$configInfo[0] not found", null, PEAR_ERROR_RETURN);
    }
    $GLOBALS['CONFIG_TYPES'][strtolower($configType)] = $configInfo;
    return true;
}

```

```

} // end func registerConfigType

/**
 * Returns the root container for this config object
 *
 * @access public
 * @return object reference to config's root container object
 */
function &getRoot()
{
    return $this->container;
} // end func getRoot

/**
 * Sets the content of the root Config_container object.
 *
 * This method will replace the current child of the root
 * Config_Container object by the given object.
 *
 * @param object $rootContainer container to be used as the first
child to root
 * @access public
 * @return mixed true on success or PEAR_Error
 */
function setRoot(&$rootContainer)
{
    if (is_object($rootContainer) &&
strtolower(get_class($rootContainer)) === 'config_container') {
        if ($rootContainer->getName() === 'root' && $rootContainer-
>getType() === 'section') {
            $this->container =& $rootContainer;
        } else {
            $this->container =& new Config_Container('section',
'root');
            $this->container->addItem($rootContainer);
        }
        return true;
    } else {
        return PEAR::raiseError("Config::setRoot only accepts object of
Config_Container type.", null, PEAR_ERROR_RETURN);
    }
} // end func setRoot

/**
 * Parses the datasource contents
 *
 * This method will parse the datasource given and fill the root
 * Config_Container object with other Config_Container objects.
 *
 * @param mixed $datasrc Datasource to parse
 * @param string $configType Type of configuration
 * @param array $options Options for the parser
 * @access public
 * @return mixed PEAR_Error on error or Config_Container object
 */
function &parseConfig($datasrc, $configType, $options = array())
{

```

```

        $configType = strtolower($configType);
        if (!$this->isConfigTypeRegistered($configType)) {
            return PEAR::raiseError("Configuration type '$configType' is
not registered in Config::parseConfig.", null, PEAR_ERROR_RETURN);
        }
        $includeFile = $GLOBALS['CONFIG_TYPES'][$configType][0];
        $className = $GLOBALS['CONFIG_TYPES'][$configType][1];
        include_once($includeFile);

        $parser = new $className($options);
        $error = $parser->parseDataSrc($dataSrc, $this);
        if ($error !== true) {
            return $error;
        }
        $this->parserOptions = $parser->options;
        $this->dataSrc = $dataSrc;
        $this->configType = $configType;
        return $this->container;
    } // end func &parseConfig

/**
 * Writes the container contents to the datasource.
 *
 * @param mixed    $dataSrc    Datasource to write to
 * @param string   $configType Type of configuration
 * @param array    $options    Options for config container
 * @access public
 * @return mixed PEAR_Error on error or true if ok
 */
function writeConfig($dataSrc = null, $configType = null, $options =
array())
{
    if (empty($dataSrc)) {
        $dataSrc = $this->dataSrc;
    }
    if (empty($configType)) {
        $configType = $this->configType;
    }
    if (empty($options)) {
        $options = $this->parserOptions;
    }
    return $this->container->writeDataSrc($dataSrc, $configType,
$options);
} // end func writeConfig
} // end class Config
?>
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// MAIL CHECKING USERS
<?php
require_once 'PEAR.php';

/**
 * PEAR's Mail:: interface. Defines the interface for implementing
 * mailers under the PEAR hierarchy, and provides supporting functions
 * useful in multiple mailer backends.
 *
 * @access public

```

```

* @version $Revision: 1.17 $
* @package Mail
*/
class Mail
{
    /**
     * Line terminator used for separating header lines.
     * @var string
     */
    var $sep = "\r\n";

    /**
     * Provides an interface for generating Mail:: objects of various
     * types
     *
     * @param string $driver The kind of Mail:: object to instantiate.
     * @param array $params The parameters to pass to the Mail:: object.
     * @return object Mail a instance of the driver class or if fails a
PEAR Error
     * @access public
     */
    function &factory($driver, $params = array())
    {
        $driver = strtolower($driver);
        @include_once 'Mail/' . $driver . '.php';
        $class = 'Mail_' . $driver;
        if (class_exists($class)) {
            $mailer = new $class($params);
            return $mailer;
        } else {
            return PEAR::raiseError('Unable to find class for driver ' .
$driver);
        }
    }

    /**
     * Implements Mail::send() function using php's built-in mail()
     * command.
     *
     * @param mixed $recipients Either a comma-seperated list of recipients
     * (RFC822 compliant), or an array of recipients,
     * each RFC822 valid. This may contain recipients not
     * specified in the headers, for Bcc:, resending
     * messages, etc.
     *
     * @param array $headers The array of headers to send with the mail, in
an
     * associative array, where the array key is the
     * header name (ie, 'Subject'), and the array value
     * is the header value (ie, 'test'). The header
     * produced from those values would be 'Subject:
     * test'.
     *
     * @param string $body The full text of the message body, including any
     * Mime parts, etc.
     *
     * @return mixed Returns true on success, or a PEAR_Error

```



```

*             containing a descriptive error message on
*             failure.
* @access public
* @deprecated use Mail_mail::send instead
*/
function send($recipients, $headers, $body)
{
    $this->_sanitizeHeaders($headers);

    // if we're passed an array of recipients, implode it.
    if (is_array($recipients)) {
        $recipients = implode(', ', $recipients);
    }

    // get the Subject out of the headers array so that we can
    // pass it as a separate argument to mail().
    $subject = '';
    if (isset($headers['Subject'])) {
        $subject = $headers['Subject'];
        unset($headers['Subject']);
    }

    // flatten the headers out.
    list($text_headers) = Mail::prepareHeaders($headers);

    return mail($recipients, $subject, $body, $text_headers);
}

/**
 * Sanitize an array of mail headers by removing any additional header
 * strings present in a legitimate header's value. The goal of this
 * filter is to prevent mail injection attacks.
 *
 * @param array $headers The associative array of headers to sanitize.
 *
 * @access private
 */
function _sanitizeHeaders(&$headers)
{
    foreach ($headers as $key => $value) {
        $headers[$key] =
preg_replace('=((<CR>|<LF>|0x0A/%0A|0x0D/%0D|\\n|\\r)\\S).*=i',
                null, $value);
    }
}

/**
 * Take an array of mail headers and return a string containing
 * text usable in sending a message.
 *
 * @param array $headers The array of headers to prepare, in an
associative
 *
 *             array, where the array key is the header name (ie,
 *             'Subject'), and the array value is the header
 *             value (ie, 'test'). The header produced from those

```

```

*           values would be 'Subject: test'.
*
* @return mixed Returns false if it encounters a bad address,
*           otherwise returns an array containing two
*           elements: Any From: address found in the headers,
*           and the plain text version of the headers.
* @access private
*/
function prepareHeaders($headers)
{
    $lines = array();
    $from = null;

    foreach ($headers as $key => $value) {
        if (strcasecmp($key, 'From') === 0) {
            include_once 'Mail/RFC822.php';
            $parser = &new Mail_RFC822();
            $addresses = $parser->parseAddressList($value, 'localhost',
false);

            if (PEAR::isError($addresses)) {
                return $addresses;
            }

            $from = $addresses[0]->mailbox . '@' . $addresses[0]->host;

            // Reject envelope From: addresses with spaces.
            if (strstr($from, ' ')) {
                return false;
            }

            $lines[] = $key . ': ' . $value;
        } elseif (strcasecmp($key, 'Received') === 0) {
            $received = array();
            if (is_array($value)) {
                foreach ($value as $line) {
                    $received[] = $key . ': ' . $line;
                }
            }
            else {
                $received[] = $key . ': ' . $value;
            }
            // Put Received: headers at the top.  Spam detectors often
            // flag messages with Received: headers after the Subject:
            // as spam.
            $lines = array_merge($received, $lines);
        } else {
            // If $value is an array (i.e., a list of addresses),
convert
            // it to a comma-delimited string of its elements
(addresses).
            if (is_array($value)) {
                $value = implode(', ', $value);
            }
            $lines[] = $key . ': ' . $value;
        }
    }
}

```

```

        return array($from, join($this->sep, $lines));
    }

    /**
     * Take a set of recipients and parse them, returning an array of
     * bare addresses (forward paths) that can be passed to sendmail
     * or an smtp server with the rcpt to: command.
     *
     * @param mixed Either a comma-seperated list of recipients
     *              (RFC822 compliant), or an array of recipients,
     *              each RFC822 valid.
     *
     * @return mixed An array of forward paths (bare addresses) or a
PEAR_Error
     *              object if the address list could not be parsed.
     * @access private
     */
    function parseRecipients($recipients)
    {
        include_once 'Mail/RFC822.php';

        // if we're passed an array, assume addresses are valid and
        // implode them before parsing.
        if (is_array($recipients)) {
            $recipients = implode(', ', $recipients);
        }

        // Parse recipients, leaving out all personal info. This is
        // for smtp recipients, etc. All relevant personal information
        // should already be in the headers.
        $addresses = Mail_RFC822::parseAddressList($recipients,
'localhost', false);

        // If parseAddressList() returned a PEAR_Error object, just return
it.
        if (PEAR::isError($addresses)) {
            return $addresses;
        }

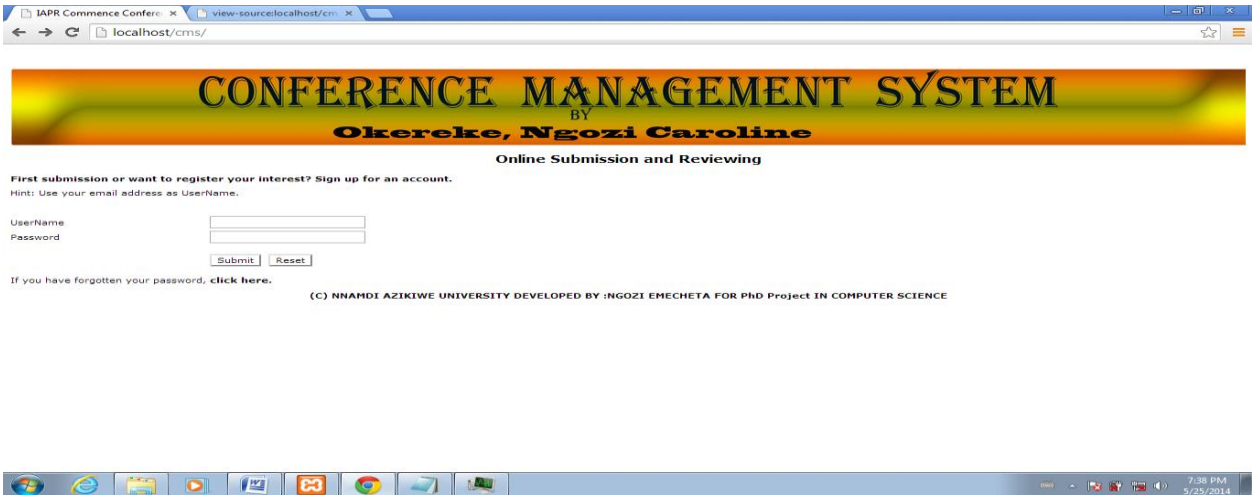
        $recipients = array();
        if (is_array($addresses)) {
            foreach ($addresses as $ob) {
                $recipients[] = $ob->mailbox . '@' . $ob->host;
            }
        }

        return $recipients;
    }
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

APPENDIX B: OUTPUT SCREEN



CONFERENCE MANAGEMENT SYSTEM
BY
Okereke, Ngozi Caroline
Online Submission and Reviewing

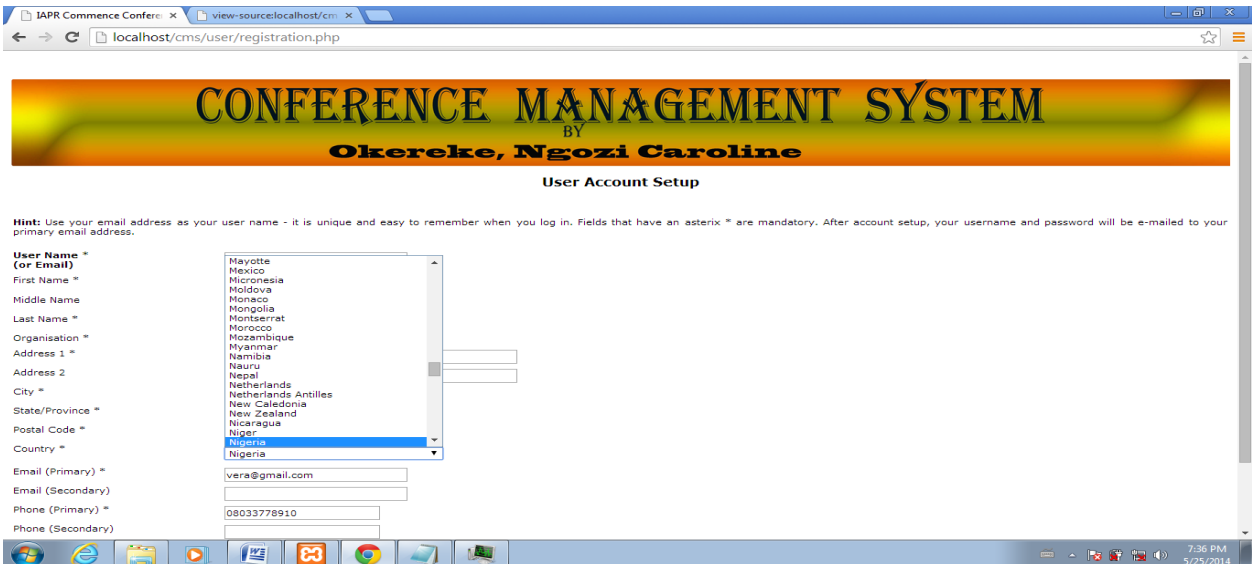
First submission or want to register your interest? Sign up for an account.
Hint: Use your email address as UserName.

UserName
Password

Submit Reset

If you have forgotten your password, [click here.](#)

(C) NNANDI AZIKIWE UNIVERSITY DEVELOPED BY INGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE



CONFERENCE MANAGEMENT SYSTEM
BY
Okereke, Ngozi Caroline
User Account Setup

Hint: Use your email address as your user name - it is unique and easy to remember when you log in. Fields that have an asterisk * are mandatory. After account setup, your username and password will be e-mailed to your primary email address.

User Name * (or Email)
First Name *
Middle Name
Last Name *
Organisation *
Address 1 *
Address 2
City *
State/Province *
Postal Code *
Country *
Email (Primary) *
Email (Secondary)
Phone (Primary) *
Phone (Secondary)

Mayotte
Mexico
Micronesia
Moldova
Monaco
Mongolia
Montserrat
Morocco
Mozambique
Myanmar
Namibia
Nauru
Nepal
Netherlands
Netherlands Antilles
New Caledonia
New Zealand
Nicaragua
Niger
Nigeria

vera@gmail.com
08033778910

localhost/cms/user/registration.php

Okereke, Ngozi Caroline

User Account Setup

Hint: Use your email address as your user name - it is unique and easy to remember when you log in. Fields that have an asterix * are mandatory. After account setup, your username and password will be e-mailed to your primary email address.

User Name * (or Email)

First Name * vera

Middle Name Akanwa

Last Name * Okorie

Organisation * Oke Polytechnic

Address 1 * 5 Awkuzu street Oke

Address 2

City * Oke

State/Province * Anambra

Postal Code * 500009

Country * Nigeria

Email (Primary) * vera@gmail.com

Email (Secondary)

Phone (Primary) * 08033778910

Phone (Secondary)

Fax No

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

Online Submission and Reviewing

First submission or want to register your interest? Sign up for an account.

Hint: Use your email address as UserName.

UserName: ngozi

Password: *****

If you have forgotten your password, [click here](#).

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

localhost/cms/admin/view_conference_info.php

Home | Conference | Papers | User Admin | Program | General Admin | Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

Conference Information

Nigeria Computer Society

Code Name:	NCS2014
Duration:	From Wed, 23 Jul 2014 To Fri, 25 Jul 2014
Location:	Port Harcourt
Host Name:	Nigeria Computer Society
Contact Email:	ncs@ncs.org.ng

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

IAPR Commence Confere X view-source:localhost/cm X

localhost/cms/admin/setup_new_account.php?accountType=Reviewer

Home Conference Papers User Admin Program General Admin Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

Reviewer Account Setup

Reviewer Login Information

Fields that have an asterix * are mandatory

Login Name *

First Name *

Middle Name

Last Name *

Email Address *

If you fill in the organization field, the new reviewer will not have to register himself.

Organization

☐ inform the user now

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

IAPR Commence Confere X

localhost/cms/admin/edit_conference_info.php

Home Conference Papers User Admin Program General Admin Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

Edit Conference Information

Name:

Code Name: (e.g., WDIC2003)

Start Date:

End Date:

Location:

Host Society Name:

Contact Email:

Logo for Conference: No file chosen (jpeg,png)

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

IAPR Commence Conferenc

localhost/cms/admin/view_phases.php

Home Conference Papers User Admin Program General Admin Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

[View Phase](#)

The current running phase is already expired. It is time to change to the next phase.

Paper Submission

Duration: From **Thu, 2 Dec 2010 To Sat, 1 Jan 2011**

Available Phases:

	Phase Name	Start Date	End Date
1.	Paper Submission	Thu, 2 Dec 2010	Sat, 1 Jan 2011
2.	Reviewer Bidding	Sat, 18 Dec 2010	Fri, 31 Dec 2010
3.	Reviewing	Sat, 11 Dec 2010	Fri, 19 Aug 2011
4.	Final Paper Submission	Mon, 10 Jan 2011	Wed, 5 Jan 2011

Phase Information

Paper Submission - Also known as the "Call for Papers" phase. Users open their accounts to register interest in the conference. Later they may upload their papers, if any. Users can make changes to their papers and upload additional papers until the next phase.

Section Chair Bidding - This is where the section chairs bid (indicate preferences) for papers they wish to review or where they have a conflict of interest. Users are prevented from submitting new papers and editing current papers in this phase. This process is designed to reduce the risk of inappropriate assignment of papers to section chairs. You may skip bidding if you wish to manually assign all section chairs to papers.

Reviewing - The users are prevented from submitting new papers and editing current papers while the section chairs are reviewing the papers. Section Chairs can submit and edit their reviews until the end of this phase.

Final Paper Submission - Users may revise existing papers but are prevented from submitting additional papers. Section Chairs are prevented from changing their reviews.

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

javahouse-ngozi.com

6:58 PM
5/25/2014

IAPR Commence Conferenc

localhost/cms/admin/view_categories.php

Home Conference Papers User Admin Program General Admin Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

[View Topic\(s\)](#)

[Add new topic](#)

Topic		
Medical Software and Applications	edit	delete
Computer vision	edit	delete
Structural pattern recognition	edit	delete
Image coding and processing	edit	delete
Biomedical pattern analysis	edit	delete
Speech recognition	edit	delete
Software Engineering	edit	delete

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

7:03 PM
5/25/2014

IAPR Commence Confer: x

localhost/cms/admin/view_all_users.php

Home Conference Papers User Admin Program General Admin Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

View All Users

Total Users: 12

Order By: Privilege - Ascending

User Name	Full Name	Organization	Privilege
william	William Shola Enck	University of Lagos	Reviewer
precilia	Precilia Barilaa Adams	University of Calabar	Reviewer
peter	Peter Borb Gilbert	University of Ibadan	Reviewer
bartho	Bartho Grandy Ekebon	University of Leads	Reviewer
brandy	Brandy Demo Ebele	University of Leads	Reviewer
emmanuel	Emmanuel Paul Gaibrel	University of Poland	Reviewer
yoyesh	Yogesh Singh Gogo	University of Poland	Reviewer
joy	Joy Mercy Nwaji	University of Portharcourt	Reviewer
kaakaki	Kaakaki Saro Aggarwai	University of Lagos	Reviewer
mary	Mary Ann Passcal	University of Portharcourt	Reviewer
eguavon	Eguavon Mark	University of Port Harcourt	Administrator
ngozi	Eguavon Mark	University of Port Harcourt	Administrator

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

7:05 PM
5/25/2014

IAPR Commence Confer: x

localhost/cms/admin/view_all_reviewers.php

Home Conference Papers User Admin Program General Admin Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

View All Users

Total Reviewers: 10

Order By: UserName - Ascending

Reviewer Name	Full Name	Organization	Delete Account
bartho	Bartho Grandy Ekebon	University of Leads	Delete Account
brandy	Brandy Demo Ebele	University of Leads	Delete Account
emmanuel	Emmanuel Paul Gaibrel	University of Poland	Delete Account
joy	Joy Mercy Nwaji	University of Portharcourt	Delete Account
kaakaki	Kaakaki Saro Aggarwai	University of Lagos	Delete Account
mary	Mary Ann Passcal	University of Portharcourt	Delete Account
peter	Peter Borb Gilbert	University of Ibadan	Delete Account
precilia	Precilia Barilaa Adams	University of Calabar	Delete Account
william	William Shola Enck	University of Lagos	Delete Account
yoyesh	Yogesh Singh Gogo	University of Poland	Delete Account

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

7:07 PM
5/25/2014

IAPR Commence Confere X

localhost/cms/admin/setup_new_account.php?accountType=Reviewer

Home Conference Papers User Admin Program General Admin Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

Reviewer Account Setup

Reviewer Login Information

Fields that have an asterisk * are mandatory

Login Name *

First Name *

Middle Name

Last Name *

Email Address *

If you fill in the organization field, the new reviewer will not have to register himself.

Organization

☐ inform the user now

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

IAPR Commence Confere X

localhost/cms/admin/name_tags_preview.php

Home Conference Papers User Admin Program General Admin Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

Name Tags Preview

From: 1 - 6

Printer Friendly Version

Prev | 1 2 3 4 5 6 7 | Next

<p>NCS2014 Nigeria Computer Society Port Harcourt, 2014-07-23 to 2014-07-25</p> <p>Eguavon Mark University of Port Harcourt</p> <p>Hosted by Nigeria Computer Society</p>	<p>NCS2014 Nigeria Computer Society Port Harcourt, 2014-07-23 to 2014-07-25</p> <p>Emmanuel Ordu University of Uyo</p> <p>Hosted by Nigeria Computer Society</p>
<p>NCS2014 Nigeria Computer Society Port Harcourt, 2014-07-23 to 2014-07-25</p> <p>Vera Uloamaka University of Port Harcourt</p> <p>Hosted by Nigeria Computer Society</p>	<p>NCS2014 Nigeria Computer Society Port Harcourt, 2014-07-23 to 2014-07-25</p> <p>Ngozi Emeheta University of port Harcourt</p> <p>Hosted by Nigeria Computer Society</p>

IAPR Commence Confer: X
localhost/cms/admin/view_letters.php

Home Conference Papers User Admin Program General Admin Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

View Formatted Letters

Total Letters: 6 Order By: LetterID - Ascending

ID	LetterID	Title	
#1		Title: Reviewer Invitation and Instructions Subject: Invitation to join Nigeria Computer Society Recipient Group: Reviewers	<ul style="list-style-type: none"> View Letter Edit Letter Send Letter
#2		Title: User Account Info Subject: \$confcode Account Information Recipient Group: Users	<ul style="list-style-type: none"> View Letter Edit Letter
#3		Title: Reviewer Account Info Subject: \$confcode Reviewer Account Information Recipient Group: Reviewers	<ul style="list-style-type: none"> View Letter Edit Letter Send Letter
#4		Title: Admin Account Info Subject: \$confcode Admin Account Information Recipient Group: Administrators	<ul style="list-style-type: none"> View Letter Edit Letter Send Letter
#5		Title: Paper Acceptance Subject: \$confcode Paper Status Recipient Group: Accepted Users	<ul style="list-style-type: none"> View Letter Edit Letter Send Letter
#6		Title: Paper Rejection Subject: \$confcode Paper Status	<ul style="list-style-type: none"> View Letter Edit Letter Send Letter

7:16 PM
5/25/2014

IAPR Commence Confer: X
view-source:localhost/cm: X
localhost/cms/admin/reviewer_preferences.php

Home Conference Papers User Admin Program General Admin Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

Reviewer Preferences

Total Reviewers: 10

Preferences for: **bartho** Batho Grandy Ekebon University of Leads

Track 1st2ndnone

Pattern Recognition ☒ ☐ ☐

Computer Vision ☐ ☐ ☒

Medical Imaging ☐ ☐ ☒

Applications ☐ ☐ ☒

Preferences for: **brandy** Brandy Demo Ebele University of Leads

Track 1st2ndnone

Pattern Recognition ☐ ☒ ☐

Computer Vision ☐ ☐ ☒

Medical Imaging ☒ ☐ ☐

Applications ☐ ☐ ☒

Preferences for: **emmanuel** Emmanuel Paul Gaibrel University of Poland

Track 1st2ndnone

Pattern Recognition ☐ ☒ ☐

Computer Vision ☒ ☐ ☐

Medical Imaging ☐ ☐ ☒

7:18 PM
5/25/2014

view-source:localhost/cm/ x

localhost/cms/admin/rooms.php

Home | Conference | Papers | User Admin | Program | General Admin | Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

Available Rooms

Add Room

Room Name			
Meeting room 1	Edit	Delete	
Meeting room 2	Edit	Delete	

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

view-source:localhost/cm/ x

localhost/cms/admin/presentation_types.php

Home | Conference | Papers | User Admin | Program | General Admin | Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

Available Presentation Types

Add Presentation Type

Presentation Type Name	Slot Length		
Oral	20	Edit	Delete
Poster	5	Edit	Delete
Invited	60	Edit	Delete

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

view-source:localhost/cm/ x

localhost/cms/admin/sessions.php

Home | Conference | Papers | User Admin | Program | General Admin | Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

Conference Sessions

Add new session | Allocate waiting papers to sessions | Deallocate all papers from sessions

Group by: Name | Track | Start Time | Room | Chairperson

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

view-source:localhost/cm: X

localhost/cms/admin/general_settings.php

Home Conference Papers User Admin Program General Admin Logout

CONFERENCE MANAGEMENT SYSTEM

BY
Okereke, Ngozi Caroline

Settings

General Settings
Website Homepage:

Upload Settings
Max Upload Paper File Size
For uploaded papers: MB
Maximum Logo File Size
For uploaded logo file: MB
Maximum Logo Dimensions
(Height x Width in pixels)
Logos larger than these dimensions will be resized.
 x

Email Settings
Email Signature: (optional)
This text will be included at the end of all outgoing email.

Localization Settings
Primary Topic Term
A paper can have only one primary topic.
Secondary Topics Term
A paper can have many secondary topics in addition to its primary topic.
Default Country:
Track:
Topic:
United States-International
7:22 PM
5/25/2014

view-source:localhost/cm: X

localhost/cms/admin/general_settings.php

Home Conference Papers User Admin Program General Admin Logout

Secondary Topics Term
A paper can have many secondary topics in addition to its primary topic.
Default Country:
Short Time Format:
Long Time Format:

Conference-Specific Settings
Abstract-only Submissions
If checked, COMMENCE will allow paper submissions that only contain an abstract.
Double Blind Reviewing
If checked, author's names will be suppressed in Reviewer interface

Colour and Style
Choose Your Own Colours
Customize your own background colour.
Customize your own foreground colour.
Apply Style Sheet

User Phase
This option allows for manual control of the user phase.
Default is the same as the administrator phase.
Default: ☒ Submission ☐ Bidding ☐ Reviewing ☐ Final Submission

Reviewer Phase
This option allows for manual control of the reviewer phase.
Default is the same as the administrator phase.
Default: ☒ Submission ☐ Bidding ☐ Reviewing ☐ Final Submission

PDF Validation Settings
Enable validating of submitted PDF-files
If checked, COMMENCE will use an external program (e.g. Pdfto) to validate pdf-files.
☐

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

7:23 PM
5/25/2014

view-source:localhost/cm: X

localhost/cms/admin/general_settings.php

Home Conference Papers User Admin Program General Admin Logout

Change Settings
Import/Export Settings
Extract All Papers
Build CD Structure
Wednesday, 31 December 1969

Secondary Topics Term
A paper can have many secondary topics in addition to its primary topic.
Default Country:
Short Time Format:
Long Time Format:

Conference-Specific Settings
Abstract-only Submissions
If checked, COMMENCE will allow paper submissions that only contain an abstract.
Double Blind Reviewing
If checked, author's names will be suppressed in Reviewer interface

Colour and Style
Choose Your Own Colours
Customize your own background colour.
Customize your own foreground colour.
Apply Style Sheet

User Phase
This option allows for manual control of the user phase.
Default is the same as the administrator phase.
Default: ☒ Submission ☐ Bidding ☐ Reviewing ☐ Final Submission

Reviewer Phase
This option allows for manual control of the reviewer phase.
Default is the same as the administrator phase.
Default: ☒ Submission ☐ Bidding ☐ Reviewing ☐ Final Submission

PDF Validation Settings
Enable validating of submitted PDF-files
If checked, COMMENCE will use an external program (e.g. Pdfto) to validate pdf-files.
☐

(C) NNAMDI AZIKIWE UNIVERSITY DEVELOPED BY :NGOZI EMECHETA FOR PhD Project IN COMPUTER SCIENCE

7:24 PM
5/25/2014

