Summer 2009

Designing and Implementing a Distributed Database for a Small Multi-Outlet Business

Joseph Grech
Regis University

Recommended Citation
Grech, Joseph, "Designing and Implementing a Distributed Database for a Small Multi-Outlet Business" (2009). *All Regis University Theses.* Paper 5.

Designing and Implementing a Distributed Database for a Small Multi-Outlet Business

Joseph Grech

Regis University

School for Professional Studies

Masters of Science in Software and Information Systems

To Isabella Farrugia

Acknowledgements

My sincere gratitude goes to my project advisor, Brad Blake, and Don Ina for their professional guidance, support and supervision throughout this project.

I would also like to thank all those who, in one way or another, were instrumental in the writing up of this dissertation, amongst whom is my uncle, the Rev. Emanuel Magro, and Kevin Muscat without whose support this dissertation would not have been possible.

The greatest debt and gratitude are owed to my fiancée; Isabella Farrugia and to my family: my parents M' Angela and Anthony who paid for my education, my sisters and my grandmother, Rita. They have all helped me endlessly.

Abstract

Data is a fundamental and necessary element for businesses. During their operations they generate a certain amount of data that [illegible] store, and later on retrieve when required. Databases provide the mean[illegible]vely retrieve data. Such a database can help a business improve its servic[illegible]ve, and ultimately increase its profits. In this paper, the system requirements [illegible]base are researched for a movie rental and sale store that has at least two outlets in different locations besides the main one. This project investigates the different stages of such a database, namely, the planning, analysis, decision, implementation and testing.

Table of Contents

List of Tables

Chapter 1 – Introduction

1.1 Statement of the problem

In Malta, there are a number of establishments that provide the rental, booking and selling of DVDs and movies. In some of these shops, all business operations are performed through the manual filling of paper forms. This method of managing data is not very efficient, and sometimes it creates inconsistent data.

To be successful, a business needs to remain competitive. Data is a valuable asset in any business. When, and if it is used wisely, it can help the business improve its services, be more competitive, and in the end increase its profits. The use of an information system, whereby the data is stored in an effective and efficient database, may provide many benefits to a business. Such a system offers shorter processing time frames than a manual one, and consequently provides quicker and better services. One can retrieve data quickly and generate reports, providing the management team with current information regarding the running of the business. As a result the management and/or directors of a business may immediately view the reports, and use this valuable information when making important decisions.

Databases are a collection of organized data, and structured in such a way as to retrieve the data easily and quickly. Well maintained databases lead to the consistency and integrity of the stored data. This is of crucial importance for the generation of reliable reports. These are only some of the advantages that databases have over manual methods for storing data. Meanwhile, one of the weaknesses of such manual methods is the time spent in gathering and retrieving the data. Documents or files may be duplicated, or not validated. This may lead to invalid reports, which ultimately may adversely affect the business itself. The use of a database reduces the time spent in gathering and retrieving the data, avoids duplication of documents or files, validates the documents or files, and provides accurate information to both owners and users of a business.

The introduction and use of a database system in these establishments will expedite their daily operations. It will provide better services to their clients, and precise and current information to the owners of these businesses. This in turn will result in better customer satisfaction, an increased turnover, and thus higher profits. Such a system would be highly beneficial, especially if these stores wanted to expand into more outlets.

In such a movie rental and sale business, data is important and vital throughout the different sectors of the business hierarchy. For example, in the rental section of the establishment, the employee asks for the customer's information to check out whether the customer is registered as a regular client, and if so, he also checks whether the client has any pending movies that still have to be returned.

Another relevant operation which is performed manually, but could easily be performed by the system is to check whether a particular movie is at the store or rented out. If an item is not available, the customer may reserve it to be picked up when it becomes available. The management may also benefit from the data collected, especially when it comes to the checking of rented and/or sold items, as well as of non-returned items, and the calculation of the daily amount of profit from rentals and sales.

Similarly, in the sales division of such a rental and selling business, the attendant at the cash register may require information regarding the item, as well as providing its price or cost for renting or selling it. When it comes to stock taking, the responsible employee may be able to view the remaining available quantities of particular items in stock, as well as decide on the number that should be ordered. Finally, the information which is gathered from the sales operation can help management to obtain a clearer idea of the preferences of their clientele with regards to the types of items which are most sought after. Consequently, the management would be in a better position to administer the stock of items, especially that of unpopular ones.

The decision to expand into different outlets in various locations brings with it different requirements besides those described above. The management needs to be able to view reports that concern all the sites regarding their performance, as well as their stock. The employees working in different locations need to know and share information regarding the renting, and selling of the items that are in different sites. These employees need this information especially when customers request a particular item that is no longer available at their location. In order to satisfy the customer, these employees need to know in real time whether the sought item is available at any other location, and to put this item on hold for the customer to pick it up. Such a system will provide better customer services.

In such a scenario, the database has to be structured in such a way that all data can be shared by all. A distributed database is the ideal solution for such a situation.

1.2 Relevance, significance or need for the project

The design and implementation of a distributed database is beneficial for such a DVD and movie rental and sale business. This type of database is to incorporate three services, namely, the renting, booking and sales of movies by this business. In order to cater for data within three different outlets, a projected distributed database is considered.

1.3 Difficulties and limitations

This project is undertaken on a theoretical level in the sense that no particular request from any such business was addressed to the author. The author had to analyze of his own accord the daily process of a business similar to the one in question. The knowledge of the business requirements is essential, since these determine which data is to be stored, how it is stored, and which reports may be created.

The second difficulty was to find information about distributed databases, and the different types available. This helped the author to compare and contrast various database solutions, and to choose the best software for this company.

The first limitation of this project is that the database was not actually installed in a rental/selling DVD and movie business. Hence, the full benefits of this project could not be evaluated. The evaluation of this project was done mainly by the testing of the database.

Another limitation is that this project deals only with the database design and implementation. In order to have a fully functional system, a web or software application is needed in order to help the users to populate, edit the data and also to generate reports from the database. Furthermore, the actual use of this application could help the author to identify more issue to fine tune the database design.

1.4 Project scope

Small businesses are becoming more conscious of the fact that proper storage and quick retrieval of valuable information can help them in their work. These requirements were identified for a DVD and movie rental/selling business leading to the design of a database to address these needs. Since such a business may have different outlets in different locations, a distributed database also needed to be designed. An evaluation of the different database management software was made in order to select the best software for such a business. The final goal of the project is to provide the most suitable distributed database for this business.

1.5 List of definitions

Database – “A shared collection of logically related data, and a description of this data, designed to meet the information needs of an organization”(Connolly & Begg, 2005).

Distributed Database – A distributed database is a logically interrelated collection of shared data (and a description of the data) physically distributed over a computer network.

Entity – “Is a distinct object (a person, place, thing, concept, or event) in the organization that is to be represented in the database” (Connolly & Begg, 2005).

Report – In computer systems, it is referred to as the output generated, especially printed, containing the business information.

Structured Query Language (SQL) – “An industry-standard language for creating, updating and querying relational database management systems” (Dictionary.com, n.d.)

SQL Script – A document containing the SQL commands to be executed.

1.6 Summary of the project

The project began by the author’s observation that a movie rental business was using paper files for logging in the data in its daily business. A database was created in order to solve the problems encountered with the paper system. First, the possible requirements of this business were deducted by analyzing its daily processes, and later the needed requirements were determined. Then, different database software solutions were evaluated, to select the best one for this business. Consequently, the database for this business was designed to cater for all the data needed in its daily process flow, and the possible reports to be used. During the design, the distribution of the data was also given special consideration.

The installation of the selected database was set on different computers representing the computers of the different outlets. Afterwards, the script for the database creation was developed and executed for the computers of the different outlets. Furthermore, the required settings were also configured in order to set up a distributed database among the different databases.

Finally, the three databases were loaded with dummy data which represented the data found in a similar business. This was followed by the testing of the database by querying it and analyzing the results.

Chapter 2 – Review of Literature and Research

2.1 Overview of all literature and research sources on the project

In this chapter, databases will be defined, as well as distributed databases and the underlying database management system (DBMS). Also in this section, some of the advantages and disadvantages of a distributed database will be discussed and compared with those of a centralized database. Furthermore, a comparison of the different types of distributed systems and their architectures will be introduced. This chapter will also include the extra design consideration one has to take when building a distributed database. Additionally, the various DBMS which are available on the market, in order to construct a distributed database for a small business, will also be compared.

2.2 Research methods used in investigating the problem and support of thesis statement

The sources of this research paper were obtained mainly from online sources and a number of publications. These sources include journals and papers which concern databases, mainly distributed databases, and its advantages/disadvantages, architectures, components, and also its design considerations.

2.3 Literature and research relevant to the project

2.3.1 Research on databases

As stated by Calero, Piattini, and Ruiz (2003), a known fact is that databases are “the centre of today’s information,” and are increasing in importance “judging by the huge business that they generate” (p. 48). Databases are extremely important since they “are the nucleus” of most systems, especially management systems (Calero, Piattini, & Ruiz, 2003, p. 48). A database is a system which is used as a repository for the user’s data. Apart from storing the user’s data, it is also responsible for structuring this data in an efficient manner. In order to achieve this efficient structuring of the data, information about the actual data, known as metadata, is needed (Rob & Coronel, 2004).

Not only should a system be able to store data in a structured manner, but it should also be able to maintain the data consistently. For this purpose, the DBMS is used for “managing and accessing large amount[s] of data consistently and efficiently” (Franklin, Halvey, & Maier, 2005, p. 27). This helps the system programmers to focus more on specific tasks regarding their applications, rather than focusing on the actual managing of the user data. The DBMS is also responsible for providing “rich data manipulation and query processing with well-understood, strong semantics” while at the same time guaranteeing that these data manipulations are concurrent and persistent (Franklin, et al., 2005, p. 27). In order for the DBMS to provide all these functionalities, it requires the data to be “under the control of a single administrative domain and to conform to a single schema” (Franklin, et al., 2005, p. 27).

There exist two main forms of database systems (DBS), known as centralized and decentralized database systems. The main distinction between these two is in reality the location of the data. Centralized database systems support data located at a single location.

On the other hand, a decentralized or distributed database system (DDS) supports data which is distributed in different locations (Rob & Coronel, 2004).

2.3.2 Research on distributed database systems

The need to access data from different data sources, a process known as data integration, has advanced in the industry market, reaching key decision makers within enterprises. Data integration "provides a competitive advantage to business" due to "provid[ing] a unified view of the data regardless of differences in data format, data location and access interfaces" (Mattos, 2003). Due to this increase in demand for data integration, application vendors have enhanced their applications accordingly. One possibility of data integration can be by using a DDS (Mattos, 2003).

A DDS is a "logically interrelated collection of shared data (and a description of the data) physically distributed over a computer network" (Connolly & Begg, 2005, p. 689). This means that a DDS resides at each site and each one shares its own data with other sites via a network. Therefore in a distributed database, a user/application is allowed to access data not found at the local site without actually knowing that the data is physically in another location (Shanker, Misra, & A.K., 2008). This is called distribution transparency because "any design complexity" such as the allocation, replication and partitioning of the data is hidden to the user (Martyn, 2000, p. 48). The software which makes this "distribution transparent to the users" is called distributed database management system (DDBMS) (Talwadker, 2003, p. 6).

C.J Date (2003), in his book "An Introduction to Database systems", mentions twelve rules which a DDBMS must possess. C.J Date states that apart from having distributed transparency, a distributed database should be locally autonomous, meaning that all operations of each site should be locally managed. He states that a DDBMS should not rely on a central site and the system must be failure independent, meaning that it is in continuous operation. Another important aspect mentioned is that a DDBMS should possess distributed query processing; that is, a query should be able to reference more than one site. Another point mentioned is that the database integrity should be maintained during distributed transactions. The final requirement considered by Date is that a DDBMS should be able to run on various hardware and operating system platforms using any possible communication network and also made up of different local DBMS. In a nutshell, according to C.J Date, a distributed database should be exactly like a centralized database within the user's perspective.

One has to be careful not to confuse distributed databases with distributed processing. The latter is a system by which a centralized database "can be accessed over a computer network" (Connolly & Begg, 2005, p. 691). A widely-used architecture which uses distributed processing is the client-server architecture which implies that "multiple user[s] access to one data server" (Vokorokos, Balaz, Adam, & Petrik, 2005). The data is stored and processed on one server, located in a single site. However, the data can be accessed and manipulated from different clients over the network. This architecture was used by database vendors to indicate distributed capabilities and used to provide a distributed database, but this architecture "by itself does not constitute a distributed DBMS" (Connolly & Begg, 2005, p. 60).

One also needs to distinguish between a distributed and a parallel DBMS. A parallel DBMS is "implemented on a tightly coupled multiprocessor" (Talwadker, 2003, p. 6). There are three main types of parallel DBMS, namely, shared nothing, shared memory, and shared disk. The shared nothing is where "each processor has its own main memory and disk units" (Talwadker, 2003, p. 6). The shared memory allows that "every processor has access to any memory module or disk unit through a high speed interconnection" (Talwadker, 2003, p. 6). The shared disk architecture derives from the architectures of both shared nothing and shared memory parallel DBMS where the processors “have their own individual main memory space but they share a common global disk storage” (Talwadker, 2003, p. 6).

The main difference between distributed and parallel DBMS is “that distributed DBMSs assume loose interconnection between processors that have their own operating systems and operate independently” (Talwadker, 2003, p. 6). On its part a parallel DBMS uses multi-processor architectures. Because of this, the latter is used to build highly available and highly efficient database servers. Apart from this difference, distributed DBMS and parallel DBMS do share some common elements.

The first similarity is that in a distributed database, each DBMS connects and operates in parallel with other DBMSs found on other sites. Similarly, a parallel DBMS connects in parallel with other processors. The designing techniques for parallel and distributed DBMS are also similar. While in parallel DBMS, “tables are partitioned (fragmented) and allocated to different disk devices,” in distributed database data is partitioned on different sites (Martyn, 2000, p. 49).

2.3.3 Advantages and Disadvantages of Distributed Databases

The main advantage that distributed database systems have over a centralized database system is the ability to distribute the data among several sites. Organizations tend to grow and databases became an important “component of everyday life in modern society” (Shanker, et al., 2008, p. 128). This data distribution allows the organization to segregate the data among all the sites (Dumitriu & Cretu, 2002).

Another advantage of DDS is performance improvement. This is mainly due to two factors. The first is that “local queries and transactions accessing data at a single site are much faster since the local database is smaller”; the second is that “transactions involving different sites can be processed concurrently, reducing execution and response time" (Corcoran & Hale, 1994, p. 248).

A third advantage is that a DDS does not have a single point of failure. If a single site fails it does not result in a complete system failure as is the case with a centralized database system. Thus, DDS has "improved reliability and availability" (Corcoran & Hale, 1994, p. 248).

On the other hand, a major disadvantage of a DDS is its complexity. Its design includes deciding on the fragmentation type, replication strategy, integrity, and concurrent transaction strategies, which are more complex to maintain especially when compared to a centralized database system. Due to its highly intricate complexity, the costs to maintain a DDS are higher. Extra hardware is also needed to build up the network and to facilitate communication between the different sites (Connolly & Begg, 2005).

In addition to its complexity, security is another disadvantage. The DDBMS have to secure the data in all sites as well in the network by which the sites communicate (Dumitriu & Cretu, 2002).

2.3.4 Research on different distributed database architectures and their components

It is useful to distinguish at this point the difference between heterogeneous and homogeneous distributed database systems. A homogenous distributed database system is a "network of two or more databases that reside on one or more machines that uses, locally, the same DBMS product" (Dumitriu & Cretu, 2002, p. 425). On the other hand, in a "heterogeneous system, sites may run different DBMS products, which need not be based on the same underlying data model" (Dumitriu & Cretu, 2002, p. 425).

There exist several architecture implementations that can be used to build a homogeneous or heterogeneous distributed database. An architecture can be described in terms of three different approaches which are based on components, functions and data. Furthermore, several implementations of DDBMS exist based on autonomy, distribution, and heterogeneity (Ozsu & Valduriez, 1999).

A possible DDBMS implementation is the client-server system. As mentioned earlier, the client-server makes use of distributed processing, by which the requests are generated from the client, while the processing and data manipulation is done at the server side. This type of system is tightly integrated and the entire “database is available to any user who wants to share the information” (Ozsu & Valduriez, 1999, p. 83). Different forms of client server implementation exist. One particular form is called multiple client-multiple server. In this sophisticated client-server architecture, a client sends a request to a server which connects to other servers if required.

According to Donald Kossmann (2000), in his paper “The state of the art in distributed query processing”, a client-server system known as the peer to peer architecture is also available. In this type of architecture “every site can act as a server that stores parts of the database and as a client that executes application programs and initiates queries” (Kossmann, 2000, p. 437). Therefore a peer to peer database system (PDBS) “is conceived as a collection of autonomous local repositories which interact (e.g., establish correspondences or exchange queries and updates requests) in a peer to peer style” (Bonifati, P.K., A.M, & K., 2008, p. 5). Therefore this type of architecture is considered as fully distributed when compared to client-server architecture (Ozsu & Valduriez, 1999). Meanwhile the client-server architecture and the PDBS architecture share the same architecture from the data logic perspective (Ozsu & Valduriez, 1999). They are composed of the local schema which is “expressed in the local data model schema” and the export schema which contains the elements which “a peer wants to share with the outside world” (Bonifati, et al., 2008, p. 6).

Another possible architecture is a federated distributed database system (FDBS). It is made up of logically related “DBSs in which operations can be applied to multiple component DBSs in a coordinated manner” (Bonifati, et al., 2008, p. 6). This type of architecture “allows users or global applications to access data stored in multiple local database systems (LDBSs), each of which is autonomously operated” (Sohn & Moon, 2000, p. 687). The federate architecture is composed of four layers. At the bottom we find the local schema. On the local schema there is the component schema “which is possibly a translation of the data model of the local DBS in a canonical model” (Bonifati, et al., 2008, p. 6). This schema contains “those elements of the component schema that the local DBS is willing to share with others” (Bonifati, et al., 2008, p. 6). Then there is also the export schema and the final topmost layer is known as the federated schema which is the “actual global schema that contains information on distribution and allocation of internal exports schema” (Bonifati, et al., 2008, p. 6). A similar but distinct architecture is the multi-database system (MDBS). The key distinction between FDBS and MDBS is their “method of integrating the component DBSs and their assumptions about the autonomy of these components. That is to say, a FDBS “rel[ies] on a single global federated schema” whereas “multiple federated schemas may coexist in MDBSs between the different cooperating component DBSs, allowing thus partial and controlled data sharing” (Bonifati, et al., 2008, p. 6).

As previously stated, architectures can also be described by the components they use. In a distributed database one can find some general components, such as the DDBMS, local DBMS (LDBMS), data communication (DC) component, and global system catalog (GSC) (Connolly & Begg, 2005). The LDBMS is basically the local DBMS which is used to control the local data in a particular site. The DC is the software that enables all the sites to communicate with one another. This usually comes with the DBMS software (Connolly & Begg, 2005). Meanwhile the GSC consists of the global, fragmented and allocation schemas.

Meghini and Thanos (1991), in their paper “The complexity of operations on fragmented relation,” define the global schema as “the definition of the relations constituting the database, as if there were no data distribution” (p.59). Subsequently the fragmentation schema contains the relationships between the fragments, and the allocation schema “gives the allocation of copies of fragments to the sites of the computer network supporting the distributed database“(Meghini & Thanos, 1991, p. 59).

Having the correct architecture for the distributed database is not enough, the design of the actual database is important as well. When developing a centralized system, the designer has to ensure that all the entities are normalized and the correct relationships are setup between the entities. When considering a DDS, the designer not only has to cater for normalization and relationships but also has to consider other aspects.

2.3.5 Research on Distributed Design considerations

The distributed database design takes into consideration three main factors, namely, fragmentation, allocation and replication. These three factors are important not just to make the database transparent to the user but also to augment the performance of the overall system.

Basically, fragmentation is the “division of the database data structures into smaller portions, called fragments” (Meghini & Thanos, 1991, p. 59), where each fragment can be distributed to several distinct sites (Getahun, Tekli, Atnafu, & Chbeir, 2007). The end result is that it enhances the performance of the DDS because fragmentation minimizes the “data transferred and accessed during the execution time,” and above all it “reduc[es] the storage overheads” (Hababech, Ramachandran, & Bowring, 2007, p. 4). Tim Martyn (2000) states that there is “no ideal distributed database” and most DBMS do not provide distributed fragmentation. The database administrator can distribute the data manually, but then the application should be aware how the data is fragmented (Connolly & Begg, 2005) .

There are three fundamental fragmentation strategies possible in a DDS. These are horizontal fragmentation, vertical fragmentation and mixed fragmentation. Horizontal fragmentation “divides data tuples into groups”; vertical fragmentation “decomposes data attributes into groups that are composed of some attributes with high affinity”; and mixed fragmentation is a hybrid fragmentation technique that “mixes the two partitioning methods, i.e., vertical partitioning followed by horizontal partitioning or vice versa” (Son & Kim, 2004, p. 551) .

Allocation is another aspect which concerns the designer when building a distributed database. It refers to “the decisions concerning where to store [the] tables” (Roosta, 2005, p. 1447) . It is important to highlight that data allocation is done after fragmenting the data in different sites (Son & Kim, 2004). It is important that the designer designs the database tables to be located at the sites where it would be mostly used so that it will not have a negative impact on performance.

Another aspect to be considered is replication. As the name implies, replication is the “process in maintaining multiple copies of data items on different locations called replicas” (Plattner & Alonso, 2004). Replication is the “most widely used method for providing high availability, fault-tolerance and good performance in distributed system” (Deris, Abawajy, & Mamat, 2008, p. 1). This is achieved because when changes occur, these are captured by the local database, and sent to the other databases. Therefore this provides the users with fast access to shared data. Since several copies exist, if a site becomes unavailable, the other sites can still continue thus making the system fault tolerant (Deris, et al., 2008). This conforms with Date’s rules. Two types of replication methods exist, namely, synchronous and asynchronous. The difference between these two is that synchronous replication “keep[s] the replicas synchronized within transactional boundaries” whereas for asynchronous replication “replicas can be updated outside the transaction boundaries” (Plattner & Alonso, 2004). In other words, the synchronous method has to wait until the update is successfully executed at all the sites. This brings a communication overhead and has to be limited to a small number of replicas. On the other hand the asynchronous method does not need to wait for all the sites to successfully execute the changes. However, this can bring inconsistencies within the replicated objects (Plattner & Alonso, 2004).

Referential integrity is one of the semantic integrity constraints which, even though it does not need to be designed, still need to be considered. It refers to the relationships by which tables in a schema are linked together. That is, “referential integrity represents the cement that keeps relational databases together,” (Ordonez, Garcia, & Chen, 2007, p. 61) where the underlying checks are done by the local DBMS in a centralized database system. These relationships guarantee the consistency and integrity of the data within the database. When considering distributed databases, distributed referential integrity is usually complicated to maintain between tables in remote databases. Triggers can be used to check these constraints but this will depend on the availability of the remote database and usually these are not provided with the DBMS solution. Another solution to this problem is that the referenced tables can be replicated to the sites so that the local DBMS is responsible for the integrity of the data. For this solution to be valid, three assumptions have to be considered. First, the replication cost does not downgrade the performance; secondly all the local databases have to possess identical integrity constraints; and third by each row should be uniquely identified within all the databases (Dye, 1999). A valid suggestion that Ordonez, et al. (2007) mentions is that “unique identifiers can be generated by concatenating the site identifier to avoid primary key duplication” (p.62).

2.3.6 Research on Concurrency

Transactions form the "interface contract" or commands between the application and the database itself (Hanssen, 2003). This transaction mechanism provides an efficient way to manage the access to the database. In order to have valid transactions the so-called ACID properties should all be present. The acronym ACID stands for:

- Atomicity - transactions execute completely or not at all
- Consistency - transactions are a correct transformation of the state
- Isolation – transaction consistency; i.e. a transaction does not view a partial change of another transaction
- Durability – committed transactions survive failure (Felber, Fetzer, Guerraoui, & Harris, 2008)

Transactions occurring at the same time, that is concurrently, are more probable to be found in a distributed database. The isolation portion of the ACID properties is fundamental in these cases so that there is consistency between the transactions. But how is this managed within a DB?

In a DB, the DBMS executes concurrency control which is the "activity of coordinating the actions of transactions that operate in parallel; access[es] shared data, and potentially interferes with one another" (Akintola, Aderounmu, Osakwe, & Adigun, 2005, p. 365) . In order to control concurrency, the system uses serializability which, as the name implies, is when transactions are executed serially, and therefore considered correct (Hanssen, 2003). Unfortunately in a DDS it is more complex to serialize the transaction when compared to a centralized database system. As a matter of fact, the DDBMS, has to "extend both the serializability argument and the concurrency control algorithms to the distributed environment" (Akintola, et al., 2005, p. 368).

There are three main methods which provide serializability and concurrency (Vasileva, Milev, & Stoyanov, 2007):

- Locking
- Time stamping ordering
- Optimistic

The locking mechanism is where transactions "lock everything you access and hold all locks until commit" (Hanssen, 2003). On the other hand, time stamping ordering mechanism does not use any locks. In this mechanism the transaction is given a unique timestamp by which these transactions are serialized. In cases of conflict during the transaction, the lowest timestamp is given priority (Hanssen, 2003). Finally, optimistic mechanism is when it "synchronize[s] concurrent execution of transactions early in their execution life cycle" (Hanssen, 2003).

2.3.7 DBMS Systems

Several DBMS solutions from various vendors exist on the market. Three of these vendors, which share two thirds of the market in regard to databases systems, are Oracle, Microsoft SQL Server and IBM DB2 Universal Database (Stamford, 2007). In this dissertation only the solutions from these three main DBMS vendors were evaluated which are Oracle 11g, Microsoft SQL server 2005 and IBM DB2 version 9.5. These three vendors offer different versions or editions of their solution which can cater for different businesses' needs. For the present study, the editions chosen are those which meet the needs of a small business. In order to carry out this evaluation, the following criteria were used:

- Fragmentation Transparency – where the application or the user is not aware how the data is partitioned or fragmented
- Location Transparency – the application or the user is not aware of the physical location of the database
- Distributed Transaction Transparency – ensures that all the transactions follow the integrity and consistency of the database by concurrency and failure recovery
- Replication Transparency – the application or the user are not aware of the replication of the data
- Support Heterogeneous System –the ability to connect with other databases which have different DBMS product
- Support Homogeneous Systems- the ability to connect with other databases which have the same DBMS product
- Operating System – the platform upon which the DBMS is capable of operating
- Cost – the cost of the solution

DBMS Function	Oracle 11g	Microsoft SQL Server 2005	IBM DB2 9.5 Universal Database
Fragmentation Transparency	No	No	No
Location Transparency	Yes	Yes	Yes

Distributed Transaction Transparency	Yes	Yes (MSDTC)	Yes (DB2 TM)
Replication Transparency	Yes Oracle Streams (Synchronous Capture only), Materialized views	Yes	Yes
Support Heterogeneous	Yes	Yes	Yes (Only by federated database)
Support Homogeneous	Yes (Database Links)	Yes (Linked Servers)	Yes
Operating System Requirements	Windows, Linux, or UNIX	Windows Server 2003, Windows Vista	Windows, Linux, or UNIX
Product Solution	Oracle Database 11g Standard Edition One	SQL Server 2005 Standard Edition	DB2 Version 9 for Linux, UNIX, and Windows
Cost	$180 per user (5 user minimum) or $5,800 per processor	Processor license $5,999 or $1,849 for 5 clients.	$5,175 per processor (100 PVU) or $175 per user

			for DB2 9 Express.

Table 1: DBMS Comparison

As one can see from Table 1, fragmentation transparency is not supported by the three DBMS solutions. These solutions do not provide the possibility for the DBMS to distribute the fragments. Therefore this distribution has to be included in the application logic. On the other hand, this does not mean that the data cannot be distributed. As a matter of fact these DBMS solutions allow the distribution of the data and also provide location transparency for the data. This is achieved by the use of synonyms and views in case of Oracle and SQL Server. Meanwhile, for DB2 it uses nicknames, which are similar to synonyms.

Another aspect evaluated was distributed transaction transparency. This type of transparency is also provided by the three solutions by using the two phase-commit mechanism. It is important to note that an extra component has to be installed and configured in case of SQL Server and DB2, that is, Microsoft distributed Transaction Coordinator (MS DTC) and DB2 Transaction Manager (TM) respectively.

In regard to replication transparency, these solutions have similar features. As in the case of Oracle Standard Edition One, Oracle Streams and materialized views are the features concerning replication which are provided. It is important to note that for this edition only synchronous capture is provided for Oracle Streams, and single master replication in the case of materialized views. SQL Server 2005 standard edition provides three types of replication which are Transactional, Merge and Snapshot replication. Meanwhile, for DB2 version 9 Express, replication is provided by SQL Replication and Q Replication. It is important to mention that for heterogeneous, Q replication, which uses WebSphere® MQ message, has to be used. Since this is an additional product, this will require an extra cost.

All the compared solutions provide connectivity with homogeneous database systems. Oracle makes use of database links while SQL Server uses linked server links to connect with other databases. On the other hand, DB2 uses IBM Homogeneous Federation Feature. This feature makes use of wrappers by which the user is able to access data in other DB2 databases.

In case of heterogeneous database connectivity, generic connectivity such as ODBC is provided by all the systems. It is important to note that Oracle provides a larger number of possible data sources when compared to SQL Server. Also when compared to DB2, it supports heterogeneity by the use of WebSphere Federation Server product. This is an additional product and thus incurs an extra cost. Regarding software requirements, Oracle and DB2 is available for standard operating systems (OS), namely, Windows, UNIX and Linux. On the contrary, SQL Server is only available for Windows OS.

Basically these three solutions provide almost the same functionality with regard to data integration and distribution. Also, the cost, which is the most determining factor, is approximately the same. Even with these similarities, there exist some differences between them. As a matter of fact, SQL Server 2005 and DB2, need an external component in order to have distributed transaction transparency. In both cases this external component has to be installed and configured. In this case Oracle can be considered as a complete solution because the components which ensure that the transactions follow a two-phase commit mechanism are part of the database itself.

Another difference highlighted is that DB2 needs an additional product in order to be able to cater for heterogeneity. This will obviously incur an extra license and therefore an increase in the total cost. In this project, homogeneous databases are preferred but the possibility to connect with heterogeneous systems can be beneficial in the future. Oracle and

SQL Server both provide connectivity to heterogeneous systems but Oracle provides a larger number of possible connections. Due to this DB2 was discarded.

Finally, interoperability, the ability to work on different OS, is another deciding factor. As seen from the table above, SQL Server solution is only available on Windows OS. So far the preferred OS which was to be used and installed on the "outlet" computers was basically Windows. A possible scenario is that in the future there may be the need to use different OS. In such a case SQL Server will prevent the user from having a homogeneous system. SQL Server was discarded, due to its lack of interoperability and due to the extra component needed for distributed transaction transparency. Therefore Oracle was chosen as the DBMS solution to be used in this project.

2.4 Summary

This project provides further knowledge on distributed databases. It also provides a solution for a small business which has several outlets and which would like to change from the manual gathering of data to a computerized system. It makes use of distributed databases, where data is gathered from different outlets, and is shared among all outlets.

Chapter 3 – Methodology

3.1 Life Cycle Model

This project mainly involves the creation of a database for a particular business. Since a database is considered part of an information system, the model used in this instance is the Database Life Cycle Model. Also known as DBLC, this is similar to the Software Development Life Cycle (SDLC) model. This is also referred to as the "Waterfall method".

The DBLC is constructed through five main stages, namely planning, analyzing, designing, implementing, and maintaining. The present researcher chose DBLC as the model for this project as each stage has first to be completed in order to proceed to the next stage. A practical example of this procedure is that the design stage could not start unless the requirements in the analysis stage were identified.

In the course of this project, the five stages were not followed in a strict sequence. Since the project had to be amended more than once, stages of the DBLC had to be reconsidered, and therefore repeated. Consequently, some of the stages followed an iterative process. The following is a detailed description of the process in each and every stage.

3.2 Specific procedures

3.2.1 Planning Stage

The planning stage began with the author identifying the deliverables for this project. The following list shows all the deliverables identified.

A solution which:

- Stores the gathered data of the various transactions. This solution has to remove data redundancy, and inconsistency. Furthermore, this solution has to provide a possibility of generating reports. This solution also has to be extended to the three different stores, and also provide the possiblity of communication with each other

- Tracks rent transactions, selling transactions, and booking transactions for the different stores

- Reports (determined in the analysis stage and implementation stage)

Primarily, the objectives for this business in using this database are to provide the customer with better service, easier and better store keeping, stock taking, and retrieval of information. These objectives have to be implemented and integrated with the other three outlets. After the objectives have been identified, the step that followed was that of observing, and analysing of a movie rental shop already in operation. The following is a detailed description of this process.

3.2.2 Analysis Stage

The analysis step began by analyzing a movie rental store in operation. The researcher noticed that movie rental stores provided three main services, namely sales, rentals, and booking of movies. The researcher noticed that the stores under analysis used a very crude method to keep records of their transaction done, be it a sale, rental or booking of movies. These stores kept records in paper form.

The author observed that every time a movie sale transaction was perfomed, obviously the customer chose the movie he wanted to buy, then the store attendant checked the price list of the movie, and requested the money. After the customer paid for the movie, the store attendant logged this sale in the sales sheet. Basically, this was a sheet of paper by which the store management could keep track of all the sales, and where the movie name and the cost were both recorded. Furthermore, this sales sheet was also used by the store keeper to see what stock needed to be reordered.

On the other hand, when a movie rental transaction was done, after the cutomer chose the movie, the store attendant checked if the customer was registered or not. This was done by retrieving the customer name in the file containing the records of the customers. For each customer there was a record, which was basically a sheet of paper containing the customer information, and the customer’s rental history. Therefore it was in this sheet where the rental details were logged in. This sheet was also used when the customer returned the movie as well as recording payment for the rental.

In the store under analysis there were customers who requested the service of booking DVDs (movies) which were already out on loan on the date of booking. This service was available but bookings were recorded manually on paper sheet forms.

After observing and analyzing the current methods used by the store in question, it was clear that the recording of the transactions which were done manually was very time consuming, and was eventually leading to long queues of customers waiting to be served. Apart from that, the data logged in manually by the store attendant could be easily duplicated by mistake, therefore containing inconsistencies, and so not very reliable. This record keeping system also made data retrieval difficult to perform, and it took a considerable amount of time.

After the analysis was conducted, the researcher concluded that the opening (or extension) of two new outlets could not afford to work with a manual record keeping system. The amount and range of services to be available at the multiple outlets would not make it possible for them to communicate, and keep each other updated when there were various transactions being done at the same time. Apart from that, more data had to be collected when considering the increased number of stores.

Therefore the main requirement was that of a storage location which held consistent data, and also was accessible in a timely manner. This requirement applied also to the different outlets. This means that each outlet would have its own data, but at the same time be able to connect, and integrate the different data in the other outlets.

Another requirement, on account of the different locations of the outlets, was that of the mobility of the rentable movies and of the employees themselves if required. In other words, the employee should be able to log in, and work in any of the outlets. Similarly, the customer should have the possibility to return the rented movie to any preferred outlet.

The second stage of the analysis was the identification of the key users involved in this store. Three main users were identified, namely the store attendant, the store keeper and the manager. For each user, requirements and types of reports were identified.

The store attendant, as the name implies, is the person who is assigned to serve customers. The main duty of this personnel is to input the data regarding the lease and selling of movies, that is all the daily transactions. In other words, it is this user who will populate the database with the appropriate data. The main requirement of the store attendant was to have available consistent data from all the outlets. Also, some report requirements were identified, which are as follows:

- A report which shows the names of the customers who have not yet returned the leased movies after the allocated period
- A report showing a record of all the movies booked by the respective customer
- A report or a list within the booking application that shows the available movies in that particular store

The second user identified was the store keeper. During the analysis it was considered that the store keeper's main role was to keep track of all the movies in the store. One of his responsibilities was to check the stock amounts and then decide accordingly if new stock was needed or not. The store keeper was also responsible for notifying other outlets about rentable movies which were returned to an outlet other than the one from where the movie was originally rented. The main requirement of this user was the facility to get the information needed for his duties in a short amount of time. The information needed was to be found in the reports identified and listed below:

- A report which shows the stock quantity in that particular store

- A report which shows the movies rented by that particular store but which were returned to a different store. With this report the store keeper can notify the other stores to send back these movies in order to be inline.

Finally, the main user who oversees the whole situation is the manager himself. Obviously the manager has to safeguard the investment of which he or she is in charge by having a clear idea of the finances of the stores so that he or she can take informed decisions. This can be done by the provision of correct and consistent reports. These are:

- A report containing the daily profit from sales transactions in all the outlets
- A report containing the daily profit from rented movies in all the outlets
- A report which shows the rented movies history in all the outlets
- A report which contains all the sold movies in a specified range of time in all the stores
- A report of the number of transactions done by each employee

3.2.3 Design Stage

The first part of the design stage was to gather all the required data elements that the system was going to use, and eventually store in the database. Four main groups were identified. These contained data elements used during the transaction of the services provided by the store, namely selling, renting and booking. One of these groups, which can be classified as that of the employee, contains data fields concerning the employee, such as the name, surname and address. Another group is that of the customer. As the name implies, it holds data related to the customer such as the name and address. Another group is the one which holds data about rentable movies. In this group data fields such as the name, actors, publishing house and movie category can be found. The final group that was identified which holds information about movies available for sale. The data in this group is similar to that found in the rentable movie group. By identifying these groups, the data required was easily identified.

Other data groups were recognized which holding data for the actual transactions during the daily services of the stores. These groups obviously have data fields such as the date of the transaction as well as the movies which were rented, sold or booked for each transaction.

The next step was to identify the business rules which can help recognize further data elements. Business rules are a narrative description of the operations which in this case are the renting, booking and selling of movies in these stores. The following list contains some of the business rules identified:

1. A customer can rent several movies at a time
2. A customer pays for the lease of the movies upon return of the movies
3. Each movie has a predefined rentable period
4. A customer has to be registered in order to be able to rent and book movies
5. A customer can return rentable movies at a different store from the one where he or she rents them
6. An employee can work in different store locations
7. A customer can book a movie at a particular store where the requested movie resides
8. Each movie has to be uniquely identified

The next step in the design stage was the creation of the actual entity relationship diagram. Obviously, entities for the employee, customer, rentable and saleable movies were created which contains the data fields identified in the previous steps. These entities would be the building components of the whole system since these entities would be referenced during the daily transactions of the stores. These entities were named employee, customer, movie rent, and movie sell respectively. When considering a database, it is beneficial to have a field which represents the physical location of the actual entity where the data resides.

This field provides distinctiveness in the records of all the databases, which will later facilitate the retrieval of the data itself. Therefore, each employee, customer, rentable and saleable movie are registered to a particular store, and this additional field will contain a reference to the location of the store where the registration took place. Additionally, this field will have to reference another entity, which was also created, and named site. It will hold the data regarding each site or store in the system. The use of this field will be combined with the primary key of the entity, in order to provide the unique identification of a movie, employee and customers in all the outlets. In the case of both movies for rent and for sale the concatenation of the movie ID (Primary key) and the Site ID, will result in a string of characters which can be printed as a barcode and attached to each movie. This will facilitate the inputting of the movie during the daily transactions just by scanning this barcode. In a business like this it is a must to have an identification of the business's assets.

In regard to the daily transactions, two entities were added for each type of transaction, that is, sales, renting and booking. One entity holds the information of the actual transaction and the other one holds the items for each transaction. The entities which hold the transaction data were named sell_transaction, rent_transaction, and book_movie respectively, while in the case of the entity which hold the items these were named sell_item, rent_item and book_item respectively. An important point to mention is that all these entities were referenced to the Site ID. As already stated this would differentiate between various transactions conducted in all three stores.

The next step was to normalize the entity diagram constructed so far. During the third stage of normalization further entities and transitive dependencies were discovered and created. One of the entities was the grade entity which contains the different grades in the store. The possible grades identified so far were, the manager, store keeper and store attendant. One could also find the salary according to the grade in this entity. Later on, each employee record was referenced to a record in this grade entity by adding the grade entity primary key. The other entity discovered was the stock entity which contains the stock types available in the store. This entity corresponds to attributes such as the selling price.

Subsequently for each user the privileges were identified for each table. At this point one should mention that in a database system, the possibilities of a user are that of selecting, inserting, updating and deleting records in the tables. According to the grade, these possibilities were identified.

After that, some of the constraints were identified in the creation of the actual databases scripts. An example of these constraints is the employment status for an employee, which can be only active or terminated. The movie type is another constraint. There are only three possibilities which are DVD, BLU-RAY and VCR. These types of constraints are important and will help to have consistent data.

Due to the requirements that each employee can work in all the stores and that the customer is free to return a movie in whichever store he/she likes brought up a critical decision during this stage of the design phase. The decision that had to be taken, regarded referential integrity. It was arguable whether to handle referential integrity by the application or by the database itself. Referential integrity is when a key in an entity is used to reference another entity. In a non-distributed database, the data have to be present in the actual database in order to be referenced. On the other hand, for a distributed database, the data does not need to be exclusively in that store's database but can be found in another database, in another store. If this has to be handled by the application, it has to check whether the key is referenced in another database located in one of the stores. In this case the application has to use distributed SQL queries, where it checks the other databases at the other store. Unfortunately this type of checking will bring with it complex logic in the application. Furthermore, the application also has to be able to detect cases where the entity is being changed or deleted by another user in order to have consistent data. Unfortunately, with the current DBMS solutions, the database itself does not handle distributed referential integrity. Fortunately, there is another solution provided by the DBMS vendors, that of replication. Replication, as the name implies, copies the data or entities to the other stores' database. Therefore the DBMS of the local store can reference to the data from the other stores more easily since it is found in the actual database. An additional advantage which replication brings is that of having a backup of the replicated entities in the other databases. Therefore, in case of one of the databases' failure, the system can still continue running. For these reasons replication was chosen for this business solution. For the requirements identified there was no need to replicate the whole schema but only a few numbers of entities.

Returning to the requirements, as regards the possibility of employees working in different stores, the employee entity was chosen to be replicated between the database sites. Therefore, the attributes found in the employee entity were found in all the databases. On the other hand, for the other requirement which allows the customer to return a movie to any store, four entities were chosen to be replicated. Obviously the customer entity was replicated since customer data was registered in the other outlets. Also, the rent_transaction and the rent_item entities were replicated, because these contain the information about the transaction. Finally the movie_rent table which contains the rentable movies was replicated due to its referral to the transactions.

The next decision to be taken was how the application was to achieve data retrieval. In other words, whether the application was going to query the database by a SQL query or just by calling a procedure. In the case where the application queried the database itself, the SQL query had to be hard-coded in the application software. This could bring problems if there were to be some changes to the database schema. Therefore these changes also had to take effect within the application code. On the other hand, if procedures were used, it would be more secure since the database checks for the user's rights. In this case only the required data is sent to the network and changes to the database have to affect only the procedures, and not the application as well. This makes the system more standardized. As one can see, the benefits of using procedures are many so it was decided to use procedures in this system.

For the procedure to be able to retrieve the data, two views were designed. One view, named sell_transaction_view, is used to gather all the information of the sales transactions from all the stores. The other view was named rent_transaction_view which, as the name implies, gathers all the data relevant to rent transactions. Afterwards, the procedures were designed to cater for the input parameters and the output data was identified.

This completed the conceptual design of the database. The next step was to choose the DBMS vendor. The researcher decided to use Oracle. Apart from the fact that Oracle is one of the most used DBMS and considered as market leader, it also provides replication functionality with the Standard One Edition. This functionality is called Oracle Streams.

The final stage of the design phase was the physical design of the actual database. This stage determines the physical storage location of the database. It was decided that the whole schema, that is, all the entities or tables mentioned above were allocated to a single tablespace which was named after the movie store. On the other hand, the procedures related to the replication functionality were to be allocated to a different tablespace.

3.2.4 Implementation and Loading

The first stage of the implementation was to install the DBMS, namely Oracle, on three computers which were to represent the three “outlet” computers. These were connected to each other by a network connection through a router. Also each computer was assigned a static IP.

The second stage was to build the actual scripts which created the database schema containing all the tables with their respective fields. All these entities were then attributed to the same tablespace. Furthermore, the links between the databases, which in the case of Oracle are called database links, were also included in this script so that the database itself could connect to the other databases.

The users mentioned in the design were created and granted the privileges for the entities. Also, synonyms were created to refer to a specific entity in a particular schema. This was beneficial in querying the data from the database since the specifying of the schema owner and the entity would not be required.

As regards to the replication functionality, the corresponding script was also created. This script created a capture queue which captured changes on the replicated tables found on the other two databases. It also created two apply queues which enabled the transfer of changes on a table of a particular database to the other databases.

The next stage was to execute the scripts on all the three computers and to load the databases with dummy data in the appropriate entity fields. The last stage was to create all the procedures mentioned in the analysis phase.

3.2.5 Testing

The testing of the system was performed by executing the procedures and verifying that the data retrieved was correct. Also consideration was taken for the time required for the data retrieval. In cases where the period of time was long, the SQL query was edited and sometimes the database was fine tuned. This was done to improve the performance of the system.

Although the DBLC's last stage regarded the maintenance of this database, discussions could not be held at that time due to the fact that the database was not implemented in a movie rental store yet. However, upon implementation this study will dedicate a discussion on how to maintain upcoming events.

3.3 Project Deliverables

The two main deliverables were the Oracle distributed database solution for the three "outlets", including the replication solution, and the database procedures which retrieve the data requested in the reports.

3.4 System Resources Requirements

The system requirements for Oracle Standard One edition are: 1GB physical memory (RAM), 5GB disk space, 256 video adapter, 550MHz processor and (on Windows Vista) 800Mhz processor minimum. The platform was decided to be Windows and this software edition requires that the operating system should be Windows 2000 or later.

The computers used all met the above requirements; two of the computers had 1 GB of RAM and ran on Windows XP Professional. The other computer had 2GB RAM and ran on Windows Vista Home Premium Edition.

3.5 Summary

The database life cycle worked well for this project. All stages of this method were closely followed. At the implementation stage, some changes and updates were implemented by the author himself. Therefore this stage turned to be a more of an iterative approach. If the application for this project had been created and implemented in a real store, new requirements would have been requested from the users. This would result in a continuation of the iterative approach bringing changes to the database solution.

Chapter 4 – Project History

4.1 How the project began.

The project began when the author noticed that some establishments which rent movies in Malta still use a paper logging system for their daily operations. These stores lack a computerized system, and could widely benefit especially from the use of a distributed database due to the fact that it stores the data more efficiently. The author decided to develop a database solution which could be used in such stores, and also manages the distribution of the data in multiple outlets.

A database system not only provides an efficient way of storing the data used in the daily operation of the store, but it also addresses other concerns. One of these concerns is the time factor. More specifically, this refers to the time required to gather the data. When using a manual system the time taken is enormous when compared to the time taken to retrieve data using a database solution. Another concern is the consistency of the data, since by correct design this can be guaranteed by the database itself.

In addition, a distributed database addresses the functionality of sharing the data found in multiple outlets. The traceability of the daily operation within the different outlets is achieved by this capability. Furthermore, several reports are needed by the different users so as to ease their daily working routine. In such case, distributed databases helps to gather all the data needed by the reports from all the outlets. Also, from the management perspective, a clearer indication of the profits and the sales trends are visible in such reports. Most importantly, the services provided are much faster and accurate, thus increasing customer satisfaction.

4.2 How the project was managed.

The project followed the different stages from planning to the implementation of the database. Unfortunately, the deployment and the maintenance stages of this project were not completed in a real movie rental store. Due to this the author was not tied down by any deadlines or changes in the requirements. If these stages had been completed, the author could have easily faced “scope creep”. Scope creep is when further requirements are added after the system specifications are completed (Russell, p. 1). This latent development could be attributed mainly to the method used to gather requirements. The author used the observation method, which involves the observation of the normal daily operation within some rental stores. The main drawback of this method is that only a partial picture of the complete behavior is seen. That is, certain anomalies or special events could not be observed and therefore these requirements were neglected. For this reason, additional requirements which were unforeseen by the author would have had to be added at a later stage.

Furthermore, at the design stage, the timeline was not met due to the lack of knowledge of Oracle Streams. Several documents had to be reviewed in order to construct scripts which involve the replication between the outlets.

4.3 Significant events/ milestones in the project

The main milestone of this project was to develop a database solution by the completion of various stages of the project life cycle. All the stages were completed, but not according to the time schedule. This can mainly be attributed due to the time required for the creation of the replication scripts was underestimated, therefore the design and implementation stages took a little bit longer to complete. Unfortunately, the stages of the implementation, and maintenance of the system in an existing movie rental store have been discarded due the fact that no rental store requested this solution.

4.4 Changes to the project plan

The overall project plan was not altered except for some changes in the design. Extra elements were added in some entities as the author realized that such fields would be beneficial if collected. Also, the introduction of the timestamp for the replicated entities was added as a requirement of the replication mechanism. Other than the minor modifications, the project plan remained the same.

4.5 Evaluation of whether project met goals

The main goal of this project was to build a distributed database system which serves as a data repository for a movie rental outlet, and also to share the data among the other two outlets. This particular goal of the project was met. Another goal, which was achieved, was the creation of several reports used by the different users within a rental store. The goal to populate the system with dummy data, and test both the complete database solution, and the reports was also achieved. However, the goal of improving the customer services could not be evaluated since this project was not deployed in an existing establishment.

4.6 Discussion of what went right and wrong with the project

Several things went well for the project. First of all, a movie rental store can have a useable database solution which stores the data used within its outlets efficiently. Reports based upon the data gathered were constructed. These reports contain information about the transactions performed in the outlet. Furthermore, this system is able to communicate, and share data found in the database systems at the other outlets. The information collected from these outlets can be shown in reports, and thus a better perspective of the situation in these stores is presented. Additionally, the tracking of the movies is recorded more easily. This type of system obviously is more convenient than using the manual logging system.

One problem encountered in this project is that the implementation stage took longer than planned. This was mainly due to the extra time taken in creating the replication scripts. This can be attributed to the lack of knowledge regarding Oracle Streams, and therefore several documents had to be reviewed. Also, the introduction of replication brought some changes to the schema by introducing new fields within the replicated entities. These schema changes resulted also in changes in the data loading scripts as the data of the extra fields had to be added manually.

Another problem encountered by the author was the method used to gather the store's requirements. The observation method only shows a snapshot of the daily operation of such stores. Therefore, some exceptions in daily routine can be missed which would necessitate changes of the project at a later stage. Unfortunately, the deployment of this project in a real working environment was not implemented, and therefore any such exceptions are still hidden.

4.7 Discussion of project variables and their impact on the project

This project was not affected by any external variables, and the development of the project itself continued smoothly. This can be mainly attributed to the fact that the deployment of this project in a real rental store had not yet taken place. Upon deployment, a discussion about the variables encountered can be done.

Another variable which may impact the project is the lack of an application. If an application was built to facilitate the population of the database, and the generation of the reports, several issues could be encountered such as missing attributes, and different data in the reports. Most importantly, the performance of the database could be evaluated more rigorously. The introduction of indexes and de-normalization of the database could be done in order to achieve a better performance.

4.8 Analysis of results

In the planning phase, three deliverables were mentioned. The first deliverable was a solution that saves gathered data from different outlets. This deliverable was met by the creation, and use of a database system. Apart from that, the database is also responsible for reducing data inconsistency, and redundancy.

The second deliverable was to track the daily transactions, namely renting, booking and selling from all the outlets. This deliverable was also met. All the transactions are captured, and stored in the respective outlet database. In the case of booking, and selling transactions, these are replicated throughout the three stores in order to cater for movie mobility.

The last deliverable is the creation of reports used by the different users in the movie rental store. This deliverable was tested by correlating the data found in the report with the actual data in the database. These tests were successful, and therefore this deliverable was considered met. An important point to note is that the generation of the reports lies in the database solution itself. This is due to the fact that for each report a procedure is used which queries the database and gathers the required data. An important advantage of using this procedure is that when changes need to be done to the reports only the procedures need to be changed. The time to generate such reports was also considered but it was only evaluated on the data loaded. This constituted only a small number of records when compared to the number of records which can be found in a real establishment.

All these three deliverables can be summed up in better service to the customers, as well as easier, and better data logging for the employees. Unfortunately, these cannot be assessed due to the fact that this solution is not yet implemented in an existing store. Following the implementation, a discussion of the outcomes, and the evaluation of the results are to be compiled.

4.9 Summary of results

The solution that was built satisfies the needs of a movie rental store with different outlets. Apart from storing the data concerning the daily operation, it can also be used for tracking movies among the stores. The design and creation of such a database system was successful, but unfortunately it could not be evaluated on a real store.

Chapter 5 – Conclusions

5.1 Lesson learned

The lessons learned from this project include:

- The importance of:
  - Obtaining clear requirements from the users

  - The benefits gained from the evaluation of the project when deployed in a real establishment.

- Development of skills in the area of:
  - Database development
  - Sharing and replicating data between databases
  - Processing of the data to build valuable reports
- Increased knowledge in:
  - SQL coding
  - Oracle DBMS
  - Oracle Distributed Capabilities
  - Oracle Streams

5.2 What would you have done differently

An aspect which the author would have done differently is the observation method used to gather the requirements. Interviewing could be another valid method which could be used to gather requirements from the different users. By interviewing, one gets a better understanding of what the user wants or would need. Nevertheless, the observation method can be used in conjunction with interviews, so that the designer gets a better perspective of the daily routine. Clear requirements are important and sometimes they determine the success or failure of a project.

5.3 Initial project expectations met

The initial project expectations were met. The store outlets can efficiently and consistently store the data gathered in its daily operations. This is achieved through the creation and use of a database, which contains different entities, and also holds the gathered data. Apart from that, valuable reports which can ease the work of the employees and the management can be generated from such a system in a short period of time. The solution also comprises the ability to connect and share the data found in different outlets' databases. Thanks to this capability, the tracking of the renting, booking and selling transactions for the different outlets as well as the location tracking of the rentable movies is possible.

5.4 Next stage of evolution

This project, however, is not a complete solution, and therefore several follow up stages are required. The first stage would be to build an application which works on top of the database systems. This application will help the users to insert, delete, and modify the data which resides in the database more easily, as well as, it enforces the business rules. Such an application can also be web based, which could benefit the customers by providing the possibility of online booking and buying of movies from these outlets.

The deployment of the database solution and the application in a real movie store is the next step. In conjunction with the deployment, a mechanism to populate the database has to be chosen. A migration strategy has to be decided on how the users are going to change from the old system to this new database system created in this project.

Afterwards, a tough evaluation of the database solution can take place based on criteria such as performance, report generation, data storage and, most importantly, customer service.

5.5 Conclusions / recommendations

This distributed database solution provides a movie rental establishment with the required data storage facility for the information gathered in its daily operation. Reports based on the stored data are generated, and different users working in such store can benefit from its contents. All this was achieved by using a commercial DBMS, namely, Oracle DBMS. Though the edition of the DBMS used was designed for small businesses, it still incurred a cost. If the business has a limited budget, the author recommends that other DBMSs which do not sustain any license fees should be used instead. The schema developed in this project can still be used, but the scripts which create the database have to be adapted to the new DBMS.

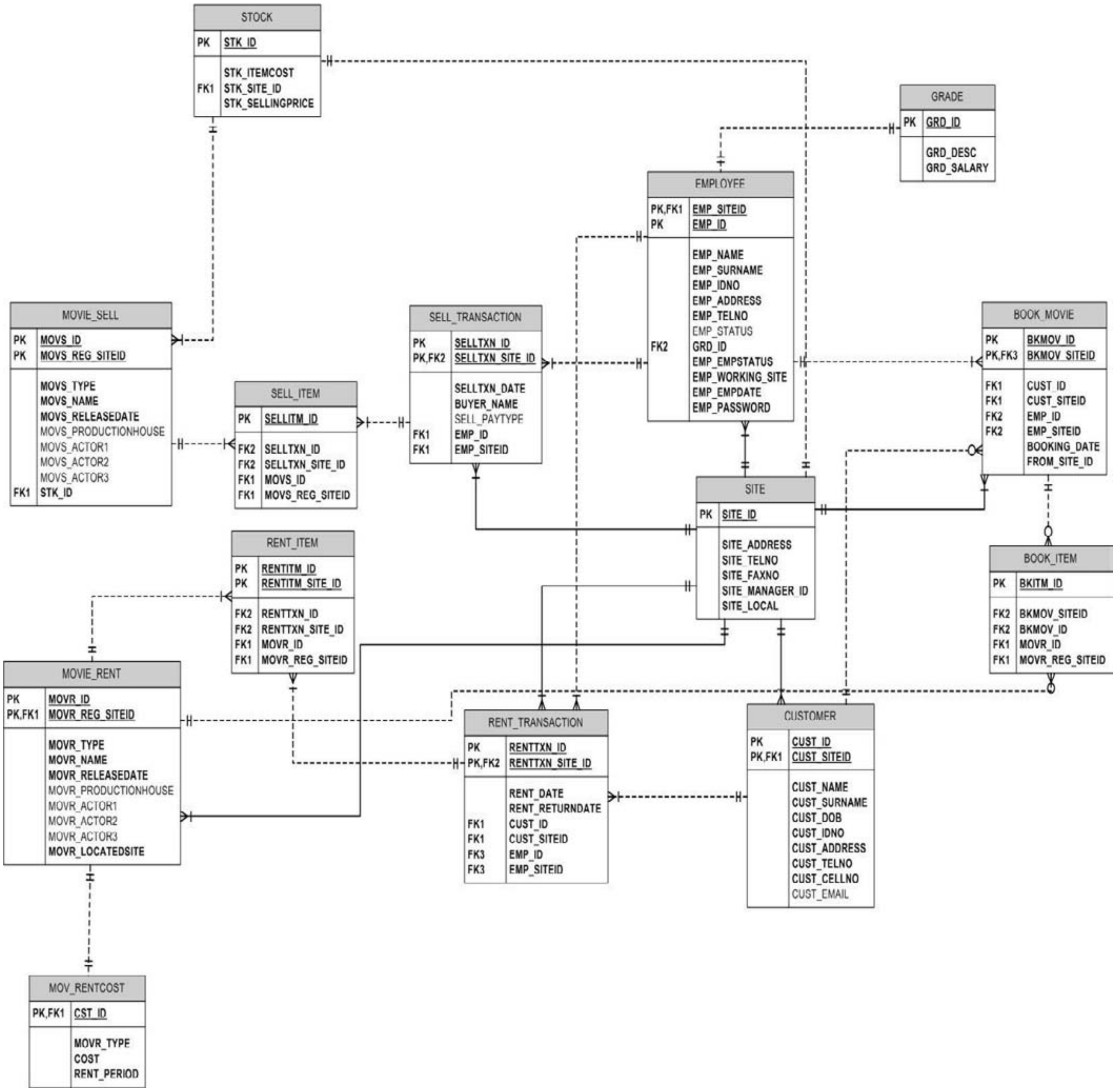
5.6 Summary of project

This project consisted of building and implementing a database system using Oracle DBMS to meet the needs of movie store. It also takes care of the different outlets of the store in question. Such stores can benefit from efficient and consistent storage of the data from their different outlets. This project also met the need of correct reports for the different users within the different outlets. The tracking of the daily transactions and the location of the movies is visible through these reports.

The project gives such stores the benefits of storing and manipulating the data in an efficient and consistent manner and also to extend these functions by sharing and accessing data located in different outlets.

Appendices

Appendix A: ERD Diagram



Appendix B: User privileges

Table User	Manager	Store Keeper	Attendant
MOV_RENTCOST	SELECT, UPDATE, INSERT, DELETE	NONE	SELECT
SELL_ITEM	SELECT, UPDATE, INSERT, DELETE	NONE	SELECT, UPDATE, INSERT
RENT_ITEM	SELECT, UPDATE, INSERT, DELETE	NONE	SELECT, UPDATE, INSERT
RENT_TRANSACTION	SELECT, UPDATE, INSERT, DELETE	NONE	SELECT, UPDATE, INSERT
SELL_TRANSACTION	SELECT, UPDATE, INSERT, DELETE	SELCET	SELECT, UPDATE, INSERT
BOOK_ITEM	SELECT, UPDATE, INSERT, DELETE	NONE	SELECT, UPDATE, INSERT
BOOK_MOVIE	SELECT, UPDATE, INSERT, DELETE	NONE	SELECT, UPDATE, INSERT
MOVIE_SELL	SELECT, UPDATE, INSERT, DELETE	SELECT, UPDATE, INSERT, DELETE	SELECT
MOVIE_RENT	SELECT, UPDATE, INSERT, DELETE	SELECT, UPDATE, INSERT, DELETE	SELECT
EMPLOYEE	SELECT, UPDATE, INSERT, DELETE	SELECT	SELECT
CUSTOMER	SELECT, UPDATE, INSERT, DELETE	NONE	SELECT, UPDATE, INSERT, DELETE
GRADE	SELECT, UPDATE, INSERT, DELETE	NONE	NONE
STOCK	SELECT, UPDATE, INSERT, DELETE	SELECT, UPDATE, INSERT, DELETE	SELECT, UPDATE, INSERT, DELETE
SITE	SELECT, UPDATE, INSERT, DELETE	SELECT	SELECT

Appendix C: Views

```
/********************************Sell Transaction View************************************/

create or replace view sell_transaction_view  as

/*********************************Query data from Site 1***********************************/

select * from (

        (select a.selltxn_id, a.selltxn_date, b.movs_id,b.movs_reg_siteid, c.MOVS_NAME,

         a.selltxn_site_id,d.stk_sellingprice,d.stk_itemcost, e.emp_id, e.emp_name,e.emp_siteid,

         e.emp_working_site

                from  sell_transaction@site1.site1 a, sell_item@site1.site1 b, movie_sell@site1.site1 c,
                stock@site1.site1d, employee@site1.site1 e

                where b.selltxn_id = a.selltxn_id and   b.movs_id = c.movs_id

            and   c.movs_reg_siteid = b.movs_reg_siteid

                and   c.stk_id = d.stk_id

            and   a.emp_id = e.emp_id

            and   a.emp_siteid = e.emp_siteid

            )

/*********************************Query data from Site 2***********************************/

        union

        ( select a.selltxn_id, a.selltxn_date, b.movs_id,b.movs_reg_siteid, c.MOVS_NAME,

        a.selltxn_site_id,d.stk_sellingprice, d.stk_itemcost, e.emp_id, e.emp_name,e.emp_siteid,
        e.emp_working_site

                from sell_transaction@site2.site2 a, sell_item@site2.site2 b, movie_sell@site2.site2 c,
                stock@site2.site2 d, employee@site2.site2 e

                 where b.selltxn_id = a.selltxn_id

                and   b.movs_id = c.movs_id

                and   c.movs_reg_siteid = b.movs_reg_siteid

                 and   c.stk_id = d.stk_id

                and   a.emp_id = e.emp_id

                and   a.emp_siteid = e.emp_siteid

                 )

/*********************************Query data from Site 3***********************************/
```

```
          union

      ( select a.selltxn_id, a.selltxn_date, b.movs_id,b.movs_reg_siteid, c.MOVS_NAME,

       a.selltxn_site_id,d.stk_sellingprice, d.stk_itemcost, e.emp_id, e.emp_name,e.emp_siteid,
e.emp_working_site

                     from sell_transaction@site3.site3 a, sell_item@site3.site3 b, movie_sell@site3.site3 c,
                     stock@site3.site3 d,  employee@site3.site3 e

               where b.selltxn_id = a.selltxn_id

                     and   b.movs_id = c.movs_id

               and   c.movs_reg_siteid = b.movs_reg_siteid

               and   c.stk_id = d.stk_id

               and   a.emp_id = e.emp_id

               and   a.emp_siteid = e.emp_siteid

          )

 ) h  order by h.selltxn_date;

/************************************Rent Transaction View*********************************/

create or replace view rent_transaction_view as

select h.renttxn_id,h.rent_date,h.rent_returndate,  h.movr_id, h.movr_reg_siteid, h.MOVR_NAME ,

          h.renttxn_site_id, h.emp_id,h.emp_name,h.emp_siteid,h.cost

 from (

/*****************************Query only Site 1 due that it is replicated**************************/

            ( select a.renttxn_id, a.rent_date, a.rent_returndate, b.movr_id,b.movr_reg_siteid, c.MOVR_NAME,

              a.renttxn_site_id, d.emp_id, d.emp_name,d.emp_siteid, e.cost

                     from rent_transaction@site1.site1 a, rent_item@site1.site1 b, movie_rent@site1.site1 c,

                     employee@site1.site1 d, MOV_RENTCOST@site1.site1 e

                     where a.renttxn_id = b.renttxn_id

               and a.renttxn_site_id = b.renttxn_site_id

                     and b.movr_id = c.movr_id

                    and b.movr_reg_siteid = c.movr_reg_siteid

                     and c.movr_type = e.movr_type

                    and a.emp_id = d.emp_id

               and a.emp_siteid = d.emp_siteid) )h;
```

Appendix D: Procedures

Sell_TXN_History Procedure

```
create or replace procedure SELL_TXN_HISTORY
(startDate in date, endDate in date,refCursor1 out SYS_REFCURSOR) as
Begin
        open refCursor1 for
         select h.selltxn_id as "Sell TXN",h.selltxn_date as "Sell Date",  h.movs_id as "Movie
                ID",h.movs_reg_siteid as "Movie Registered Site", h.MOVS_NAME as "Movie
Name",
                h.selltxn_site_id as "TXN Site ID", h.stk_sellingprice as "Selling Price"
         from sell_transaction_view h
         where h.selltxn_date >= startDate and h.selltxn_date <= endDate
         order by h.selltxn_date, h.selltxn_id;

EXCEPTION WHEN OTHERS THEN
     RAISE;
END;
```

SELL_PROFIT_HISTORY Procedure

```
Create or replace procedure SELL_PROFIT_HISTORY

(startDate in date, endDate in date, refCursor1 out SYS_REFCURSOR, vTotalProfit out number,
vTotalSales out NUMBER) as

 Begin
        open refCursor1 for
        select h.selltxn_id as "Sell TXN", h.selltxn_date as "Sell Date", h.movs_id as "MOVIE ID",
                h.MOVS_NAME as "MOVIE Name" , h.selltxn_site_id as "SITE ID", h.emp_id as
                "EMPLOYEE ID", (h.stk_sellingprice - h.stk_itemcost) as "Profit"
         from sell_transaction_view h
         where h.selltxn_date >= startdate and h.selltxn_date <= enddate
         order by h.selltxn_date, h.selltxn_id;

          /*Calculates Total Profit*/
         select sum(h.stk_sellingprice - h.stk_itemcost) into vTotalProfit
         from sell_transaction_view h
        where h.selltxn_date >= startDate and h.selltxn_date <= endDate;

        /*Calculates Total Sales*/
         select count (*) into vTotalSales from sell_transaction_view h
         where h.selltxn_date >= startDate and h.selltxn_date <= endDate;

EXCEPTION WHEN OTHERS THEN
     RAISE;
END;
```

RENT_TXN_HISTORY Procedure

```
create or replace procedure RENT_TXN_HISTORY
(startDate in date, endDate in date, refCursor2 out SYS_REFCURSOR) as

Begin
        open refCursor2 for
         select h.renttxn_id as "Rent ID",h.rent_date as "Rent Date",h.rent_returndate as "Return
Date",
               h.movr_id as "Movie ID", h.movr_reg_siteid as "Movie Reg Site", h.MOVR_NAME
as
               "Movie Name", h.renttxn_site_id as "TXN Site ID"
         from rent_transaction_view h
         where h.rent_date >= startdate and h.rent_date <= enddate
         and h.rent_returndate is not null
         order by h.rent_date, h.renttxn_id;

EXCEPTION WHEN OTHERS THEN
     RAISE;
END;
```

RENT_PROFIT_HISTORY

```
create or replace procedure RENT_PROFIT_HISTORY
(startDate in date, endDate in date, refCursor2 out SYS_REFCURSOR, vTotalProfit out number,
vTotalTXN out number) as

Begin
        open refCursor2 for
         select h.renttxn_id as "Rent ID",h.rent_date as "Rent Date",h.rent_returndate as "Return
Date",
               h.movr_id as "Movie ID", h.MOVR_NAME as "Movie Name", h.renttxn_site_id as
"Site
               ID", h.cost as "Payment"
         from rent_transaction_view h
         where h.rent_date >= startdate and h.rent_date <= enddate
         and h.rent_returndate is not null
         order by h.rent_date, h.renttxn_id;

         select sum (h.cost) into vtotalprofit from rent_transaction_view h
         where h.rent_date >= startdate and h.rent_date <= enddate
         and h.rent_returndate is not null;

         select count(*) into vtotaltxn from rent_transaction_view h
        where h.rent_date >= startdate and h.rent_date <= enddate
         and h.rent_returndate is not null;

EXCEPTION WHEN OTHERS THEN
     RAISE;
END;
```

TXN_EMPLOYEE procedure

```
create or replace procedure TXN_EMPLOYEE
(startDate in date, endDate in date, rentCursor out SYS_REFCURSOR, sellCursor out
SYS_REFCURSOR) as

Begin
        open rentCursor for
          select count (a.renttxn_id) as "RentTXN", a.emp_id as "EMPLOYEE ID", a.emp_siteid as
                  "REGISTERED SITE ID", a.emp_name as "Name"
          from rent_transaction_view a
          where (a.emp_id ,a.emp_siteid) in (select b.emp_id, b.emp_siteid from employee b)
          and a.rent_date >=  startDate and a.rent_date <=  endDate and a.rent_returndate is not null
          group by a.emp_id, a.emp_siteid, a.emp_name;


        open sellCursor for
           select count (a.selltxn_id) as "SellTXN", a.emp_id as "EMPLOYEE ID", a.emp_siteid as
                  "REGISTERED SITE ID", a.emp_name as "Name"
          from sell_transaction_view a
          where (a.emp_id ,a.emp_siteid) in (select b.emp_id, b.emp_siteid from employee b)
          and a.selltxn_date >=  startDate and a.selltxn_date <=  endDate
          group by a.emp_id, a.emp_siteid, a.emp_name;


EXCEPTION WHEN OTHERS THEN
     RAISE;
END;
```

STOCK_QNTY procedure

```
create or replace procedure STOCK_QNTY
(stockCursor out SYS_REFCURSOR) as

Begin
        open stockCursor for

        select count(*), a.stk_id from movie_sell a, stock b
        where a.stk_id = b.stk_id
        and (a.movs_id, a.movs_reg_siteid) not in (select d.movs_id, d.movs_reg_siteid from
sell_item d)
        group by a.stk_id
        order by a.stk_id;


EXCEPTION WHEN OTHERS THEN
     RAISE;
END;
```

MOVIE_LOCATEDSITE procedure

```
create or replace procedure MOVIE_LOCATEDSITE
(movieCursor out SYS_REFCURSOR) as

Begin
        open movieCursor for
        select a.movr_id, a.movr_name, a.movr_type, a.movr_locatedsite
        from movie_rent a
        where a.movr_reg_siteid = (select b.site_id from site b where b.site_local = 'T')
        and a.movr_locatedsite != a.movr_reg_siteid
        and a.movr_id not in
                (select c.movr_id from rent_transaction_view c where c.rent_returndate is NULL)
        group by a.movr_id,a.movr_name, a.movr_type, a.movr_locatedsite;


EXCEPTION WHEN OTHERS THEN
     RAISE;
END;
```

DUE_MOVIES procedure

```
create or replace procedure DUE_MOVIES
(movieCursor out SYS_REFCURSOR) as

Begin
        open movieCursor for
        select e.cust_id, e.cust_name, e.cust_surname, e.cust_cellno, d.movr_id, d.movr_name,
              a.rent_date, a.renttxn_site_id
        from rent_transaction a, mov_rentcost f, rent_item c, movie_rent d, customer e
        where a.rent_returndate is NULL
        and a.renttxn_site_id = (select b.site_id from site b where b.site_local = 'T')
        and a.renttxn_id = c.renttxn_id
        and a.renttxn_site_id = c.renttxn_site_id
        and c.movr_id = d.movr_id
        and c.movr_reg_siteid = d.movr_reg_siteid
        and d.movr_type = f.movr_type
        and SYSDATE > (a.rent_date + f.rent_period)
        and a.cust_id = e.cust_id
        and a.cust_siteid = e.cust_siteid
        order by a.cust_id;

EXCEPTION WHEN OTHERS THEN
     RAISE;
END;
```

BOOKED_MOVIES procedure

```
create or replace procedure BOOKED_MOVIES
(bkmovieCursor out SYS_REFCURSOR) as

Begin
         open bkmovieCursor for
        select a.bkmov_id, b.movr_id, c.movr_name, e.cust_id, e.cust_name, e.cust_surname,
                a.booking_date
        from book_movie a, book_item b, movie_rent c, customer e
        where a.bkmov_id = b.bkmov_id
        and b.movr_id = c.movr_id
        and b.movr_reg_siteid = c.movr_reg_siteid
        and a.cust_id = e.cust_id
        and a.cust_siteid = e.cust_siteid
        order by a.bkmov_id, a.booking_date;

EXCEPTION WHEN OTHERS THEN
     RAISE;
END;
```

AVAILABLE_MOVIES procedure

```
create or replace procedure AVAILABLE_MOVIES
(avmovieCursor out SYS_REFCURSOR) as

Begin
         open avmovieCursor for
         select a.movr_id, a.movr_name, a.movr_type, a.movr_reg_siteid from movie_rent a
         where a.movr_id not in (select c.movr_id from rent_transaction b, rent_item c
                                             where b.renttxn_id = c.renttxn_id
                                             and b.renttxn_site_id = c.renttxn_site_id
                                             and b.rent_returndate is null)
         and (a.movr_id,a.movr_reg_siteid) not in (select e.movr_id,e.movr_reg_siteid
                                                  from book_movie d, book_item e
                                                  where d.bkmov_id = e.bkmov_id)
        and a.movr_locatedsite = (select f.site_id from site f where f.site_local = 'T')
        order by a.movr_id;
EXCEPTION WHEN OTHERS THEN
     RAISE;
END;
```

Annotated Bibliography

Akintola, A. A., Aderounmu, G. A., Osakwe, A. U., & Adigun, M. O. (2005). Performance modeling of an enhanced optimistic locking architecture for concurrency control in a Distributed Database systems. Journal of Research and Practice in Information Technology, 37(4), 365-380.

In this journal, an overview of the importance, and as well as the complexity of concurrency within distributed databases, are highlighted. The authors contiunued this research by analyzing a technique, called optimistic locking architecture, which can be used in such systems.

Bonifati, A., P.K., C., A.M, O., & K., S. (2008). Distributed Databases and peer-to-peer databases: past and present. ACM SIGMOD Record, 37(1), 5-11.

The authors of this paper gave a defention of P2P database systems, and a comparison with other types of distributed database, such as multi-database systems and federated database systems. Within this comparison, the different architectures of these system, is also highlighted. Furthermore several features of P2P systems are also distinguished, and a taxonomy of the existing P2P database systems is given.

Calero, C., Piattini, M., & Ruiz, F. (2003). Towards a database body of knowledge: a study from Spain. ACM SIGMOD Record, 32(2), 48-53.

In this paper, the authors mentioned the increase in importance of databases systems and the huge business that they generate. This was demonstrated by the huge increase in annual growth of the database market, and databases are included in all international curricula. The authors continued their paper by proposing a first draft of a systemized database of knowledge called DBBOK.

Connolly, T., & Begg, C. (2005). Database Systems: A Practical Approach to Design, Implementation, and Management (4 ed.): Addison-Wesley.

Both Thomas Connolly and Carolyn Begg share experience on database design with in the industry. At the moment they are teaching at the university of Paisley in Scotland.

This book foucuses on databases and discusses several aspects in databases design. There are also some chapters which are specifially dedicated to distribted databases. In these chapters the concepts, design considerations, distributed concurrency and replication are some of the topics considered.

Corcoran, A. L., & Hale, J. (1994). A genetic algorithm for fragment allocation in a distributed database system. Proceedings of the 1994 ACM symposium on Applied computing, 247-250.

This article gives an overview of distributed databases and a description of the advantages and disadvantages of such databases.

Date, C. J. (2003). An Introduction to Database systems (8th ed.): Addison Wesley.

This book by C.J. Date, gives an introduction and explanation of the different aspects of a databases. Furthermore, the author gives a good explanation of distributed databases. The twelve rules which distinguish a distributed database are also discussed.

Deris, M. M., Abawajy, J. H., & Mamat, A. (2008). An efficient replicated data acess approach for large-scal distributed systems. Future Generation Computer Systems, 24(1), 1-9.

The importance of replication in data intensive distributed database is highlighted in this journal. It mentions, specifically that replication is used to have high data availablilty and fault tolerant system. In this work the authors also proposes a protocol to maintain consistency between the replicas, and at the same time provids high avalabilty and fault toerance.

Dumitriu, F., & Cretu, L. G. (2002). Distributed Database Technology: A Management Perspective. Proceedings of Economics and Management of Transformation International Symposium.

This paper discuss in depth distributed databases and its advantages and disadvantages with respect to management issues.

Dye, C. (1999). Oracle Distributed Systems (1 ed.): O'Reilly.

> This book describe in detail the distibuted capabilites of Oracle database systems. The author explains in detail, especially the configuration needed to use these distributed capabilites. Its worth mentioning that one of these capabilites explained is actaully the replication possiblites within Oracle DBMS.

Felber, P., Fetzer, C., Guerraoui, R., & Harris, T. (2008). Transactions are back -- but are they the same? ACM SIGACT News, 39(1), 48-58.

> A brief overview of concurrency control within database is given by the authors. They continued this paper by proposing the same concurrency control mechanism for mutli threaded application to access shared memory.

Franklin, M., Halvey, A., & Maier, D. (2005). From databases to dataspaces: a new abstraction for information management ACM SIGMOD Record, 34(4), 27-33.

> The authors of this paper gave an overview of data management systems, where database management systems were also discussed. Furthermore, the authors introduced the use of data spaces as a new form of data management system where they also proposed the design and development of a DataSpace support platforms.

Getahun, F., Tekli, J., Atnafu, S., & Chbeir, R. (2007). The use of semantic-based predicates implication to improve horizontal multimedia database fragmentation. Workshop on multiledia information retrieval on The many faces of multimedia semantics, 29-38.

> The different forms of fragmentation techqniques used within distributed databases, more specifically in multimedia databases, are discussed in this paper.

Hababech, I. O., Ramachandran, M., & Bowring, N. (2007). A high-performance computing method for data in distributed database systems. The Journal of Supercomputing, 39(1), 3-18.

> The performance of a distributed database can be enchanced by efficient use of clusters and data allocation. An approach that computes the descision values for allocating data fragments and a logical way to group sites in clusters is presented in this paper.

Hanssen, G. (2003). Concurrency control in distributed geographical database systems.

Gjermund Hanssen discussed the transaction mechanisms and concurrency control in distributed database systems. Also, the different concurrency control mechanisms used, are also mentioned with in this paper.

Kossmann, D. (2000). The state of the art in distributed query processing. ACM Computing Surveys (CSUR), 32(4), 422-469.

Donald Kossmann, explains in detail distributed query processing and presents some techniques useful for distributed databases systems. Some of the architectures used in distributed databases such as client-server architeure is also discussed and explains how distributed query processing is used in such systems.

Martyn, T. (2000). Implementing Design for Databases: The 'Forgotten' Step. IT Professional, 42-49.

Tim Martyn is a faculty member at the Rensselaer University at Hartford, where in this paper he gives a description of the design and implementation of databases. He also provides a description of the different types of databases including distributed and parallel databases. In this paper Tim proposes to extend the three step methodology when designing databases to a four step in order to include the implementation design.

Mattos, N. M. (2003). Integrating information for on demand computing. Proceedings of the 29th international conference on Very large data base, 29, 8-14.

The importance of information integration was highlighted by Nelson Mattos. In this paper the author describes the motivation and the requirements of information integration. Consequently some scenarios of information integration are given.

Meghini, C., & Thanos, C. (1991). The complexity of operations on a fragmented relation. ACM Transactions on Database Systems (TODS), 16(1), 56-87.

The authors of this paper discuss, that an aspect in designing a distributed databases, is data fragmentation. Different types of schemas that can be found in a distributed architecture are explained.

Ordonez, C., Garcia, J. G., & Chen, Z. (2007). Measuring referential integrity in distributed databases. Proceedings of the ACM first workshop on CyberInfrastructure: information management in eScience, Lisbon, Portugal, 61-66.

> The metrics, involved in measuring the referential integrity in distributed databases, are proposed within this article. The authors also explain several query optimization issues. Consecutively, an overview of a prototype application which computes referential completeness, and consistency metrics is given. These application is used in distributed scentific databases.

Ozsu, M. T., & Valduriez, P. (1999). Principles of Distributed Databases (2 ed.). New Jersey: Prentice-Hall.

> Distributed databases are the main focus of this book by M.T. Ozsu and P. Valduriez. Different topics related to specifically distributed databases, and the types of architecuteres, with their respective components used to build such databases are explained in some detail.

Plattner, C., & Alonso, G. (2004). Ganymed: Scalable replication for transactional web apllications. Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, Toronto Canada, 155-174.

> In this paper, the authors give a defention of replication and an overview of the different types of replcation systems used in database systems.

Rob, P., & Coronel, C. (2004). Database Systems: Design, Implementation, & Management (6 ed.): Thomson Course Technology.

> As the name implies, this book provides a good description of the design, implementation and the management of databases system. The authors cover in some detail the design aspects and procedures in desiging the actual databases. Distributed databases are also discussed in this book.

Roosta, S. H. (2005). A new model for distributed database systems. International Journal of computer Mathematics, 82(12), 1447-1454.

Seyed Roosta mentions the decsicions and consideration when designing a distributed database. A special consideration is given to fragmentation and allocation of the data. Also a model for allocating tables in distributed databases is developed in this paper.

Russell, L. Dealing with "Scope Creep" in Software Development Projects Retrieved 10th February, 2009, from http://www.projectsmart.co.uk/dealing-with-scope-creep-in-software-development-projects.html

Linda Russll, gives a description of scope creep and some suggestions how to deal with it.

Shanker, U., Misra, M., & A.K., S. (2008). Distributed real time database systems: background and literature review. Distributed and Parallel Databases, 23(2), 127-149.

The authors give a description of a real time database focusing mainly on the performance of such systems. Moreover a research about the issues encountered in transaction processing of a distributed real time database systems is also mentioned.

Sohn, K., & Moon, S. (2000). Achieving high degree of concurrency in multidatabase transaction scheduling: MTOS. Journal of Systems Architecture, 46(8), 687-698.

This paper is focused on multi-database systems, where a defention of this database system is given. The authors focuses mainly on the concurrency of these systems, where they also proposed a ticket based system which can be used inorder to obtain concurrency control.

Son, J. H., & Kim, M. H. (2004). An adaptable vertical partitioning method in distributed systems. Journal of Systems and Software, 73(3), 551-561.

The authors describe the types of partioning, that is, vertical, horizontal and hybrid in a distributed systesm. Apart from a proposal of an adpative vertical partionining method is given with this paper.

Stamford, C. (2007). Gartner Says Worldwide Relational Database Market Increased 14 Percent in 2006 Granter Press Release Retrieved 24th January, 2009, from http://www.gartner.com/it/page.jsp?id=507466

In this web article, the author gives an overview of the market as regards relational database. a comparison of the popularity and growth of some database vendors is also given.

structured query language. (n.d.). Dictionary.com Unabridged (v 1.1). Retrieved April 23, 2008, from Dictionary.com website:http://dictionary.reference.com/browse/SQL

Dictionary.com was used as a resource to clearly define the term structured query language (SQL).

Talwadker, A. S. (2003). Survey of Computing Sciences in Colleges. Journal of Computing Sciences in Colleges, 18(6), 5-9.

The author gives a description of parallel database systems and also includes the different architectures for parallel databases. He continues this paper by considering the data placement and the query optimization in these database systems.

Vasileva, S., Milev, P., & Stoyanov, B. (2007). Some Models of a Distributed Database Management Systems with Data Replication. International Conference on Computer Systems and Technologies - CompSysTech'07.

In this paper, mentiones the three different methods for transaction concurrency control focusing mainly 2PL method. The author also explains the different forms of the 2PL method in distributed databases.

Vokorokos, L., Balaz, A., Adam, N., & Petrik, S. (2005). Dataflow distributed database systems. (Technical report). [Magazine/Journal]. Annals of DAAAM & Proceedings (373).

In this article, the authors describe the forms of distribution and the different DDBMS architectures. It also explains the client-server architecture and peer to peer distributed database architectures.

References

1. Ault, M. Distributed Database Management Retrieved 20th January 2008, 2008, from http://www.praetoriate.com/t_grid_rac_distributed_db.htm
2. DB2 servers and IBM data server clients. *DB2 Version 9.5 for Linux, UNIX, and Windows* Retrieved 30th July 2008, from http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.luw.qb.server.doc/doc/r0006867.html
3. Distributed Database Concepts. *Oracle® Database Administrator's Guide 10g Release 2 (10.2)* Retrieved 28th July 2008, 2008, from http://download.oracle.com/docs/cd/B19306_01/server.102/b14231/ds_concepts.htm
4. Ensor, D., & Stevenson, I. (2003). Introduction to Oracle design. *ORACLE DATABASE ADMINISTRATOR*, from http://searchoracle.techtarget.com/tip/1,289483,sid41_gci904246,00.html
5. Fogel, S. (2008). Oracle Database Administration Guide11g Release Retrieved 12th August, 2008, from http://www.oracle.com/pls/db111/to_pdf?pathname=server.111/b28310.pdf
6. Greenwald, R., Stackowiak, R., & Stern, J. (2004). *Oracle essentials: Oracle Database 10g* (Third ed.): O'Reilly.
7. Moore, S. (2008). Oracle Database Advanced Application Developer's Guide 11g Release Retrieved 26th September, 2008, from http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28424.pdf
8. Morales, T. (2008). Oracle Database Reference 11g Release Retrieved 1st October, 2008, from http://download.oracle.com/docs/cd/B28359_01/server.111/b28320.pdf
9. Morris-Murphy, L. L. (2003). *Oracle 9i: SQL with an Introduction to PL/SQL*. Canada: Thomson.

10. Oracle Database 11*g*. Retrieved 28th July 2008, 2008, from http://www.oracle.com/technology/products/database/oracle11g/index.html

11. Powell, G., & McCullough-Dieter, C. (2007). *Oracle 10g Database Administration: Implementation & Administration*: Thomson.

12. SQL Server 2008. Retrieved 30th July 2008, 2008, from http://technet.microsoft.com/en-us/library/bb418491.aspx

13. Strohm, R. (2007). Oracle Database Concepts 11g Release Retrieved 12th August, 2008, from http://download.oracle.com/docs/cd/B28359_01/server.111/b28318.pdf

14. Urbano, R. (2008a). Oracle Database 2 Day + Data Replication and Integration Guide Retrieved 1st October 2008, from http://download.oracle.com/docs/cd/B28359_01/server.111/b28324.pdf

15. Urbano, R. (2008b). Oracle Database Advanced Replication 11g Release Retrieved 26th September 2008, from http://download.oracle.com/docs/cd/B28359_01/server.111/b28326.pdf

16. Urbano, R. (2008c). Oracle Streams Concepts and Administration 11g Release Retrieved 10th October 2008, from http://download.oracle.com/docs/cd/B28359_01/server.111/b28321.pdf

17. Urbano, R. (2008d). Oracle Streams Replication Administration's Guide 11g Release Retrieved 2nd October, 2008, from http://download.oracle.com/docs/cd/B28359_01/server.111/b28322.pdf