6-15-1999

Analysis of N-Tier Architecture Applied to Distributed-Database Systems

Alexandre G. Valente

ANALYSIS OF N-TIER ARCHITECTURE APPLIED TO DISTRIBUTED-DATABASE SYSTEMS

THESIS

Alexandre G. Valente, 1st Lt., BAF

AFIT/GCS/ENG/99J-04

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCE/ENG/99J-04

ANALYSIS OF N-TIER ARCHITECTURE APPLIED TO DISTRIBUTED-DATABASE SYSTEMS

THESIS

Alexandre G. Valente, 1st Lt, BAF

AFIT/GCE/ENG/99J-04

AFIT/GCE/ENG/99J-04

ANALYSIS OF N-TIER ARCHITECTURE APPLIED TO DISTRIBUTED-DATABASE SYSTEMS

THESIS

Presented to the Graduate School of Engineering

of Air Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Engineering

Alexandre Valente, B.S.C.E.

1st Lieutenant, Brazilian Air Force

June, 1999

AFIT/GCE/ENG/99J-04

ANALYSIS OF N-TIER ARCHITECTURE APPLIED TO DISTRIBUTED-DATABASE SYSTEMS

THESIS

Alexandre G. Valente, 1st Lt, BAF

Approved:

Gary B. Lamont, Ph. D. (Chairman) — 9 JUNE '99 — Date

Maj. Richard A. Raines, Ph.D. — 9 Jun 99 — Date

Maj. Michael L. Talbert, Ph.D. — 9 Jun 99 — Date

Acknowledgements

I wish to thank my Thesis Advisor, Dr. Gary B. Lamont for all the guidance provided during my research effort. Also, I want to thank my Thesis Committee and everybody who contributed to the success of this work.

Thanks to my colleagues of work in the AFIT Parallel lab, especially to Cap. Luiz Fernando Silva.

Special thanks to my family, my parents, and particularly, to my wife Helen, who always provided support and encouragement.

Table of Contents

List of Figures

List of Tables

Abstract

N-tier architecture has been more commonly used as a methodology for developing large database applications. This work evaluates the use of this architecture instead of the classical Client/Server architecture in developing corporate applications based on distributed databases. The comparison between architectures is performed using applications that execute transactions similar to those defined in the Transaction Process Council Type C benchmark (TPC-C). The environment used for development and testing was the AFIT Bimodal Cluster (ABC) – an heterogeneous cluster of PCs, running Microsoft Windows NT 4.0 OS. The comparative experimental analysis demonstrated that the N-tier architecture allows more efficient bandwidth utilization between client and server machines, with similar performance. Results led to conclusion that the N-tier architecture is better suited than the Client/Server for use in corporate systems interconnected by low-bandwidth Wide-Area-Networks (WANs), such as the Internet.

ANALYSIS OF N-TIER ARCHITECTURE APPLIED TO DISTRIBUTED-DATABASE SYSTEMS

I. Introduction

The Brazilian Air Force's Computer Science and Statistics Directorate (DIRINFE), among other tasks, is responsible for creating standards and defining the way software development should be done in the Brazilian Air Force (BAF). These standards usually specify hardware platform, such as network servers, application servers and workstations; and the software platform to be used, such as Network Operating Systems (NOS) and Database Management Systems (DBMSs).

But computer technology is a fast moving target; keeping up with new releases and creating the appropriate standards to deal with it is an overwhelming task. Therefore, technicians and engineers from DIRINFE spend a large part of its time learning how to use and apply new software methodologies.

In the last few years, one new technology, generically known as *Distributed Systems*, has become commercially available and has been subject of "evangelization" by the largest software companies such as Microsoft [3] and Sun [44].

The concept behind Distributed Systems is to develop software as a set of small components. These components are called distributed objects and they can run at different application Servers, accessing different databases. In this type of

system, tasks are divided in 3 or more tiers: a thin interface layer, which interacts with the user; one or more middle tiers that hold the application business logic; and a data tier. Because of this disposition, this technology is also called N-Tier development model [25, 27, 31, 33, 60]. In this thesis investigation, the terms *Distributed Systems, Distributed Objects* or *N-tier model* will be interchangeable.

Distributed systems advocates affirm that this technology could provide several advantages over the existent Client/Server model, such as better software maintenance, smaller development time and better resource utilization and scalability [27, 31, 60]. Of course, new technologies also have disadvantages, which are not usually clear because of the marketing hype.

If Distributed Systems benefits could be validated, this technology could be especially useful to BAF's large corporate systems. These systems have to deal with Brazil's continental distances and usually low-bandwidth network connections.

The goal of this research effort is to compare the use of the Distributed Objects in large distributed database systems against the standard Client/Server methodology. Among the items being measured are network utilization, scalability and easy of implementation and use of this type of solution.

This chapter provides a background on BAF networks and on one of its corporate systems. It also describes the specific problem, research objectives, methodology, assumptions, scope and limitations, significance of research, and expected results.

1.1 Background Information

1.1.1 Client/Server Systems Development in the Brazilian Air Force

Except for the last 6 years, all corporate systems in the Brazilian Air Force (BAF) have been developed to work in some kind of mainframe. There were three computer centers (CCA – Air Force Computing Center) - CCA-SJ, CCA-RJ and CCA-BR – that had mainframes that hosted corporate software. These centers also were responsible for maintaining hardware and developing software. The computer center at Rio (CCA-RJ), has a large number of programmers which sole purpose was to maintain corporate software in use at BAF [1].

In the early 90's, LANs started to become widespread in the BAF. Novell Netware was the standard LAN Network Operating System and MS Windows were the standard PC OS. With the available PCs, some systems started to be developed targeting PCs only. Usually these systems were developed in DBASE III or Clipper, and later, in FoxPro[1].

With PCs becoming more and more common, BAF started to plan the development of corporate systems totally based on PCs. To be consistent with the "downsizing" wave, common at that time, it was decided by DIRINFE that the system would be based on the Client/Server model, using a relational database server. It was also decided that it would be constructed using a 4th generation development tool, and the Oracle CASE Designer was selected in 1991. Since the CASE used was Oracle, the database server chosen was also Oracle, to minimize

possible software "impedances". The first system to be built using this new technology and paradigms would be a critical system to the BAF, the SILOMS - Services, Material and Logistics Integrated System.

Following the tradition of the mainframe, BAF decided that it would develop all its corporate computer systems in an "in-house" manner. The reasons for this type of development are mainly concerns about security and control of the source code of these systems. Although this has been proven to work in some cases, this model also brought to the BAF all the problems related to developing and maintaining large corporate programs.

The advent of new environments such as the Internet posed new challenges to the development of the BAF corporate systems. The necessity of interconnection of the different bases and to provide up-do-date information caused the review and adaptation of most of BAF projects.

1.1.2 Current BAF Status in System Development

All major BAF corporate systems are today being ported to the Client/Server model and the problems encountered are basically the same in all them: Client/Server technology alone wasn't enough to guarantee the success of the new "downsized" systems. Problems such as lack of experience using the Client/Server model or overstatement of its capabilities caused several projects

[1] Information derived from the author's own experience working in the DIRINFE

to go over budget and over the expected development time. SILOMS is still under construction after more than 5 years of work[2].

BAF currently has standardized Oracle as the relational Database Server, and Oracle Designer 2000 as the tool for development of Client/Server corporate systems. The standard NOS changed from Novell Netware to Microsoft Windows NT, basically because of the incompatibilities encountered when running Oracle in older Netware servers. The use of Windows in desktops brought with it some new RAD tools such as Delphi, Visual Basic and PowerBuilder [1]. Also, due to the use of NT, Microsoft SQL Server is sometimes used as the intermediate relational database server. Today, in non-corporate systems, Delphi or PowerBuilder are being used to construct front-ends. These tools are also used to provide alternative front-ends to corporate databases.

But even with these RAD development tools, the adopted model is still the standard Client/Server model. The typical scenario in this model is a "fat" client program running everything, from presentation to business logic; and Oracle or other relational database as the backend, running many complex stored procedures. This model of development has been causing many problems with deployment, maintenance and Internet compatibility. New technologies and models have to be applied to solve part of these problems and this research effort try to address some of these issues.

---

[2] Information based on author's own experience working in the SILOMS Project

Another problem encountered by BAF developers is how to effectively integrate distributed databases. In the standard Client/Server approach, Oracle is used to replicate data among various data sites. But this makes it difficult to have up-to-date information, since it takes time to synchronize the data. Also, having multiple data sites implies the use of mechanisms such as "two-phase commit", which slows down considerably the system. Finally, many locations don't have a fast network connection, therefore replication and two-phase commit mechanisms do not work well, leading to data inconsistency and poor system response time.

Finally, the greater problem is scalability. Scaling Client/Server systems means being able to support a large number of users accessing a single server. Today, the only solution of this problem is to increase the processing power of the server and its available bandwidth. But this solution is an expensive one, since it means buying expensive server hardware and high-speed channels. This problem is also addressed in this research effort.

1.1.3 BAF Network Backbone

In order to analyze the distributed systems in use in BAF, it is necessary to know the layout of the backbone of BAF as well as the available bandwidth and resources.

The corporate BAF network is based on a backbone that is available to most of BAF's units [2]. One of its largest customers is the Materiel Command, because its SILOMS is the largest Client/Server being used today [1].

Since Brazil has the fourth largest territory in the world in continuous surface (exceeded only by Russia, Canada and China), BAF has huge distances to cover in its network backbone. SILOMS, for example, has to reach the BAF's main five depots and several bases located in regions of difficult access, such as in the Amazonian rain forest. The distances between points of presence can reach several thousand kilometers.

Currently, BAF employs several X.25 links between the units in São Paulo, Rio Grande do Sul, Brasília, Pernambuco, Pará, and Amazonas to the concentrator located at Rio de Janeiro. This network backbone is called RCDMA (Air Force Data Communications Network) and it is supposed to connect all BAF's LANs and MANs [1, 2].

The commands that are using the RCDMA today are DAC (Civil Aviation Department), DIRMA (Materiel Directorate), DIRINT (Administration Directorate), DIRSA (Directorate), DIRENG (Engineering Directorate), and all units of the COMGAP (Support Command).

A recent study by [2] proposed a layout for the RCDMA, shown in Figure 1.1. This research assumes that this layout will be adopted and the necessary infrastructure is in place. The reason for this assumption is the lack of up-to-date information of the RCDMA, since it is being currently upgraded and the ultimate goal is the layout proposed by [2] or something very similar.

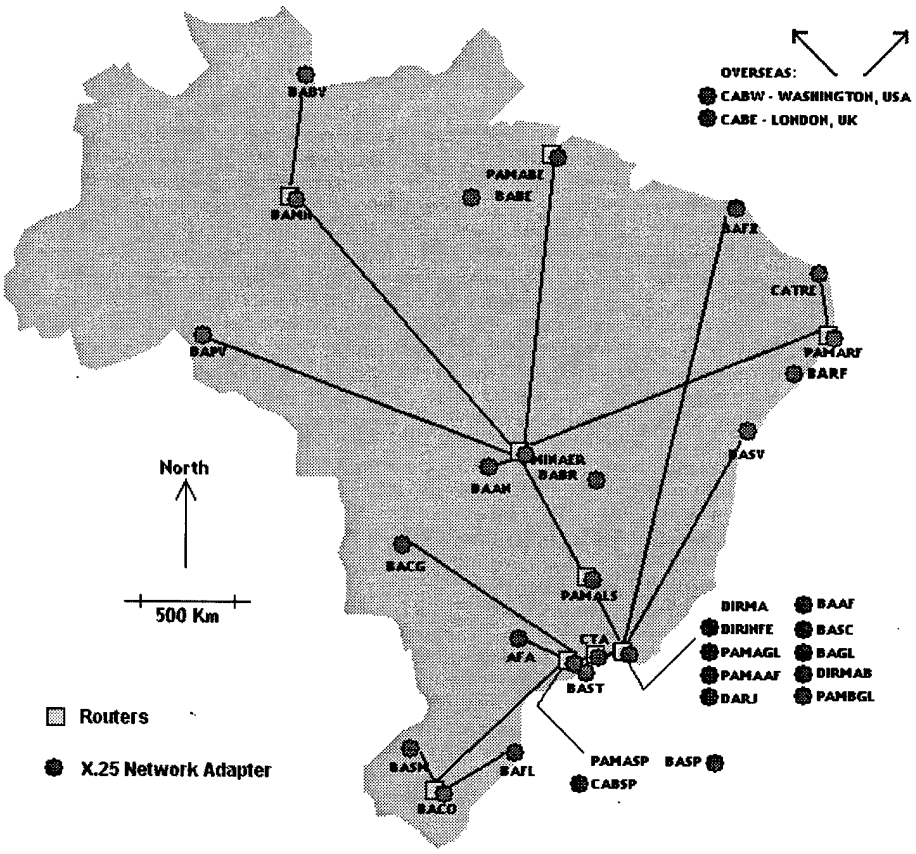

Figure 1.1 - Proposed Brazilian Air Force RCDMA [2]

1.2 Research Objectives

This research effort addresses one issue that has been debated in BAF in the last years: the advantages or disadvantages of adopting a N-tier development model for corporate systems, instead of using the standard Client/Server approach. This research uses a model of a typical a corporate system as data set. Thus, the conclusions that are derived could be extended to any other major system being used or developed today in BAF.

1.3 Significance of the Research

This research by BAF as a first analysis of the use Distributed Systems model in corporate systems. The objective is to have to provide some experimental insights of how a Distributed Systems uses network bandwidth and computer resources, compared to the standard Client/Server model. These results could serve to justify or not changes in the way BAF develops software.

As an example, one of the most important corporate systems in use in BAF is the SILOMS – Services, Material and Logistics Integrated System. This system is still in development and it is supposed to provide online Logistics information where needed [1].

SILOMS is being developed using the standard Client/Server model. Therefore, for the reasons described in Chapter 2, it needs high-bandwidth connections and demands considerable local computing resources.

Figure 1.2 shows all units that have to be integrated with the main Depots by SILOMS. Most of these units have low-bandwidth network connections and small local computer resources. Therefore, it a huge investment is necessary to be able to make all these units part of SILOMS.

If N-tier technology could minimize these necessities, it would help making corporate systems faster and cheaper to deploy and maintain. Like SILOMS, many other applications could potentially benefit from this technology. This research is a first step to validate (or not!) the use of Distributed Objects in BAF.

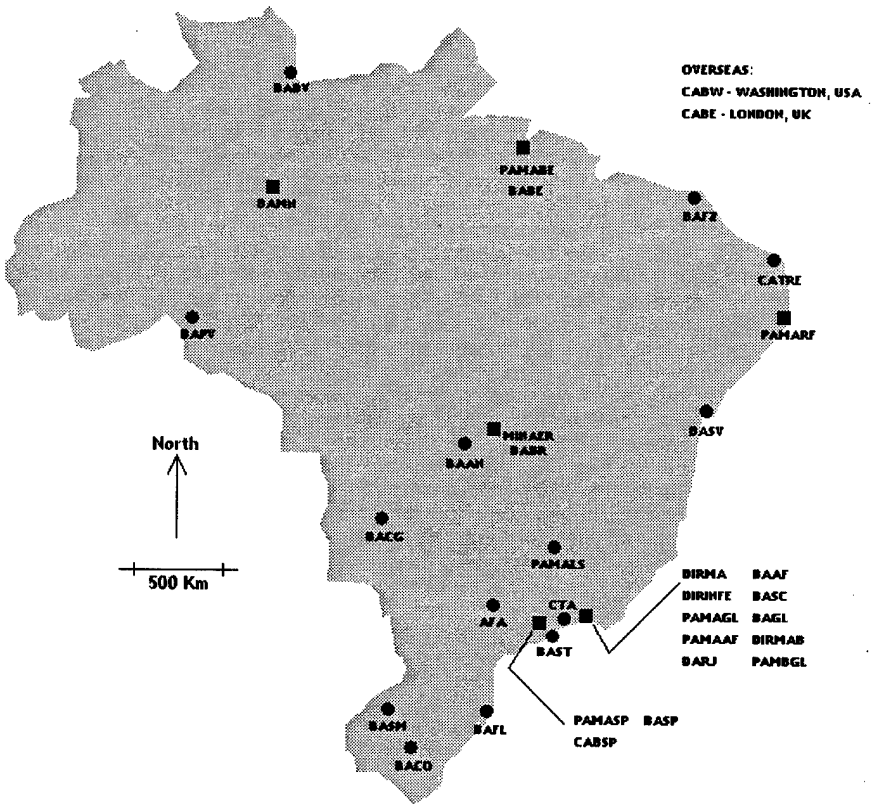

Figure 1.2 - Brazilian Air Force Materiel System Units [2]

1.4 Approach and Organization

To compare the use of the N-tier architecture with the standard Client/Server, the following steps are required:

1. Design a database model that exemplifies the workload and the data distribution of a typical corporate system.
2. Design an application using the chosen model, and implement it, using the Client/Server and the Distributed Objects methodologies, in an environment similar to the ones in use in BAF.

3. Design a set of experiments to reflect the tasks commonly executed in a corporate database and the respective metrics.
4. Obtain statistical results for these metrics, for both methodologies.
5. Analyze the data and derive conclusions.

All these steps are present in the following chapters of this research effort. Chapter 2 provides a theoretical background about the technologies used in this research; Chapter 3 details the adopted methodology; Chapter 4 describes metrics and experiments implementation; Chapter 5 contains the results; and in the Chapter 6 are the conclusions and recommendations.

1.5 Assumptions, Scope and Limitations

All the BAF systems references used in this thesis investigation do not necessarily reflect exact systems currently in use in BAF. This research assumes a common-sense solution where there is no sufficient information about the real system.

Also, all conclusions derived in this research may not be directly applicable to production systems. Different hardware and software platforms could significantly alter the results obtained in this research platform.

Although there are several different factors that affect performance and resource utilization, due to a limited time frame, only the most common were used in this research. Chapter 4 details the chosen factors and the reasoning behind the decisions.

This research also assumes that developers could freely switch between the Client/Server and Distributed Objects model. This does not entirely reflect a real system, were developers would have to be trained in the new technologies. This could take a considerable time, since usually new technologies have a steep learning curve.

1.6 Summary

This chapter provides a general overall description of this research effort. The research objectives and significance are explained. Also, this chapter details the research approach and how this research work is organized. Finally, some assumptions and scope limitations are listed.

II. Literature Review

2.1 Introduction

This chapter covers the literature background necessary to this research effort. Three main topics are addressed: parallel and distributed database systems, standard Client/Server development model and n-tier development. Standards such as CORBA [35] and DCOM [49] are also detailed.

2.2 Parallel and Distributed Database Systems

The idea behind parallel systems is to divide a task among different *workers* in order to achieve better execution time or to be able to deal with problems of a greater size [22]. By using parallel processing, an application can achieve some speedup in parts of the task, allowing better response times and better use of resources. Usually, parallelism can be exploited at different levels. For example, in a DBMS, parallelism can be used at a query level, by using many processors to join or filter some relation; or at the application level, by using multiple threads to attend concurrent users.

This research effort is based on the use of large, distributed databases. Therefore, this discussion of parallel systems will be limited to the use of parallelism in databases and distributed database systems.

Parallel Database Systems, are, according to [22], MIMD systems – Multiple instruction-streams, Multiple data-streams. This means that different data sets are used as processing base for multiple processors. In a modern Database

System, many servers work cooperatively, each one with its own portion of the data.

According to [21], two main measures of performance of a database system are: response time and throughput. By performing subtasks in parallel, a database system can improve response time because the task can be done faster and concurrently; and throughput, because more tasks can be executed at the same time.

It is not possible to parallelize all tasks executed by a DBMS, though. Even at the query level, the type of query and the physical distribution of the data dictates the amount of parallelism that can be achieved.

Three architectural models [21] are relevant in a Parallel Database System: Shared Memory, where all tasks share a common memory, Shared Disk, where the processors share a common disk; and Shared nothing. An example of the first model is multiple threads executing a single query in a Symmetrical Multiprocessor (SMP) machine. Multiple instances of a server accessing the same database in some shared physical media is an example of the second model, and different DBMSes integrated in a single database application by some network exemplify the last. Of course, all these models can occur in any large distributed database system.

All modern DBMSes exploit parallelism at multiple levels. It is common to find database servers that have many processors (SMPs), using shared memory

architecture. This type of machine is expensive and usually there is a limit on the number of processors that the DBMS or OS can support.

Shared disk is also a very common type of architecture in parallel database systems, where Redundant Array of Inexpensive Disks (RAID) is shared among different servers. This architecture usually does not scale beyond a certain point due to the disk bottleneck. The last type of architecture, shared nothing, is the most common one. In this category are included systems based on Client/Server or N-tier models.

On Client/Server or N-tier systems, processing tasks are divided among the existing processors. Client/Server systems usually have one or more database servers that are responsible for heavy processing tasks and multiple clients providing user interface and validating data. N-tier systems also have multiple databases and multiple clients but usually heavy processing and data validation are performed in application servers.

When multiple databases, at different physical locations, are part of the same application, this application is called a Distributed Database System. All modern DBMSes are capable of dealing with multiple databases and integrating them in some form to provide distributed capabilities.

The use of distributed and parallel technologies in database systems has been a common place in recent versions of commercial DBMSes, such as Microsoft SQL Server [47] and Oracle [48]. These DBMSes make use of the multithread capabilities of Windows NT and Unix OSes to be able to exploit local parallel

processing; and they make use of the network infrastructure to provide distributed database capabilities.

The following sections describe in detail the two models being comparing in this research effort: Client/Server and N-tier. Also, Distributed Database systems are detailed.

2.3 Client/Server Systems

2.3.1 Historical Notes

The Client/Server model originated from the old concept of centralized computers, where "dumb" terminals accessed a large mainframe. This model was used because computational power was very expensive and it was necessary to share it among many simultaneous users in order to minimize the mainframe cost.

After the PC revolution, in the middle 80's, computational cost dropped considerably and users started to use their own machines to run their programs. This brought some new problems to companies, since each PC started to accumulate its own data in a particular program format, making cross-application data exchange a difficult task. Later in the decade, LAN's started to proliferate and, with them, the need to put together all this dispersed data in a common file server.

In the beginning of the 90's, LAN's file servers were used to store data files, which were used by applications running on workstations. Therefore, these

servers often became an application bottleneck since they worked like scaled down mainframes, providing data integrity, concurrency control, etc. Workstations hosted all the applications, therefore networks have to support high network traffic and it was necessary to have workstations with significant computational power. These problems were the main reason for the rising of the Client/Server model.

2.3.2 Client/Server Model

According to [8], the "Client/Server technology is a paradigm, or model, for the interaction between concurrently executing software processes". This model applies to one or more machines. This means that it is possible to have a Client/Server system in a single machine, if there are different processes for client and server tasks.

Although the Client/Server model can be used in different contexts and infrastructures, such as inside the local operating system or network, the most frequent use of it is in networked database environment. In this case, a server processes are running in the database server, and the client process is running in some workstation, connected to the database server by some network.

The Client/Server model solved the problem of inter-application communication by storing all data in a single server database and providing standard data retrieving methods. The server also alleviated the need of powerful workstations because it hosted all the application's heavy-processing tasks.

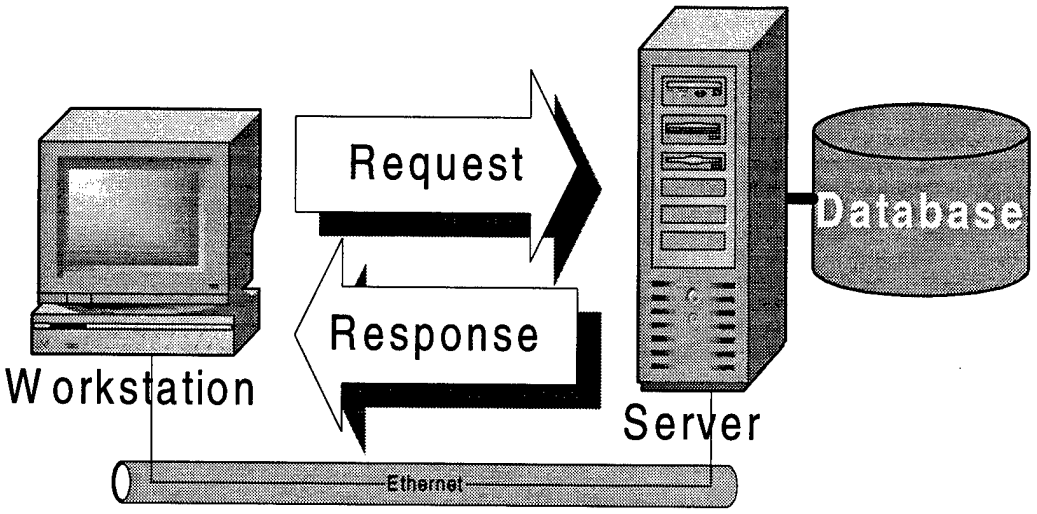

Figure 2.1 – Classical Client/Server Model

In the Client/Server model, a **client** send requests and a **server** responds to these requests by doing some processing and returning results (see Fig. 2.1). This "interaction between the Client and Server is a cooperative, transactional exchange in which the client is the proactive and the server is reactive"[8].

In Client/Server Database Systems, data is stored in the data server in a relational and specialized application. This application is often called SQL Server, and it is responsible for providing access to multiple, concurrently users. Usually, multiple applications such as spreadsheets or business graphic software are connected to a database at the same time, each one requesting data and posting transactions. The database server also is responsible for maintaining data integrity, dealing with error recovery and security.

The client program, in a Client/Server database system, provides the following functions:

- Windows and screen manipulations – such as dialog box controls;
- Keyboard or mouse entry;
- Sound and video displaying and management;
- Data entry and some level of data validation;
- Error and Help displaying.

Users do not have to know that there is a network and a database server running in the background, he or she only interacts with the client console.

According to [12], the server program has the following main attributes:

- Provides a method of data access to the client – normally done by using a standard data access language, such as Structured Query Language (SQL);
- Provides some Data Definition Language (DDL) to retrieve meta-information about the stored data and to create and destroy data objects;
- Has the ability to measure the data access performance and provide means of changing critical parameters;
- Controls Data integrity and guarantees entity and referential consistency of the data;
- Process Transactions and guarantees that data updates occur in a consistent manner;

- Controls Concurrency to allow a large number of users to work at the same time in the same database;
- Provides Security and authorization checking;
- Provides Backup, recovery and other database administration functions.

2.3.3 Distributed Databases

Distributed Databases is the term used to describe a collection of data which is logically viewed as one but actually is physically located at different, connected nodes [7]. These nodes are loosely coupled [21] and can be connected by any physical mean. This interconnection medium is usually a WAN, such as the Internet. Each node that participates in a Distributed Database may issue transactions that span many one or more other nodes.

A Distributed Database Management System – DDBMS – is a DBMS server that is capable of managing many connected databases to create a centralized, unique view of all connected databases. To be able to perform this task, the DBMS has to deal with many factors, such as Distributed Update Propagation, Distributed Catalog Management, Distributed Concurrency Control and Distributed Query Optimization [7]. To correctly address all these issues, the DDBMS has to have local autonomy, sustain continuous operation, allow multiple control sites, support data location independence, and provide distributed transaction management.

A Distributed Database System relies on a mechanism called *two-phase commit*. This allows an update to happen in the correct order in multiple locations. By using two-phase commit mechanism, the DBMS ensures that the data referential integrity is preserved in all locations affected by the transaction [19].

In a Distributed Database, transactions that span more than one database have to maintain the same characteristics of single database transaction. In other words, a distributed database transaction has to be: **Atomic** – that means that the transaction happens in all nodes or in none; **Consistent** – meaning than a transaction will always execute in the same manner, being reproducible; **Isolated** - which means that all data involved in a transaction is protected from external changes during the transaction; and **Durable** – meaning that once the transaction is committed, the data is secured and the changes can't be reversed [6].

According to [21], there are several ways of storing data in a distributed database: through **replication, fragmentation** or using both. In the following section these concepts are explained in more detail.

2.3.3.1 Replication

Data Replication is the mechanism of copying data from one to location to one or more destinations. With this mechanism, the data located in one node can be provided to multiple users. Therefore, the Data Replication mechanism is used in Distributed Database to move the data among the various nodes.

Data Replication mechanisms also guarantee that each update is propagated to all the copies; therefore preserving the data consistency. The synchroni-

zation of the replicated data can be achieved by different ways: **Immediately** – when the updated is executed, the DDBMS starts copying the changed data to its replicated copies; **Scheduled** – all the changes are propagated at a specified time; **Triggered** – happens when a certain event occurs (such as the number of updated entries reaches a certain limit); and **Manual** – the administrator starts the process [6].

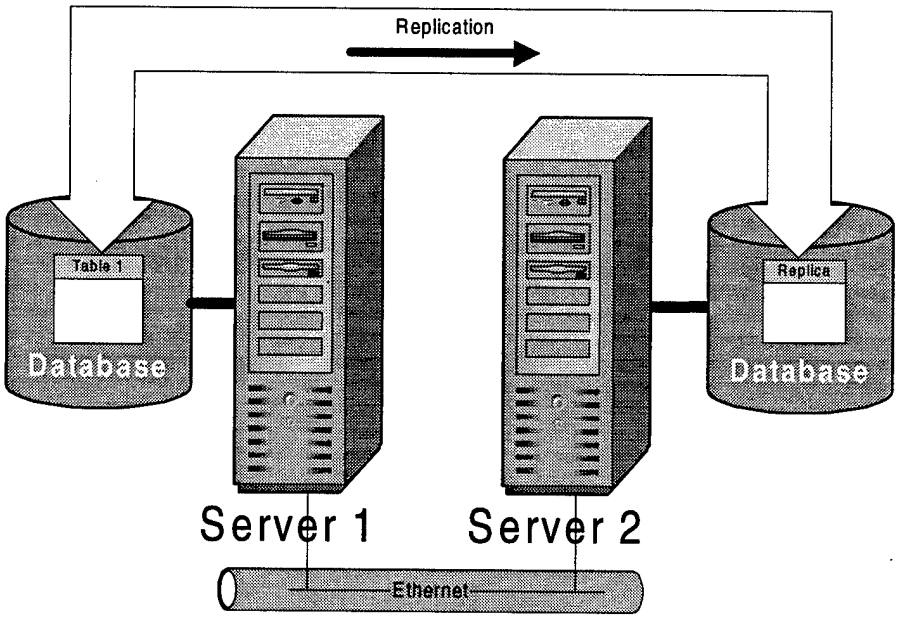

Figure 2.2 – One-Way Data Replication

When two or more different nodes update a record that is physically replicated at two or multiple sites, a **collision** or **conflict** is said to happen. In this case, the DDBMS has to decide which copy will prevail over the other. There are different ways of solving this problem; the common approach is to implement rules (based on timestamps or other priorities) to resolve the conflicts; or to have

the database administrator to resolve manually each conflict. There are DDBMS tools that automate part of this process.

The main disadvantage of the Data Replication mechanism is that one cannot be assured of the real state of the global database at a given time. The data can be different on the various locations and only after synchronization; the data at all sites will reflect the actual global state. But since the data is always changing, the local copy is almost never up-to-date. Therefore, data replication cannot be used on critical systems, such as in a seat reservation system; otherwise two customers could reserve the same seat.

2.3.3.2 Fragmentation

Fragmentation is the other way to store data in a distributed database. Instead of copying relations among the nodes, the relation is split and the relation itself can be dispersed through many nodes.

Depending on the way we split the data, the fragmentation is called **horizontal** or **vertical**. In the horizontal fragmentation, different tuples are assigned to different nodes. On the other hand, vertical fragmentation breaks the relation by assigning columns to different locations, decomposing the original relation scheme [21].

In real system, a mixture of vertical and horizontal fragmentation is used where appropriate, depending on the type of the system and how the data is dispersed.

2.3.3.3 Two-Phase Commit

To ensure atomicity, when a transaction is issued in a distributed system, all nodes must agree on the final result of the transaction [21]. The common protocols used to achieve this atomicity are the two-phase commit and the three-phase commit protocols. The three-phase commit has some advantages over the two-phase commit but the major database vendors usually only implement the two-phase commit protocol in their products.

The two-phase commit protocol occurs in the following way (Figure 2.3):

1. The first node starts the transaction, writes the *prepare* statement in the local log and sends a *prepare* message to all other nodes involved in the transaction. All nodes that receive the prepare message, add a *ready* statement in their local log and reply with a *ready* message. If any node cannot execute the *prepare* command, it sends back an *abort* message.
2. If all the nodes respond positively to the *prepare* message before a timeout, the initial node issues a commit to its log and sends a *commit* message to all the participating nodes. All nodes that receive this message also commit their local databases. If any of the nodes issued an *abort* message in the previous phase, the initial node aborts its transaction and sends an *abort* message to all participating nodes.

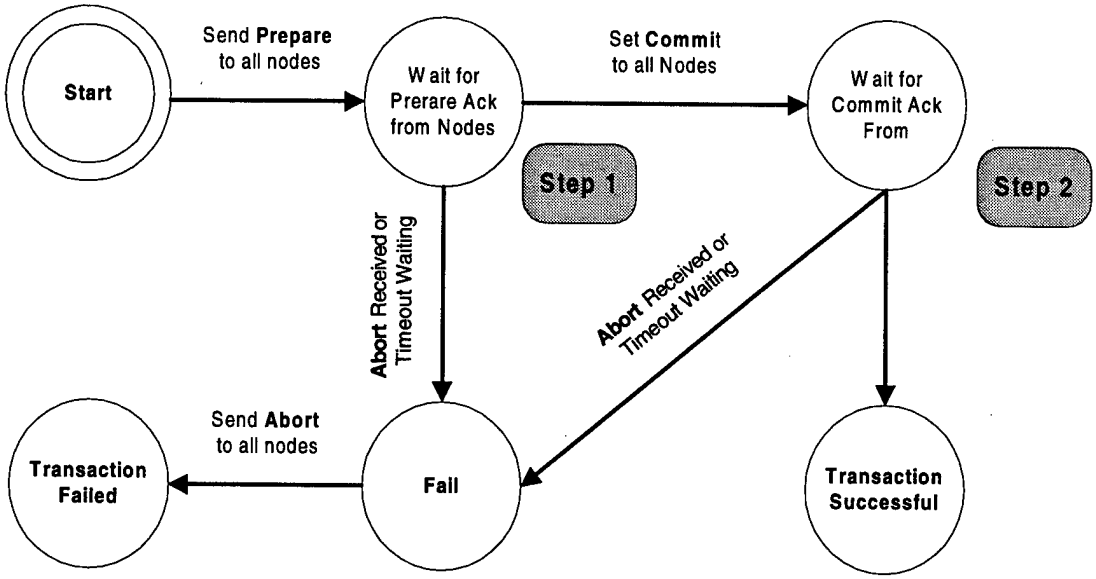

Figure 2.3 – State Diagram of Two-Phase Commit Protocol

2.4 N-tier Architecture

As discussed in the previous section, Client/Server has several advantages, but it also has several disadvantages, as described below [27]:

- There is no middleware involved in the Client/Server model; therefore integrating different vendors is a very difficult task, since each server has to maintain its own copy of application logic, often in different languages.
- There is no support for rich data such as images or videos in the standard relational database servers. Although some vendors do provide some types of functionality, integrating those in the Client/Server model is not a straightforward operation.

- The server has to be increasingly powerful as the application scales. This leads to a very expensive, mainframe-like hardware.
- Since the client usually plays an active role in the business logic implementation, it has the necessity of a high-bandwidth network. And this necessity grows as the number of simultaneous users increases.
- Scalability in a Client/Server means increasing the processing and bandwidth power and thus increasing the server price. Also, in this type of system the server becomes a single point of failure, increasing the price of disaster recovery solutions.

To address these disadvantages, N-tier model uses the concept of distributed application processing. The idea is to exploit the available computational power of different servers by breaking the application in several components.

The N-tier architecture can solve many of the listed C/S problems by introducing intermediate layers of software between the client and the server. These layers will act as middleware that implement all the applications business logic providing application scalability by increasing the number of application servers. In the following sections these concepts are explained in more detail.

2.4.1 Distributed Objects Architecture

The middle layers of a N-tier application are comprised by Business Objects. These objects act as a "bridge" between the client and the server, being responsible to carry out transactions that can span across multiple servers.

Each of these objects usually implements a "Business Rule". Business Rule, or Logic, is any type of function that executes one or more task of the company's application. Data validation, database transactions and query processing are example of business rules [32].

The business objects can act alone or in cooperation among with other business objects. Each one can be viewed as a single entity; therefore they are usually implemented as a binary software object. The collection of these objects in the middle-layer of a N-tier application is called Distributed Objects Architecture.

Distributed Objects Architecture is model where software is developed using Component-Based development. Component-Based Development is an evolution of previous paradigms of software development, such as modular development, subroutine libraries, Client/Server and object-oriented development.

In the pure object-oriented approach, the reuse and inheritance is restricted to the source code level. If a developer has to change a class definition, he or she would have to change and recompile the entire application. The idea of components is to promote the binary reuse of software. In a Distributed Objects Environment, the component is a unit of packing, distribution, maintenance and development [28]. The application is a composed by a collection of run time interconnected components. Each component can be modified and replaced without the need to recompile the entire application. The component supports all characteristics of objects, such as polymorphism, encapsulation and inheritance.

The use of objects provides the possibility of fine-grained tuning in the computing architecture by moving or copying objects to appropriate nodes of the network, hence the term "Distributed". Also, components can be located at several different servers to achieve load-balancing capabilities.

Distributed objects communicate with each other using messages and specified **interfaces**. The component acts as a service server, by responding to messages addressed to its interfaces; the implementation of these interfaces is hidden from the clients. Components may change independently and transparently, provided that their interfaces are maintained.

To support a Distributed Objects application, it's necessary to have an infrastructure to handle tasks such as object creation, destruction and intercommunication. This infrastructure acts like a bus, connecting the different components and providing a common interface that exposes the component services.

OMG's Object Management Architecture (OMA) is an example of such architecture. It is intended to support distributed enterprise computing applications [35] and includes the following components:

- A global *object model* to define how the heterogeneous resources that makes up the system can be modeled as objects.
- The *Object Request Broker* (ORB), an object messaging bus that enables distributed objects to transparently send and received requests and responses.

- *Object Services*, which support basic functions such as program queries, transactions, and event notification, for using and implementing objects;
- *Common Facilities*, which provide end-user oriented capabilities that useful across multiple application domains.
- *Domain Objects*, which are likely to be used only in specific vertical application domains, such as telecommunications or manufacturing.
- *Application Objects*, which are built specifically for a particular application.

The bus that interconnects the objects also provides mechanisms that let components exchange metadata and discover each other.

Three commercial architectures are currently widely used as an infrastructure to Distributed Objects: Microsoft DCOM [35], Object Management Group (OMG) CORBA [49] and Java Enterprise Java Beans (EBJ) [51], by Sun Corporation. In the following sections, the first two architectures are described. Since EJB was released during this research effort, it is not analyzed here.

2.4.2 CORBA

CORBA stands for *Common Object Request Broker Architecture* [36]. It is controlled by OMG, which have over 700 member companies, such as IBM, SUN and Oracle. The most recent CORBA specification is the 2.1. Many products, such

as Iona Orbix, IBM SOM and Inprise's (former Borland) Visibroker, have an ORB that adheres to this specification.

CORBA Objects are packed binary components that remote clients can access via method invocations [35]. The language and compiler used to create server objects are totally transparent to clients. The clients don't need to know in what operating system or computer the component resides.

The CORBA components publish an interface that acts as a binding between clients and servers. The *Interface Definition Language (IDL)* is used to specify the published interfaces. The IDL-specified methods could be written in Smalltalk, C, C++ or Java. IDL provides operating system and programming language independent interfaces to all services that a component offers.

The *Object Request Broker (ORB)* provides the object bus. It also provides a set of distributed services to let objects discover each other at run time and invoke each other's services[1]. It does that by mediating the transfer of messages from an object to another. When a client invokes a service from a CORBA object, the ORB redirects the function call across the network to the target object (see Figure 2.4).

The ORB offers some *object services* that are used to do maintenance functions. The most important ones are:

- Life Cycle Services – used for creating, copying, moving and deleting components;

- Persistence Service – provides interface for storing components persistently;
- Naming Service – allows the components to locate each other;
- Event Service – allows components to register and unregister itself for receiving events;
- Concurrency Control Service – provides a resource lock manager;
- Transactional Service – provides two-phase commit using transactions;
- Relationship Service – allows the creation of dynamic relations among components;
- Externalization Service – provides a way of getting data in or out a component;
- Query Service – provides query operations for objects;
- Licensing Service – controls the use of objects;
- Properties Service – provides a mechanism to alter component's attributes.

To call a member function of a CORBA object, the client needs only to know the standard ORB Services and the object IDL. The creation of a CORBA Application involves the following steps:

1. Define the interfaces to the objects, using the CORBA IDL.
2. Compile these interfaces using a IDL Compiler, which produces a *stub code* for the client objects and a *skeleton code* fort the server object;
3. Develop Server programs that will implement the defined interfaces;

4. Register the Server object in the ORB;

5. Develop Client programs that use the defined interfaces;

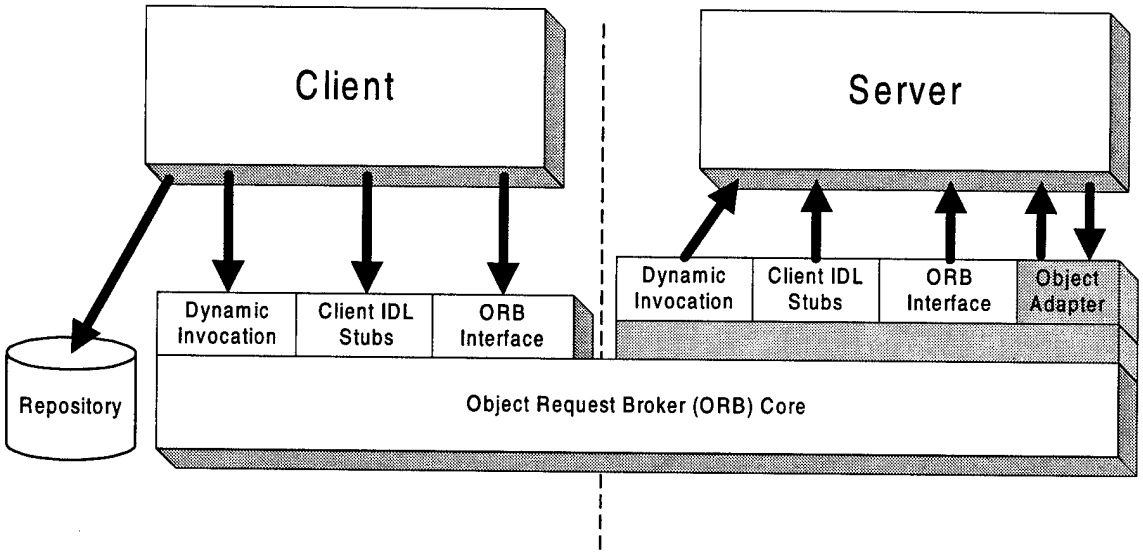

Figure 2.4 – CORBA Architecture

2.4.5 DCOM

COM stands for *Component Object Model* [49]. COM is a Microsoft's binary standard and it specifies how to build components that can dynamically interact. DCOM stands for *Distributed COM* and it is an extension to the COM model that allows the objects to exist across a network. DCOM simply replaces the standard COM inter-process communication by a network protocol. Usually the terms COM and DCOM are interchangeable, but COM is more adequate to a single machine application and DCOM to a network application.

Similarly to CORBA, a COM object exposes its services by defining the interfaces through a Interface Definition Language (IDL). Although COM IDL is very similar to CORBA IDL, they have some differences that prevent one for being compiled by the other.

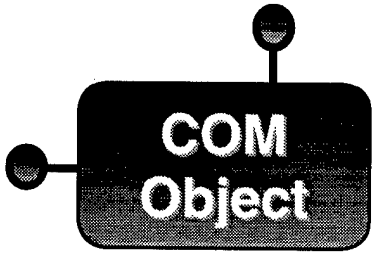

Figure 2.5 – Typical COM object representation.

COM Components are created as an executable code, distributed either as Win32 dynamic link libraries (DLLs) or as executables (EXEs). A COM component also supports the usual object-oriented characteristics, such as polymorphism, encapsulation and interface inheritance. One thing that COM does not support is the implementation inheritance, but it supports binary reuse through Containment and Aggregation. I diagrams, a COM component is usually represented as in Figure 2.5. The little "lollypops" represent interfaces that the object exposes.

COM components are language independent and most commercial development environments support it, including C++, Visual Basic, Delphi and Java. COM library API provides the common component management services. This COM infrastructure, shown in Figure 2.6, is present in all Microsoft OSes, such Windows 98 and Windows NT.

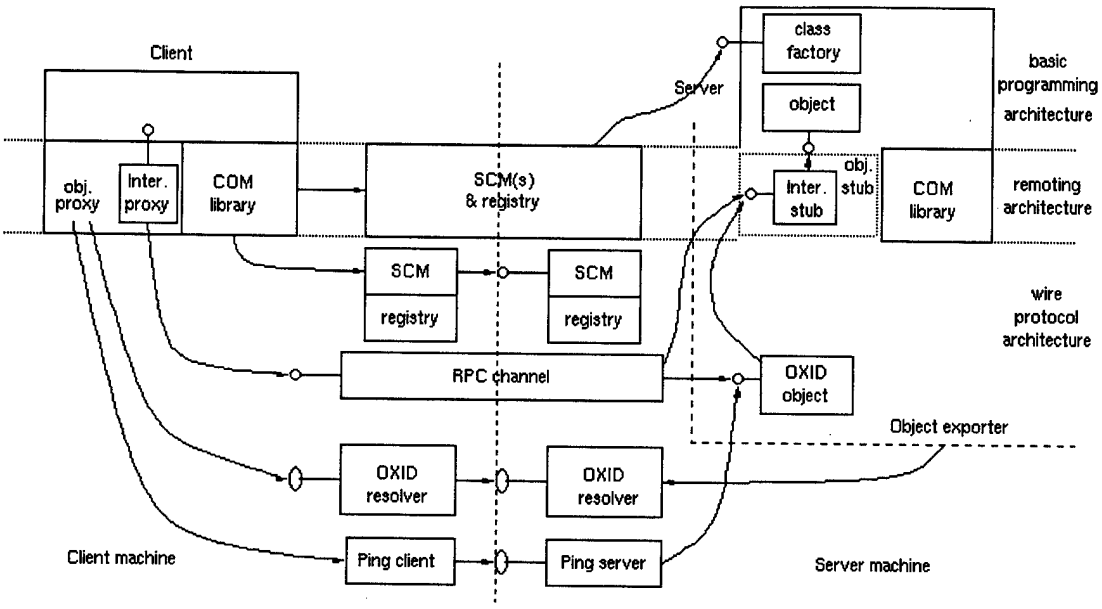

Figure 2.6 – DCOM overall architecture [50].

The client application uses COM objects through COM interfaces. During the first request (or at a time specified by the client), the server object is activated and the requested interface is sent back to the client. All COM interfaces are derived from a standard interface: *IUnknow*. An object can implement one or more interfaces.

There are two types of server objects: **in-process** and **out-of-process** objects. The former executes inside the client's address space and is packaged by a DLL.

Out-of-process objects can reside in the same machine, in a different address space, or in a different machine. To allow a client accesses an interface of this type of object, it's necessary to use piece of COM infrastructure called *proxy-*

stub pair. Their purpose is to transfer the parameters and return values across the different address spaces or machines. This process is called *marshaling.*

All COM components are registered in the Microsoft Operating System registry. This enables the client to find the objects that they require. All interfaces have a unique identifier number called *IID* (interface ID) and the object package has a *CLSID* (class Ids). These IDs are Global Unique Identifiers (GUID), generated by an algorithm that uses the network board physical address, time and other variables to ensure that the generated ID is unique.

Differently from CORBA, where calls can be defined as synchronous or asynchronous, all COM calls are synchronous. Therefore, COM calls are not scalable by themselves, since a client must wait a complete method execution before being able to execute a subsequent task. To address this issue, most current COM-based systems also use Microsoft Message Queue Server (MSMQ) [65]. The MSMQ Server is a message-based server middleware that can provide asynchronous capabilities to COM applications. The upcoming COM version, COM+, will have MSMQ integrated into the core COM framework. This will make asynchronous COM calls transparent to users. COM+ will be available with Microsoft Windows 2000, which is currently in Beta test.

It is also common to implement COM systems in conjunction with Microsoft Transaction Server (MTS) [66]. MTS provides better scalability to COM systems by using resource pooling and object caching. Also, MTS can coordinate transactions among different database servers, making it possible to business

objects in middle-tier to issue multi-database transactions. As MSMQ, MTS will be also part of the COM+ framework.

2.4.6 Comparing CORBA and DCOM

COM and CORBA define objects as a collection of methods and data. Both allow access to an object only through specific interfaces, and both provide an Interface Definition Language (IDL) that can be used to define that interface. But they have some differences, and in the following paragraphs some of this differences will be described.

The main difference between the two architectures is the use of interfaces. In CORBA, each object presents a single interface to its clients, and each client holds one object reference to the object as a whole. In COM, an object can present two or more interfaces to its clients. A client usually holds multiple interface pointers to the same object. Unlike CORBA, COM clients invoke methods through a specific pointer to the interface containing that method rather than via a single reference to the entire object.

One other difference is the way of creating and managing objects. In CORBA, an object is typically created by a call to the ORB. This call generates an object reference to the new object, a reference that can be used by clients to invoke methods on that object. When a client invokes a method on an object that's currently active (i.e., the object's code and data are in memory), the ORB passes the request to the running object. If the target object is not currently active, the ORB loads it, then hands it the client's request. Clients don't need to inform an

object when they are done using it, and exactly when an object stop running is not defined by the CORBA standard. Instead, some CORBA implementations require the client to explicitly tell the ORB that an object should be deleted. Until this is done, the ORB is perfectly willing to start and stop the object as needed.

In COM, a client can create an object via a call to the standard COM library. Among other parameters to this call, the client specifies the CLSID of the object it wants to create and the desired IID. To efficiently create many objects of the same class, a client can instead acquire a pointer to a class factory for that class. A client gets its first interface pointer to a new object as part of the creation process. It then gets any other pointers as it needs, by asking the object for them directly. When the client is finished using the object, it informs this fact by calling the *Release* method on the interface pointer. When all clients have released all pointers on all of an object's interfaces, the object usually destroys itself.

A third, and perhaps, the most controversial architectural difference between CORBA and COM, refers to one aspect of object-orientation: inheritance. COM does not support for multiple interface inheritance, due to COM implementation specifics. Although it is very rare to find applications that need multiple interface inheritance, this is a major argument of CORBA followers against COM.

2.5 Summary

This chapter reviews the theoretical background for the topics used in this research effort. The concept of parallel and distributed systems architectures applied to large database systems is discussed.

The two main distributed objects standards, CORBA and DCOM, are explained. These two standards are also compared and its main differences highlighted.

III. Methodology

3.1 Introduction

As described in the historical background (Chapter 1), the Client/Server model alone hasn't been enough to guarantee the success of the corporate systems developed at BAF. This research effort analyzes the consequences of using the N-tier development model instead of the Client/Server. The particular interest is with respect to changes in the system scalability and network utilization.

One of main steps when developing a N-tier development is to isolate all the business processes in the corporation and implement them as middle-layer components, using one of the available distributed-objects technologies (see previous chapter). The N-tier client application is usually designed to be a thin interface, which communicates with the business objects in the middle layers. The business objects implement the corporate business logic, and communicate with other business objects or use the data layer for storing and retrieving data.

As advocated by N-tier vendors [3, 35, 44], by using the N-tier model, better scalability could be possible because business objects could be replicated to different servers and activated by some server load balancing mechanism. They also say that N-tier systems require less network bandwidth between clients and servers because all data manipulation occurs between the middle and the data tier.

But this is a relatively new model and development teams are afraid of what impact on overall system performance would this technology bring. Also,

although the development time could in theory be reduced, all development teams would have to be trained to use new development tools and models. Finally, what would be the real benefits in the scalability of the produced software? The improvement in scalability would have to be big enough to justify such a transition.

In this chapter, the design of two models is detailed: a standard Client/Server model and an N-tier model. Both models use the same underlying database, which is also detailed. Finally, the metrics to compare these models and the experiments to measure them are explained.

3.2 Detailed Research Objectives

In this research effort, the comparison between Client/Server and N-tier architectures is being addressed. Specifically, trying to identify and measure the advantages or disadvantages of adopting a N-tier development model for corporate systems instead of using the standard Client/Server approach. The items considered in order to achieve this major goal are:

1. Investigate the current research in component-based development.

2. Investigate the current research in parallel computer systems.

3. Investigate and learn how to use components to develop a large N-tier corporate system.

4. Design and install an environment to simulate a corporate distributed database system.

5. Design and implement a program using the standard Client/Server methodology.

6. Design and implement a program using the N-tier methodology.

7. Design a set of metrics to be used to compare the two methodologies and the respective experiments to measure them.

8. Develop a testing plan to be used to run the proposed experiments and measure the developed metrics in both models.

3.2.1 Discussion of Objectives

Component based development is developed around the two standards being used by the industry, CORBA and DCOM. The knowledge of these standards is useful to BAF because it can provide a better ground for discussion if BAF decides to pursue this technology.

The second objective is important to understand the some of the issues related to the development of parallel systems, and to apply these concepts in the development of database distributed systems.

Objective 3 is necessary to know how to use component-based software to build N-tier systems, with emphasis in the development of large corporate systems.

The objectives 4 to 6 are accomplished when setting up an environment and building up the models for both the standard Client/Server and the N-tier systems. The environment set up includes the installation and configuration of a

Relational Database Server and clients, and a component-based infrastructure using Windows NT, SQL Server and DCOM

Finally, the last two objectives are important to be able to develop the metrics and the experiments to compare the models, by gathering statistical data.

3.3 Environment

3.3.1 Platform

Corporate systems are based on multiple LANs, each one with multiple servers (file server, database and application servers, etc), interconnected by a WAN. This research effort uses the ABC Bimodal PC Cluster in the Parallel Lab, for development and testing. It was chosen because of it availability and because it has the necessary number of server to resemble a corporate LAN. Although it is a single LAN, WAN traffic can be estimated by measuring network traffic between client and servers and among servers.

The ABC NT Cluster consists of 4 Pentium II 333 MHz, 7 Pentium II 400 MHz computers and a Pentium II 450MHz, all with at least 128M of memory and interconnected by a Fast-Ethernet network (100Mbits/s) using a central Intel switch [70]. Each machine is able to dual-boot to Linux [70] or Windows NT [71].

3.3.2 Operating System

This research effort uses only the NT operating system version 4.0, Service Pack 4. The reason for this choice is primarily to be consistent with BAF's environment, which uses NT as OS for database servers, as discussed in Chapter 1.

Also, the workload to install and conFigure a DCOM or a CORBA framework in a mixed environment, such as Linux-NT, is beyond the scope of this work.

3.3.3 Use of DCOM in this Research Effort

The alternatives to implement distributed objects, as discussed in chapter 2, are CORBA, DCOM or EJB. DCOM is the choice for this research effort because of:

1. Most corporate systems in the BAF run on Windows Operating Systems (Windows 98 and NT). The exceptions are legacy systems that are currently in process of conversion to a standard C/S application.
2. BAF's environment is almost all based on Microsoft Operating Systems and Development Tools. The only exceptions are the Oracle Database Server [4] and some development tools such as Delphi or PowerBuilder [43]. But even these tools have the necessary support for DCOM.
3. Developer tools for DCOM are much more common than the ones for CORBA. Particularly, the Microsoft tools that are used in this research effort, such as Visual C++ and Visual Basic [3], have native support for DCOM development.
4. The use of CORBA ORB instead of DCOM would have to be tied to one specific vendor, since different vendor ORBs hardly interface with each other. To choose one of the available ORBs would make this re-

search too specific, since BAF does not currently use any ORB implementation and a possible choice is not known at this time.

5. Finally, the research results can be extrapolated to any Distributed Objects Architecture, such as CORBA or EJB, since they all share the common methodology.

3.3.4 Use of Microsoft SQL Server as RDBMS

PC-based corporate system uses one of the commercially available SGBDs for this platform, such as Oracle SQL Server [48], Microsoft SQL Server [47] and others. This research effort uses MS SQL Server as the Relational Database Server. Although this is not the main RDBMS in use at BAF (the most common is Oracle, as discussed in chapter 1), it was the choice because it has a better integration with Windows NT (such as administration tools, DCOM support, network and security integration [47]) and because of the author's familiarity with it

The results derived from this research effort do not depend on the relational DBMS used; therefore this choice is irrelevant to the achievement of the research objectives.

3.3.5 Development Tools

In this research effort, two different software systems have to be designed and built: the Client/Server front-end and the DCOM Objects. There are many available software tools that can be used to build front-ends, such as Inprise's Delphi [43], Sybase's PowerBuilder [6], Microsoft Visual Basic [3], etc. For this

research effort, MS Visual Basic 6.0 Service Pack 1 is chosen to build the front-ends of the Client/Server and the N-tier model. The reason for this choice is simply due to the author's familiarity with it. The research results are independent of the front-end used.

For building DCOM Objects, Visual Basic and Microsoft Visual C++ 6.0 Service Pack 1 are used. Visual Basic is used when performance was not a critical issue, since, based on this author's experience, it is simpler to create DCOM objects in VB than in VC. When VC is used, the Automation Template Library (ATL) [51] was used to create the DCOM framework. There was not much choice in this case; of the available C++ frameworks such as Inprise's C++ Builder [43], Symantec C++ [72], only MS Visual C++ has a library to automate the process of creating COM objects.

The choice of a development tool is not a factor that affects the results of this research effort; the Distributed Objects can be built using any development tool that support these architectural concepts.

3.4 Design and Implementation

To evaluate and compare N-tier against the Client/Server models, it is necessary to have some set of metrics that can be applied independently of the model being used. These metrics have to be based on distributed databases and have to simulate real-world scenarios. The Transaction Processing Council [37] TPC-C benchmark has exactly these characteristics, therefore its metrics were chosen to be used in this research effort.

TPC-C benchmark basically measures transaction response time, as described later in this chapter. In this research effort, bandwidth utilization is also an important factor to be measured. Therefore, network-monitoring tools were used for determining the bandwidth utilization of both models.

All TPC-C benchmark's transactions use an underlying database, which design is specified in the TPC-C benchmark. This database models a warehouse, with sub-districts, items, stock and clients. The following section details this database and its implementation.

3.4.1 Database Design

The database specified in the TPC-C benchmarks is one that represents a business that "manage, sell or distribute a product or service" [37]. The database models a company that has many districts, in different locations, associated with a central warehouse. The warehouse has 10,000 items in stock and 10 different districts, and each district has 3,000 consumers. Customers place new orders, with an average of 10 order lines in average. They can also request the status of any existing order [37]. The database diagram is shown in Figure 3.2. The Order table in the TPC-C specification was changed to District_Order because in some tools, "Order" is a reserved word.

Some TPC-C clauses, described in sections 1.5 and 2.3 of [73], concerning integrity, isolation and some ACID properties were not considered in this research effort because all commercial RDBMS already ensure these characteristics.

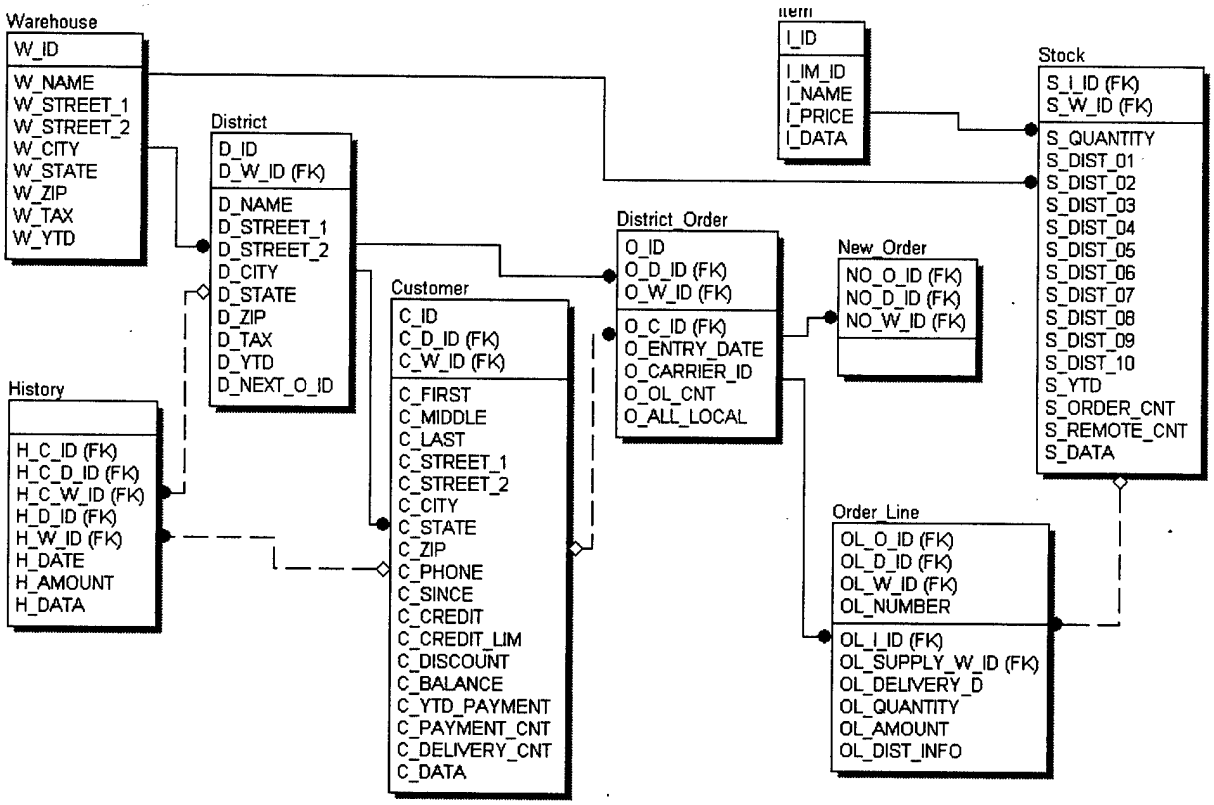

Figure 3.1 – Database Diagram

Table Name	Cardinality	Size (bytes)
Warehouse	1	89
District	10	950
Customer	30,000	19,650,000
History	30,000	1,380,000
Order	30,000	720,000
New_Order	9,000	72,000
Order_Line	300,000	16,200,000
Stock	100,000	30,600,000
Item	100,000	8,200,000
	Total	76,823,039

Table 3.1 – Database Cardinality

3.4.1.1 Consistency Rules

The TPC-C benchmark specifies how the database has to be populated. The cardinality of the tables and the expected table size are shown in table 3.1. The actual physical size can be different due to index implementations. It also specifies some integrity rules that have to be enforced at the database all the time to ensure database consistency. These consistency requirements are:

1) Warehouse Entity: **W_YTD = sum(D_YTD)**
2) District, District_Order and New_Order entities: **D_NEXT_ID –1 = max(O_ID) = max(No_ID)**
3) District, District_Order and New_Order entities: **D_NEXT_ID –1 = max(O_ID) = max(No_ID)**
4) New_Order Entity: **max(NO_O_ID) = min(NO_O_ID) + 1 = [number of rows in New_Order for this district]**
5) District_Order and Order_Line entities: **sum(O_OL_CNT) = [number of rows in the Order_Line for this District]**
6) District_Order Table: **O_CARRIER_ID = Null ⇔ There is a entry in New_Order such as (O_W_ID, O_D_ID, O_ID) = (NO_W_ID, NO_D_ID, NO_O_ID)**

7) Order_Line Table:

OL_DELIVERY_ID = Null ⇔ O_CARRIER_ID = Null if (O_W_ID, O_D_ID, O_ID) = (OL_W_ID, OL_D_ID, OL_O_ID)

8) Warehouse and History entities:

W_YTD = sum(H_AMOUNT)

9) District and History entities:

D_YTD = sum(H_AMOUNT) when (D_W_ID, D_ID) = (H_W_ID, H_D_ID)

10) Customer, History, District_Order and Order_Line entities:

C_BALANCE = sum(OL_AMOUNT) – sum(H_AMOUNT) where

(C_W_ID, C_D_ID, C_ID) = (H_C_W_ID, H_C_D_ID, H_C_ID)

(OL_W_ID, OL_D_ID, OL_O_ID) = (O_W_ID, O_D_ID, O_ID)

(O_W_ID, O_D_ID, O_C_ID) = (C_W_ID, C_D_ID, C_ID)

OL_DELIVERY_ID is not Null

11) Customer, District_Order and New_Order entities:

(count(*) from District_Order) – (count(*) from New_Order) = sum(C_DELIVERY_CNT) where

(O_W_ID, O_D_ID) = (NO_W_ID, NO_D_ID) = (C_W_ID, C_D_ID)

12) Customer and Order_Line entities:

C_BALANCE + C_YTD_PAYMENT = sum(OL_AMOUNT) where

OL_DELIVERY_ID is not null

3.4.1.2 Transactions

The five transactions specified in the TPC-C benchmark are *New Order, Payment, Order Status, Delivery* and *Stock Level.* For a detailed explanation of the transactions and the intermediate steps, refer to [73] and [37].

The New Order is a read-write, high processing transaction that represents the act of entering a new order by some customer. It affects almost all tables of the database and it is the most resource demanding transaction of all specified TPC-C transactions.

The Payment transaction is also a read-write transaction, but not as heavy as the New Order. It affects the District_Order, Order_Line, History and Customer tables.

The Order Status and Stock Level are read only transactions that return few records. The Stock Level transaction, though, is a processing intensive transaction because it requires queries that may scan many entries in the Stock Table.

The Delivery is a read-write transaction with medium resource usage. But it has to be spanned in an asynchronous manner, activated by the user.

3.4.2 Client/Server Model

The Client/Server (see chapter 2) model is constructed using the standard 2-tier architecture, client front-end and database server. Its transaction routines are constructed using database triggers whenever possible, to optimize performance. Figure 3.3 shows the standard Client/Server topology.

3.4.2.1 Front-End

The front-end is responsible for acquiring user data, spanning the transaction and displaying the results. The front-end may also do some processing whenever transactions are too complex to construct using a trigger or a stored procedure. In this case, all actions occur inside the same transaction.

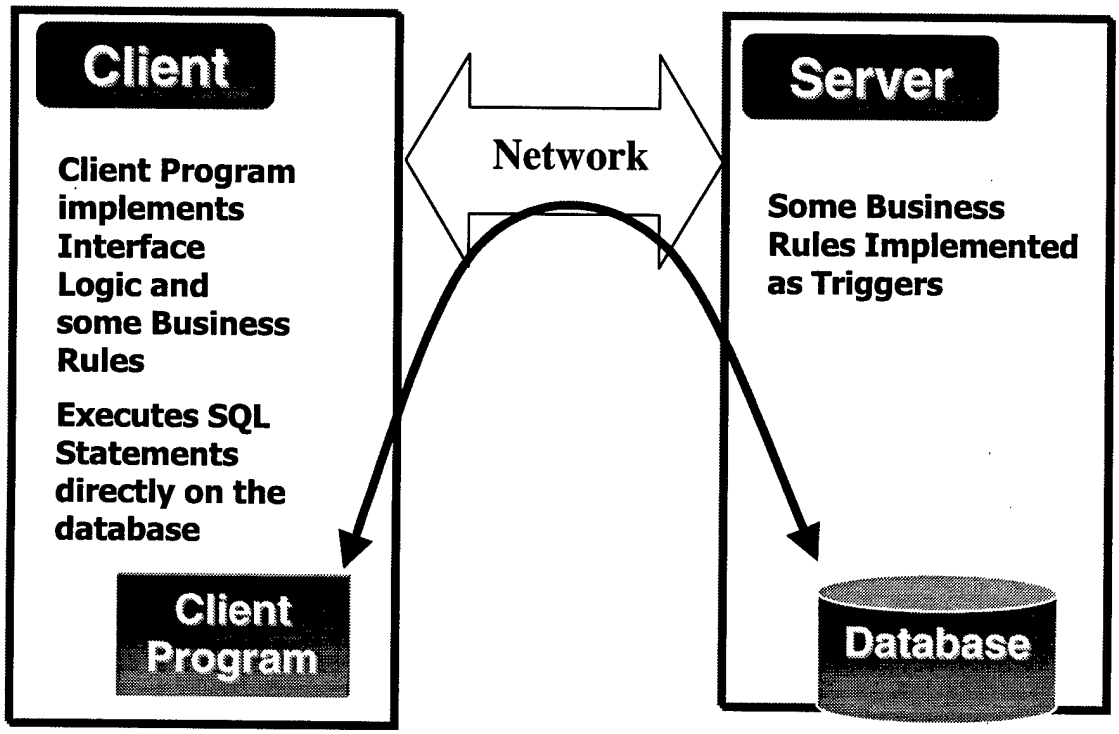

Figure 3.2 – Standard Client/Server Topology

3.4.2.2 Database

The database server is responsible for holding all the data, maintaining ACID properties and running all stored procedures and triggers. All application programs are written using Transact-SQL and batch instructions. Asynchronous routines are constructed using logged events and database alerts. When more than one server is used, distributed transactions are coordinated by the DTC –

Distributed Transaction Coordinator, using two-phase commit protocols and data replication.

3.4.3 N-tier Model

The N-tier model has at least 3 layers: the data layer, comprised by the RDBMS and the necessary Data Objects; the business layer, with its Business Objects; and the interface layer. More than one intermediate layers can exist, and Business Objects can invoke methods from different middle layers. These layers can be located at one or more computers, and the this location topology is discussed in the subsequent sections.

All objects from the different layers are designed using the Microsoft Visual Modeler [3], which is present in the Visual Studio 6.0. This tool is used to generate the code for the objects implementation and it also provides object Unified Modeling Language (UML) diagrams [39].

3.4.3.1 Data Layer

The Data layer, as in the Client/Server model, has one or more database servers that act as data storage. But differently from the Client/Server Model, database servers don't perform business actions. There are few or no stored procedures and few functions are implemented as triggers. The only function performed by the RDBMS is to maintain data and referential integrity. The access to the database server is done through the Data Objects, shown in Figure 3.4.

The Data Objects are used by the Business Objects to perform transactions and by the interface to query and access results. They are designed to resemble the data structure of the system database (the TPC-C database). There are objects for each of the entities, and all data relationships are reflect in the objects relationships.

In an object-oriented system, an Object-Oriented Database Management Server (OODBMS) would be a better model to implement the data services [45], since it is also object-oriented. However, there is no widely used commercial version of an OODBMS for Windows NT and, since OODBMSs are still a new technology, there is no available information about its benefits in corporate systems [75]. Therefore, in this research effort uses a standard RDBMS as underlying data storage for the N-tier system.

To be able to use an RDBMS as underlying storage, all data Objects use special methods specially designed to create the object from a specific record in a table, and to store itself back into the table. A data object can only be altered inside a transaction operation. Although this is not enforced by the architecture itself, all the systems layers are designed to work following this rule.

3.4.3.2 Business Layer

The Business Objects are responsible for executing all business operations. In this research scenario, they implement all the 5 TPC-C transactions. The interface layer triggers the transactions, and the objects from the Data Layer are used

as elements of data during the transaction. The diagram of the Business layer, with its business objects is shown in Figure 3.5.

There are 3 objects in the Business Layer: *GenRand*, *LastNameGen* and *Transactions*. The *GenRand* is responsible for providing all the random functions that are necessary for the diverse TPC-C functions. *LastNameGen* is responsible for encapsulating all the functions that generates the Last Name of the Customer field, as specified in the TPC-C Benchmark. Finally, the *Transactions* object is responsible to implement all the TPC-C transactions, as specified in the TPC-C benchmark.

Microsoft Transaction server can encapsulate business objects, providing database resource pooling and object caching. Therefore, it could improve the system scalability. This alternative is explored in the design of the topologies, as described in the following sections.

3.4.3.3 Interface Layer

The interface layer in the N-tier model is a very thin one. It doesn't implement any business functions; its only purpose is to span these transactions by activating the appropriate business object. Also, this layer is responsible to display the transactions results and to accept user interaction.

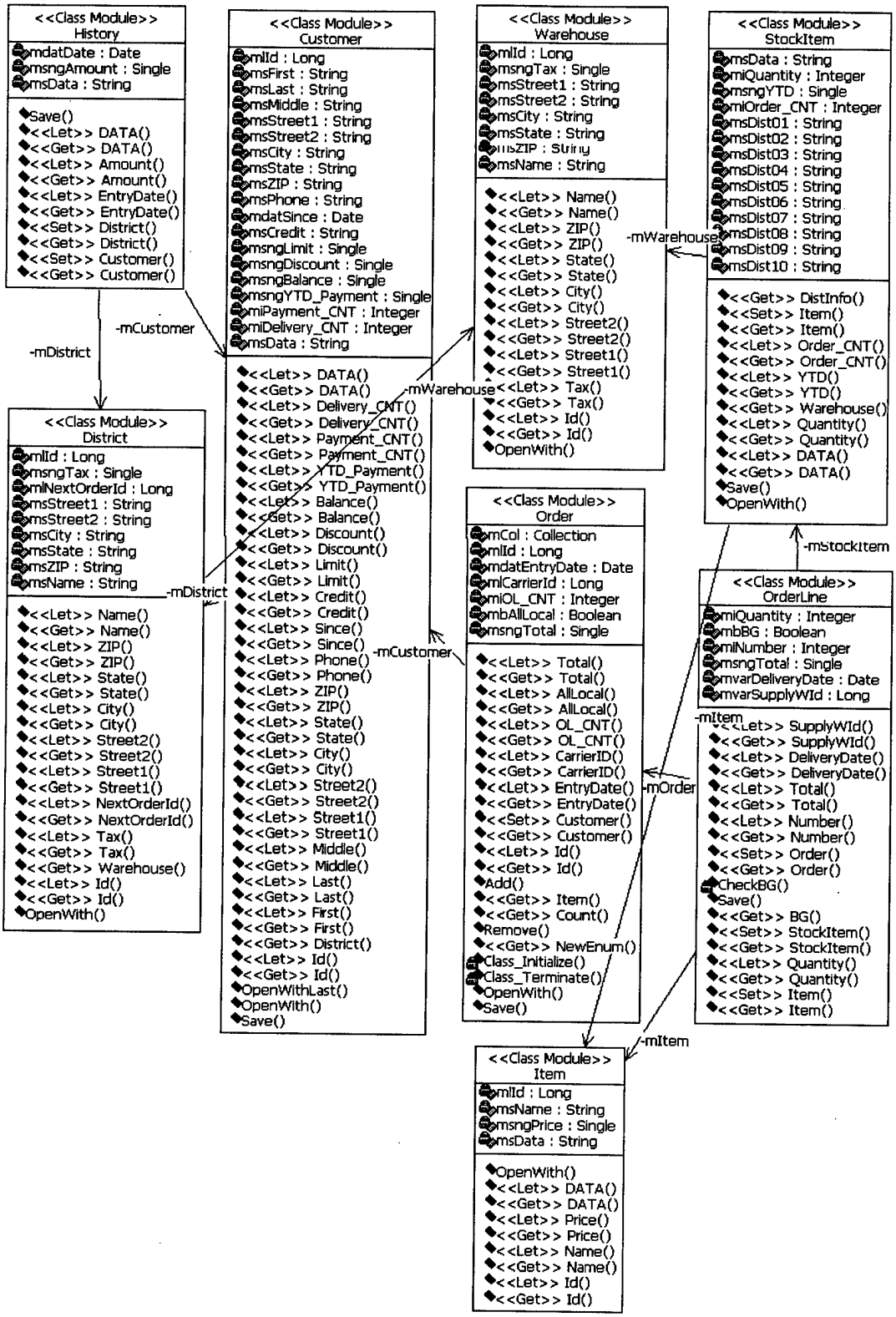

Figure 3.3 – Data Objects Layer

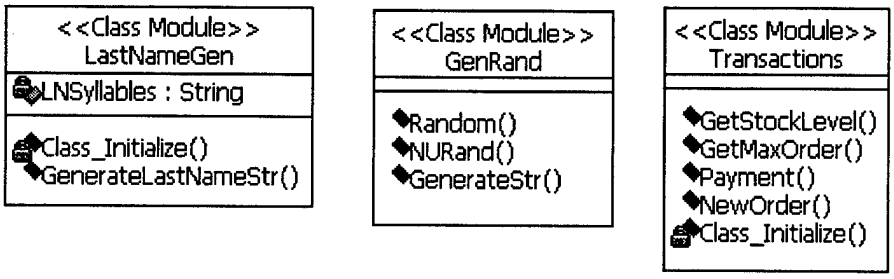

Figure 3.4 – Data Objects Layer

3.4.3.4 Topologies

Although in a typical N-tier system, business objects are located at a dedicated application server, this research also analyses other topologies. The goal is to measure the overhead caused by the use of objects in the client and server machines. Therefore, four different topologies are evaluated:

1) **Topology 1** – In this topology (Figure 3.6), all objects reside in the client machine; the server only contains the RDMS. This topology is not common in N-tier systems but it will serve to evaluate the overhead of using objects as opposed to the Client/Server model.

2) **Topology 2** – This is a common N-tier implementation for small applications. The Client contains only the interface and the server contains the database and all the remaining layers (see Figure 7). The server may use a Transaction Server to encapsulate the Business Objects. This topology resembles the Client/Server model, the only differences are that all business logic is in the server and the server ap-

plication is multi-layered. Small intranet web applications usually use this topology.

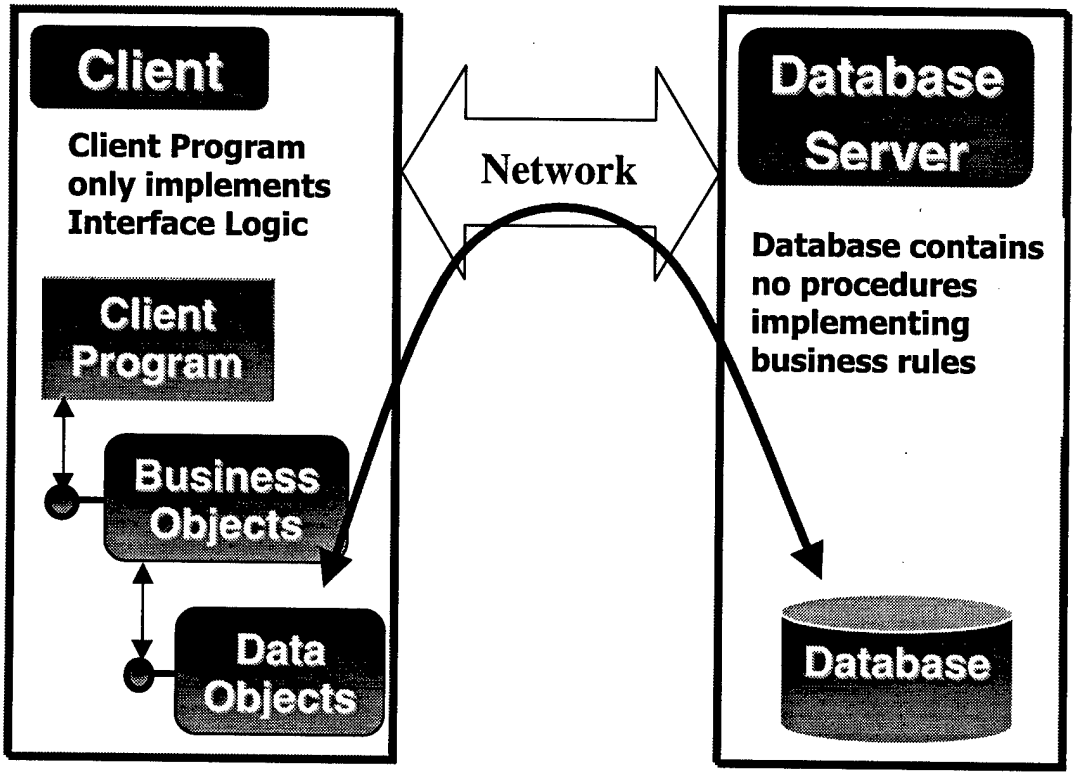

Figure 3.5 – N-tier Topology 1

3) **Topology 3** – This topology, shown in Figure 3.7, is the typical N-tier system using a Transaction server. In this topology, the Data Objects are located at the database server and a dedicated Application Server hosts the Business Objects (Figure 3.7).

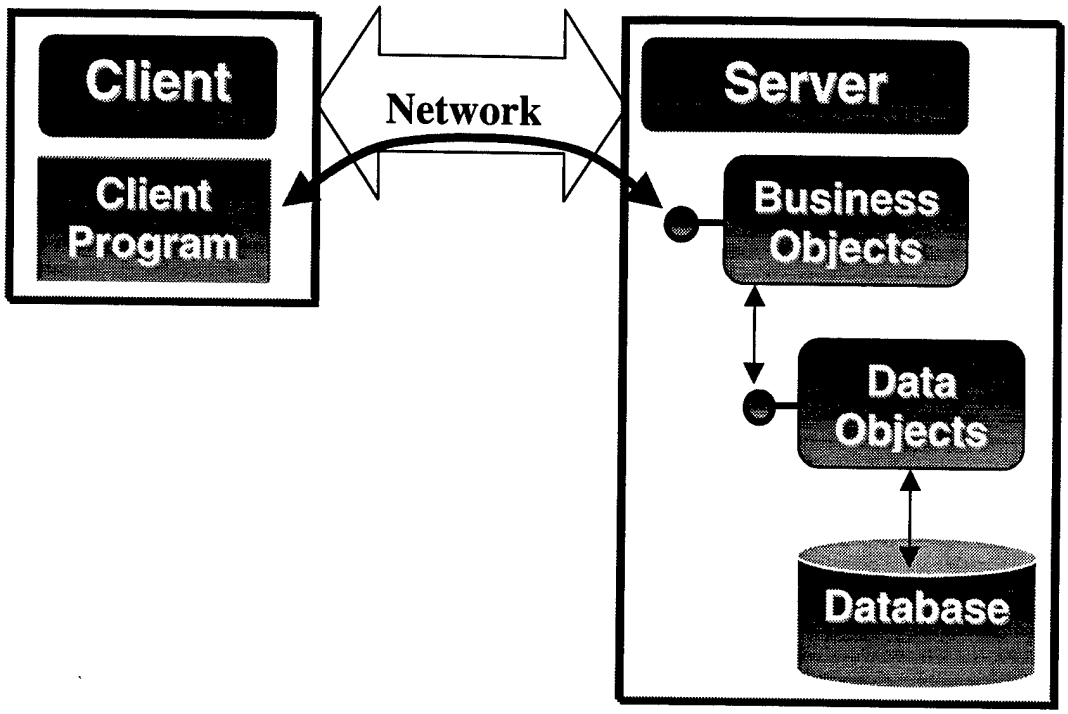

Figure 3.6 – N-tier Topology 2

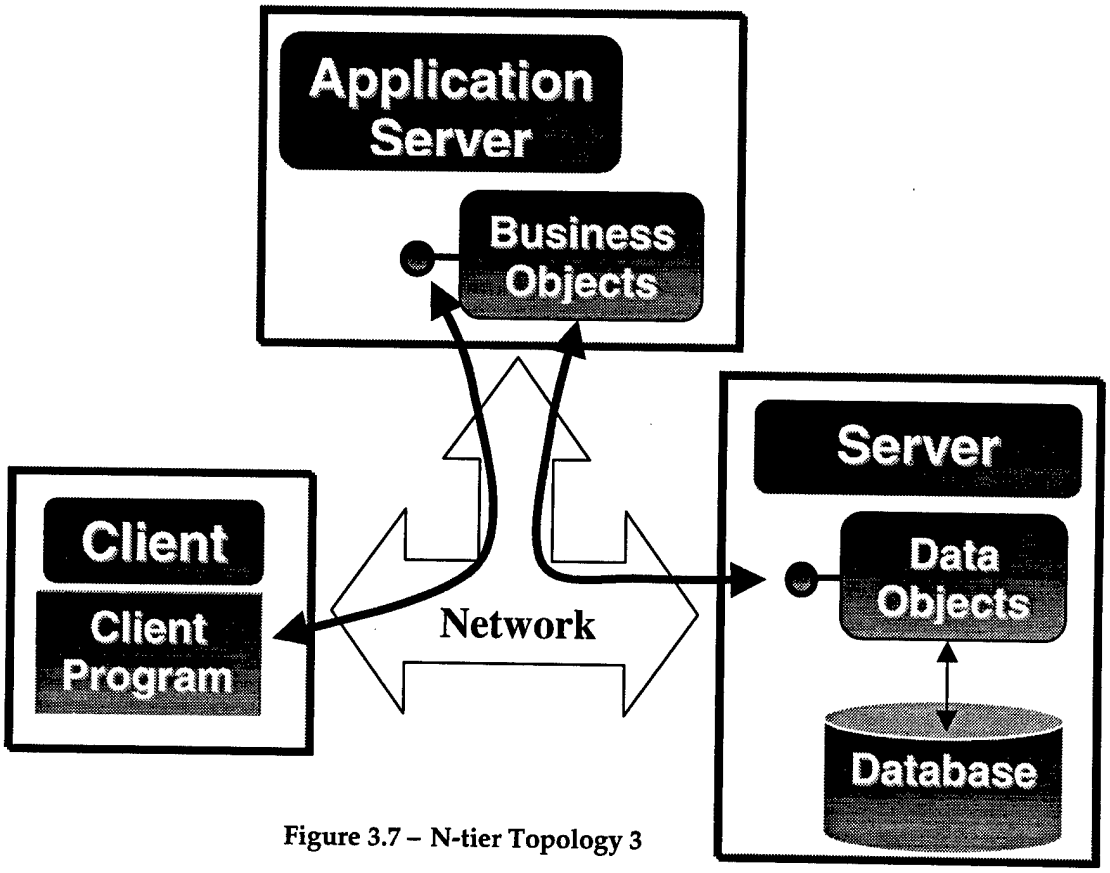

Figure 3.7 – N-tier Topology 3

4) **Topology 4** – This is the most advanced topology and it's the one supposed to have better scalability (Figure 3.8). Clients communicate to Business Objects directly or through MSMSQ as appropriate, and the Business Objects are running inside MTS in an application server. Multiple application servers and database servers can be used.

3.5 Design of Experiments

The method chosen to evaluate and compare the Client/Server with the N-tier model is the TPC-C benchmark. To accomplish this, all the TPC-C benchmark transactions are implemented in both models, using the methodologies appropriate to each one.

This section describes the set of experiments designed to investigate these implementations by measuring execution time and used network bandwidth.

The experiments are divided in two parts: Part A is uses the 5 TPC-C benchmarks in both models against a single database server. This part serves to investigate both models in local network environment.

Part B has only the first TPC-C transaction and has the purpose of investigating both models in a multi-database environment. Two TPC-C databases are interconnected according to the appropriate architecture and network bandwidth utilization between the servers is also measured. This experiment simulates an application executing in a WAN environment, with multiple servers. Transactions 2 to 5 were not used because they don't demand enough database resources

to cause enough inter-server communication. In Part B, only Topology 4 is used since it is the one that implements fully the N-tier architecture.

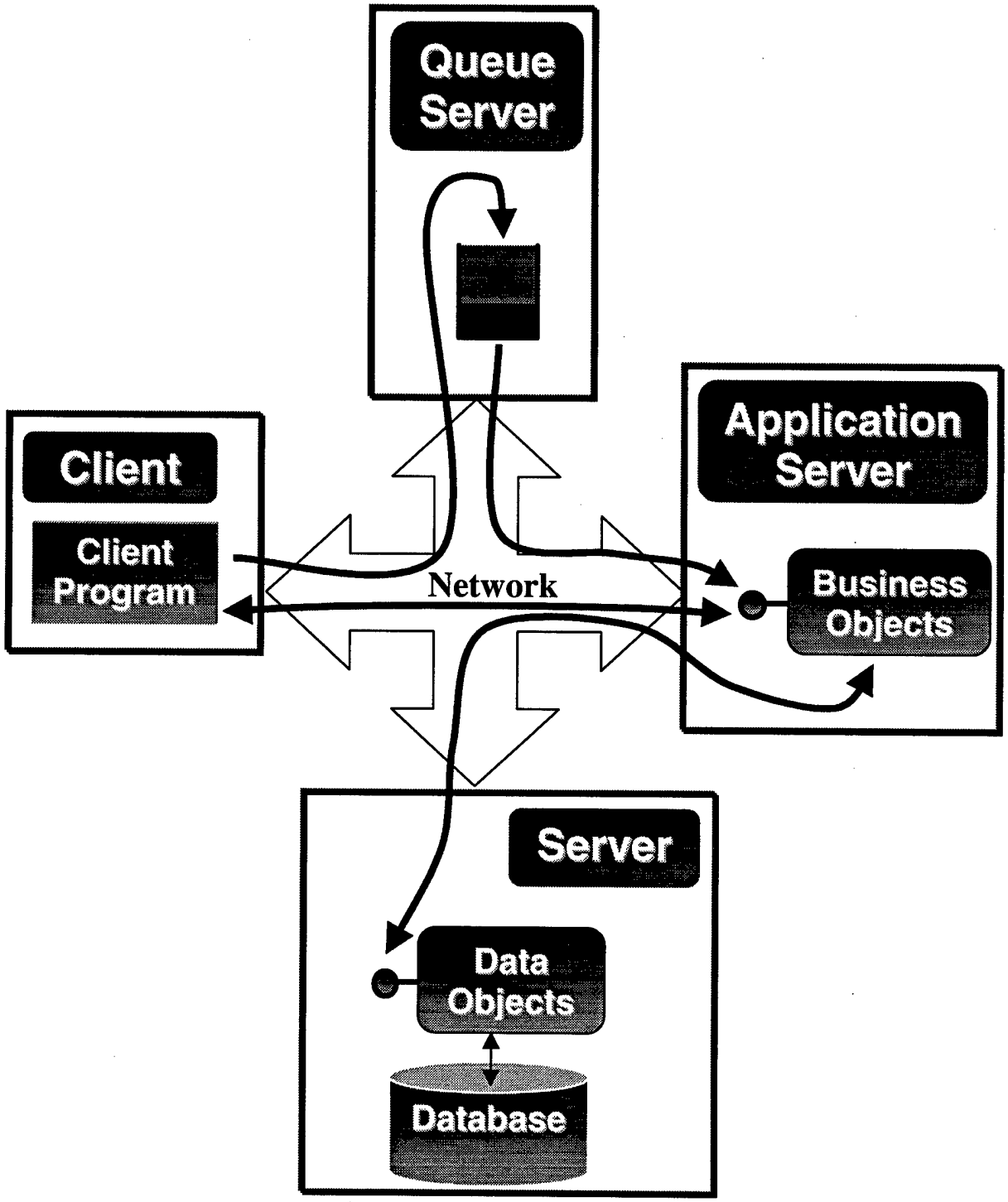

Figure 3.8 – N-tier Topology 4

3.5.1 Parameters

The parameters for both Part A and B experiments are the following:

1) TPC-C Transaction: One of the 5 transactions of the TPC-C benchmark that is being executed.
2) Application Model being used: Client/Server of N-tier application;
3) N-tier topology (in Part B, only the Topology 4 is used);
4) Software: Environment software as described, SQL Server, Transaction Server;
5) Compilers: Compilers used to build the applications, such as Visual Basic or Visual C++;
6) Network: Type of network being used and network utilization during the experiment.
7) Operating System: OS Software, such as version of the Windows NT, and OS utilization.

3.5.2 Factors

The factors selected from the available parameters are the first three items – TPC-C Transactions, Application Model and N-tier topology. All other parameters are defined as described in the start of this chapter.

To try to obtain unbiased results, the network utilization is kept to minimum during the execution of the experiments. The same is valid for the Operating System; no user applications are running other than the experiment itself.

3.5.3 Metrics

To achieve the proposed objectives, the following metrics are used in this research effort (see discussion in section 3.4):

1) Client to Server Network Bandwidth: the amount of network bandwidth used by each model between the client and servers when executing each of the TPC-C benchmark transactions.
2) Server to server Network Bandwidth: In Part B, bandwidth between database servers is also measured.
3) Response Time: the interval between the user's request and the system response, measured until the receipt of the last character of the system's response [44].

3.5.3.1 TPC-C Transactions

The 5 transactions defined in the TPC-C benchmark are used as factors to the experiments. The workload provided is different for each transaction; the first one is a heavy load transaction and the 3rd and 4th transactions are lightweight ones.

3.5.3.2 Application Model and Topology

The Application Model, Client/Server or N-tier, is also used as a factor to the experiments. The implementations of the transactions are different in each model although they accomplish exactly the same results. This leads to different execution times and different bandwidth usage.

The N-tier application can be used with different topologies, the difference among then being the location of the Data and Business Objects. Therefore, another factor in the experiments is the N-tier topology used in the N-tier Model design, as described, earlier in this chapter in section 3.4.3.4.

The first 3 topologies are used to measure the potential overhead of using a N-tier technology. But only Topology 4 has all elements of a N-tier system, therefore the final comparison between the Client/Server and an N-tier model is made using this topology.

3.5.4 Experiments

The experiments are devised using the combination of all specified factors. The resulting grid and number of experiments for Part A are shown in table 3.2. Table 3.3 shows the grid of Part B experiments.

3.5.4.1 Measurement Confidence

To achieve a high level of confidence it is necessary to have a "reasonable" number of experiment executions. As described in [44], the number of necessary

experiments to achieve a specified confidence level is given by the following equation:

$$n = \left(\frac{100zs}{r\bar{x}} \right)^2$$

Experiments Part A	TPC-C Transaction	Model	Topology
1	New Order	Client/Server	N/A
2		N-Tier	Topology 1
3			Topology 2
4			Topology 3
5			Topology 4
6	Payment	Client/Server	N/A
7		N-Tier	Topology 1
8			Topology 2
9			Topology 3
10			Topology 4
11	Order-Status	Client/Server	N/A
12		N-Tier	Topology 1
13			Topology 2
14			Topology 3
15			Topology 4
16	Delivery	Client/Server	N/A
17		N-Tier	Topology 1
18			Topology 2
19			Topology 3
20			Topology 4
21	Stock-Level	Client/Server	N/A
22		N-Tier	Topology 1
23			Topology 2
24			Topology 3
25			Topology 4

Table 3.2 – Part A Experiments

Experiments Part B	TPC-C Transaction	Model	Topology
1	New Order	Client/Server	N/A
2		N-Tier	Topology 4

Table 3.3 – Part B Experiments

In this equation, z is the normal quantile for the desired confidence level, s is the standard deviation of the samples, r is the desired accuracy, and x is the mean of the samples collected.

The complete analysis of the necessary number of experiments and the associated confidence level associated with mean and variance is presented in the next chapter.

3.6 Summary

This chapter describes the methodology of this research effort. It provided details about the models being compared, Client/Server and N-tier, and their respective topologies. This chapter also explains the method of comparison, the TPC-C benchmark, and the factors being measured. The design of the TPC-C transactions implementation in both models is explained, according with the topology used.

The last part of the chapter lists the experiments being performed and details the measurement confidence and number of experiments.

IV. Implementation

4.1 Introduction

This chapter describes the implementation of both Client/Server and N-tier models, based on the design discussed in Chapter 3. It also details the implementation of the supporting databases and replication schema.

Also addressed are the configuration of software applications and frameworks such as DCOM, MTS and MSMQ, as well as the techniques used to gather experimental. The network monitoring tools used to measure network utilization are also detailed.

4.2 Database Implementation

Logic Works Erwin® 2.6 modeling tool [67] is used to model the database described in the TPC-C specification. This application also generates the SQL Script that was executed in the SQL Server to generate the database structure.

In the Client/Server model, all integrity constraints where implemented using triggers. The resulting SQL Code is shown in Appendix B, which contains all the scripts to produce the database.

All the primary keys and foreign keys are indexed in the database. Although those indexes require a large amount of disk space, they improve execution times in most query executions.

4.2.1 Database Population

According to TPC-C specification [37], the database has to be populated using a certain pattern of random strings and numbers, combined to some specific data. These rules are implemented in a Visual Basic program that is listed in Appendix C.

The execution of this program in an environment such as the ABCNT Cluster takes 3 to 8 hours, depending on the rate of utilization of the server and the network. The resulting database is about 168 Mbytes in size, and it is conFigured to be located on a single SQL Server 7 data file.

4.2.2 Experiment Platform

To execute all defined experiments, it was necessary to allocate 5 different ABCNT machines. To ensure a fair comparison, 5 machines were chosen with the exact same configuration: ABNCT05, ABCNT06, ABCNT07, ABCNT08 and ABCNT09. All are Pentium II – 400MHz machines, with 128M of memory and 4G available disk space (after software installation). They are all interconnected by the ABC 100Mbps switch. All machines were part of the ABCNT domain, which has ABNCT01 as domain controller.

The machines are classified according with its role in the experiment. In the Client/Server experiments, ABCNT05 is the client; ACNT09 and ABNCT08 are the database servers (Figure 4.1). In the N-Tier experiments, ABCNT05 is the client; ABCNT09 and ABCNT08 are database servers; and ABNCT06 and ABCNT07 are application servers (Figure 4.2).

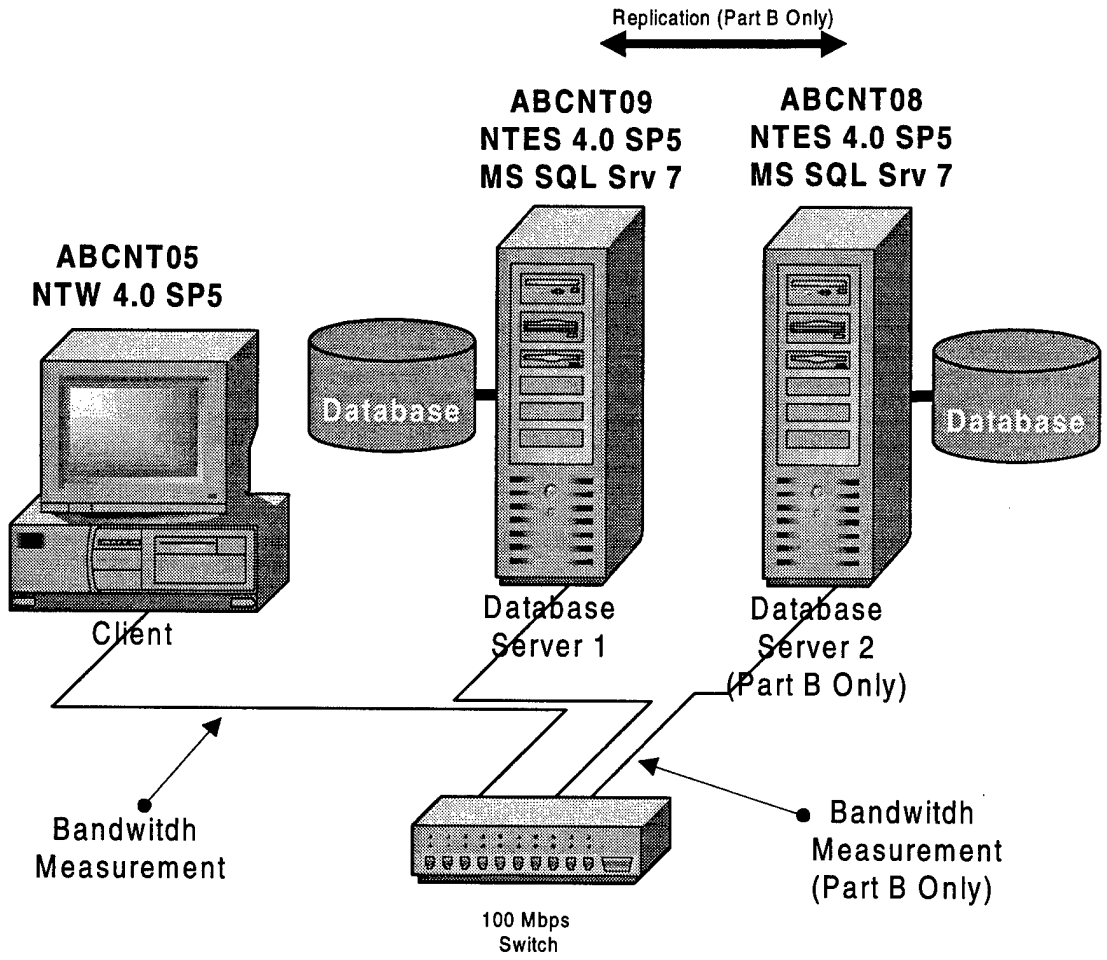

Figure 4.1 – Client/Server Model Layout

The OS of all database and application servers is Windows NT Enterprise version 4.0, with Service Pack 5. All database servers have MS SQL Server 7 Enterprise version and also MSMQ and MTS 2.0 (from Option Pack 4.0). The Client machine has Windows NT Workstation 4.0, SP5, and administrative tools for SQL Server 7, MSMQ and MTS. In the N-Tier experiments, the database servers have a MSMQ client installation and the MSMQ controller is located at the application servers.

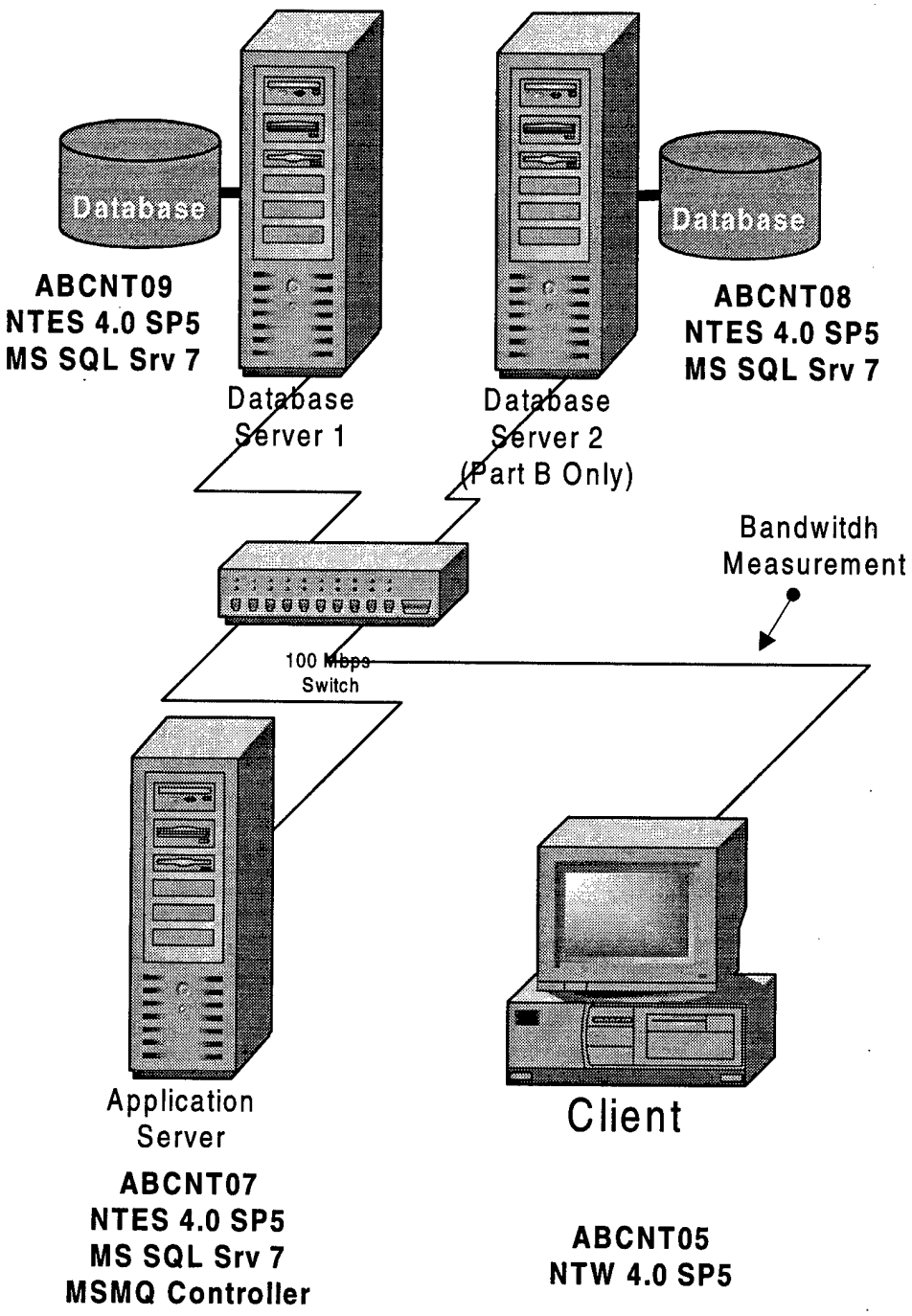

Figure 4.2 – N-Tier Model Layout

All database and application servers were also running Network Monitoring Tools from Windows NT, for bandwidth measurement. Although this

poses some communication overhead, it does not affect the comparison, since it equally impairs all servers during all experiments

4.3 Client/Server System Model Implementation

4.3.1 Database Implementation

In the Client/Server model, the database server has to execute most of the transaction processing tasks. This is accomplished by using triggers and stored procedures that are activated when some database action occurs, such as a table insert or delete. To implement the transactions specified in the TPC-C protocol, triggers were built and associated with appropriate events. All triggers and stored procedures used are listed in the Appendix B.

4.3.1.1 Replication

In the experiments of Part B, two similar databases, A and B, were constructed in different servers. The only difference between them is the warehouse identification, W_ID, which is "1" in database A and "2" in database B.

MS SQL Server 7 Enterprise version supports 3 types of replication: Snapshot, Transactional and Merge. Snapshot and Merge are the models described in chapter 2, with read-only and read-write replicas, respectively. The Transactional is a special type that provides one-way replication but integrated with a 2-phase commit protocol (see chapter 2).

Since the only TPC-C transaction tested in a multi-database environment was Transaction 1 (New Order), a replication scenario was installed to support it.

The difference between the standard and distributed version of Transaction 1 is that in the distributed version, an order can be filled with order items of a different supplier warehouse. Therefore, a new order could update stock tables in the local and remote databases.

Therefore, it's necessary for the databases to have access to the Warehouse, District and Stock tables of each other. Warehouse and District is a read-only replica, with low frequency of updates. Stock is read-write in both servers, with a high update frequency. To support this scenario, the following replication steps were implemented:

1. Warehouse and District were implemented as a Transactional replication type, with horizontal fragmentation based on the W_ID.
2. Stock was implemented as a Merge replication type, from ABCNT08 to ABCNT09. Replication conflicts were set to be stored for further resolution.

The replication chosen for Stock could let to replication conflicts, since two clients could update the same stock item at about the same time. In a real-world system, some complex conflict resolution procedure would have to be implemented to deal with this situation.

4.3.1.2 New Order Transaction

The new order transaction is a resource demanding transaction because it affects most of the tables in the database. When a user inserts a new order, the following actions occur: [37]

1) The chosen Warehouse Tax is retrieved;
2) The chosen District Tax and Next_Order_Id are retrieved;
3) The Customer matching the last name is selected, with the respective Discount and Credit Status;
4) A new record is inserted in the District_Order, table, with the O_ID matching the Next_Order_Id;
5) For each Order_Line:
    1) In Part B, randomly choose the supplier warehouse.
    2) The chosen Item Price is retrieved, the Amount calculated, and a new record is inserted in the Order_Line table;
    3) Update the Stock Quantity appropriately;
    4) Search the Item and the Stock information for a specific string;
6) Compute the total amount of the order and display all the order information;
7) A new entry is placed at the New_Order table;
8) Update the District Next_Order_Id.

An insert trigger in the Order_Line table performs the Stock, New_Order and District alterations. The client performs all order computations, by issuing multiple select statements to the database.

In the experiments of Part B, the products' W_ID is chosen to be the one of the remote server. Therefore, the customer is local but the products in the transaction are remote.

4.3.1.3 Payment Transaction

The payment transaction is a read-write transaction that has a medium resource demand. The steps of this transaction are:

1) The chosen Warehouse address and YTD is retrieved;
2) The chosen District address and YTD is retrieved;
3) The Customer is retrieved based on a last name search, the address, credit limit, YTD, credit status and discount are retrieved;
4) The Customer Balance is decreased by the Amount;
5) An entry in the History table is made;
6) If the Customer has a Bad Credit, a specific string is added to the Customer DATA;
7) The Transaction details are shown to the user.

A trigger in the History table does the Customer alteration. All other calculations are performed by the Front-End.

4.3.1.4 Order Status Transaction

This is a read-only transaction, having the following steps:

1) A Customer is selected by a last name search or by a random selected C_ID;
2) The most recent order is retrieved from the District_Order table;
3) The corresponding Order Lines are retrieved;
4) The results are shown to the user.

This transaction is executed by using multiple SQL statements, defined in the Front-End.

4.3.1.5 Delivery Transaction

This is a read-write transaction with medium demand of processing resources. It must be set up to executed in deferred mode, triggered by a user action. At least 90% of the transactions have to be completed in an 80 second interval. The steps in this transaction are:

1) The user starts the transaction by issuing a specific command specifying the Warehouse and the Carrier;
2) In deferred mode, the following actions are executed:
    1) For the chosen warehouse, for each of the 10 districts, chose the oldest order placed by searching the New_Order table;
    2) The entry in New_Order is deleted;
    3) The respective Order is updated with the Carrier;
    4) The corresponding Order Lines have the Delivery date updated;
    5) The respective Customer has the Balance and Delivery Counter updated.

This transaction is implemented by using a stored procedure. This procedure is triggered by a database alert that happens when a record is inserted in the Scheduled_Jobs, a table constructed specifically for this purpose. Since this is an alert, not a trigger, the execution occurs after some seconds and the control

returns immediately to the user after the insertion. The results of the Store Procedure executed are stored in the Excuted_Jobs, a table built to store this data.

4.3.1.6 Stock Level Transaction

This is a read-only, processing intensive transaction that scans the stock table for items that are below a specified threshold. The steps are:

1) Chose a random Warehouse, District and Threshold (between 10 and 20);
2) Retrieve the last 20 Orders for the given District, and for each Order Line, check if the item stock level is below the chosen threshold;
3) Display the results.

Although this transaction is implemented basically with a single SQL Statement, it takes a reasonable amount of time to execute because it has to scan about 200 entries in the Stock table.

4.3.2 Front-End Implementation

The front end is implemented in the Visual Basic 6.0, using a standard Win32 Exe file. The transactions were implemented as subroutines and the listings are shown in the Appendix C.

The client user interface for transaction testing purposes is shown in Figure 4.1. Each button can trigger one specific transaction, and the time used to execute that transaction is shown in the screen after the transaction results.

The results were displayed using the terminal layout according to the TPC-C specification. The transaction output can be disabled or redirected to a file.

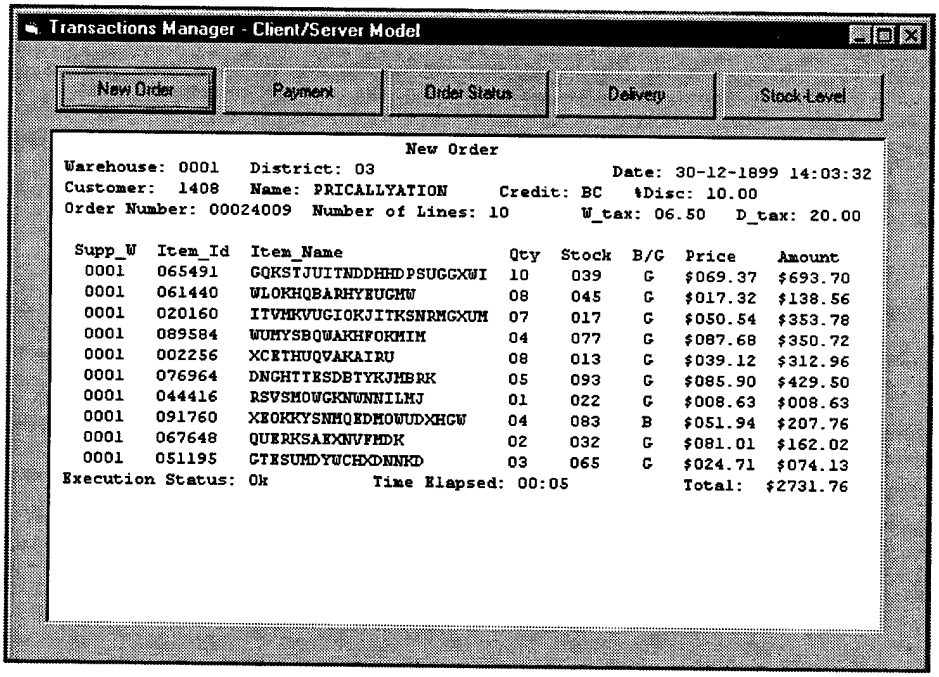

Figure 4.3 – Testing Client/Server Model User Interface

The database connections used in the program were the Microsoft ActiveX Data Objects (ADO) [68]. ADO is a library of COM objects that provides access to any database supporting OLEDB or ODBC standards. Although ADO calls are not as fast as the one from a pure RDBMS driver, it was chosen because it provides the easiest programming environment and it is well integrated with Visual Basic and Visual C++.

4.4 N-tier System Model Implementation

4.4.1 Database Implementation

In the N-Tier model, the database server has no participation in transaction processing tasks. It only acts as persistent data storage for objects. Therefore, the database was implemented with only the definitions of the data structures (tables and indexes) from the TPC-C specifications.

In Part B, no replication was set up for this model. Multi-database transactions used objects located at different application servers, each accessing its own database server. In a real-world scenario, some type of read-only replication could be used to allow faster query execution.

4.4.2 Data-Tier Implementation

The data-tier objects are responsible for communicating with the database, storing and retrieving data as appropriate. The objects described in chapter 3 were implemented using Visual Basic in a single Din-process COM module, **ThesisDO.DLL**.

All data objects share a single database connection, implemented using ADO. The database connection is only open during storing or retrieving operations. After the data retrieval, the object manipulates the data using ADO disconnected recordsets.

There exist two versions of ThesisDO.DLL, one for out-of-process COM servers (internally called ThesisDO) and one for MTS servers (called Thesis-

DOMTS). The first one doesn't have implicit transaction handling, therefore in ThesisDO an external ADO database connection is used to handle database transactions; this connection is passed to the data objects for use in the data update.

On the other hand, ThesisDOMTS doesn't have to deal with database transactions, since MTS is responsible for handling it. Therefore, in this version, no ADO connection is necessary during data storage or retrieval operations. Figure 4.2 shows the complete set of modules developed and the listings are shown in Appendix C.

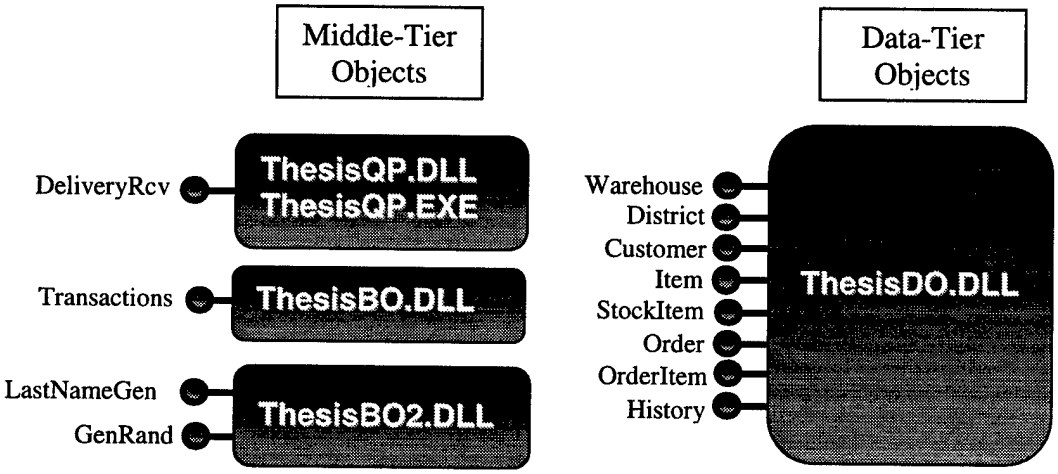

Figure 4.4 – Testing Client/Server Model User Interface

4.4.3 Middle-Tier Implementation

All transactions in the N-tier model are implemented using middle-tier objects. Each transaction is considered to be a "Business Logic" that is activated by the client process.

4.4.3.1 New Order Transaction

This transaction executes the same actions described in the Client/Server implementation. In the N-Tier implementation, the following processing steps are taken:

1) The Warehouse object is instantiated with the appropriate W_ID;
2) The chosen District object is instantiated;
3) A customer object matching the last name is instantiated;
4) A new object Order is created, using the District and Customer objects as parameters;
5) Several Order_Line objects are created and appended in the Order object. For each Order_Line:
    5) A specified Item object is instantiated;
    6) The correspondent Stock object is instantiated and its quantity changed appropriately;
6) The new Order object is saved.
7) Display all the order information.

When the object is saved, it executes all the underlying functions described in the TPC-C specification, such as inserting a record in the New_Order table, altering the Warehouse's Stock, etc.

4.4.3.2 Payment Transaction

The steps of this transaction in the N-Tier model are:

1) The chosen Warehouse object is instantiated;

2) The chosen District object is instantiated;

3) The Customer object is instantiated based on the provided last name;

4) The Customer Balance is decreased by the Amount;

5) The Transaction details are shown to the user.

The participating objects are responsible for making the appropriate changes, such as inserting the record in the History table and altering the Customer properties, according to the TPC-C specification.

4.4.3.3 Order Status Transaction

The steps performed are:

1) A Customer object is instantiated by using the last name or C_ID provided;

2) The most recent Order object is instantiated from a specified District_Order object;

3) The results are shown to the user.

4.4.3.4 Delivery Transaction

Since this must be executed in deferred mode, it is implemented in the N-tier model as an asynchronous function call. MSMQ is used to store the customer request, and a middle tier object is responsible for reading from the queue and processing the transaction. The steps in this transaction are:

1) The user starts the transaction by sending a message to the appropriate MSMQ queue;

2) The middle-tier object receives the message and executes the transaction as described in the Client/Server model;

4.4.3.5 Stock Level Transaction

The steps in the N-Tier model are:

1) Instantiate a random Warehouse, District objects and choose a Threshold value between 10 and 20;
2) Instantiate the last 20 Orders for the given District, and for each Order Line, check if the item stock level is below the chosen threshold;
3) Display the results.

4.4.3.6 Modules

A single object, called *Transactions*, implements all above transactions. This object is implemented inside a single module, **ThesisBO.DLL** or **ThesisBO.EXE**. The executable is an out-of-process COM component to be executed in remote machines. There are two versions of the DLL, one for use in the client process and the other for use with MTS.

The in-process and the out-of-process versions are exactly the same. The difference in the MTS version is the way the object handles transactions: in the MTS version, the MTS Context object controls transactions, while in the other versions, transactions use the standard *BeginTrans* and *CommitTrans* database methods.

There are other two modules defined in the middle-tier. **ThesisBO2.exe** contains the code responsible for generating random numbers and last names, according to the TPC-C specification. **ThesisQP.DLL** (with its out-of-process equivalent, **ThesisQP.EXE**) is responsible for receiving the request from the MSMQ and executing the TPC-C Transaction 4. Figure 4.2 shows the middle-tier modules and the listings are shown in Appendix C.

4.4.4 Front-End Implementation

The front end is implemented in the Visual Basic 6.0, using a standard Win32 Exe file. The front end executes the transactions by issuing the appropriate calls to the middle-tier object *Transactions*. Listings are shown in the Appendix C.

4.5 Experiments Implementation and Measurements

The machines ABCNT05 to ABCNT09 are used to implement the experiments described in Chapter 3. For all topologies, the number of experiments are calculated using the formula $n = (100zs / rx)^2$. Some pilot experiments were executed to find x, s and r. From this data, to achieve at least 95% precision in within about 1% of the mean value, a minimum of 300 experiments is necessary. Although this is the worst case (for most experiments, a lower number would suffice), this number of experiments is used in all topologies and all cases.

A special program, developed for this purpose, executes and gathers the execution time for all experiments, since it is necessary to run 300x(5 topologies x 5 transactions + 2 Part B) = 6,600 experiments. The program automates the task of starting the experiment, executing it and recording the data. It also automatically stores the data in an Excel spreadsheet, filling in the appropriate locations. The spreadsheet is automated to calculate and show standard deviations, variance and comparison charts. The code for this driver program is listed in Appendix C.

MS Windows NT Performance Monitor is used to measure the network bandwidth. Two services have to be installed in the NT Server to measure network utilization: SMP Service and Network Monitoring Tools. To gather the data, the Performance Monitor is conFigured to run in Log mode, recording all data related to the Network Segment. This data is then converted to a chart view and exported to an Excel Spreadsheet.

The execution of all experiments takes about 5 hours to execute. Some user interaction is necessary when changing from one topology to another. Part B was not executed in the same batch due the required database configuration.

Also, the location of the middle-tier objects has to be changed from one topology to another. This is accomplished by using the MTS administrator and the DCOM configuration utility (dcomcnfg). The experiment driver program interrupted the experiment batch and prompted the user to manually execute these configurations when switching from one topology to another.

4.6 Summary

This chapter addresses the implementation of both Client/Server and N-tier models. The software and hardware platform used are detailed and the required software configuration listed. This chapter also provides details about the modules of the Client/Server system and the layers of the N-Tier implementation.

The database implementation used to host these models and the TPC-C Transactions and the replication mechanisms used to simulate a distributed-database environment are also covered.

The last sections of this chapter explain the experiments implementation and the tools used to collect and store the data.

V. Data Analysis

5.1 Introduction

This chapter describes the results of the experiments executed, according to the design and implementation seen in the previous chapters, in sections 3.4, 3.5 and 4.2 to 4.5. The general partial results of each transaction are analyzed, comparing the Client/Server and N-tier models in terms of execution time and network utilization.

In the final section, a general analysis of the results is performed, discussing the advantages and disadvantages of each model in executing TPC-C Transactions, and in real-world applications in general.

5.2 Collected Data Analysis

As explained in chapter 4, all execution time data is consolidated in an Excel spreadsheet. The summary or execution times for Part A and B are shown in Table 5.1 and Table 5.2 respectively.

The bandwidth data collected are also consolidated in an Excel spreadsheet and the resulting charts are show in the following sections. All execution-time charts show error bars corresponding to the standard deviation of the experiments.

TPC-C Transaction	Item	Client/ Server	N-Tier Client	N-tier Server	N-Tier App Server	N-Tier MTS Server
1	Iterations:	300	300	300	300	300
	Mean (sec):	0.68	0.73	0.672	0.678	2.328
	Variance:	0.03119	0.00126	0.00316	0.00126	0.08745
	Std Deviation:	0.17662	0.03554	0.05621	0.03548	0.29573
	Precision (95%):	0.01999	0.00402	0.00636	0.00401	0.11234
2	Iterations:	300	300	300	300	300
	Mean (sec):	0.156	0.175	0.161	0.165	0.704
	Variance:	0.0009	0.00244	0.00233	0.002	0.00741
	Std Deviation:	0.02995	0.04941	0.04825	0.04475	0.0861
	Precision (95%):	0.00339	0.05246	0.06085	0.04896	0.0524
3	Iterations:	300	300	300	300	300
	Mean (sec):	0.108	0.149	0.125	0.130	1.172
	Variance:	0.00062	0.00156	0.00175	0.00168	0.06274
	Std Deviation:	0.02499	0.03951	0.04178	0.04095	0.25049
	Precision (95%):	0.00283	0.00447	0.00473	0.00463	0.02834
4	Iterations:	300	300	300	300	300
	Mean (sec):	0.040	0.0055	0.0046	0.0050	0.0065
	Variance:	0.00075	2.2E-05	3.1E-08	6.3E-07	4.9E-05
	Std Deviation:	0.02737	0.00464	0.00017	0.00079	0.00703
	Precision (95%):	0.0031	0.00053	2E-05	9E-05	0.0008
5	Iterations:	300	300	300	300	300
	Mean (sec):	0.032	0.035	0.031	0.033	0.261
	Variance:	0.00024	0.00035	0.00074	0.00024	0.00695
	Std Deviation:	0.01539	0.01871	0.02715	0.01548	0.08335
	Precision (95%):	0.00174	0.00212	0.00307	0.00175	0.00943

Table 5.1 – Results of Part A - Execution Times

TPC-C Transaction	Item	Client/ Server	N-Tier MTS Server
1	Iterations:	300	300
	Mean (sec):	0.67	1.453
	Variance:	0.047	0.00785
	Std Deviation:	0.2167	0.08863
	Precision (95%):	0.0245	0.10114

Table 5.2 – Results of Part B - Execution Times

5.2.1 Data Analysis

5.2.1.1 Part A - Transaction 1 – New Order

This is the most demanding TPC-C Transaction in terms of processing resources. Therefore, its execution time is the highest among all five transactions. The execution times in the Client/Server and in the four N-Tier topologies are shown in Figure 5.1. The used bandwidth is shown in Figure 5.2.

The Client/Server implementation of this transaction is very different from the N-Tier one. In the Client/Server, processing is about evenly distributed between Client and Server. But, since the client has to execute several SQL statements, the network bandwidth used is high, with a mean value of 0.6% - although this is a low number, it represents a high utilization, relatively speaking, since 0.6% in a 100 Mbps network is equal to 600 Kbps, more than what's available in most WANS.

On the other hand, the N-Tier implementation has all the processing performed by the middle Tier. Therefore, the segment between client and middle tier requires less network bandwidth than the segment between the Middle-Tier and the Data-Tier.

From Figure 5.1, it's possible to conclude that there is not much variation in execution times among the Client/Server and N-Tier topologies 1 to 3. Among those, N-Tier topology 1 has a slight worse execution time, of 0.73 sec. per transaction, compared to the 0.68 sec. per transaction in the Client/Server model. This is explained by the overhead of having all layers in the Client-Machine.

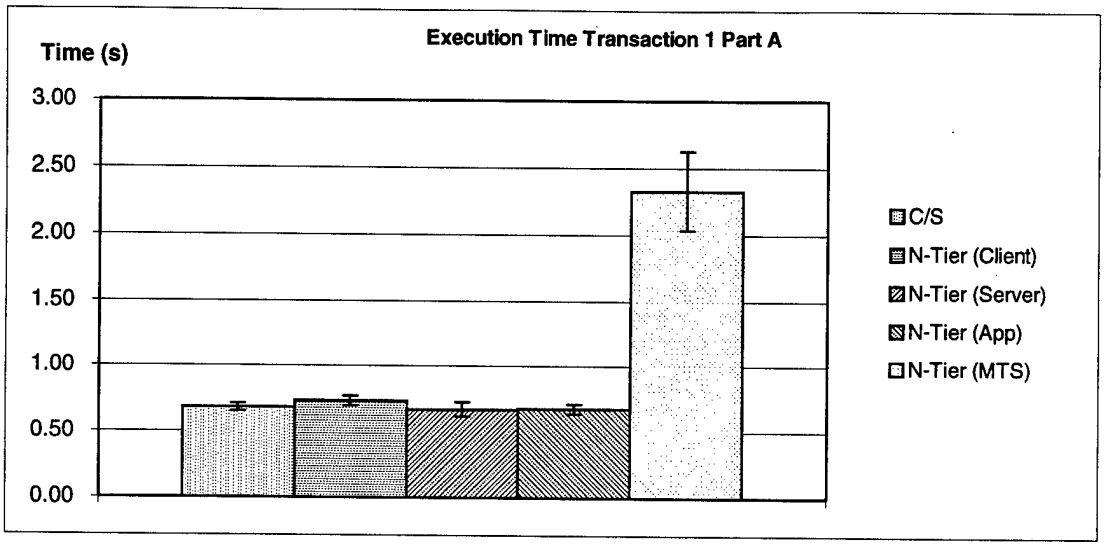

Figure 5.1 – Transaction 1 (Part A) Execution Times

N-Tier Topology 2 has the best execution time, 0.67 sec. (although within the standard deviation of the Client/Server model and Topology 3), which is explained by little communication overhead (all layers are located at the server) of this implementation. This result could only happen in a lightly loaded server; if processing resources start to become an issue, this topology could potentially start to have worse results.

The N-tier Topology 4 has worst execution time, of 2.34 sec. per transaction. This is almost 4 times the execution time of the Client/Server version. This is caused by the overhead of using MTS. Since MTS has to handle generic, multi-database transactions, it cannot be as efficient as a transaction executed with native SQL Server 7 drivers. Although this was the worst topology in terms of execution time, it was the one that used the least network bandwidth, with mean of 0.01% (10 Kbps).

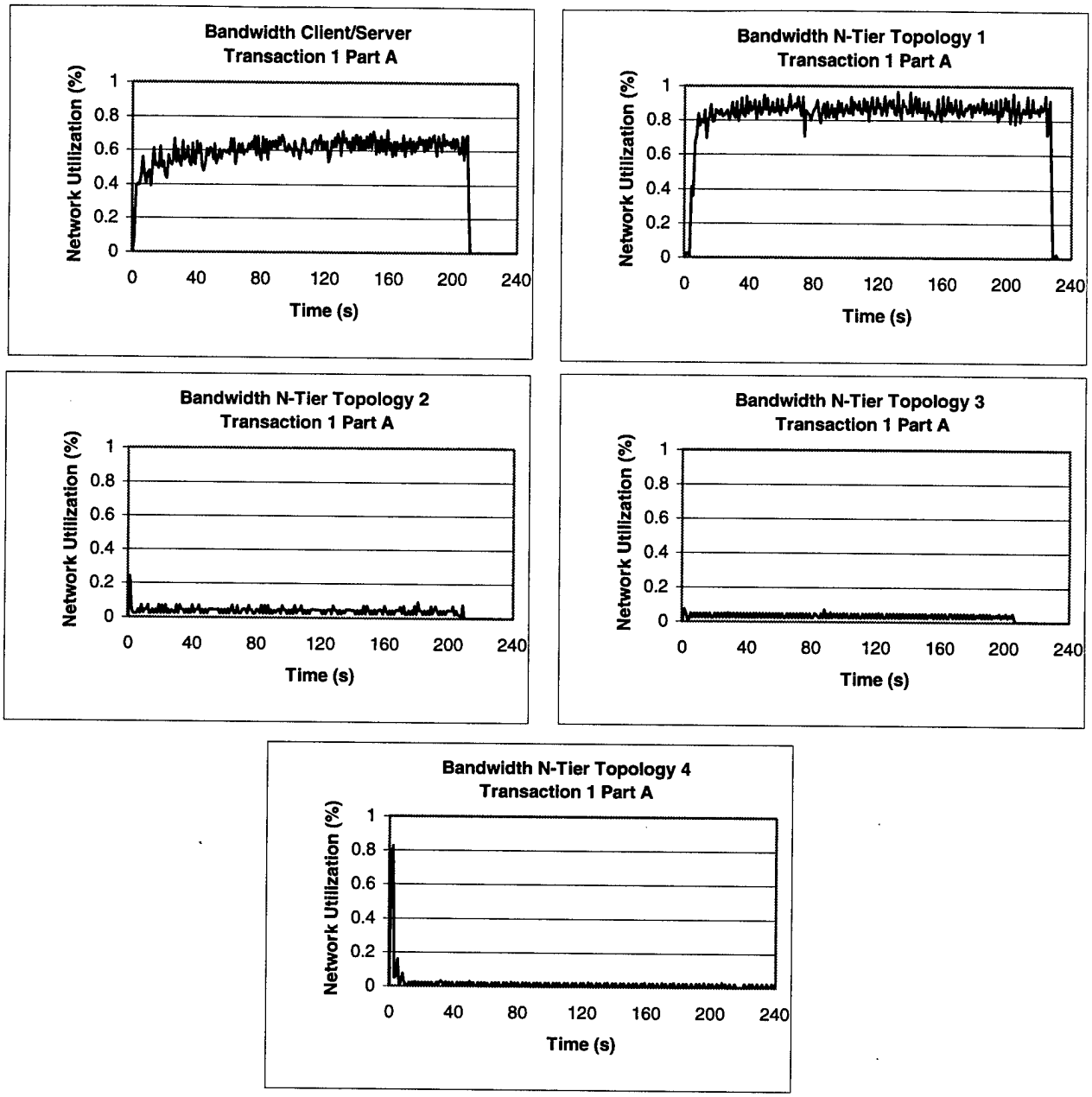

Figure 5.2 – Transaction 1 (Part A) Bandwidth Utilization

As shown in Figure 5.2, the topologies 2, 3 and 4 have a network utilization of about 10 times lower than the Client/Server version. This is explained by the position of the middle-tier. Since it is not in the client, all the heavy database

communication occurs among the servers, the client only issues requests and receives results.

The spike in the network utilization (about 1% utilization) in the beginning of the program in Topology 4 is caused by the first activation of MTS objects. Posterior activations don't cause this behavior, since MTS caches the used objects.

In terms of cost/benefit, the best architecture for this particular transaction is Topology 3. It has the second best execution time, 0.678 s, second best bandwidth utilization (0.14%) and has no scalability impacts on the database server because the processing is done in the application server. But 0.14% utilization (140 Kbps) is still a high value for most WANs. In environments such as the Internet, only Topology 4 is viable.

5.2.1.2 Part A - Transaction 2 – Payment

This is the second most demanding TPC-C Transaction in terms of processing resources. The execution times in the Client/Server and in the four N-Tier topologies are shown in Figure 5.3. The used bandwidth is shown in Figure 5.4.

The results for this Transaction are very similar to the ones of the Transaction 1. In this Transaction, the Client/Server version has the best execution time, 0.16 sec. per transaction, but within the standard deviation of the 3 first N-Tier topologies.

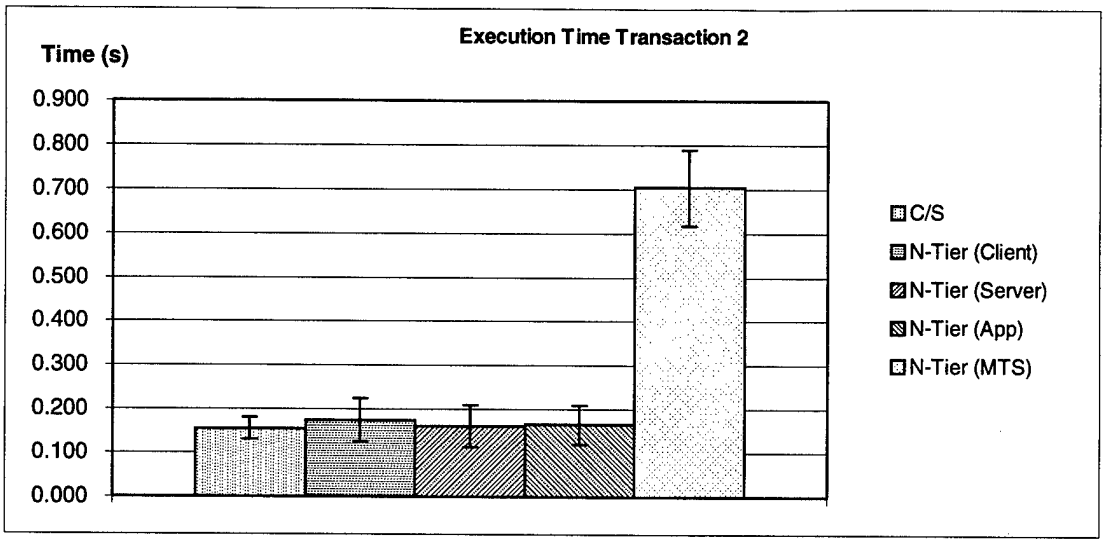

Figure 5.3 – Transaction 2 Execution Times

For the same reasons explained in Transaction 1, the worst execution time is the N-Tier Topology 4, 0.70 sec. per transaction, with more than 4 times the execution time of the Client/Server version. Again, Topology 4 was the one that uses least bandwidth, mean of 0.03% (30 Kbps), while the Client/Server version has a mean of 0.84% (840 Kbps).

In this experiment, the spike in the beginning of the execution of N-tier Topology 4 is less significant than in transaction 1, only about 0.3%. This is because this transaction is simpler, requiring fewer object instantiations. Comparing N-tier topologies 3 and 4, it is also noticeable that the former uses about two times more network bandwidth, although both use an application server to host the middle-tier. MTS object caching is reason of this behavior.

The best topology for this transaction, in terms of cost/benefit, is again Topology 3. It presents the third best result, 0.165 sec. per transaction, and the second best network utilization, 0.1% (100 Kbps).

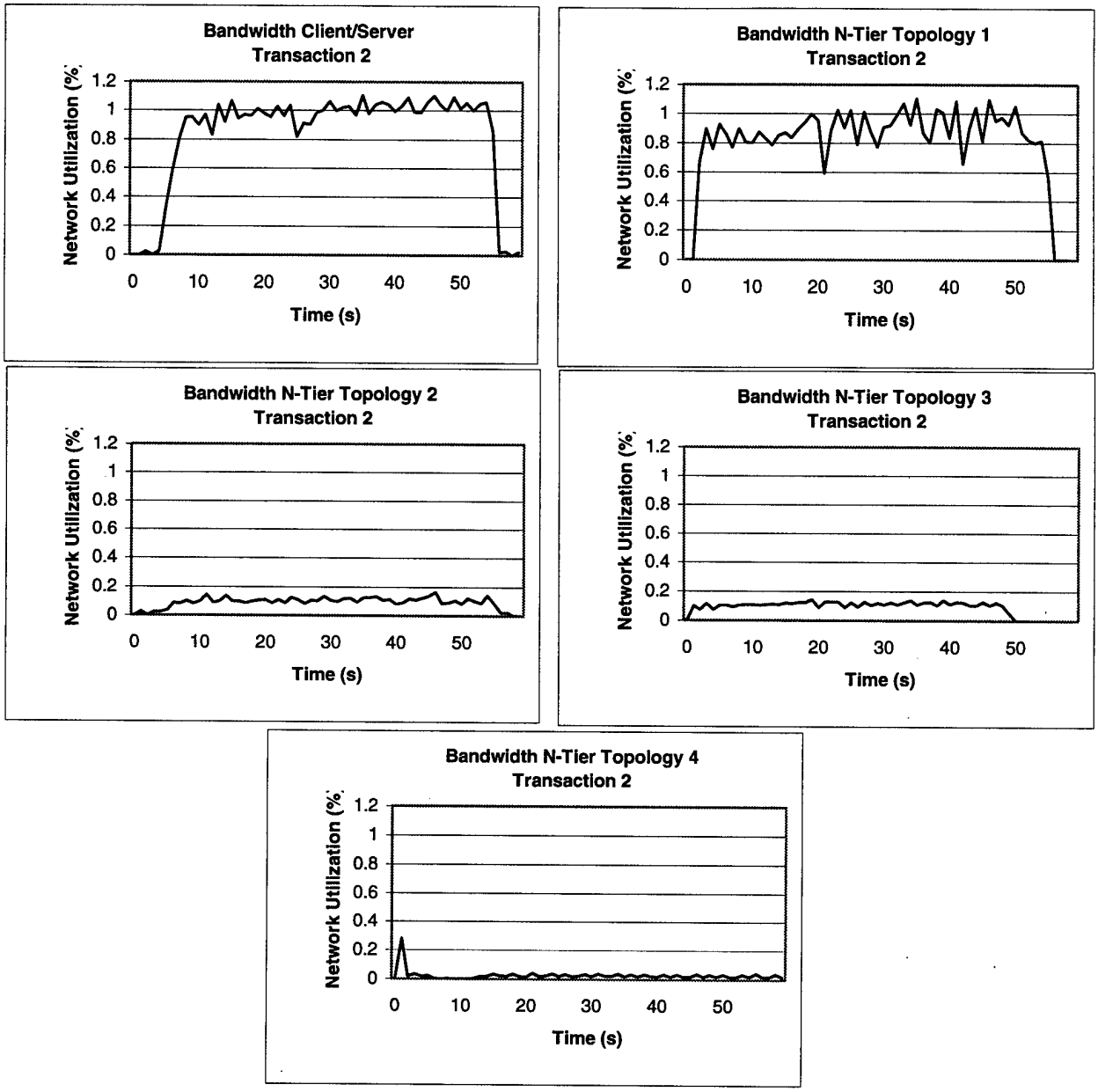

Figure 5.4 – Transaction 2 Bandwidth Utilization

5.2.1.3 Part A - Transaction 3 – Order Status

This is a light transaction in terms of processing resources. It basically sends a series of SQL Statements to the database. The execution times in the Cli-

ent/Server and in the four N-Tier topologies are shown in Figure 5.5. Bandwidth use is shown in Figure 5.6.

Again the same execution times pattern of the previous transactions happens in this one. The Client/Server and the first 3 N-Tier topologies have very similar execution times, with the Client/Server version slightly faster, 0.108 sec. per transaction.

But N-Tier Topology 4 did not present the same behavior of the other transactions. Client/Server and N-Tier topologies 1 to 3 shows improved execution times compared to the previous executions which is expected; since the transaction is not actually updated data, the data is simple read from the database.

But, as can be seen in Figure 5.5, the N-Tier Topology 4 execution time increased, compared to the previous transaction, to 1.17 sec. per transaction. This behavior is explained by the characteristics of MTS and the way the *Transactions* object is implemented (see section 4.4.3).

The *Transactions* object is a single DCOM component, which responsible for executing all 5 transactions. Since there are some transactions that require database updates, this component was registered in MTS as a component that "requires transaction". Therefore, MTS starts a transaction every time a method is executed, even if the transaction is a simple database read, as happens in this case. This is why N-Tier Topology 4 has the highest execution time; it is executing a database transaction while the other versions are not.

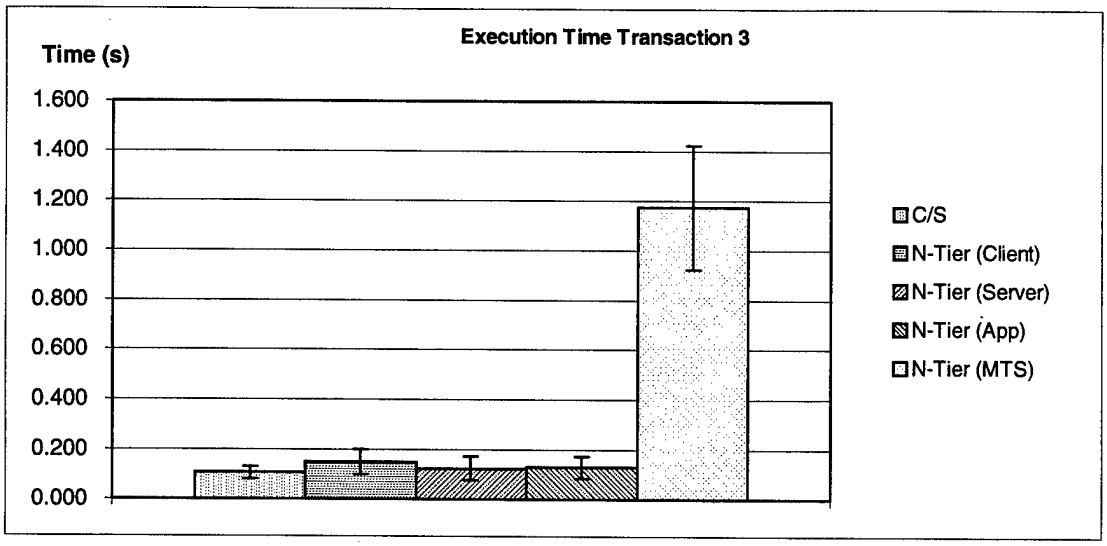

Figure 5.5 – Transaction 3 Execution Times

An analysis of network bandwidth utilization shows that Client/Server uses 0.48% bandwidth, Topology 1, 1.25% and Topology 2, 0.13%. And since this transaction requires extensive data reading, the overhead of using objects instead plain SQL statements is more easily noticeable. Comparing the Client/Server against N-Tier topology 1 shows that the later requires almost 3 times more network bandwidth. Again, MTS object caching cause low network utilization for this transaction, about 0.018%.

The better architecture for this transaction is again N-Tier topology 3, although the Client/Server version, with use of more efficient SQL Statements, can be also considered. In a real world scenario, where the data retrieved would be far more complex, a pure SQL solution is probably be the best solution for this type of transaction.

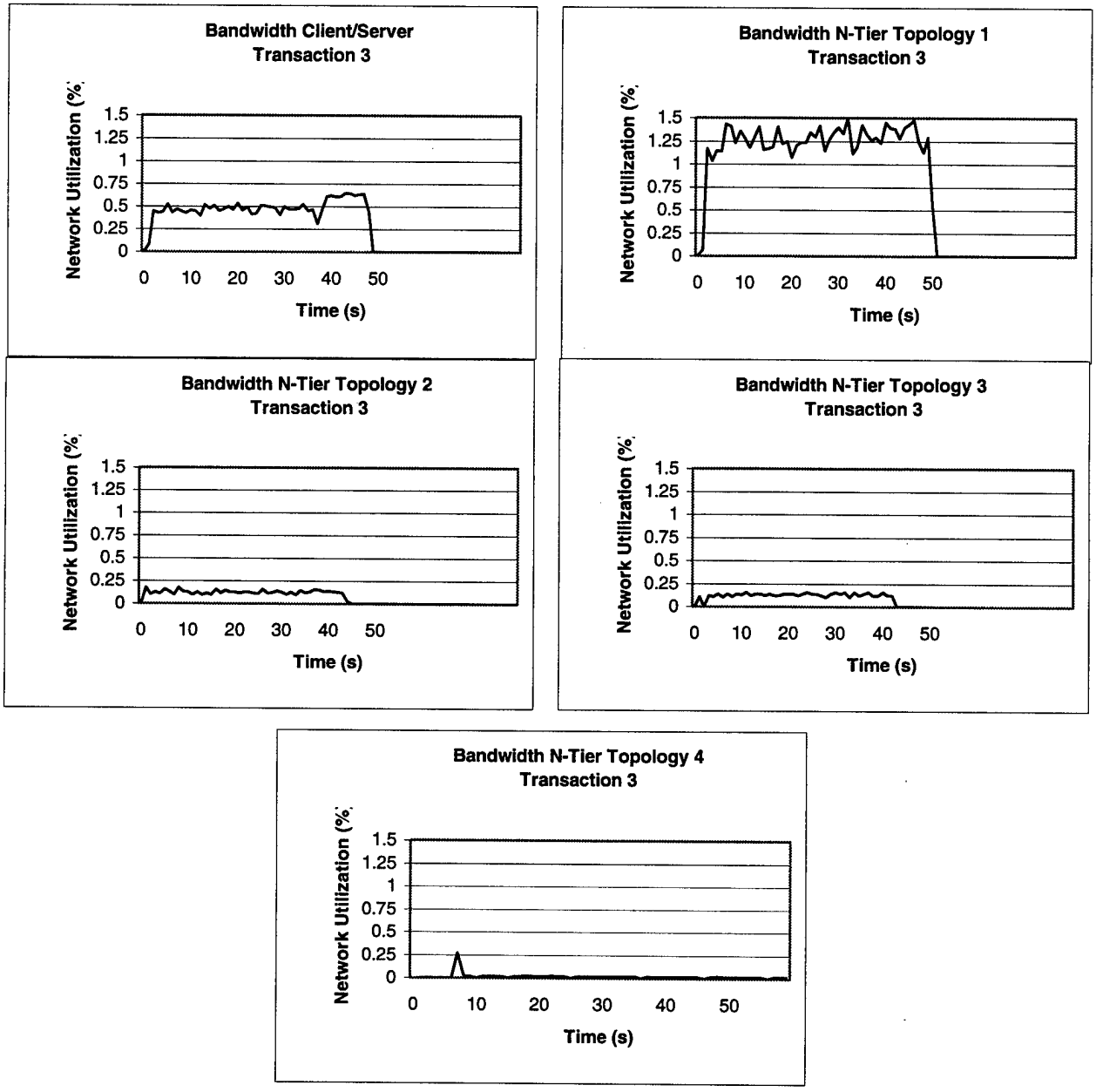

Figure 5.6 – Transaction 3 Bandwidth Utilization

5.2.1.4 Part A - Transaction 4 – Delivery

This is a deferred transaction, the request is queued and the some server component executes the process at a later time. Figure 5.7 shows the execution time and Figure 5.8 shows the bandwidth utilization.

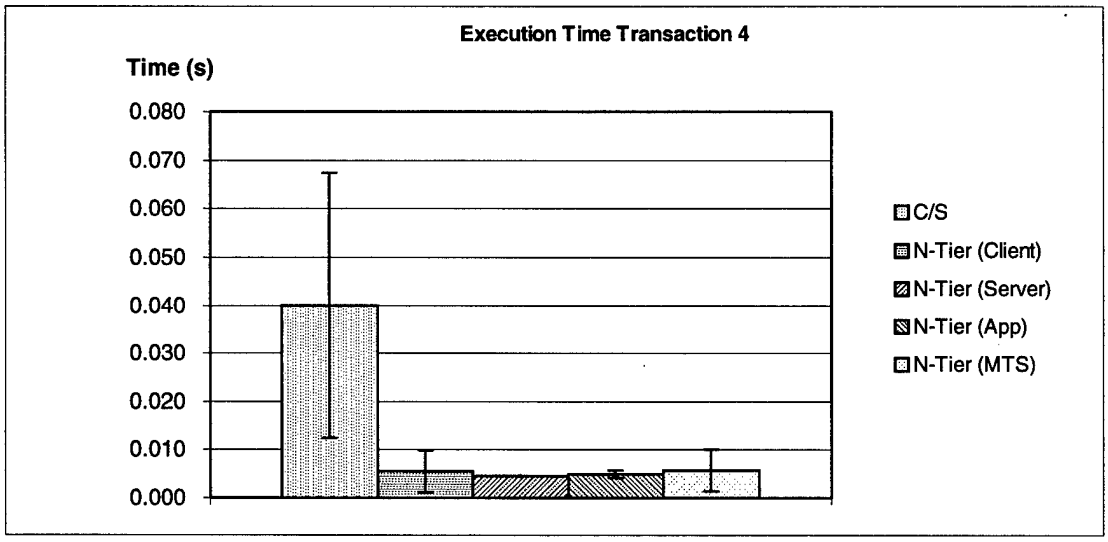

Figure 5.7 – Transaction 4 Execution Times

The implementation of the Client/Server version is very different from the N-Tier topologies for this transaction. In the Client/Server model, the request is logged in a table alert activates the server agent. In the N-Tier model, a MSMQ message is sent to a specific object in the middle tier, which in turn executes the transaction asynchronously. So, while the Client/Server model executes a table update, the N-Tier model just uses an asynchronous.

This difference of implementation justifies the time results. The Client/Server version takes 0.04 sec. per transaction, while all the N-tier topologies takes from 0.046 sec. to 0.065 s. The error bar in the Client/Server in Figure 5.7 is

very significant. This is caused by the server background execution of the agent code, which competes for the same processing resources as the transaction itself.

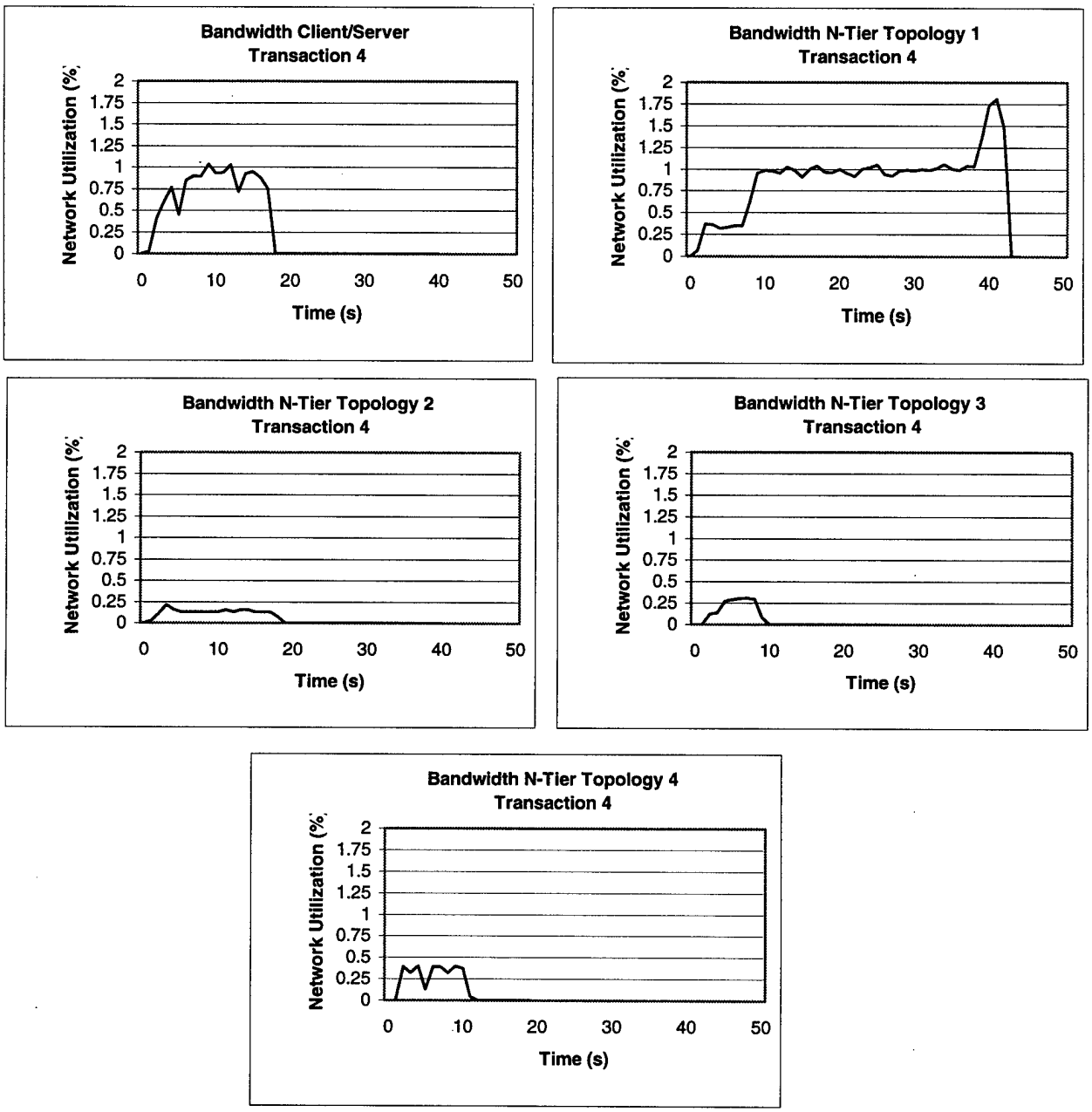

Figure 5.8 – Transaction 4 Bandwidth Utilization

In this transaction, the last 3 N-Tier Topologies have the best execution times, between 0.046 sec. and 0.065 sec. per transaction. This is expected since the use the basically the same code.

The analysis of the bandwidth utilization shows that an MSMQ message uses very little bandwidth. N-tier Topology 2 is the one that uses more bandwidth among all N-tier versions. This reason is that the middle-tier server that processes MSMQ messages is located at the client machine. Therefore, all messages are returned to the sender machine for processing. This explains the higher bandwidth usage, 0.89%, and why the network continues to be used long after the transactions are over (the 300 iterations takes less than 10 seconds).

The best choice for this transaction is Topology 3. It is the one has the best execution time, 0.046s per transaction, and least bandwidth utilization, 0.12% (120 Kbps).

5.2.1.5 Part A - Transaction 5 – Stock Level

This is a read-only transaction that executes a series of SQL Statements. The execution times in the Client/Server and in the four N-Tier topologies are shown in Figure 5.9. Bandwidth use is shown in Figure 5.10.

These results show basically the same behavior seen in Transactions 2 and 3: the Client/Server version is slightly faster than the N-Tier, and the MTS presented the worst performance, about 10 times the Client/Server execution time.

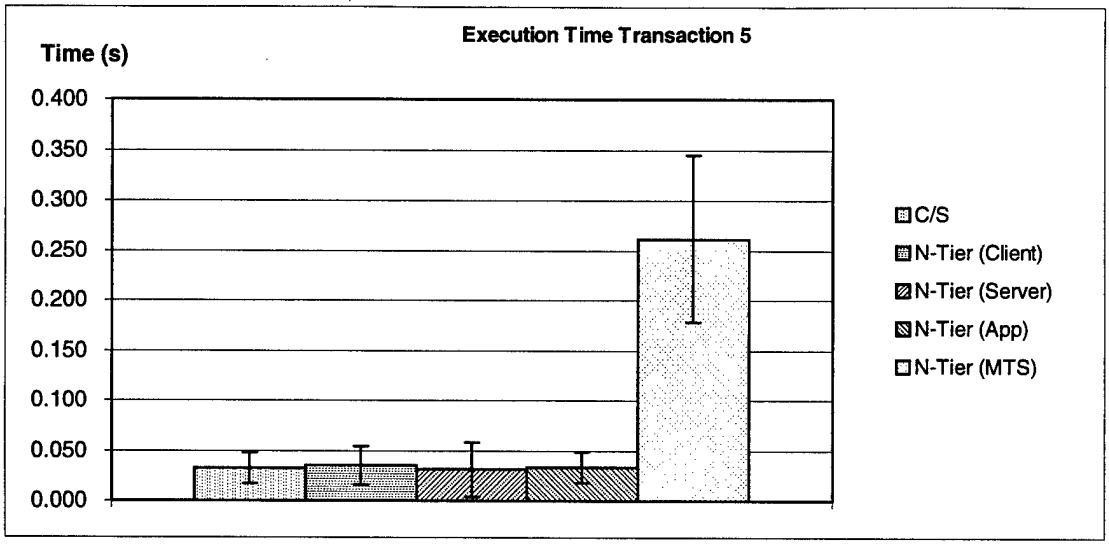

Figure 5.9– Transaction 5 Execution Times

Also in this case, MTS enforces the transaction, causing the N-Tier Topology 4 to perform poorly (0.261 sec. per transaction) compared to the others (about 0.03 sec. per transaction). The bandwidth utilization pattern is also very similar to Transaction 3. Again, the caching mechanisms of MTS make a very efficient use of the network bandwidth cause Topology 4 to use 0.026% while the other version use between 0.12 and 0.50% network bandwidth.

As in the previous transactions, the N-Tier topology 3 is the one that provides the best cost/benefit ration, with a good execution time, 0.033 sec. per transaction, and the second lowest network utilization, 0.128%.

5.2.1.6 Part B - Transaction 1 – New Order

This transaction access and updates records in two databases. It simulates a real world scenario of a distributed database application. The Client/Server version uses replication and two-phase commit protocol to update data and en-

sure data integrity. The N-tier uses the distributed transaction capabilities of MTS to coordinate the transactions.

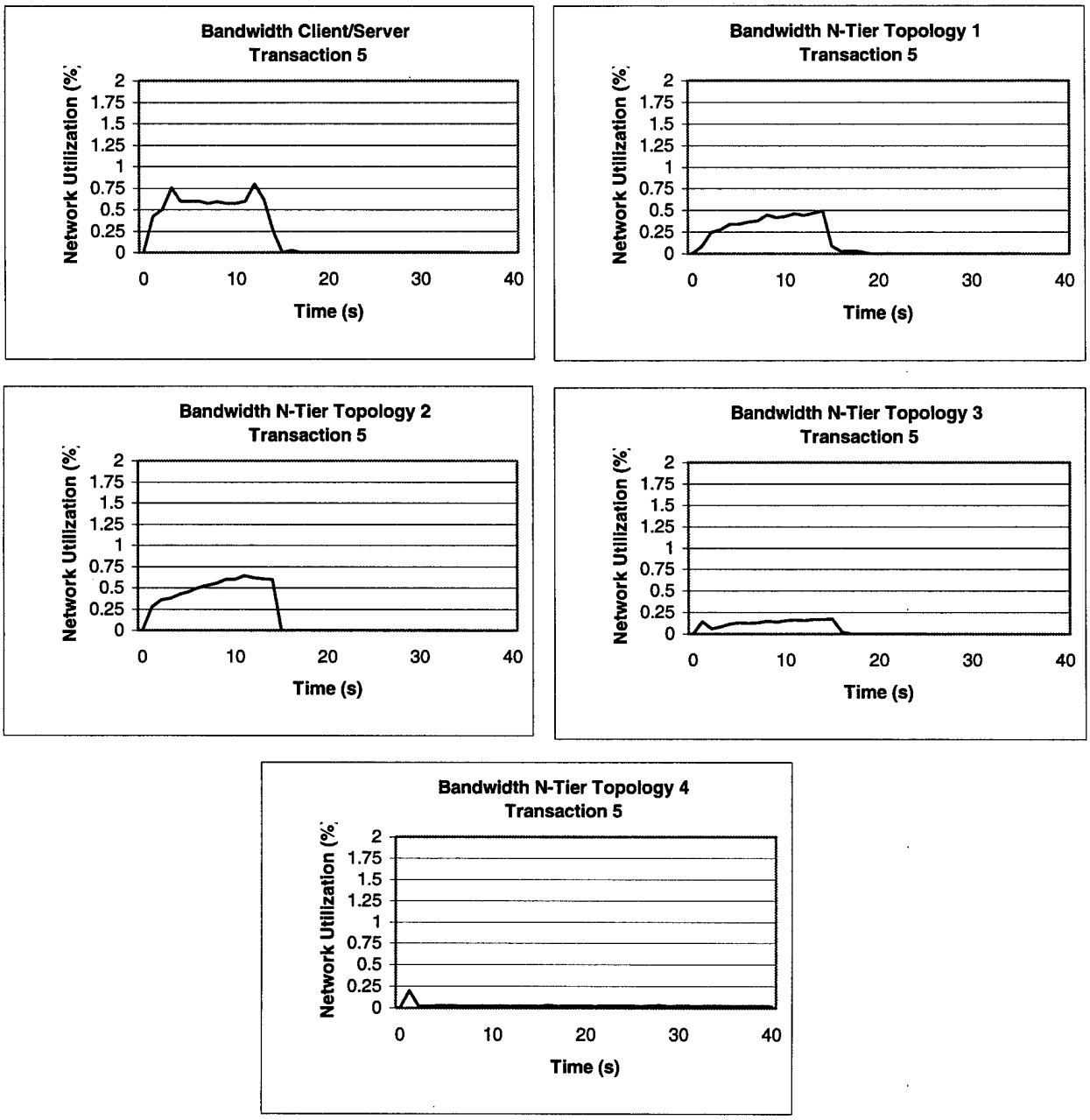

Figure 5.10 – Transaction 5 Bandwidth Utilization

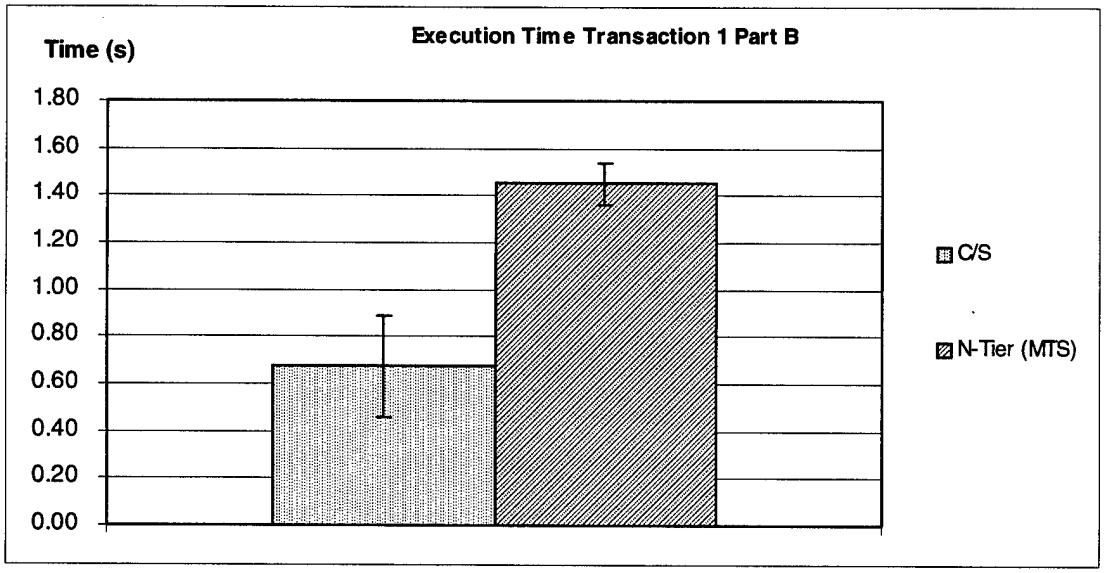

Figure 5.11 – Transaction 1 (Part B) Execution Times

Figure 5.11 shows the execution times of the Client/Server and N-Tier implementation. The Client/Server version has a clear advantage; with an execution time almost three times lower than the N-Tier version.

The execution times are 0.67 sec. in the Client/Server, and 1.453 sec. in the N-tier. This difference is explained by two factors: first, the MTS distributed transaction coordinator (MS-DTC) is not as efficient as the SQL Server 7 drivers, since MTS has to work with any generic database. Second, the SQL Server 7 has advantages in the way it implements the transaction in the distributed scenario. It first updates the local database, allowing the client to continue in its work, and then executes the transaction in the remote database. If some merge problem occurs, the SQL Server rollbacks the transaction and generates a replication merge conflict. This conflict has to be solved by the administrator or by a specialized

routine. In MTS, once the transaction is committed, it is guaranteed to be committed in all participating database.

This fact can also explain the network bandwidth utilization, shown in Figures 5.12 to 5.14. It can be seen in Figure 5.13 that the network continues to be used long after the client finishes its work. The chart shows the exact times the Replication Agent in the SQL Server process a batch of records, sending them to the remote database.

The charts also show that the network utilization of the N-Tier client, mean value of 0.02% (20 Kbps), is much lower compared to the Client/Server version, which has a mean value of 0.60%. As explained in the previous transactions, this is happens because most traffic occurs between the application server and the database server; the client only issues the command and receives the results.

It can be seen in Figure 5.12 that the N-Tier version uses 30 times less bandwidth than the Client/Server counterpart. When the system is activated, the usual object activation can reach up to 2% network utilization, but after that, it stays at about 0.02%.

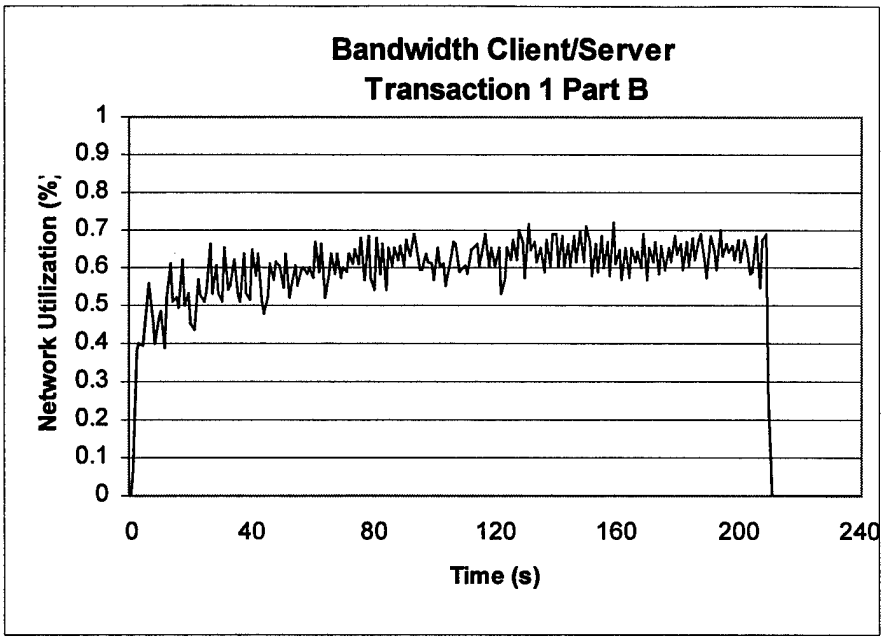

Figure 5.12 - C/S Front-End Bandwidth Utilization

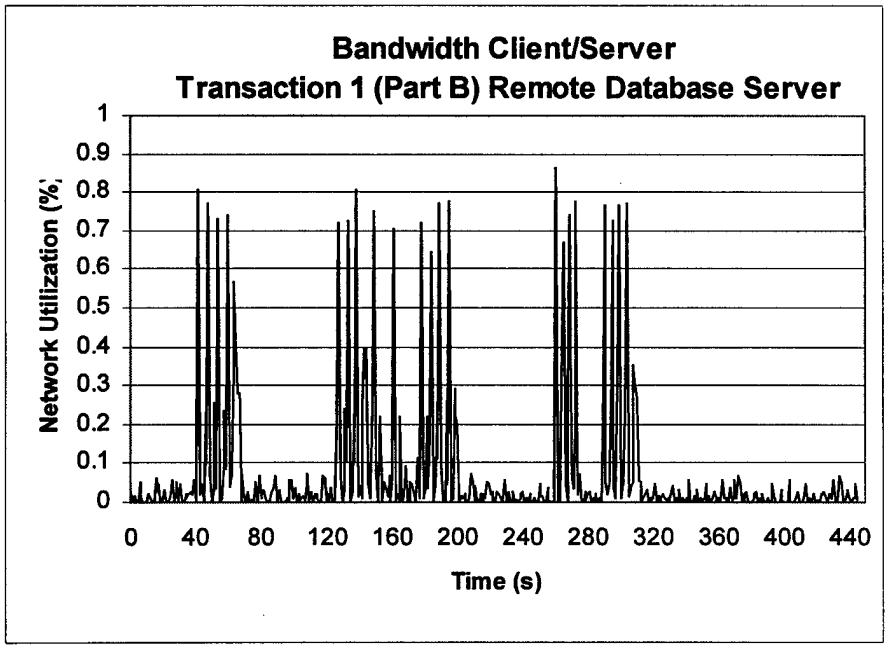

Figure 5.13 - C/S Remote Server Bandwidth Utilization

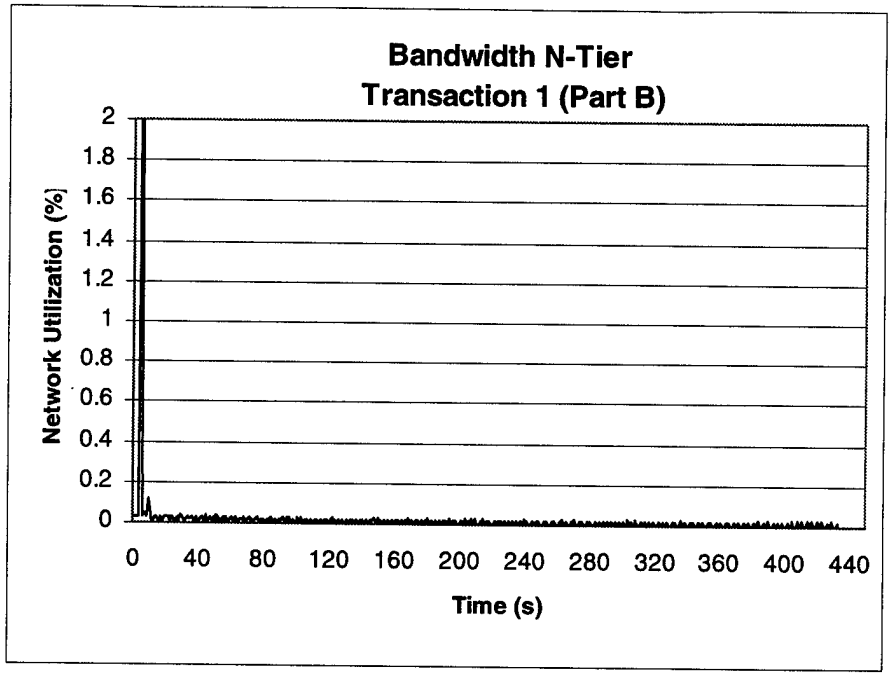

Figure 5.12 – N-Tier Front-End Bandwidth Utilization

In this transaction, the best model is not clearly defined, since one is faster but the other uses less network bandwidth. The best solution in a real world scenario would depend on the application requirements (speed vs. available network bandwidth).

5.3 General Analysis

From the collected data detailed in the previous sections, it is possible to derive some general conclusions about the models being compared.

First, the Client/Server model uses more bandwidth than the N-tier topologies in almost all scenarios (exceptions are small, read-only transactions, such as Transaction 3). When compared to MTS implementation, it uses much more bandwidth the N-tier versions, in all cases tested.

Second, the N-Tier topologies with middle-tier located apart from the front-end performs as well, or better in some cases, than the Client/Server model. The COM overhead is only noticeable in quick transactions, with low processing requirements.

Third, MTS poses a serious overhead in all tested cases. Its use of a generic distributed transaction coordinator causes it to perform poorly compared to transactions using native SQL Server drivers. The advantages are an easier to implement solution to distributed databases and more efficient network utilization, due to caching mechanisms. The replication capabilities of SQL Server do not provide the same functionality of the two-phase commit characteristics of MTS.

Fourth, MSMQ brings good benefits to asynchronous method calls. It poses no noticeable overhead and provides a very efficient use of network bandwidth, compared to Client/Server solutions.

Of course these findings apply to the specific transactions used in this research effort. But since TPC-C is based on real-world applications, these findings can be applied to other distributed database systems with reasonable confidence.

5.4 Efficiency Discussion

Although it would be interesting to compare the efficiency of the Client/Server and N-tier topologies, this analysis, with the measurement tools used in the research effort, is currently not possible.

In Client/Server or N-tier systems, a more efficient system is the one that can support a larger number of simultaneous clients in a single server (or servers). Therefore, the less processing resources an implementation uses, the more capable of supporting clients it will be (assuming that each new client uses the same amount of processing resources), and therefore, more efficient.

But to measure the processing utilization of a server in a lightly loaded environment such as the one in this research effort is an impractical task. The experiments described in this chapter do not cause the SGBDs, which have to support all clients, to use enough processing resources to make it possible to perform accurate measurements. Therefore, a processor utilization analysis would be not conclusive.

To solve this problem, it would be necessary to develop some specific measurement technique or to create a workload to cause a higher server utilization (using many simultaneous clients, for example). Both alternatives are beyond the scope of this research.

5.5 Summary

This chapter provides an analysis of the experimental data collected. Based on the resulting charts for each TPC-C Transaction, the Client/Server and N-Tier models are compared, and their advantages or disadvantages listed.

The last sections provide a general summary of the findings derived from the data analysis, about the Client/Server and the N-Tier topologies. A discussion of efficiency of the models being tested is also performed.

VI. Conclusions and Recommendations

The goal of this research effort is to analyze the use of Distributed Objects against the standard Client/Server model in Distributed databases. The implementation of the models and metrics chosen provided quantitative and qualitative results that serve as basis for a series of conclusions, meeting the detailed objectives described in section 3.2.

A general statistical finding is that the N-Tier model uses lower network bandwidth than standard Client/Server. This is due to the location of the business logic implementation – in application servers, instead of in the client or in the database server. In corporate systems that are based on low-speed communication links, such as the ones being developed in BAF, this implies that the distributed objects technology can provide better results than standard the Client/Server architecture.

The common conception that distributed-objects overhead causes N-Tier systems to perform worse than Client/Server systems was not validated: in all tested cases, the N-Tier topologies, without using MTS, presented execution times close to the Client/Server implementation in most cases, with better results in particular types of transactions. Even in the case where all N-Tier layers are located at the client, the execution times were mostly within the standard deviation of the ones from Client/Server implementation. This result implies that N-Tier technology could be used in most corporate database systems without seri-

ous performance penalties. The typical advantages of the N-tier objected-oriented development, such as code reuse, easier maintenance, and better abstraction, could justify the overhead encountered.

Another conclusion derived from the experimental data is that the MTS does pose a significant overhead. Its Distributed Transaction Coordinator causes the execution to be several times slower than the client/server or other N-tier topologies. This overhead should be weighted against the capabilities demonstrated by MTS, which is capable of handling transactions across multiple databases with no source code changes. This can be especially useful when dealing with high-volume, multi-database systems. To implement these functionalities without MTS would mean to write code to manage simultaneous two-phase commit transactions and specialized procedures to deal with replication conflicts. Another qualitative fact to be pondered is that the upcoming Windows 2000 incorporates MTS and the DCOM infrastructure at the operating system level, possibly diminishing the MTS overhead.

Although the use of pure DCOM instead of MTS provided execution times comparable to the Client/Server one, this solution is not as scalable, since objects are not shared among clients, and there are no MTS-provided database resource and object caching.

N-Tier systems were found to handle asynchronous and deferred transactions more efficiently than the Client/Server implementations, in terms of execution time and network utilization. The use of a message application server

such as MSMQ can bring benefits even to standard Client/Server systems. It poses no noticeable overhead and it uses less network bandwidth. In standard Client/Server systems, the only way to implement deferred transactions is by using database internal alerts and agents [47]. This implies developing specialized stored procedures and using complex configuration steps in each database server. MSMQ can be used to provide a single point of management with little coding.

A general disadvantage of the N-Tier model is the learning curve. In the DCOM framework, a developer has to be familiar with Windows NT, MTS, MSQM, DCOM configuration, database configuration and NT integrated security. The same is valid for any other distributed-object architecture, such as CORBA or EJB. In the client/server model, usually knowledge of the DBMS is sufficient for developing applications. This situation tends to gets worse as distributed objects vendors increment their framework with more layers of applications, making the environment even more complex.

Another disadvantage is the environmental configuration. To properly setup a DCOM infrastructure, one has to conFigure Windows NT (with is multiple Service Packs, installed in the proper order), SQL Server, Option Pack, MTS, MSMQ, DCOM security and client policies. This was found to be an overwhelming task; a considerable part of this research effort was spent learning the proper configuration and installation of these components. Again, these configu-

ration procedures are supposed to be easier in Windows 2000, since DCOM, MTS and MSQM will already be provided by the operating system.

As a final conclusion, both models were found to be an effective model for development corporate systems. The main advantage of Distributed Objects architecture is the more efficient use of the network bandwidth. Therefore, this model is recommended to be used in situations were network bandwidth is an issue. In other situations, several factors such as technical expertise, number and type of database servers, type of development methodology (objected-oriented or not), and scalability should be considered.

6.1 Future Directions

There are several lines of research that could follow this research work. An analysis of different Distributed Objects frameworks such as CORBA or EJB would be useful to compare performance issues and easy of use of the different solutions. A cross-platform study could also provide useful insights of if Distributed Technology could be successfully applied to mixed environments.

The use of Windows 2000 could also be considered as a research effort. Since the new OS will incorporate the entire DCOM framework, an analysis of performance and configuration issues could extend the conclusion of this work. Also, different implementations, using other database models, could be used for further investigation.

Although MSMQ is a relatively old product (was release more than 2 years ago), very few real world systems use it as a transactional component.

Transactional queues could provide better performance is some type of transactions by allowing than to execute asynchronously. An analysis of the situations where this technology could apply and the possible benefits from it could prove to be valuable.

Bibliography

1) Brazilian Air Force. Description of the Brazilian Air Force Information System. WWWeb, http://www.maer.mil.br. August 1998.

2) Guerra, A. & Silva, G. N. Feasibility Study on the Use of the Internet for Traffic of Unclassified Data. MS thesis, AFIT/GLM/LAL/98S-7. School of Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, 1998.

3) Microsoft Web Site. Microsoft Developer Network. WWWeb, http://www.microsoft.com/msdn.

4) Oracle Web Site. Oracle White Papers. WWWeb, http://www.oracle.com.

5) 15 Seconds. 15 Seconds Forums and Papers. WWWeb, http://www.15seconds.com.

6) Sybase Web Site. Sybase White Papers. WWWeb, http://www.sybase.com.

7) Dewire, D. Application Development for Distributed Environments. McGraw-Hill, 1994

8) Renaud, P. Introduction to Client/Server Systems: a practical guide for systems professionals. John Willey & Sons, 1993.

9) Smith, P. Client/Server Computing. SAMS Publishing, 1992.

10) Bell, D & Grimson, J. Distributed Database Systems. Addison-Wesley, 1992.

11) Silvio P. et all. Distributed Relational Database. Prentice-Hall, 1996.

12) Khoshafian, S. et all. Client/Server SQL Applications. Morgan-Kaufmann, 1992.

13) Vaugh, W. R. Hitchhicker's Guide to Visual Basic & SQL Server. 5th edition, Microsoft Press, 1997.

14) Thompson, Charles. Database Replication, DBMS Magazine, May 1997

15) Bobrowski, Steve. Implementing Data Replication, Oracle Magazine, May/June WWWeb, http://www.oramag.com/archives/36client.html

16) Fradkov, Sergey. Current Issues in Data Replication, WWWeb, http://www.unifx.com/article.html

17) Clegg, Peter. Avoid Data Distribution Pitfalls - Lantimes online, WWWeb http://www.wcmh.com/lantimes/archive/501b039a.html

18) Allan, Tony et al. Duplicate Data in a Distributed Document Database, WWWeb http://yallara.cs.rmit.edu.au/~junweic/link_cs445/Distributed/index.html#authors

19) Avital, Orly G & Avital, Oren. Distributed Databases, WWWeb http://techst02.technion.ac.il/~s2490610/dbe/

20) Beel, David & Grimson, Jane. Distributed Database Systems, Addison-Wesley, 1992

21) Siblerschatz, A.; Korth, H.; Sudarshan, S. Database Systems Concepts. McGraw-Hill, 1997.

22) Flynn, Michael J. Very High-Speed Computing Systems, Proceedings of the IEEE 54:12 (December 1966), pp1901-1909.

23) Rogerson, D. Inside COM. Microsoft Press, 1997

24) Li, S. & Economopoulos, P. Professional COM Applications with ATL. Wrox Press, 1998.

25) Lhotka, R. Professional Visual Basic 5 Business Objects. Wrox Press, 1997.

26) Platt, D. The Essence of COM with ActiveX. 2nd Ed. Prentice-Hall, 1998.

27) Orfali, R.; Harkey, D. & Edwards, J. The Essential Distributed Objects Survival Guide. John Willey & Sons, 1996.

28) Box, D. Essential COM. Addison-Wesley, 1998.

29) Eddon G. & Eddon H. Inside Distributed DCOM. Microsoft Press, 1998.

30) Grimmes, R. Professional DCOM Programming. Wrox Press, 1997.

31) Session, R. COM and DCOM: Microsoft's Vision for Distributed Objects. John Willey & Sons, 1998.

32) Pinnock, J. Professional DCOM Application Development. Wrox Press, 1998.

33) Redmond III, F. DCOM: Microsoft Distributed Component Object Model. IDG Books, 1997.

34) Homer, Alex & Sussman, David. Professional MTS and MSMQ with VB and ASP. Wrox Press, 1998.

35) Object Management Group - OMG & X/Open. The Common Object Request Broker - Architecture and Specification. OMG, 1992

36) CORBA Internet Site. WWWeb, http//www.corba.org.

37) TPC-C Specification. TPC - Transaction Processing Performance Council. WWWeb, http://www.tpc.org

38) Grimes, R. ATL COM Programming. Wrox Press, 1998.

39) Muller, P. Instant UML. Wrox Press, 1997.

40) Vinoski, S. New Features for CORBA 3.0. Communications of the ACM, Vol. 41, pag. 10, pg. 44-52.

41) Miller, K. Professional NT Services, Wrox Press 1998.

42) Grimes, R *et al*, Beginning ATL COM Programming, Wrox Press 1998.

43) Inprise Web Site. Applications, PowerBuilder and Delphi. WWWeb, http://www.inprise.com.

44) Sun Web Site. WWWeb: http://wwww.sun.com.

45) Jain, Raj, The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation, and Modeling. John Wiley & Sons, 1991.

46) Won, K - Modern Database Systems. ACM Press, 1995

47) Microsoft SQL Server Site. WWWeb: http://www.microsoft.com/sql.

48) Oracle Database Server. WWWeb: http://www.oracle.com/database.

49) DCOM Specifications. WWWeb: http://www.microsoft.com/com.

50) Chung, P. E. et al, DCOM and CORBA Side by Side, Step By Step, and Layer by Layer, C++ Report, Sept. 1997.

51) Rector B., Sells C., ATL Internals. Addison-Wesley, 1999.

52) Box, D. et al, Effective COM. Addison-Wesley, 1999.

53) Otey, M. & Conte, P., SQL Server 7 Developer's Guide. Osborne-McGraw Hill, 1999.

54) Dickman, A., Designing Applications with MSMQ. Addison-Wesley, 1999.

55) Schildt, H., STL Programming. Osborne-McGraw Hill,1999.

56) Meyers, S., Effective C++ CD. Addison-Wesley, 1999.

57) Major, A., COM IDL & Interface Design. Wrox Press, 1999.

58) Spenik, M. et al, Microsoft SQL Server 7 DBA Survival Guide. SAMS Publishing, 1999.

59) Pattison, T., Programming Distributed Applications with COM and Microsoft Visual Basic 6.0. Microsoft Press, 1999.

60) Kirtland, M., Designing Component-Based Applications. Microsoft Press, 1999.

61) Soukoup, R. & Delaney, K., Inside Microsoft SQL Server 7.0. Microsoft Press, 1999.

62) McGehee, B. et al, Using SQL Server 7.0. QUE, 1999.

63) Gamma, E. et al, Design Patterns - Elements of Reusable Objected-Oriented Software. Addison-Wesley, 1995.

64) McConnel, S., Code Complete: A Practical Handbook of Software Construction. Microsoft Press, 1993.

65) Microsoft Message Queue Server. WWWeb, http://www.microsoft.com/ntserver/appservice/exec/overview/MSMQ_Overview.asp.

66) Microsoft Transaction Server. WWWeb, http://www.microsoft.com/ntserver/appservice/exec/overview/Trans_Overview.asp.

67) ErWin Web Page. WWWeb, http://www.platinum.com/products/appdev/erwin_ps.htm.

68) ADO Web Page. WWWEb, http://www.microsoft.com/data.

69) Intel web Page. WWWeb, http:///www.intel.com.

70) Linux Web Site. WWWeb, http://www.linux.org.

71) NT Server Web Page. WWWeb,
http://www.microsoft.com/ntserver.

72) Symantec Web Site. WWWeb, http://www.symantec.com.

73) TPC Benchmark Specification, Revision 3.4. WWWeb,
http://www.tpc.org. August 15, 1998

74) Kim, Won. Modern Database Systems : The Object Model,
Interoperability, and Beyond. Addison-Wesley, 1995.

Appendix A – Acronyms

ACID	Atomicity, Consistency, Isolation and Durability
ADO	ActiveX Database Objects
AFA	Brazilian Air Force Academy
BAF	Brazilian Air Force
C/S	Client/Server
CAB-SP	Brazilian Air Force Procurement Commission at São Paulo
CAB-L	Brazilian Air Force Procurement Commission at London
CAB-W	Brazilian Air Force Procurement Commission at Washington
CASE	Computer Aided Software Engineering
CCA-RJ	Brazilian Air Force Computing Center at Rio de Janeiro
CCA-SJ	Brazilian Air Force Computing Center at São José dos Campos
CCA-BR	Brazilian Air Force Computing Center at Brasília
CATRE	Tactical Air Force Training Center
COM	Component Object Model
COMGAP	Brazilian Air Force Support Command
CORBA	Common Object Request Broker Architecture
DAC	Brazilian Air Force Civil Aviation Department
DAO	Database Access Objetcs (Jet Library)
DBMS	Database Management System
DCOM	Distributed Component Object Model

DDBMS	Distributed Database Management System
DDL	Data Definition Language
DEPV	Brazilian Air Force Air Traffic Directorate
DIRENG	Brazilian Air Force Engineering Directorate
DIRINFE	Brazilian Air Force Computer Science and Statistics Directorate
DIRMA	Brazilian Air Force Materiel Directorate
DIRMAB	Brazilian Air Force Munitions Directorate
DIRSA	Brazilian Air Force Healthcare Directorate
DTC	Distributed Transaction Coordinator
LAN	Local-Area Network
MAN	Metropolitan-Area Network
MINAER	Brazilian Air Force Administrative Headquarters
MTS	Microsoft Transaction Server
MSMQ	Microsoft Message Queue Server
NOS	Network Operating System
ORB	Object Request Broker
ODBC	Open Database Connectivity Library
OODBMS	Object-Oriented DataBase Management Server
PAMA-AF	Brazilian Air Force Aeronautical Depot at Afonsos
PAMA-GL	Brazilian Air Force Aeronautical Depot at Galeão
PAMA-LS	Brazilian Air Force Aeronautical Depot at Lagoa Santa
PAMA-SP	Brazilian Air Force Aeronautical Depot at São Paulo

PAMA-RF	Brazilian Air Force Aeronautical Depot at Recife
RCDMA	Brazilian Air Force Data Communications Network
SILOMS	Brazilian Air Force Logistics, Materiel, and Services Information System
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
VB	Microsoft Visual Basic
VB	Microsoft Visual C++
WAN	Wide-Area Network
WWW	World Wide Web
X.25	Packet-switching CCITT protocol standard

Appendix B – Database SQL Scripts

```
/* Microsoft SQL Server - Scripting                    */
/* Database: Thesis
            */
/* Creation Date 12/11/1998 12:58:32 PM
            */

set quoted_identifier  OFF
GO

/* Microsoft SQL Server - Scripting                    */
/* Server: HELEN
            */
/* Database: Thesis
            */
/* Creation Date 12/11/1998 11:05:25 AM
            */

CREATE RULE ALL_LOCAL_VAL
            AS @col BETWEEN 0 AND 9

GO
CREATE RULE CREDIT_VAL
            AS @col IN ('GC', 'BC')

GO
CREATE RULE DISTRICT_VAL
            AS @col BETWEEN 1 AND 20

GO
CREATE RULE ORDER_LINE_VAL
            AS @col BETWEEN 0 AND 15

GO
CREATE RULE QUANTITY_VAL
            AS @col BETWEEN 1 AND 99

GO

setuser N'dbo'
GO

create default [ZERO_VALUE] as 0
GO

create default [ONE_VALUE] as 1
GO


EXEC sp_addtype N'ALL_LOCAL_DOMAIN', N'char (18)', N'null'
GO

setuser
GO

setuser N'dbo'
GO

EXEC sp_bindrule N'[dbo].[ALL_LOCAL_VAL]', N'[ALL_LOCAL_DOMAIN]'
GO

setuser
GO

setuser N'dbo'
GO

EXEC sp_addtype N'CREDIT_DOMAIN', N'char (18)', N'null'
GO

setuser
GO

setuser N'dbo'
GO

EXEC sp_bindrule N'[dbo].[CREDIT_VAL]', N'[CREDIT_DOMAIN]'
GO

setuser
GO

setuser N'dbo'
GO

EXEC sp_addtype N'DISTRICT_DOMAIN', N'char (18)', N'null'
GO

setuser
GO

setuser N'dbo'
GO

EXEC sp_bindrule N'[dbo].[DISTRICT_VAL]', N'[DISTRICT_DOMAIN]'
GO

setuser
GO

setuser N'dbo'
GO

EXEC sp_addtype N'ORDER_LINE_DOMAIN', N'char (18)', N'null'
GO

setuser
GO

setuser N'dbo'
GO

EXEC sp_bindrule N'[dbo].[ORDER_LINE_VAL]',
N'[ORDER_LINE_DOMAIN]'
GO

setuser
GO

setuser N'dbo'
GO

EXEC sp_addtype N'QUANTITY_DOMAIN', N'smallint', N'null'
GO

setuser
GO

setuser N'dbo'
GO

EXEC sp_bindrule N'[dbo].[QUANTITY_VAL]', N'[QUANTITY_DOMAIN]'
GO

setuser
GO

CREATE TABLE [dbo].[Warehouse] (
            [W_ID] [int] NOT NULL ,
            [W_NAME] [varchar] (10) NOT NULL ,
            [W_STREET_1] [varchar] (20) NOT NULL ,
            [W_STREET_2] [varchar] (20) NULL ,
            [W_CITY] [varchar] (20) NOT NULL ,
            [W_STATE] [varchar] (2) NOT NULL ,
            [W_ZIP] [varchar] (9) NOT NULL ,
            [W_TAX] [real] NULL ,
            [W_YTD] [real] NULL ,
             PRIMARY KEY  CLUSTERED
            (
                        [W_ID]
            )  ON [PRIMARY]
)
GO

setuser N'dbo'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]', N'[Warehouse].[W_YTD]'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]', N'[Warehouse].[W_TAX]'
GO

setuser
GO

CREATE TABLE [dbo].[District] (
            [D_ID] [smallint] NOT NULL ,
            [D_W_ID] [int] NOT NULL ,
            [D_NAME] [varchar] (10) NOT NULL ,
            [D_STREET_1] [varchar] (20) NOT NULL ,
            [D_STREET_2] [varchar] (20) NULL ,
            [D_CITY] [varchar] (20) NOT NULL ,
            [D_STATE] [varchar] (2) NOT NULL ,
            [D_ZIP] [varchar] (9) NOT NULL ,
            [D_TAX] [real] NULL ,
            [D_YTD] [real] NULL ,
            [D_NEXT_O_ID] [int] NULL ,
             PRIMARY KEY  CLUSTERED
            (
                        [D_ID],
                        [D_W_ID]
            )  ON [PRIMARY] ,
             FOREIGN KEY
            (
                        [D_W_ID]
            ) REFERENCES [dbo].[Warehouse] (
                        [W_ID]
            )
)
GO

 CREATE  INDEX [DistrictWarehouse] ON [dbo].[District]([D_W_ID])
ON [PRIMARY]
GO

setuser N'dbo'
GO
```

```
EXEC sp_bindrule N'[dbo].[DISTRICT_VAL]', N'[District].[D_ID]'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]', N'[District].[D_TAX]'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]', N'[DISTRICT].[D_YTD]'
GO

EXEC sp_bindefault N'[dbo].[ONE_VALUE]',
N'[DISTRICT].[D_NEXT_O_ID]'
GO

setuser
GO

CREATE TABLE [dbo].[Customer] (
            [C_ID] [int] NOT NULL ,
            [C_D_ID] [smallint] NOT NULL ,
            [C_W_ID] [int] NOT NULL ,
            [C_FIRST] [varchar] (16) NOT NULL ,
            [C_MIDDLE] [varchar] (2) NULL ,
            [C_LAST] [varchar] (16) NOT NULL ,
            [C_STREET_1] [varchar] (20) NOT NULL ,
            [C_STREET_2] [varchar] (20) NULL ,
            [C_CITY] [varchar] (20) NOT NULL ,
            [C_STATE] [varchar] (2) NOT NULL ,
            [C_ZIP] [varchar] (9) NOT NULL ,
            [C_PHONE] [varchar] (16) NULL ,
            [C_SINCE] [datetime] NULL ,
            [C_CREDIT] [varchar] (2) NULL ,
            [C_CREDIT_LIM] [real] NULL ,
            [C_DISCOUNT] [real] NULL ,
            [C_BALANCE] [real] NULL ,
            [C_YTD_PAYMENT] [real] NULL ,
            [C_PAYMENT_CNT] [real] NULL ,
            [C_DELIVERY_CNT] [real] NULL ,
            [C_DATA] [text] NULL ,
             PRIMARY KEY  CLUSTERED
            (
                        [C_ID],
                        [C_D_ID],
                        [C_W_ID]
            )  ON [PRIMARY] ,
             FOREIGN KEY
            (
                        [C_D_ID],
                        [C_W_ID]
            ) REFERENCES [dbo].[District] (
                        [D_ID],
                        [D_W_ID]
            )
)
GO

 CREATE  INDEX [CustomerDistrict] ON [dbo].[Customer]([C_D_ID],
[C_W_ID]) ON [PRIMARY]
GO

setuser N'dbo'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]',
N'[Customer].[C_CREDIT_LIM]'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]',
N'[Customer].[C_DISCOUNT]'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]',
N'[Customer].[C_BALANCE]'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]',
N'[Customer].[C_YTD_PAYMENT]'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]',
N'[Customer].[C_PAYMENT_CNT]'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]',
N'[Customer].[C_DELIVERY_CNT]'
GO

EXEC sp_bindrule N'[dbo].[CREDIT_VAL]', N'[Customer].[C_CREDIT]'
GO

EXEC sp_bindrule N'[dbo].[DISTRICT_VAL]', N'[Customer].[C_D_ID]'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]',
N'[Customer].[C_DELIVERY_CNT]'
GO

setuser
GO

CREATE TABLE [dbo].[District_Order] (
            [O_ID] [int] NOT NULL ,
            [O_D_ID] [smallint] NOT NULL ,
            [O_W_ID] [int] NOT NULL ,
            [O_C_ID] [int] NULL ,
            [O_ENTRY_DATE] [datetime] NULL ,
            [O_CARRIER_ID] [char] (18) NULL ,
            [O_OL_CNT] [smallint] NULL,
            [O_ALL_LOCAL] [tinyint] NULL ,
             PRIMARY KEY  CLUSTERED
            (
                        [O_ID],
                        [O_D_ID],
                        [O_W_ID]
            )  ON [PRIMARY] ,
             FOREIGN KEY
            (
                        [O_C_ID],
                        [O_D_ID],
                        [O_W_ID]
            ) REFERENCES [dbo].[Customer] (
                        [C_ID],
                        [C_D_ID],
                        [C_W_ID]
            ),
             FOREIGN KEY
            (
                        [O_D_ID],
                        [O_W_ID]
            ) REFERENCES [dbo].[District] (
                        [D_ID],
                        [D_W_ID]
            )
)
GO

 CREATE  INDEX [DistrictOrderDistrict] ON
[dbo].[District_Order]([O_D_ID], [O_W_ID]) ON [PRIMARY]
GO

 CREATE  INDEX [DistrictOrderOrder] ON
[dbo].[District_Order]([O_D_ID], [O_W_ID], [O_C_ID]) ON [PRIMARY]
GO

setuser N'dbo'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]',
N'[District_Order].[O_OL_CNT]'
GO

EXEC sp_bindrule N'[dbo].[ALL_LOCAL_VAL]',
N'[District_Order].[O_ALL_LOCAL]'
GO

EXEC sp_bindrule N'[dbo].[DISTRICT_VAL]',
N'[District_Order].[O_D_ID]'
GO

setuser
GO

CREATE TABLE [dbo].[History] (
            [H_C_ID] [int] NULL ,
            [H_C_D_ID] [smallint] NULL ,
            [H_C_W_ID] [int] NULL ,
            [H_D_ID] [smallint] NULL ,
            [H_W_ID] [int] NULL ,
            [H_DATE] [datetime] NULL ,
            [H_AMOUNT] [real] NULL ,
            [H_DATA] [varchar] (24) NULL ,
             FOREIGN KEY
            (
                        [H_D_ID],
                        [H_W_ID]
            ) REFERENCES [dbo].[District] (
                        [D_ID],
                        [D_W_ID]
            ),
             FOREIGN KEY
            (
                        [H_C_ID],
                        [H_C_D_ID],
                        [H_C_W_ID]
            ) REFERENCES [dbo].[Customer] (
                        [C_ID],
                        [C_D_ID],
                        [C_W_ID]
            )
) ON [PRIMARY]
GO

 CREATE  INDEX [Historydistrict] ON [dbo].[History]([H_D_ID],
[H_W_ID]) ON [PRIMARY]
GO

 CREATE  INDEX [HistoryCustomer] ON [dbo].[History]([H_C_ID],
[H_C_D_ID], [H_C_W_ID]) ON [PRIMARY]
GO

setuser N'dbo'
GO

EXEC sp_bindrule N'[dbo].[DISTRICT_VAL]', N'[History].[H_C_D_ID]'
GO

EXEC sp_bindrule N'[dbo].[DISTRICT_VAL]', N'[History].[H_D_ID]'
GO

setuser
GO

CREATE TABLE [dbo].[Item] (
            [I_ID] [int] NOT NULL ,
            [I_IM_ID] [char] (18) NULL ,
            [I_NAME] [varchar] (24) NULL ,
            [I_PRICE] [real] NULL ,
            [I_DATA] [varchar] (50) NULL ,
             PRIMARY KEY  CLUSTERED
            (
                        [I_ID]
            )  ON [PRIMARY]
)
GO


setuser
```

```
GO

CREATE TABLE [dbo].[Stock] (
            [S_I_ID] [int] NOT NULL ,
            [S_W_ID] [int] NOT NULL ,
            [S_QUANTITY] [smallint] NULL ,
            [S_DIST_01] [varchar] (24) NULL ,
            [S_DIST_02] [varchar] (24) NULL ,
            [S_DIST_03] [varchar] (24) NULL ,
            [S_DIST_04] [varchar] (24) NULL ,
            [S_DIST_05] [varchar] (24) NULL ,
            [S_DIST_06] [varchar] (24) NULL ,
            [S_DIST_07] [varchar] (24) NULL ,
            [S_DIST_08] [varchar] (24) NULL ,
            [S_DIST_09] [varchar] (24) NULL ,
            [S_DIST_10] [varchar] (24) NULL ,
            [S_YTD] [real] NULL ,
            [S_ORDER_CNT] [smallint] NULL ,
            [S_REMOTE_CNT] [smallint] NULL ,
            [S_DATA] [varchar] (50) NULL ,
             PRIMARY KEY  CLUSTERED
            (
                        [S_I_ID],
                        [S_W_ID]
            )  ON [PRIMARY] ,
             FOREIGN KEY
            (
                        [S_I_ID]
            ) REFERENCES [dbo].[Item] (
                        [I_ID]
            ),
             FOREIGN KEY
            (
                        [S_W_ID]
            ) REFERENCES [dbo].[Warehouse] (
                        [W_ID]
            )
)
GO

setuser N'dbo'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]', N'[Stock].[S_YTD]'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]',
N'[Stock].[S_ORDER_CNT]'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]',
N'[Stock].[S_REMOTE_CNT]'
GO

EXEC sp_bindefault N'[dbo].[ZERO_VALUE]', N'[Stock].[S_QUANTITY]'
GO

setuser
GO

 CREATE  INDEX [StockItem] ON [dbo].[Stock]([S_I_ID]) ON
[PRIMARY]
GO

 CREATE  INDEX [StockWarehouse] ON [dbo].[Stock]([S_W_ID]) ON
[PRIMARY]
GO

CREATE TABLE [dbo].[New_Order] (
            [NO_O_ID] [int] NOT NULL ,
            [NO_D_ID] [smallint] NOT NULL ,
            [NO_W_ID] [int] NOT NULL ,
             PRIMARY KEY  CLUSTERED
            (
                        [NO_O_ID],
                        [NO_D_ID],
                        [NO_W_ID]
            )  ON [PRIMARY] ,
             FOREIGN KEY
            (
                        [NO_O_ID],
                        [NO_D_ID],
                        [NO_W_ID]
            ) REFERENCES [dbo].[District_Order] (
                        [O_ID],
                        [O_D_ID],
                        [O_W_ID]
            )
)
GO

 CREATE  INDEX [New_OrderDistrictOrder] ON
[dbo].[New_Order]([NO_O_ID], [NO_D_ID], [NO_W_ID]) ON [PRIMARY]
GO

 CREATE  INDEX [IX_NO_O_ID] ON [dbo].[New_Order]([NO_O_ID]) WITH
FILLFACTOR = 50 ON [PRIMARY]
GO

setuser N'dbo'
GO

EXEC sp_bindrule N'[dbo].[DISTRICT_VAL]',
N'[New_Order].[NO_D_ID]'
GO

setuser
GO

CREATE TABLE [dbo].[Order_Line] (
            [OL_O_ID] [int] NOT NULL ,
            [OL_D_ID] [smallint] NOT NULL ,
            [OL_W_ID] [int] NOT NULL ,
            [OL_NUMBER] [char] (18) NOT NULL ,
            [OL_I_ID] [int] NULL ,
            [OL_SUPPLY_W_ID] [int] NULL ,
            [OL_DELIVERY_D] [datetime] NULL ,
            [OL_QUANTITY] [smallint] NULL ,
            [OL_AMOUNT] [real] NULL ,
            [OL_DIST_INFO] [varchar] (24) NULL ,
             PRIMARY KEY  CLUSTERED
            (
                        [OL_O_ID],
                        [OL_D_ID],
                        [OL_W_ID],
                        [OL_NUMBER]
            )  ON [PRIMARY] ,
             FOREIGN KEY
            (
                        [OL_I_ID],
                        [OL_SUPPLY_W_ID]
            ) REFERENCES [dbo].[Stock] (
                        [S_I_ID],
                        [S_W_ID]
            ),
             FOREIGN KEY
            (
                        [OL_O_ID],
                        [OL_D_ID],
                        [OL_W_ID]
            ) REFERENCES [dbo].[District_Order] (
                        [O_ID],
                        [O_D_ID],
                        [O_W_ID]
            )
)
GO

 CREATE  INDEX [Order_LineDistrict_Order] ON
[dbo].[Order_Line]([OL_O_ID], [OL_D_ID], [OL_W_ID]) ON [PRIMARY]
GO

 CREATE  INDEX [Order_LineStock] ON [dbo].[Order_Line]([OL_I_ID],
[OL_SUPPLY_W_ID]) ON [PRIMARY]
GO

setuser N'dbo'
GO

EXEC sp_bindrule N'[dbo].[DISTRICT_VAL]',
N'[Order_Line].[OL_D_ID]'
GO

EXEC sp_bindrule N'[dbo].[QUANTITY_VAL]',
N'[Order_Line].[OL_QUANTITY]'
GO



create trigger tD_Customer on Customer for DELETE as
/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
/* DELETE trigger on Customer */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
    /* Customer R/22 District_Order ON PARENT DELETE SET NULL */
    if exists(select O_ID
      from District_Order,deleted
      where
        /* %JoinFKPK(District_Order,deleted," = "," and") */
        District_Order.O_C_ID = deleted.C_ID and
        District_Order.O_D_ID = deleted.C_D_ID and
        District_Order.O_W_ID = deleted.C_W_ID)
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot DELETE "Customer" because
"District_Order" exists.'
      goto error
    end

    /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
    /* Customer R/9 History ON PARENT DELETE SET NULL */
    if exists(select H_C_ID
      from History,deleted
      where
        /* %JoinFKPK(History,deleted," = "," and") */
        History.H_C_ID = deleted.C_ID and
        History.H_C_D_ID = deleted.C_D_ID and
        History.H_C_W_ID = deleted.C_W_ID)
      begin
                select @errno  = 30007,
                @errmsg = 'Cannot DELETE "Customer" because
"History" exists.'
                goto error
      end


    /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
create trigger tI_Customer on Customer for INSERT as
/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
/* INSERT trigger on Customer */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
```

```
  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  /* District R/8 Customer ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(C_D_ID) or
    update(C_W_ID)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,District
        where
          /* %JoinFKPK(inserted,District) */
          inserted.C_D_ID = District.D_ID and
          inserted.C_W_ID = District.D_W_ID
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "Customer" because
"District" does not exist.'
      goto error
    end
  end


  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
create trigger tU_Customer on Customer for UPDATE as
/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
/* UPDATE trigger on Customer */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insC_ID int,
           @insC_D_ID smallint,
           @insC_W_ID int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  /* Customer R/22 District_Order ON PARENT UPDATE SET NULL */
  if
    /* %ParentPK(" or",update) */
    update(C_ID) or
    update(C_D_ID) or
    update(C_W_ID)
     begin
         select @errno  = 30007,
                @errmsg = 'Cannot UPDATE "Customer" primary key.'
              goto error
     end

  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
CREATE trigger tD_District on District for DELETE as
/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
/* DELETE trigger on District */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
    /* District R/21 District_Order ON PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,District_Order
      where
        /*  %JoinFKPK(District_Order,deleted," = "," and") */
        District_Order.O_D_ID = deleted.D_ID and
        District_Order.O_W_ID = deleted.D_W_ID
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "District" because
"District_Order" exists.'
      goto error
    end

    /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
    /* District R/11 History ON PARENT DELETE SET NULL */
    if exists(select H_D_ID
      from History,deleted
      where
        History.H_D_ID = deleted.D_ID and
        History.H_W_ID = deleted.D_W_ID)
        /* %JoinFKPK(History,deleted," = "," and") */
      begin
       select @errno  = 30001,
              @errmsg = 'Cannot DELETE "District" because
"History" exists.'
       goto error
      end

    /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
    /* District R/8 Customer ON PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,Customer
      where
        /*  %JoinFKPK(Customer,deleted," = "," and") */
        Customer.C_D_ID = deleted.D_ID and
        Customer.C_W_ID = deleted.D_W_ID
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "District" because
"Customer" exists.'
      goto error
    end

      update Warehouse set W_YTD = W_YTD - deleted.D_YTD
       from Warehouse, deleted
      where Warehouse.W_ID = deleted.D_W_ID

    /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end


GO
CREATE   trigger tI_District on District for INSERT as
/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
/* INSERT trigger on District */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  /* Warehouse R/5 District ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(D_W_ID)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,Warehouse
        where
          /* %JoinFKPK(inserted,Warehouse) */
          inserted.D_W_ID = Warehouse.W_ID
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "District" because
"Warehouse" does not exist.'
      goto error
    end

     update Warehouse set W_YTD = W_YTD + inserted.D_YTD
     from Warehouse, inserted
     where Warehouse.W_ID = inserted.D_W_ID

  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end


GO
CREATE  trigger tU_District on District for UPDATE as
/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
/* UPDATE trigger on District */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insD_ID smallint,
           @insD_W_ID int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  /* District R/21 District_Order ON PARENT UPDATE RESTRICT */
  if
    /* %ParentPK(" or",update) */
    update(D_ID) or
    update(D_W_ID)
       begin
        select @errno  = 30005,
               @errmsg = 'Cannot UPDATE "District" primary key.'
        goto error
       end


/* Warehouse Constraint YTD */
if update(D_W_ID) or update(D_YTD)
   begin
      update Warehouse set W_YTD = W_YTD - deleted.D_YTD
       from Warehouse, deleted
      where Warehouse.W_ID = deleted.D_W_ID
     update Warehouse set W_YTD = W_YTD + inserted.D_YTD
     from Warehouse, inserted
     where Warehouse.W_ID = inserted.D_W_ID
   end

  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  return
```

```
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
CREATE  trigger tD_District_Order on District_Order for DELETE as
/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
/* DELETE trigger on District_Order */
begin
  declare  @errno   int,
           @numrows int,
           @errmsg  varchar(255)

    select @numrows = @@rowcount

/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
    /* District_Order R/29 Order_Line ON PARENT DELETE RESTRICT
*/
    if exists (
      select * from deleted,Order_Line
      where
        /*  %JoinFKPK(Order_Line,deleted," = "," and") */
        Order_Line.OL_O_ID = deleted.O_ID and
        Order_Line.OL_D_ID = deleted.O_D_ID and
        Order_Line.OL_W_ID = deleted.O_W_ID
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "District_Order" because
"Order_Line" exists.'
      goto error
    end

    /* ERwin Builtin Wed Nov'11 13:37:55 1998 */
    /* District_Order R/28 New_Order ON PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,New_Order
      where
        /*  %JoinFKPK(New_Order,deleted," = "," and") */
        New_Order.NO_O_ID = deleted.O_ID and
        New_Order.NO_D_ID = deleted.O_D_ID and
        New_Order.NO_W_ID = deleted.O_W_ID
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "District_Order" because
"New_Order" exists.'
      goto error
    end

  update Customer
  set C_DELIVERY_CNT = C_DELIVERY_CNT + @numrows
  from Customer, deleted
  where Customer.C_W_ID = deleted.O_W_ID and
        Customer.C_D_ID = deleted.O_D_ID and
        Customer.C_ID = deleted.O_C_ID

    /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end


GO
CREATE      trigger tI_District_Order on District_Order for
INSERT as
/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
/* INSERT trigger on District_Order */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  /* Customer R/22 District_Order ON CHILD INSERT SET NULL */

  if not exists( select  D_ID from District, inserted where
District.D_NEXT_O_ID = inserted.O_ID)
    begin
      select @errno  = 30502,
             @errmsg = 'Cannot INSERT "District_Order" because
"O_ID" does not match with "D_NEXT_O_ID".'
      goto error
    end

    update District
    set D_NEXT_O_ID = D_NEXT_O_ID + 1
    from District, District_Order
    where District.D_ID = District_Order.O_D_ID

    insert into New_Order  select O_ID, O_D_ID, O_W_ID from
inserted

  if
    /* %ChildFK(" or",update) */
    update(O_C_ID) or
    update(O_D_ID) or
    update(O_W_ID)
  begin
    if not exists (
          select * from Customer, inserted
          where
            /* %JoinFKPK(inserted,Customer," = "," and") */
            inserted.O_C_ID = Customer.C_ID and
            inserted.O_D_ID = Customer.C_D_ID and
            inserted.O_W_ID = Customer.C_W_ID
        )
      begin
        select @errno  = 30007,
               @errmsg = 'Cannot INSERT "District_Order" because
"CUSTOMER" does exist.'
        goto error
      end

  end

  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  /* District R/21 District_Order ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(O_D_ID) or
    update(O_W_ID)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,District
        where
          /* %JoinFKPK(inserted,District) */
          inserted.O_D_ID = District.D_ID and
          inserted.O_W_ID = District.D_W_ID
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "District_Order" because
"District" does not exist.'
      goto error
    end
  end

  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end


GO
create trigger tU_District_Order on District_Order for UPDATE as
/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
/* UPDATE trigger on District_Order */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insO_ID int,
           @insO_D_ID smallint,
           @insO_W_ID int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  /* District_Order R/29 Order_Line ON PARENT UPDATE RESTRICT */
  if
    /* %ParentPK(" or",update) */
    update(O_ID) or
    update(O_D_ID) or
    update(O_W_ID)
  begin
    if exists (
      select * from deleted,Order_Line
      where
        /*  %JoinFKPK(Order_Line,deleted," = "," and") */
        Order_Line.OL_O_ID = deleted.O_ID and
        Order_Line.OL_D_ID = deleted.O_D_ID and
        Order_Line.OL_W_ID = deleted.O_W_ID
    )
    begin
      select @errno  = 30005,
             @errmsg = 'Cannot UPDATE "District_Order" because
"Order_Line" exists.'
      goto error
    end
  end

  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  /* District_Order R/28 New_Order ON PARENT UPDATE RESTRICT */
  if
    /* %ParentPK(" or",update) */
    update(O_ID) or
    update(O_D_ID) or
    update(O_W_ID)
  begin
    if exists (
      select * from deleted,New_Order
      where
        /*  %JoinFKPK(New_Order,deleted," = "," and") */
        New_Order.NO_O_ID = deleted.O_ID and
        New_Order.NO_D_ID = deleted.O_D_ID and
        New_Order.NO_W_ID = deleted.O_W_ID
    )
    begin
      select @errno  = 30005,
             @errmsg = 'Cannot UPDATE "District_Order" because
"New_Order" exists.'
      goto error
    end
  end

  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  /* Customer R/22 District_Order ON CHILD UPDATE SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(O_C_ID) or
    update(O_D_ID) or
    update(O_W_ID)
```

```
  begin
    update District_Order
      set
        /* %SetFK(District_Order,NULL) */
        District_Order.O_C_ID = NULL,
        District_Order.O_D_ID = NULL,
        District_Order.O_W_ID = NULL
      from District_Order,inserted
      where
        /* %JoinPKPK(District_Order,inserted," = "," and") */
        District_Order.O_ID = inserted.O_ID and
        District_Order.O_D_ID = inserted.O_D_ID and
        District_Order.O_W_ID = inserted.O_W_ID and
        not exists (
          select * from Customer
          where
            /* %JoinFKPK(inserted,Customer," = "," and") */
            inserted.O_C_ID = Customer.C_ID and
            inserted.O_D_ID = Customer.C_D_ID and
            inserted.O_W_ID = Customer.C_W_ID
        )
  end

  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  /* District R/21 District_Order ON CHILD UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(O_D_ID) or
    update(O_W_ID)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,District
        where
          /* %JoinFKPK(inserted,District) */
          inserted.O_D_ID = District.D_ID and
          inserted.O_W_ID = District.D_W_ID
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "District_Order" because
"District" does not exist.'
      goto error
    end
  end

 if update(O_C_ID) or
     update(O_W_ID) or
     update(O_D_ID)
     begin
        if not exists (
          select * from Customer, inserted
          where
            /* %JoinFKPK(inserted,Customer," = "," and") */
            inserted.O_C_ID = Customer.C_ID and
            inserted.O_D_ID = Customer.C_D_ID and
            inserted.O_W_ID = Customer.C_W_ID)
        begin
            select @errno = 30502,
                   @errmsg = 'Cannot UPDATE "District_Order"
because "Customer" does not exist.'
            goto error
        end
        update Customer
        set C_DELIVERY_CNT = C_DELIVERY_CNT - 1
        from Customer, deleted
        where
            deleted.O_C_ID = Customer.C_ID and
            deleted.O_D_ID = Customer.C_D_ID and
            deleted.O_W_ID = Customer.C_W_ID

        insert into New_Order  select O_ID, O_D_ID, O_W_ID from
inserted
     end


  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
CREATE  trigger tU_History on History for UPDATE as
/* ERwin Builtin Wed Nov 11 13:38:03 1998 */
/* UPDATE trigger on History */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount

  /* ERwin Builtin Wed Nov 11 13:38:03 1998 */
  /* Customer R/9 History ON CHILD UPDATE SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(H_C_ID) or
    update(H_C_D_ID) or
    update(H_C_W_ID) or
    update(H_D_ID) or
    update(H_W_ID)
  begin
            select @errno = 30804,
                   @errmsg = 'Cannot change "History" district or
customer.'
        goto error
  end
  if update(H_AMOUNT)
  begin
     update District
     set D_YTD = D_YTD - deleted.H_AMOUNT
     from District, deleted
     where District.D_ID = deleted.H_D_ID and
               District.D_W_ID = deleted.H_W_ID

     update District
     set D_YTD = D_YTD + inserted.H_AMOUNT
     from District, inserted
     where District.D_ID = inserted.H_D_ID and
               District.D_W_ID = inserted.H_W_ID

    update Customer
    set C_BALANCE = C_BALANCE + deleted.H_AMOUNT
    from customer, deleted
    where  Customer.C_W_ID = deleted.H_C_W_ID and
           Customer.C_D_ID = deleted.H_C_D_ID and
               Customer.C_ID = deleted.H_C_ID
    update Customer

    set C_BALANCE = C_BALANCE - inserted.H_AMOUNT
    from customer, inserted
    where  Customer.C_W_ID = inserted.H_C_W_ID and
           Customer.C_D_ID = inserted.H_C_D_ID and
               Customer.C_ID = inserted.H_C_ID
  end


  /* ERwin Builtin Wed Nov 11 13:38:03 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end


GO
CREATE  trigger tI_History on History for INSERT as
/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
/* INSERT trigger on History */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  /* District R/11 History ON CHILD INSERT SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(H_D_ID) or
    update(H_W_ID)
  begin
            if not exists (
           select * from District, inserted
           where
             /* %JoinFKPK(inserted,District," = "," and") */
             inserted.H_D_ID = District.D_ID and
             inserted.H_W_ID = District.D_W_ID
         )
            begin
               select @errno = 30456,
                             @errmsg = 'Cannot insert "History"
because "District does not exist.'
               goto error
            end
  end

  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  /* Customer R/9 History ON CHILD INSERT SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(H_C_ID) or
    update(H_C_D_ID) or
    update(H_C_W_ID)
  begin
            if not exists (
           select * from Customer, inserted
           where
             /* %JoinFKPK(inserted,Customer," = "," and") */
             inserted.H_C_ID = Customer.C_ID and
             inserted.H_C_D_ID = Customer.C_D_ID and
             inserted.H_C_W_ID = Customer.C_W_ID
         )
            begin
               select @errno = 30945,
                             @errmsg = 'Cannot insert "History"
because "Customer" does not exist.'
               goto error
            end
  end

     update District
     set D_YTD = D_YTD + inserted.H_AMOUNT
     from District, inserted
     where District.D_ID = inserted.H_D_ID and
               District.D_W_ID = inserted.H_W_ID

     update Customer
     set C_BALANCE = C_BALANCE - inserted.H_AMOUNT
     from Customer, inserted
     where
        /* %JoinFKPK(inserted,Customer," = "," and") */
        inserted.H_C_ID = Customer.C_ID and
        inserted.H_C_D_ID = Customer.C_D_ID and
        inserted.H_C_W_ID = Customer.C_W_ID
```

```
  /* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end


GO
CREATE   trigger tD_History on History for DELETE as
/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
/* INSERT trigger on History */
begin
  declare  @numrows int,
               @errno   int,
               @errmsg  varchar(255)

     update District
     set D_YTD = D_YTD - deleted.H_AMOUNT
     from District, deleted
     where District.D_ID = deleted.H_D_ID and
               District.D_W_ID = deleted.H_W_ID

     update Customer
     set C_BALANCE = C_BALANCE + deleted.H_AMOUNT
     from Customer, deleted
     where
        /* %JoinFKPK(inserted,Customer," = "," and") */
        deleted.H_C_ID = Customer.C_ID and
        deleted.H_C_D_ID = Customer.C_D_ID and
        deleted.H_C_W_ID = Customer.C_W_ID


/* ERwin Builtin Wed Nov 11 13:37:55 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end


GO
create trigger tD_Item on Item for DELETE as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* DELETE trigger on Item */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
    /* Item R/30 Stock ON PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,Stock
      where
        /*  %JoinFKPK(Stock,deleted," = "," and") */
        Stock.S_I_ID = deleted.I_ID
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "Item" because "Stock"
exists.'
      goto error
    end


    /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
create trigger tU_Item on Item for UPDATE as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* UPDATE trigger on Item */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insI_ID int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* Item R/30 Stock ON PARENT UPDATE RESTRICT */
  if
    /* %ParentPK(" or",update) */
    update(I_ID)
  begin
    if exists (
      select * from deleted,Stock
      where
        /*  %JoinFKPK(Stock,deleted," = "," and") */
        Stock.S_I_ID = deleted.I_ID
    )
    begin
      select @errno  = 30005,
             @errmsg = 'Cannot UPDATE "Item" because "Stock"
exists.'
      goto error
    end
  end


  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
```

```
create trigger tI_New_Order on New_Order for INSERT as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* INSERT trigger on New_Order */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* District_Order R/28 New_Order ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(NO_O_ID) or
    update(NO_D_ID) or
    update(NO_W_ID)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,District_Order
        where
          /* %JoinFKPK(inserted,District_Order) */
          inserted.NO_O_ID = District_Order.O_ID and
          inserted.NO_D_ID = District_Order.O_D_ID and
          inserted.NO_W_ID = District_Order.O_W_ID
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "New_Order" because
"District_Order" does not exist.'
      goto error
    end
  end


  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
create trigger tU_New_Order on New_Order for UPDATE as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* UPDATE trigger on New_Order */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insNO_O_ID int,
           @insNO_D_ID smallint,
           @insNO_W_ID int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* District_Order R/28 New_Order ON CHILD UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(NO_O_ID) or
    update(NO_D_ID) or
    update(NO_W_ID)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,District_Order
        where
          /* %JoinFKPK(inserted,District_Order) */
          inserted.NO_O_ID = District_Order.O_ID and
          inserted.NO_D_ID = District_Order.O_D_ID and
          inserted.NO_W_ID = District_Order.O_W_ID
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "New_Order" because
"District_Order" does not exist.'
      goto error
    end
  end


  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
CREATE    trigger tD_New_Order on New_Order for DELETE as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* INSERT trigger on New_Order */
begin
  declare  @numrows int,
               @errno   int,
               @errmsg  varchar(255)

  select @numrows = @@rowcount

  update Customer
  set C_DELIVERY_CNT = C_DELIVERY_CNT + @numrows
  from Customer, deleted, District_Order
  where District_Order.O_W_ID = deleted.NO_W_ID and
```

```
            District_Order.O_D_ID = deleted.NO_D_ID and
            District_Order.O_ID = deleted.NO_O_ID and
            Customer.C_W_ID = deleted.NO_W_ID and
            Customer.C_D_ID = deleted.NO_D_ID and
            Customer.C_ID = District_Order.O_C_ID


  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
CREATE  trigger tI_Order_Line on Order_Line for INSERT as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* INSERT trigger on Order_Line */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* Stock R/32 Order_Line ON CHILD INSERT SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(OL_I_ID) or
    update(OL_SUPPLY_W_ID)
  begin
   if not exists (
          select * from Stock, inserted
          where
            /* %JoinFKPK(inserted,Stock," = "," and") */
            inserted.OL_I_ID = Stock.S_I_ID and
            inserted.OL_SUPPLY_W_ID = Stock.S_W_ID)
   begin
      select @errno  = 30007,
             @errmsg = 'Cannot INSERT "Order_Line" because
"Stock" does not exist.'
      goto error
    end
  end

  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* District_Order R/29 Order_Line ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(OL_O_ID) or
    update(OL_D_ID) or
    update(OL_W_ID)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,District_Order
        where
          /* %JoinFKPK(inserted,District_Order) */
          inserted.OL_O_ID = District_Order.O_ID and
          inserted.OL_D_ID = District_Order.O_D_ID and
          inserted.OL_W_ID = District_Order.O_W_ID
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "Order_Line" because
"District_Order" does not exist.'
      goto error
    end
  end

  update District_Order
  set O_OL_CNT = O_OL_CNT + @numrows
  from District_Order, inserted
  where District_Order.O_W_ID = inserted.OL_W_ID and
              District_Order.O_D_ID = inserted.OL_D_ID and
              District_Order.O_ID = inserted.OL_O_ID

  update Customer
  set C_BALANCE = C_BALANCE + inserted.OL_AMOUNT
  from Customer, inserted, District_Order
  where
           Customer.C_ID = District_Order.O_C_ID and
           Customer.C_W_ID = District_Order.O_W_ID and
           Customer.C_D_ID = District_Order.O_D_ID and
           inserted.OL_W_ID = District_Order.O_W_ID and
           inserted.OL_D_ID = District_Order.O_D_ID and
           inserted.OL_O_ID = District_Order.O_ID

  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end


GO
CREATE  trigger tU_Order_Line on Order_Line for UPDATE as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* UPDATE trigger on Order_Line */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insOL_O_ID int,
           @insOL_D_ID smallint,
           @insOL_W_ID int,
           @insOL_NUMBER char(18),
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* Stock R/32 Order_Line ON CHILD UPDATE SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(OL_I_ID) or
    update(OL_SUPPLY_W_ID)
  begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "Order_Line" because
"Stock" cannot change.'
      goto error
  end

  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* District_Order R/29 Order_Line ON CHILD UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(OL_O_ID) or
    update(OL_D_ID) or
    update(OL_W_ID)
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "Order_Line" because
"District_Order" cannot change.'
      goto error
    end

  update Customer
  set C_BALANCE = C_BALANCE + inserted.OL_AMOUNT
  from Customer, inserted, District_Order
  where
           Customer.C_ID = District_Order.O_C_ID and
           Customer.C_W_ID = District_Order.O_W_ID and
           Customer.C_D_ID = District_Order.O_D_ID and
           inserted.OL_W_ID = District_Order.O_W_ID and
           inserted.OL_D_ID = District_Order.O_D_ID and
           inserted.OL_O_ID = District_Order.O_ID

  update Customer
  set C_BALANCE = C_BALANCE - deleted.OL_AMOUNT
  from Customer, deleted, District_Order
  where
           Customer.C_ID = District_Order.O_C_ID and
           Customer.C_W_ID = District_Order.O_W_ID and
           Customer.C_D_ID = District_Order.O_D_ID and
           deleted.OL_W_ID = District_Order.O_W_ID and
           deleted.OL_D_ID = District_Order.O_D_ID and
           deleted.OL_O_ID = District_Order.O_ID

  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end


GO
CREATE  trigger tD_Order_Line on Order_Line for DELETE as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* INSERT trigger on Order_Line */
begin

  declare  @numrows int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount

  update District_Order
  set O_OL_CNT = O_OL_CNT - @numrows
  from District_Order, deleted
  where District_Order.O_W_ID = deleted.OL_W_ID and
              District_Order.O_D_ID = deleted.OL_D_ID and
              District_Order.O_ID = deleted.OL_O_ID

  update Customer
  set C_BALANCE = C_BALANCE - deleted.OL_AMOUNT
  from Customer, deleted, District_Order
  where
           Customer.C_ID = District_Order.O_C_ID and
           Customer.C_W_ID = District_Order.O_W_ID and
           Customer.C_D_ID = District_Order.O_D_ID and
           deleted.OL_W_ID = District_Order.O_W_ID and
           deleted.OL_D_ID = District_Order.O_D_ID and
           deleted.OL_O_ID = District_Order.O_ID


  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end


GO
create trigger tD_Stock on Stock for DELETE as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* DELETE trigger on Stock */
begin
  declare  @errno   int,
           @errmsg  varchar(255)

    /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
```

```
end

GO
create trigger tI_Stock on Stock for INSERT as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* INSERT trigger on Stock */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* Warehouse R/31 Stock ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(S_W_ID)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,Warehouse
        where
          /* %JoinFKPK(inserted,Warehouse) */
          inserted.S_W_ID = Warehouse.W_ID
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "Stock" because "Warehouse"
does not exist.'
      goto error
    end
  end

  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* Item R/30 Stock ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(S_I_ID)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,Item
        where
          /* %JoinFKPK(inserted,Item) */
          inserted.S_I_ID = Item.I_ID
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "Stock" because "Item" does
not exist.'
      goto error
    end
  end


  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
create trigger tU_Stock on Stock for UPDATE as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* UPDATE trigger on Stock */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insS_I_ID int,
           @insS_W_ID int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* Stock R/32 Order_Line ON PARENT UPDATE SET NULL */
  if
    /* %ParentPK(" or",update) */
    update(S_I_ID) or
    update(S_W_ID)
  begin
    update Order_Line
      set
        /* %SetFK(Order_Line,NULL) */
        Order_Line.OL_I_ID = NULL,
        Order_Line.OL_SUPPLY_W_ID = NULL
      from Order_Line,deleted
      where
        /* %JoinFKPK(Order_Line,deleted," = "," and") */
        Order_Line.OL_I_ID = deleted.S_I_ID and
        Order_Line.OL_SUPPLY_W_ID = deleted.S_W_ID
  end

  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* Warehouse R/31 Stock ON CHILD UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(S_W_ID)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,Warehouse
        where
          /* %JoinFKPK(inserted,Warehouse) */
          inserted.S_W_ID = Warehouse.W_ID
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "Stock" because "Warehouse"
does not exist.'
      goto error
    end
  end

  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* Item R/30 Stock ON CHILD UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(S_I_ID)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,Item
        where
          /* %JoinFKPK(inserted,Item) */
          inserted.S_I_ID = Item.I_ID
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "Stock" because "Item" does
not exist.'
      goto error
    end
  end


  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
create trigger tD_Warehouse on Warehouse for DELETE as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* DELETE trigger on Warehouse */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
    /* Warehouse R/31 Stock ON PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,Stock
      where
        /*  %JoinFKPK(Stock,deleted," = "," and") */
        Stock.S_W_ID = deleted.W_ID
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "Warehouse" because "Stock"
exists.'
      goto error
    end

    /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
    /* Warehouse R/5 District ON PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,District
      where
        /*  %JoinFKPK(District,deleted," = "," and") */
        District.D_W_ID = deleted.W_ID
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "Warehouse" because
"District" exists.'
      goto error
    end


    /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
create trigger tU_Warehouse on Warehouse for UPDATE as
/* ERwin Builtin Wed Nov 11 13:38:20 1998 */
/* UPDATE trigger on Warehouse */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insW_ID int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* Warehouse R/31 Stock ON PARENT UPDATE RESTRICT */
  if
    /* %ParentPK(" or",update) */
    update(W_ID)
  begin
    if exists (
      select * from deleted,Stock
      where
```

```
        /*  %JoinFKPK(Stock,deleted," = "," and") */
        Stock.S_W_ID = deleted.W_ID
    )
    begin
      select @errno  = 30005,
             @errmsg = 'Cannot UPDATE "Warehouse" because "Stock"
exists.'
      goto error
    end
  end

  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  /* Warehouse R/5 District ON PARENT UPDATE RESTRICT */
  if
    /* %ParentPK(" or",update) */
    update(W_ID)
  begin
    if exists (
      select * from deleted,District
      where
        /*  %JoinFKPK(District,deleted," = "," and") */
        District.D_W_ID = deleted.W_ID
    )
    begin
      select @errno  = 30005,
             @errmsg = 'Cannot UPDATE "Warehouse" because
"District" exists.'
      goto error
    end
  end


  /* ERwin Builtin Wed Nov 11 13:38:20 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

GO
```

Deivery_Job

```
begin
  Declare @D_ID int,
              @Skipped int,
              @Job_Id int ,
              @NO_ID int,
              @CAR_ID int,
              @W_ID int ,
              @OL_Sum float,
              @ErrNum int

  select @D_ID = 1
  select @ErrNum = 0
  select @Skipped =0

  begin transaction
  While @D_ID<=10
  begin
            select @Job_ID = min(SC_ID), @W_ID = min(SC_W_ID),
                        @CAR_ID = min(SC_CARRIER_ID) from
Scheduled_Jobs

            if @@Error >0 Select @ErrNum = @ErrNum + 1

            if exists(select NO_O_ID from New_Order where
                        NO_D_ID = @D_ID and NO_W_ID = @W_ID)
            begin
              select @NO_ID = min(NO_O_ID) from New_Order  where
                        NO_D_ID = @D_ID and NO_W_ID = @W_ID

              if @@Error >0 Select @ErrNum = @ErrNum + 1

              update District_Order
                        set O_CARRIER_ID = @CAR_ID
                        from District_Order
                        where O_ID = @NO_ID and O_D_ID = @D_ID
and O_W_ID = @W_ID

              if @@Error >0 Select @ErrNum = @ErrNum + 1

              select @OL_Sum = SUM(OL_AMOUNT)
                        from  Order_Line
                        where OL_O_ID = @NO_ID and OL_D_ID =
@D_ID and OL_W_ID = @W_ID

              if @@Error >0 Select @ErrNum = @ErrNum + 1

update Customer
            set C_BALANCE = @OL_SUM,
                        C_DELIVERY_CNT = C_DELIVERY_CNT + 1
                  from  Customer, Order_Line,
District_Order
                  where O_ID = @NO_ID and O_D_ID = @D_ID
and O_W_ID = @W_ID and
                        O_C_ID = C_ID and O_D_ID = C_D_ID
and O_W_ID = C_W_ID and
                        OL_O_ID = @NO_ID and OL_D_ID =
@D_ID and OL_W_ID = @W_ID

              if @@Error >0 Select @ErrNum = @ErrNum + 1

              update Order_Line
                        set OL_DELIVERY_D = Getdate()
                        from  Order_Line
                        where OL_O_ID = @NO_ID and OL_D_ID =
@D_ID and OL_W_ID = @W_ID

              delete New_Order where NO_O_ID = @NO_Id and NO_D_ID
= @D_ID and NO_W_ID = @W_ID

              if @@Error >0 Select @ErrNum = @ErrNum + 1
            end
            else select @Skipped = @Skipped + 1
            select @D_ID = @D_ID + 1
  end

  insert into Executed_Jobs (J_EXEC_D, J_SKIPPED, J_SCHEDULED_D,
J_W_ID, J_CARRIER_ID)
  select getdate(), @Skipped , SC_DATE, SC_W_ID, SC_CARRIER_ID
from Scheduled_Jobs
            where SC_ID = @Job_Id

  if @@Error >0 Select @ErrNum = @ErrNum + 1

  delete Scheduled_Jobs where SC_ID = @Job_Id

  if @@Error >0 Select @ErrNum = @ErrNum + 1

  if @ErrNum > 0
    begin
            rollback transaction
            delete Scheduled_Jobs where SC_ID = @Job_Id
    end
  else
            commit transaction
end
```

Appendix C – Visual Basic Programs

1. Database Data Generator

```
Option Explicit
Const MAX_ITEM = 100000#
Const MAX_CUSTOMER = 3000
Const MAX_ORDER = 3000
Const ORDER_THRESHOLD = 2101
Const CUSTOMER_THRESHOLD = 999

Private LNSyllables(0 To 9) As String

Private Function GenerateLastNameStr(sCode As String) As String
Dim iIndex As Long
Dim sAux As String

iIndex = Val(Right(sCode, 1))
sAux = LNSyllables(iIndex)
If Len(sCode) = 2 Then
    iIndex = Val(Left(sCode, 1))
    sAux = LNSyllables(iIndex) & sAux
ElseIf Len(sCode) > 2 Then
    iIndex = Val(Mid(sCode, 2, 1))
    sAux = LNSyllables(iIndex) & sAux
    iIndex = Val(Left(sCode, 1))
    sAux = LNSyllables(iIndex) & sAux
End If
GenerateLastNameStr = sAux

End Function

Private Function GenerateStr(iLen As Long) As String
Dim i As Long
Dim sAux As String
Dim cAux As String

For i = 1 To iLen
    cAux = Chr(Int(58 * Rnd) + 32)
    sAux = sAux & cAux
Next
GenerateStr = sAux
End Function

Private Function GenerateNumberStr(iLen As Long) As String
Dim i As Long
Dim sAux As String
Dim cAux As String

For i = 1 To iLen
    cAux = Chr(Int(10 * Rnd) + 48)
    sAux = sAux & cAux
Next
GenerateNumberStr = sAux
End Function

Private Function GenerateNameStr(iLen As Long) As String
Dim i As Long
Dim sAux As String

For i = 1 To iLen
    sAux = sAux & Chr(Int(25 * Rnd) + 65)
Next
GenerateNameStr = sAux
End Function
Private Sub cmdGen_Click(index As Integer)
Dim i As Long
Dim j As Long
Dim k As Long
Dim l As Long
Dim iLength As Long
Dim iPos As String
Dim sAux1 As String
Dim sAux2 As String
Static bNoMessages As Boolean

Screen.MousePointer = vbHourglass
Select Case index
    Case 0:
        '// Delete all
       prgComplete.Max = 9
       ThesisEnv.OrderLineDelete
       prgComplete = 1
       DoEvents
       ThesisEnv.NewOrderDelete
       prgComplete = 2
       DoEvents
       ThesisEnv.OrderDelete
       prgComplete = 3
       DoEvents
       ThesisEnv.HistoryDelete
       prgComplete = 4
       DoEvents
       ThesisEnv.CustomerDelete
       prgComplete = 5
       DoEvents
       ThesisEnv.DistrictDelete
       prgComplete = 6
       DoEvents
       ThesisEnv.StockDelete
       prgComplete = 7
       DoEvents
       ThesisEnv.ItemDelete
       prgComplete = 8
       DoEvents
       ThesisEnv.WarehouseDelete
       prgComplete = 9
       DoEvents
       prgComplete = 0
       If Not bNoMessages Then
        MsgBox "Data deleted!", vbInformation
       End If
    Case 1:
        '// Item
        With ThesisEnv.rsItemInput
        prgComplete.Max = MAX_ITEM
        .Open
        For i = 1 To MAX_ITEM
            prgComplete = i
            DoEvents
            sAux1 = ""
            sAux2 = ""
            .AddNew
                Randomize
                !I_ID = i
                !I_IM_ID = Int(10001 * Rnd + 1)
'// [1 .. 10,000]
                !I_NAME = GenerateNameStr(Int(11 * Rnd + 14))
'// [14 .. 24]
                !I_PRICE = Int(10001 * Rnd + 1) / 100
'// [1.00 .. 10.00]
                iLength = Int(25 * Rnd + 26)
'// [26 .. 50]
                sAux1 = GenerateStr(iLength)
                If Int(100 * Rnd + 1) <= 10 Then
                    '// 10% of the cases
                    iPos = Int(iLength * Rnd + 1)
                    If iPos > 42 Then iPos = 42
'// Otherwise, ORIGINAL wouldn't fit
                    If iPos > 1 Then
                        sAux2 = Left(sAux1, iPos)
                    End If
                    sAux2 = sAux2 & "ORIGINAL"
                    If iPos + 8 < iLength Then
                        sAux2 = sAux2 & Right(sAux1, iLength -
iPos - 8)
                    End If
                    sAux1 = sAux2
                End If
                !I_DATA = sAux1
            .Update
        Next
        .Close
        DoEvents
        prgComplete = 0
        End With
        If Not bNoMessages Then
            MsgBox "Item generated!", vbInformation
        End If
    Case 2:
        '// Warehouse
        With ThesisEnv.rsWarehouseInput
        .Open
        .AddNew
        !W_ID = 1
        !W_NAME = "Warehouse1"
        !W_STREET_1 = "1234 Noname St."
        !W_STREET_2 = "12th Floor Room 1345"
        !W_CITY = "Washington"
        !W_STATE = "OH"
        !W_ZIP = 454311111
        !W_TAX = 0.065
        !W_YTD = 0
        .Update
        .Close
        If Not bNoMessages Then
            MsgBox "Warehouse generated!", vbInformation
        End If
        End With
    Case 3:
        '// Stock
        With ThesisEnv.rsStockInput
        prgComplete.Max = MAX_ITEM
        .Open
        For i = 1 To MAX_ITEM
            prgComplete = i
            DoEvents
            sAux1 = ""
            sAux2 = ""
            .AddNew
                Randomize
                !S_I_ID = i
                !S_W_ID = 1
                !S_QUANTITY = Int(90 * Rnd + 10)
'// [10 .. 100]
                !S_DIST_01 = GenerateNameStr(24)
```

```
                !S_DIST_02 = GenerateNameStr(24)
                !S_DIST_03 = GenerateNameStr(24)
                !S_DIST_04 = GenerateNameStr(24)
                !S_DIST_05 = GenerateNameStr(24)
                !S_DIST_06 = GenerateNameStr(24)
                !S_DIST_07 = GenerateNameStr(24)
                !S_DIST_08 = GenerateNameStr(24)
                !S_DIST_09 = GenerateNameStr(24)
                !S_DIST_10 = GenerateNameStr(24)
                iLength = Int(25 * Rnd + 26)
'// [26 .. 50]
                sAux1 = GenerateStr(iLength)
                If Int(100 * Rnd + 1) <= 10 Then
                    '// 10% of the cases
                    iPos = Int(iLength * Rnd + 1)
                    If iPos > 42 Then iPos = 42
'// Otherwise, ORIGINAL wouldn't fit
                    If iPos > 1 Then
                        sAux2 = Left(sAux1, iPos)
                    End If
                    sAux2 = sAux2 & "ORIGINAL"
                    If iPos + 8 < iLength Then
                        sAux2 = sAux2 & Right(sAux1, iLength -
iPos - 8)
                    End If
                    sAux1 = sAux2
                End If
                !S_DATA = sAux1
                !S_YTD = 0
                !S_ORDER_CNT = 0
                !S_REMOTE_CNT = 0
            .Update
        Next
        .Close
        DoEvents
        prgComplete = 0
        End With
        If Not bNoMessages Then
            MsgBox "Stock generated!", vbInformation
        End If
    Case 4:
        '// District
        With ThesisEnv.rsDistrictInput
        prgComplete.Max = 10
        .Open
        For i = 1 To 10
            Randomize
            prgComplete = i
            .AddNew
            !D_ID = i
            !D_W_ID = 1
            !D_NAME = GenerateNameStr(Int(5 * Rnd + 6))
'// [6 .. 10]
            !D_STREET_1 = GenerateNameStr(Int(11 * Rnd + 10))
'// [10 .. 20]
            !D_STREET_2 = GenerateNameStr(Int(11 * Rnd + 10))
'// [10 .. 20]
            !D_CITY = GenerateNameStr(Int(11 * Rnd + 10))
'// [10 .. 20]
            !D_STATE = GenerateNameStr(2)
            !D_ZIP = Int(89999 * Rnd + 10000) & "1111"
            !D_TAX = Int(3 * Rnd) / 10
'// [0.0 .. 0.2]
            !D_YTD = 30000
            !D_NEXT_O_ID = 1
            .Update
        Next
        .Close
        If Not bNoMessages Then
            MsgBox "District generated!", vbInformation
        End If
        prgComplete = 0
        End With
    Case 5:
        '// Customer
        With ThesisEnv.rsCustomerInput
        prgComplete.Max = 10 * MAX_CUSTOMER
        .Open
        For i = 1 To 10
         For j = 1 To MAX_CUSTOMER
            DoEvents
            Randomize
            prgComplete = (i - 1) * MAX_CUSTOMER + j
            .AddNew
            !C_ID = j
            !C_D_ID = i
            !C_W_ID = 1
            !C_FIRST = GenerateNameStr(Int(9 * Rnd + 8))
'// [8 .. 16]
            !C_MIDDLE = "OE"
            If j > CUSTOMER_THRESHOLD Then
                !C_LAST = GenerateLastNameStr(Str(j))
            Else
                !C_LAST = GenerateLastNameStr(Str(Int(999 *
Rnd))) '// [000 .. 999]
            End If
            !C_STREET_1 = GenerateNameStr(Int(11 * Rnd + 10))
'// [10 .. 20]
            !C_STREET_2 = GenerateNameStr(Int(11 * Rnd + 10))
'// [10 .. 20]
            !C_CITY = GenerateNameStr(Int(11 * Rnd + 10))
'// [10 .. 20]
            !C_STATE = GenerateNameStr(2)
            !C_ZIP = Int(90000 * Rnd + 10000) & "1111"
'// [10000 .. 99999]
            !C_PHONE = GenerateNumberStr(16)
            !C_SINCE = Format(Date, "mm/dd/yyyy")
            If Int(100 * Rnd + 1) <= 10 Then
                '// 10% of the cases
                !C_CREDIT = "BC"
            Else
                !C_CREDIT = "GC"
            End If
            !C_CREDIT_LIM = 50000#
            !C_DISCOUNT = Int(6 * Rnd) / 10
'// [0.00 .. 0.5]
            !C_BALANCE = 0
            !C_YTD_PAYMENT = 10
            !C_PAYMENT_CNT = 1
            !C_DELIVERY_CNT = 0
            !C_DATA = GenerateStr(Int(201 * Rnd + 300))
'// [200 .. 500]
            .Update
         Next
        Next
        .Close
        If Not bNoMessages Then
            MsgBox "Customer generated!", vbInformation
        End If
        prgComplete = 0
        End With
    Case 6:
        '// History
        With ThesisEnv.rsHistoryInput
        prgComplete.Max = 10 * MAX_CUSTOMER
        .Open
        For i = 1 To 10
         For j = 1 To MAX_CUSTOMER
            DoEvents
            Randomize
            prgComplete = (i - 1) * MAX_CUSTOMER + j
            .AddNew
            !H_C_ID = j
            !H_C_D_ID = i
            !H_C_W_ID = 1
            !H_D_ID = i
            !H_W_ID = 1
            !H_DATE = Format(Date, "mm/dd/yyyy")
            !H_AMOUNT = 10
            !H_DATA = GenerateStr(Int(13 * Rnd + 12))
'// [12 .. 24]
            .Update
         Next
        Next
        .Close
        If Not bNoMessages Then
            MsgBox "History generated!", vbInformation
        End If
        prgComplete = 0
        End With
    Case 7:
        '// Order
        With ThesisEnv.rsOrderInput
        prgComplete.Max = MAX_ORDER * 10
        .Open
        For i = 1 To 10
          For j = 1 To MAX_ORDER
            DoEvents
            Randomize
            prgComplete = (i - 1) * MAX_ORDER + j
            .AddNew
            !O_ID = j
            !O_C_ID = Int(MAX_CUSTOMER * Rnd + 1)      '// [1 ..
MAX_CUSTOMER]
            !O_D_ID = i
            !O_W_ID = 1
            !O_ENTRY_DATE = Format(Date, "mm/dd/yyyy")
            If j < ORDER_THRESHOLD Then
                !O_CARRIER_ID = Int(11 * Rnd + 1)          '//
[1 .. 10]
            End If
            !O_ALL_LOCAL = 1
            !O_OL_CNT = 0
            .Update
         Next
        Next
        .Close
        If Not bNoMessages Then
            MsgBox "Order generated!", vbInformation
        End If
        prgComplete = 0
        End With
    Case 8:
        '// New_Order
        ThesisEnv.NewOrderAdjust ORDER_THRESHOLD
        If Not bNoMessages Then
            MsgBox "New Order adjusted!", vbInformation
        End If
    Case 9:
        '// Order_Line
        With ThesisEnv.rsOrderLineInput
        prgComplete.Max = MAX_ORDER * 10
        .Open
        For i = 1 To 10
          For j = 1 To MAX_ORDER
          k = Int(11 * Rnd + 5)                          '// [5 ..
15]
           For l = 1 To k
            DoEvents
            Randomize
            prgComplete = (i - 1) * MAX_ORDER + j
            .AddNew
            !OL_O_ID = j
            !OL_D_ID = i
            !OL_W_ID = 1
            !OL_NUMBER = l
            !OL_I_ID = Int(MAX_ITEM * Rnd + 1)       '// [1 ..
MAX_ITEM]
            !OL_SUPPLY_W_ID = 1
            If j < ORDER_THRESHOLD Then
                !OL_DELIVERY_D = Date
                !OL_AMOUNT = 0
            Else
                !OL_AMOUNT = Int(999999 * Rnd + 1) / 100   '//
[0.1 .. 9,999.99]
            End If
            !OL_QUANTITY = 5
            !OL_DIST_INFO = GenerateStr(24)
```

```
            .Update
          Next
        Next
      Next
      .Close
      If Not bNoMessages Then
          MsgBox "Order_Line generated!", vbInformation
      End If
      prgComplete = 0
      End With
  Case 10:
      bNoMessages = True
      For i = 0 To 9
          cmdGen_Click (i)
      Next
      bNoMessages = False
```

```
Client/Server Transactions Front-End
Option Explicit
Const STOCK_QUERY_NUMBER = 24
Private LNSyllables(0 To 9) As String

Private Sub PrintResult(sText, Optional iSpaces As Integer = 0,
Optional SizeToFit As Integer = 0, Optional bLineFeed As Boolean
= False)
Static bOldLine As Boolean

    If Not bOldLine Then
        iSpaces = iSpaces + 1
        bOldLine = True
    End If
    txtResult = txtResult & Space(iSpaces)
    txtResult = txtResult & sText
    If SizeToFit > 0 Then
        If Len(sText) < SizeToFit Then
            txtResult = txtResult & Space(SizeToFit - Len(sText))
        End If
    End If
    If bLineFeed Then
        txtResult = txtResult & vbCrLf
        bOldLine = False
    End If
End Sub
Private Function Stock_Level(W_ID As Long) As Boolean
On Error GoTo SError
Dim iMinThreshold As Integer
Dim lD_ID As Long
Dim lNO_ID As Long
Dim bTransaction As Boolean
Dim datStartTime As Date
Dim lSeconds As Long
Dim iMinutes As Integer

datStartTime = Time
DoEvents
Randomize
Stock_Level = False

iMinThreshold = Int(Rnd * 11) + 10
'// District
lD_ID = Int(Rnd() * 10) + 1

'// Begin Transaction
ThesisEnv.ThesisConn.BeginTrans
bTransaction = True
DoEvents

'// Get Next_Order_ID
ThesisEnv.DistrictQuery lD_ID, W_ID
lNO_ID = ThesisEnv.rsDistrictQuery!D_NEXT_O_ID - 21

'// Check Stock
ThesisEnv.StockLevelQuery lD_ID, W_ID, lNO_ID, iMinThreshold

PrintResult "Stock-Level", 32, , True
PrintResult "Warehouse: " & Format(W_ID, "0000")
PrintResult "District: " & Format(lD_ID, "00"), 3, , True
PrintResult "", , , True
PrintResult "Stock Level Threshold: " & Format(iMinThreshold,
"00"), , , True
PrintResult "", , , True
PrintResult "Low Stock: " &
Format(ThesisEnv.rsStockLevelQuery!Low_Stock, "00"), , , True
PrintResult "", , , True

Stock_Level = True

lSeconds = DateDiff("s", datStartTime, Time)
iMinutes = lSeconds / 60
lSeconds = lSeconds - iMinutes * 60
PrintResult "Time Elapsed: " & Format(iMinutes, "00") & ":" &
Format(lSeconds, "00")

bTransaction = False
ThesisEnv.ThesisConn.CommitTrans

SEnd:
On Error Resume Next
With ThesisEnv
.rsDistrictQuery.Close
.rsStockLevelQuery.Close
End With
Exit Function

SError:
If bTransaction Then
    bTransaction = False
    ThesisEnv.ThesisConn.RollbackTrans
End If
MsgBox Error, vbCritical
Resume SEnd
```

```
End Select
Screen.MousePointer = vbNormal
End Sub
Private Sub Form_Load()
LNSyllables(0) = "BAR"
LNSyllables(1) = "OUGHT"
LNSyllables(2) = "ABLE"
LNSyllables(3) = "PRI"
LNSyllables(4) = "PRES"
LNSyllables(5) = "ESE"
LNSyllables(6) = "ANTI"
LNSyllables(7) = "CALLY"
LNSyllables(8) = "ATION"
LNSyllables(9) = "EING"
End Sub
```

```
End Function

Private Function Delivery(W_ID As Long) As Boolean
On Error GoTo DError
Dim bTransaction As Boolean
Dim lCarrier_ID As Long
Dim datStartTime As Date
Dim lSeconds As Long
Dim iMinutes As Integer

datStartTime = Time
DoEvents
Randomize
Delivery = False

lCarrier_ID = Int(Rnd() * 10) + 1

With ThesisEnv.rsScheduledJobsInput
    .Open
    .AddNew
    !SC_W_ID = W_ID
    !SC_CARRIER_ID = lCarrier_ID
    !SC_DATE = Time
    .Update
End With

PrintResult "Order-Status", 35, , True
PrintResult "Warehouse: " & Format(W_ID, "0000"), , , True
PrintResult "", , , True
PrintResult "Carrier Number: " & Format(lCarrier_ID, "00"), , ,
True
PrintResult "", , , True
PrintResult "Execution Statuos: Delivery has been queued.", , ,
True
PrintResult "", , , True

Delivery = True
'MsgBox lD_ID & " " & lC_ID

lSeconds = DateDiff("s", datStartTime, Time)
iMinutes = lSeconds / 60
lSeconds = lSeconds - iMinutes * 60
PrintResult "Time Elapsed: " & Format(iMinutes, "00") & ":" &
Format(lSeconds, "00")

DEnd:
On Error Resume Next
With ThesisEnv
.rsScheduledJobsInput.Close
End With
Exit Function

DError:
If bTransaction Then
    bTransaction = False
    ThesisEnv.ThesisConn.RollbackTrans
End If
MsgBox Error, vbCritical
Resume DEnd

End Function

Private Function Order_Status(W_ID As Long) As Boolean
On Error GoTo OSError
Dim bTransaction As Boolean
Dim lD_ID As Long
Dim lC_ID As Long
Dim lO_ID As Long
Dim i As Integer
Dim j As Integer
Dim sLastName As String
Dim datStartTime As Date
Dim lSeconds As Long
Dim iMinutes As Integer
Dim sSql As String
Dim rstOrder As New ADODB.Recordset
Dim rstOrderLine As New ADODB.Recordset

Randomize
Order_Status = False

Do
    '// District
    lD_ID = Int(Rnd() * 10) + 1

    '// Customer
    'If Int(Rnd() * 10) + 1 > 6 Then
        '// Uses ID
        lC_ID = NURand(1023, 1, 3000)
    'Else
        '// Query Last Name
    '    lC_ID = 0
    '    Do While lC_ID = 0
    '        j = NURand(255, 0, 999)
```

```
'        sLastName = GenerateLastNameStr(Str(j))
'        ThesisEnv.CustomerQueryLast sLastName, W_ID, lD_ID
'        With ThesisEnv.rsCustomerQueryLast
'            If .RecordCount > 0 Then
'                i = .RecordCount / 2
'                .Move i
'                lC_ID = !C_ID
'            End If
'            .Close
'        End With
'    Loop
'End If

    '// Get the order with max O_ID
    ThesisEnv.OrderQuery lC_ID, W_ID, lD_ID
    If Not IsNull(ThesisEnv.rsOrderQuery!MaxId) Then
        Exit Do
    Else
        ThesisEnv.rsOrderQuery.Close
    End If
Loop

datStartTime = Time
DoEvents
lO_ID = ThesisEnv.rsOrderQuery!MaxId

'// Open Customer
ThesisEnv.CustomerQuery2 lC_ID, W_ID, lD_ID

'// Get the Order
sSql = "Select O_ID, O_ENTRY_DATE, O_CARRIER_ID from
district_order where "
sSql = sSql & "O_ID =" & lO_ID & " and O_W_ID=" & W_ID & " and
O_D_ID=" & lD_ID
rstOrder.Open sSql, ThesisEnv.ThesisConn, adOpenForwardOnly,
adLockReadOnly

'// Get the order line items
sSql = "Select OL_SUPPLY_W_ID, OL_I_ID, OL_QUANTITY, OL_AMOUNT,
OL_DELIVERY_D from Order_Line "
sSql = sSql & "where OL_O_ID=" & lO_ID & " and OL_W_ID=" & W_ID &
" and OL_D_ID=" & lD_ID
rstOrderLine.Open sSql, ThesisEnv.ThesisConn, adOpenForwardOnly,
adLockReadOnly

'// Print Results
PrintResult "Order-Status", 33, , True
PrintResult "Warehouse: " & Format(W_ID, "0000")
PrintResult "District: " & Format(lD_ID, "00"), 3, , True
PrintResult "Customer:  " & Format(lC_ID, "0000")
PrintResult ThesisEnv.rsCustomerQuery2!C_FIRST, 3, 16
PrintResult ThesisEnv.rsCustomerQuery2!C_MIDDLE, 1
PrintResult ThesisEnv.rsCustomerQuery2!C_LAST, 1, 16, True
PrintResult "Cust-Balance: "
PrintResult "$" & Format(ThesisEnv.rsCustomerQuery2!C_BALANCE,
"000000000.00"), , , True
PrintResult "", , , True
PrintResult "Order-Number: " & Format(rstOrder!o_id, "00000000")
PrintResult "Entry-Date: " & Format(rstOrder!o_entry_date, "dd-
mm-yyyy hh:mm:ss"), 3
PrintResult "Carrier-Number: ", 2
If Not IsNull(rstOrder!o_carrier_id) Then
    PrintResult Format(rstOrder!o_carrier_id, "00"), , , True
Else
    PrintResult "NULL", , , True
End If
PrintResult "Supp_W"
PrintResult "Item_Id", 7
PrintResult "Qty", 4
PrintResult "Amount", 5
PrintResult "Delivery-Date", 6, , True
Do While Not rstOrderLine.EOF
    With rstOrderLine
    PrintResult Format(!OL_SUPPLY_W_ID, "0000"), 1
    PrintResult Format(!OL_I_ID, "000000"), 8
    PrintResult Format(!OL_QUANTITY, "00"), 5
    PrintResult Format(!OL_AMOUNT, "$00000.00"), 5
    If Not IsNull(!OL_DELIVERY_D) Then
        PrintResult Format(!OL_DELIVERY_D, "dd-mm-yyyy"), 6, ,
True
    Else
        PrintResult "NULL", 6, , True
    End If
    .MoveNext
    End With
Loop

lSeconds = DateDiff("s", datStartTime, Time)
iMinutes = lSeconds / 60
lSeconds = lSeconds - iMinutes * 60
PrintResult "Time Elapsed: " & Format(iMinutes, "00") & ":" &
Format(lSeconds, "00")

Order_Status = True
'MsgBox lD_ID & " " & lC_ID

OSEnd:
On Error Resume Next
With ThesisEnv
.rsCustomerQuery2.Close
.rsOrderQuery.Close
End With
rstOrderLine.Close
rstOrder.Close
Exit Function

OSError:
If bTransaction Then
    bTransaction = False
    ThesisEnv.ThesisConn.RollbackTrans
End If
MsgBox Error, vbCritical
Resume OSEnd

End Function

Private Function Payment(W_ID As Long) As Boolean
On Error GoTo PError
Dim bTransaction As Boolean
Dim lD_ID As Long
Dim lC_ID As Long
Dim i As Integer
Dim j As Integer
Dim sAux As String
Dim sLastName As String
Dim datStartTime As Date
Dim lSeconds As Long
Dim iMinutes As Integer
Dim sngAmount As Single
Dim sData As String

datStartTime = Time
DoEvents
Randomize
Payment = False

'// District
lD_ID = Int(Rnd() * 10) + 1

'// Customer
'If Int(Rnd() * 10) + 1 > 6 Then
    '// Uses ID
'   lC_ID = NURand(1023, 1, 3000)
'Else
    '// Query Last Name
    lC_ID = 0
    Do While lC_ID = 0
        j = NURand(255, 0, 999)
        sLastName = GenerateLastNameStr(Str(j))
        ThesisEnv.CustomerQueryLast sLastName, W_ID, lD_ID
        With ThesisEnv.rsCustomerQueryLast
            If .RecordCount > 0 Then
                i = .RecordCount / 2
                .Move i
                lC_ID = !C_ID
            End If
            .Close
        End With
    Loop
'endif

'// Amount
sngAmount = Int(Rnd() * 500000) / 100 + 1

'// Begin Transaction
ThesisEnv.ThesisConn.BeginTrans
bTransaction = True
DoEvents

PrintResult "Payment", 33, , True
PrintResult "Date: " & Format(Time, "dd-mm-yyyy hh:mm:ss"), , ,
True
PrintResult "", , , True

'// Open Warehouse
ThesisEnv.WarehouseQuery2 W_ID
PrintResult "Warehouse: " & Format(W_ID, "0000")

'// Get Next Order ID
ThesisEnv.DistrictQuery2 W_ID, lD_ID
PrintResult "District: " & Format(lD_ID, "00"), 25, , True

PrintResult ThesisEnv.rsWarehouseQuery2!W_STREET_1, , 20
PrintResult ThesisEnv.rsDistrictQuery2!D_STREET_1, 20, 20, True
PrintResult ThesisEnv.rsWarehouseQuery2!W_STREET_2, , 20
PrintResult ThesisEnv.rsDistrictQuery2!D_STREET_2, 20, 20, True
PrintResult ThesisEnv.rsWarehouseQuery2!W_CITY, , 20
PrintResult ThesisEnv.rsWarehouseQuery2!W_STATE, 1
PrintResult Left(ThesisEnv.rsWarehouseQuery2!W_ZIP, 5) & "-" &
Right(ThesisEnv.rsWarehouseQuery2!W_ZIP, 4), 1
PrintResult ThesisEnv.rsDistrictQuery2!D_CITY, 6, 20
PrintResult ThesisEnv.rsDistrictQuery2!D_STATE, 1
PrintResult Left(ThesisEnv.rsDistrictQuery2!D_ZIP, 5) & "-" &
Right(ThesisEnv.rsDistrictQuery2!D_ZIP, 3), 1, , True
PrintResult "", , , True

'// Open Customer
ThesisEnv.CustomerQuery2 lC_ID, W_ID, lD_ID
sData = ThesisEnv.rsCustomerQuery2!C_DATA

'// Check Customer Credit
If ThesisEnv.rsCustomerQuery2!C_CREDIT = "BC" Then
    sAux = "Entry: " & lC_ID & ", " & lD_ID & "; " & W_ID & "; "
& lD_ID & "; " & W_ID & "; " & sngAmount & "."
    If Len(sData) - Len(sAux) > 500 Then
        ThesisEnv.rsCustomerQuery2!C_DATA = sAux & Left(sData,
500 - Len(sAux))
        sData = sAux & Left(sData, 500 - Len(sAux))
    Else
        ThesisEnv.rsCustomerQuery2!C_DATA = sAux & sData
        sData = sAux & sData
    End If
    ThesisEnv.rsCustomerQuery2.Update
End If

PrintResult "Customer:  " & Format(lC_ID, "0000"), , , True
PrintResult "Name:   "
PrintResult ThesisEnv.rsCustomerQuery2!C_FIRST, , 16
PrintResult ThesisEnv.rsCustomerQuery2!C_MIDDLE, 1
PrintResult ThesisEnv.rsCustomerQuery2!C_LAST, 1, 16
PrintResult "Since:  " &
Format(ThesisEnv.rsCustomerQuery2!C_SINCE, "dd-mm-yyyy"), 5, ,
True
PrintResult ThesisEnv.rsCustomerQuery2!C_STREET_1, 8, 20
PrintResult "Credit: " & ThesisEnv.rsCustomerQuery2!C_CREDIT, 21,
, True
PrintResult ThesisEnv.rsCustomerQuery2!C_STREET_2, 8, 20
```

```
PrintResult "%Disc:  " &
Format(ThesisEnv.rsCustomerQuery2!C_DISCOUNT * 100, "00.00"), 21,
, True
PrintResult ThesisEnv.rsCustomerQuery2!C_CITY, 8, 20
PrintResult ThesisEnv.rsCustomerQuery2!C_STATE, 1
PrintResult Left(ThesisEnv.rsCustomerQuery2!C_ZIP, 5) & "-" &
Right(ThesisEnv.rsCustomerQuery2!C_ZIP, 3), 1
PrintResult "Phone:  " & Left(ThesisEnv.rsCustomerQuery2!C_PHONE,
6) & "-" & Mid(ThesisEnv.rsCustomerQuery2!C_PHONE, 7, 3) & "-" &
Mid(ThesisEnv.rsCustomerQuery2!C_PHONE, 10, 3) & "-" &
Right(ThesisEnv.rsCustomerQuery2!C_PHONE, 4), 8, , True
PrintResult "", , , True

PrintResult "Amount Paid:"
PrintResult Format(sngAmount, "$0000.00"), 10
PrintResult "New Cust-Balance:", 6
PrintResult "$" & Format(ThesisEnv.rsCustomerQuery2!C_BALANCE -
sngAmount, "000000000.00"), 1, , True
PrintResult "Credit Limit:"
PrintResult "$" & Format(ThesisEnv.rsCustomerQuery2!C_CREDIT_LIM,
"000000000.00"), 4, , True
PrintResult "", , , True
If ThesisEnv.rsCustomerQuery2!C_CREDIT = "BC" Then
    PrintResult "Cust-Data:"
    PrintResult Left(sData, 50), 1, , True
    PrintResult Mid(sData, 51, 50), 11, , True
    PrintResult Mid(sData, 101, 50), 11, , True
    PrintResult Mid(sData, 151, 50), 11, , True
    PrintResult "", , , True
End If

With ThesisEnv.rsHistoryInput
    .Open
    .AddNew
    !H_C_ID = lC_ID
    !H_C_D_ID = lD_ID
    !H_C_W_ID = W_ID
    !H_D_ID = lD_ID
    !H_W_ID = W_ID
    !H_DATE = Date
    !H_AMOUNT = sngAmount
    !H_DATA = ThesisEnv.rsWarehouseQuery2!W_NAME & "    " &
ThesisEnv.rsDistrictQuery2!D_NAME
    .Update
End With

lSeconds = DateDiff("s", datStartTime, Time)
iMinutes = lSeconds / 60
lSeconds = lSeconds - iMinutes * 60
PrintResult "Time Elapsed: " & Format(iMinutes, "00") & ":" &
Format(lSeconds, "00")

bTransaction = False
ThesisEnv.ThesisConn.CommitTrans

Payment = True
'MsgBox lD_ID & " " & lC_ID

PEnd:
On Error Resume Next
With ThesisEnv
.rsCustomerQuery2.Close
.rsHistoryInput.Close
.rsDistrictQuery2.Close
.rsWarehouseQuery2.Close
End With
Exit Function

PError:
If bTransaction Then
    bTransaction = False
    ThesisEnv.ThesisConn.RollbackTrans
End If
MsgBox Error, vbCritical
Resume 'PEnd

End Function

Private Function ClientTransaction(W_ID As Long) As Boolean
On Error GoTo CTError
Dim bTransaction As Boolean
Dim iOrder_Cnt As Integer
Dim lD_ID As Long
Dim lC_ID As Long
Dim lO_ID As Long
Dim lI_ID As Long
Dim i As Integer
Dim j As Integer
Dim bBg As Boolean
Dim iQuantity As Integer
Dim sSql As String
Dim sngTotal As Single
Dim datStartTime As Date
Dim lSeconds As Long
Dim iMinutes As Integer

datStartTime = Time
DoEvents
Randomize
ClientTransaction = False

'// Number of lines  [5..15]
iOrder_Cnt = Int(Rnd() * 11) + 5

'// District
lD_ID = Int(Rnd() * 10) + 1

'// Customer
lC_ID = NURand(1023, 1, 3000)

'// Fix SQL Statement for Stock
sSql = "Select S_QUANTITY, S_DATA, S_ORDER_CNT, S_YTD, S_DIST_" &
Format(lD_ID, "00") & " as S_DIST"
sSql = sSql & " from Stock where S_I_ID=? and S_W_ID=?"

ThesisEnv.Commands(STOCK_QUERY_NUMBER).CommandText = sSql

'// Begin Transaction
ThesisEnv.ThesisConn.BeginTrans
bTransaction = True
DoEvents

PrintResult "New Order", 33, , True

'// Open Warehouse
ThesisEnv.WarehouseQuery W_ID
PrintResult "Warehouse: " & Format(W_ID, "0000")

'// OpenCustomer
ThesisEnv.CustomerQuery lC_ID, W_ID, lD_ID

'// Get Next Order ID
ThesisEnv.DistrictQuery lD_ID, W_ID
lO_ID = ThesisEnv.rsDistrictQuery!D_NEXT_O_ID
PrintResult "District: " & Format(lD_ID, "00"), 3

PrintResult "Date: " & Format(Time, "dd-mm-yyyy hh:mm:ss"), 23, ,
True
PrintResult "Customer:  " & Format(lC_ID, "0000")
PrintResult "Name: " & Left(ThesisEnv.rsCustomerQuery!C_LAST,
16), 3, 21
PrintResult "Credit: " & ThesisEnv.rsCustomerQuery!C_CREDIT, 3
PrintResult "%Disc: " &
Format(ThesisEnv.rsCustomerQuery!C_DISCOUNT * 100, "00.00"), 3, ,
True
PrintResult "Order Number: " & Format(lO_ID, "00000000")
PrintResult "Number of Lines: " & Format(iOrder_Cnt, "00"), 2
PrintResult "W_tax: " & Format(ThesisEnv.rsWarehouseQuery!W_TAX *
100, "00.00"), 7
PrintResult "D_tax: " & Format(ThesisEnv.rsDistrictQuery!D_TAX *
100, "00.00"), 3, , True
PrintResult "", , , True
PrintResult "Supp_W", 1
PrintResult "Item_Id", 2
PrintResult "Item_Name", 2
PrintResult "Qty", 16
PrintResult "Stock", 2
PrintResult "B/G", 2
PrintResult "Price", 2
PrintResult "Amount", 4, , True

'// Insert District_Order
With ThesisEnv.rsOrderInput
    .Open
    .AddNew
    !o_id = lO_ID
    !O_C_ID = lC_ID
    !O_D_ID = lD_ID
    !O_W_ID = W_ID
    !o_entry_date = Format(Date, "mm/dd/yyyy")
    !O_ALL_LOCAL = 1
    !O_OL_CNT = 0
    .Update
End With

With ThesisEnv.rsOrderLineInput
.Open
sngTotal = 0
For i = 1 To iOrder_Cnt
    '// Find the item
    lI_ID = NURand(8191, 1, 100000)
    '// Quantity
    iQuantity = Int(Rnd() * 10) + 1

    '// Open Item
    ThesisEnv.ItemQuery lI_ID

    '// Open Stock
    ThesisEnv.StockQuery lI_ID, W_ID

    '// Update STOCK table
    If ThesisEnv.rsStockQuery!S_QUANTITY >= iQuantity + 10 Then
        ThesisEnv.rsStockQuery!S_QUANTITY =
ThesisEnv.rsStockQuery!S_QUANTITY - iQuantity
    Else
        ThesisEnv.rsStockQuery!S_QUANTITY =
ThesisEnv.rsStockQuery!S_QUANTITY + 91
    End If
    ThesisEnv.rsStockQuery!S_YTD = ThesisEnv.rsStockQuery!S_YTD +
iQuantity
    ThesisEnv.rsStockQuery!S_ORDER_CNT =
ThesisEnv.rsStockQuery!S_ORDER_CNT + 1
    ThesisEnv.rsStockQuery.Update

    '// Check for B/G
    bBg = False
    If InStr(ThesisEnv.rsItemQuery!I_DATA, "ORIGINAL") > 0 Then
        If InStr(ThesisEnv.rsStockQuery!S_DATA, "ORIGINAL") > 0
Then
            bBg = True
        End If
    End If

    '// Insert Order_Line
    .AddNew
    !OL_O_ID = lO_ID
    !OL_D_ID = lD_ID
    !OL_W_ID = W_ID
    !OL_NUMBER = i
    !OL_I_ID = lI_ID
    !OL_SUPPLY_W_ID = W_ID
    !OL_AMOUNT = iQuantity * ThesisEnv.rsItemQuery!I_PRICE
    !OL_QUANTITY = iQuantity
    !OL_DIST_INFO = ThesisEnv.rsStockQuery!S_DIST
    .Update

    sngTotal = sngTotal + iQuantity *
ThesisEnv.rsItemQuery!I_PRICE
```

```
    PrintResult Format(W_ID, "0000"), 2
    PrintResult Format(lI_ID, "000000"), 3
    PrintResult Left(ThesisEnv.rsItemQuery!I_NAME, 23), 3, 23
    PrintResult Format(iQuantity, "00"), 2
    PrintResult Format(ThesisEnv.rsStockQuery!S_QUANTITY, "000"),
4
    PrintResult IIf(bBg, "B", "G"), 4
    PrintResult Format(ThesisEnv.rsItemQuery!I_PRICE, "$000.00"),
3
    PrintResult Format(ThesisEnv.rsItemQuery!I_PRICE * iQuantity,
"$000.00"), 2, , True
    DoEvents

    '// Close Item and Stock
    ThesisEnv.rsItemQuery.Close
    ThesisEnv.rsStockQuery.Close
Next

lSeconds = DateDiff("s", datStartTime, Time)
iMinutes = lSeconds / 60
lSeconds = lSeconds - iMinutes * 60
PrintResult "Execution Status: Ok"
PrintResult "Time Elapsed: " & Format(iMinutes, "00") & ":" &
Format(lSeconds, "00"), 10
PrintResult "Total:  " & Format(sngTotal, "$0000.00"), 11, , True


End With

bTransaction = False
ThesisEnv.ThesisConn.CommitTrans

ClientTransaction = True
'MsgBox lD_ID & " " & lC_ID

CTEnd:
On Error Resume Next
With ThesisEnv
.rsWarehouseQuery.Close
.rsDistrictQuery.Close
.rsCustomerQuery.Close
.rsOrderLineInput.Close
.rsOrderInput.Close
End With
Exit Function

CTError:
If bTransaction Then
    bTransaction = False
    ThesisEnv.ThesisConn.RollbackTrans
End If
MsgBox Error, vbCritical
Resume CTEnd
```

```
End Function

Private Sub cmdTransaction_Click(Index As Integer)
Dim bResult As Boolean

txtResult = ""
Screen.MousePointer = vbHourglass
Select Case Index
    Case 0
        bResult = ClientTransaction(1)
    Case 1
        bResult = Payment(1)
    Case 2
        bResult = Order_Status(1)
    Case 3
        bResult = Delivery(1)
    Case 4
        bResult = Stock_Level(1)
End Select
Screen.MousePointer = vbNormal

End Sub
Private Function GenerateLastNameStr(sCode As String) As String
Dim iIndex As Long
Dim sAux As String

iIndex = Val(Right(Trim(sCode), 1))
sAux = LNSyllables(iIndex)
If Len(sCode) = 2 Then
    iIndex = Val(Left(Trim(sCode), 1))
    sAux = LNSyllables(iIndex) & sAux
ElseIf Len(sCode) > 2 Then
    iIndex = Val(Mid(Trim(sCode), 2, 1))
    sAux = LNSyllables(iIndex) & sAux
    iIndex = Val(Left(Trim(sCode), 1))
    sAux = LNSyllables(iIndex) & sAux
End If
GenerateLastNameStr = sAux

End Function
Private Sub Form_Load()
ThesisEnv.ThesisConn.Open
LNSyllables(0) = "BAR"
LNSyllables(1) = "OUGHT"
LNSyllables(2) = "ABLE"
LNSyllables(3) = "PRI"
LNSyllables(4) = "PRES"
LNSyllables(5) = "ESE"
LNSyllables(6) = "ANTI"
LNSyllables(7) = "CALLY"
LNSyllables(8) = "ATION"
LNSyllables(9) = "EING"
End Sub
```

2. Client/Server Transactions Front-End

```
Option Explicit '// Module General

Public Function Random(x As Long, y As Long) As Long
Randomize
Random = Int(Rnd() * (y - x)) + x
End Function

Public Function NURand(A As Long, x As Long, y As Long) As Long
Dim C As Long

C = A / 2
NURand = (((Random(0, A) Or Random(x, y)) + C) Mod (y - x + 1)) +
x
End Function

Private Function GenerateStr(iLen As Long) As String
Dim i As Long
Dim sAux As String
Dim cAux As String

For i = 1 To iLen
    cAux = Chr(Int(58 * Rnd) + 32)
    sAux = sAux & cAux
Next
GenerateStr = sAux
End Function

Option Explicit
Const STOCK_QUERY_NUMBER = 24
Private LNSyllables(0 To 9) As String

Private Sub PrintResult(sText, Optional iSpaces As Integer = 0,
Optional SizeToFit As Integer = 0, Optional bLineFeed As Boolean
= False)
Static bOldLine As Boolean

    If Not bOldLine Then
        iSpaces = iSpaces + 1
        bOldLine = True
    End If
    txtResult = txtResult & Space(iSpaces)
    txtResult = txtResult & sText
    If SizeToFit > 0 Then
        If Len(sText) < SizeToFit Then
            txtResult = txtResult & Space(SizeToFit - Len(sText))
        End If
    End If
    If bLineFeed Then
        txtResult = txtResult & vbCrLf
        bOldLine = False
    End If
End Sub
Private Function Stock_Level(W_ID As Long) As Boolean
On Error GoTo SError
Dim iMinThreshold As Integer
Dim lD_ID As Long
Dim lNO_ID As Long
Dim bTransaction As Boolean
Dim datStartTime As Date
Dim lSeconds As Long
Dim iMinutes As Integer

datStartTime = Time

Randomize
Stock_Level = False

iMinThreshold = Int(Rnd * 11) + 10
'// District
lD_ID = Int(Rnd() * 10) + 1


'// Get Next_Order_ID
ThesisEnv.DistrictQuery lD_ID, W_ID
lNO_ID = ThesisEnv.rsDistrictQuery!D_NEXT_O_ID - 21

'// Check Stock
ThesisEnv.StockLevelQuery lD_ID, W_ID, lNO_ID, iMinThreshold

PrintResult "Stock-Level", 32, , True
PrintResult "Warehouse: " & Format(W_ID, "0000")
PrintResult "District: " & Format(lD_ID, "00"), 3, , True
PrintResult "", , , True
PrintResult "Stock Level Threshold: " & Format(iMinThreshold,
"00"), , , True
PrintResult "", , , True
PrintResult "Low Stock: " &
Format(ThesisEnv.rsStockLevelQuery!Low_Stock, "00"), , , True
PrintResult "", , , True

Stock_Level = True


SEnd:
On Error Resume Next
With ThesisEnv
.rsDistrictQuery.Close
.rsStockLevelQuery.Close
End With
Exit Function

SError:
If bTransaction Then
    bTransaction = False
    ThesisEnv.ThesisConn.RollbackTrans
End If
MsgBox Error, vbCritical
Resume SEnd

End Function

Private Function Delivery(W_ID As Long) As Boolean
On Error GoTo DError
Dim bTransaction As Boolean
Dim lCarrier_ID As Long
Dim datStartTime As Date
Dim lSeconds As Long
Dim iMinutes As Integer

datStartTime = Time

Randomize
Delivery = False

lCarrier_ID = Int(Rnd() * 10) + 1

With ThesisEnv.rsScheduledJobsInput
    .Open
    .AddNew
    !SC_W_ID = W_ID
    !SC_CARRIER_ID = lCarrier_ID
    !SC_DATE = Time
    .Update
End With

PrintResult "Order-Status", 35, , True
PrintResult "Warehouse: " & Format(W_ID, "0000"), , , True
PrintResult "", , , True
PrintResult "Carrier Number: " & Format(lCarrier_ID, "00"), , ,
True
PrintResult "", , , True
PrintResult "Execution Statuos: Delivery has been queued.", , ,
True
PrintResult "", , , True

Delivery = True
'MsgBox lD_ID & " " & lC_ID

lSeconds = DateDiff("s", datStartTime, Time)
iMinutes = lSeconds / 60
lSeconds = lSeconds - iMinutes * 60
PrintResult "Time Elapsed: " & Format(iMinutes, "00") & ":" &
Format(lSeconds, "00")

DEnd:
On Error Resume Next
With ThesisEnv
.rsScheduledJobsInput.Close
End With
Exit Function

DError:
If bTransaction Then
    bTransaction = False
    ThesisEnv.ThesisConn.RollbackTrans
End If
MsgBox Error, vbCritical
Resume DEnd

End Function

Private Function Order_Status(W_ID As Long) As Boolean
On Error GoTo OSError
Dim bTransaction As Boolean
Dim lD_ID As Long
Dim lC_ID As Long
Dim lO_ID As Long
Dim i As Integer
Dim j As Integer
Dim sLastName As String
Dim datStartTime As Date
Dim lSeconds As Long
Dim iMinutes As Integer
Dim sSql As String
Dim rstOrder As New ADODB.Recordset
Dim rstOrderLine As New ADODB.Recordset

datStartTime = Time

Randomize
Order_Status = False

Do
    '// District
    lD_ID = Int(Rnd() * 10) + 1

    '// Customer
    'If Int(Rnd() * 10) + 1 > 6 Then
        '// Uses ID
        lC_ID = NURand(1023, 1, 3000)
    'Else
        '// Query Last Name
    '    lC_ID = 0
    '    Do While lC_ID = 0
    '        j = NURand(255, 0, 999)
    '        sLastName = GenerateLastNameStr(Str(j))
```

```
'        ThesisEnv.CustomerQueryLast sLastName, W_ID, lD_ID
'        With ThesisEnv.rsCustomerQueryLast
'            If .RecordCount > 0 Then
'                i = .RecordCount / 2
'                .Move i
'                lC_ID = !C_ID
'            End If
'            .Close
'        End With
'    Loop
'End If

    '// Get the order with max O_ID
    ThesisEnv.OrderQuery lC_ID, W_ID, lD_ID
    If Not IsNull(ThesisEnv.rsOrderQuery!MaxId) Then
        Exit Do
    Else
        ThesisEnv.rsOrderQuery.Close
    End If
Loop

lO_ID = ThesisEnv.rsOrderQuery!MaxId

'// Open Customer
ThesisEnv.CustomerQuery2 lC_ID, W_ID, lD_ID

'// Get the Order
sSql = "Select O_ID, O_ENTRY_DATE, O_CARRIER_ID from
district_order where "
sSql = sSql & "O_ID =" & lO_ID & " and O_W_ID=" & W_ID & " and
O_D_ID=" & lD_ID
rstOrder.Open sSql, ThesisEnv.ThesisConn, adOpenForwardOnly,
adLockReadOnly

'// Get the order line items
sSql = "Select OL_SUPPLY_W_ID, OL_I_ID, OL_QUANTITY, OL_AMOUNT,
OL_DELIVERY_D from Order_Line "
sSql = sSql & "where OL_O_ID=" & lO_ID & " and OL_W_ID=" & W_ID &
" and OL_D_ID=" & lD_ID
rstOrderLine.Open sSql, ThesisEnv.ThesisConn, adOpenForwardOnly,
adLockReadOnly

'// Print Results
PrintResult "Order-Status", 33, , True
PrintResult "Warehouse: " & Format(W_ID, "0000")
PrintResult "District: " & Format(lD_ID, "00"), 3, , True
PrintResult "Customer:  " & Format(lC_ID, "0000")
PrintResult ThesisEnv.rsCustomerQuery2!C_FIRST, 3, 16
PrintResult ThesisEnv.rsCustomerQuery2!C_MIDDLE, 1
PrintResult ThesisEnv.rsCustomerQuery2!C_LAST, 1, 16, True
PrintResult "Cust-Balance: "
PrintResult "$" & Format(ThesisEnv.rsCustomerQuery2!C_BALANCE,
"000000000.00"), , , True
PrintResult "", , , True
PrintResult "Order-Number: " & Format(rstOrder!o_id, "00000000")
PrintResult "Entry-Date: " & Format(rstOrder!o_entry_date, "dd-
mm-yyyy hh:mm:ss"), 3
PrintResult "Carrier-Number: ", 2
If Not IsNull(rstOrder!o_carrier_id) Then
    PrintResult Format(rstOrder!o_carrier_id, "00"), , , True
Else
    PrintResult "NULL", , , True
End If
PrintResult "Supp_W"
PrintResult "Item_Id", 7
PrintResult "Qty", 4
PrintResult "Amount", 5
PrintResult "Delivery-Date", 6, , True
Do While Not rstOrderLine.EOF
    With rstOrderLine
    PrintResult Format(!OL_SUPPLY_W_ID, "0000"), 1
    PrintResult Format(!OL_I_ID, "000000"), 8
    PrintResult Format(!OL_QUANTITY, "00"), 5
    PrintResult Format(!OL_AMOUNT, "$00000.00"), 5
    If Not IsNull(!OL_DELIVERY_D) Then
        PrintResult Format(!OL_DELIVERY_D, "dd-mm-yyyy"), 6, ,
True
    Else
        PrintResult "NULL", 6, , True
    End If
    .MoveNext
    End With
Loop

lSeconds = DateDiff("s", datStartTime, Time)
iMinutes = lSeconds / 60
lSeconds = lSeconds - iMinutes * 60
PrintResult "Time Elapsed: " & Format(iMinutes, "00") & ":" &
Format(lSeconds, "00")

Order_Status = True
'MsgBox lD_ID & " " & lC_ID

OSEnd:
On Error Resume Next
With ThesisEnv
.rsCustomerQuery2.Close
.rsOrderQuery.Close
End With
rstOrderLine.Close
rstOrder.Close
Exit Function

OSError:
If bTransaction Then
    bTransaction = False
    ThesisEnv.ThesisConn.RollbackTrans
End If
MsgBox Error, vbCritical
Resume OSEnd

End Function
```

```
Private Function Payment(W_ID As Long) As Boolean
On Error GoTo PError
Dim bTransaction As Boolean
Dim lD_ID As Long
Dim lC_ID As Long
Dim i As Integer
Dim j As Integer
Dim sAux As String
Dim sLastName As String
Dim datStartTime As Date
Dim lSeconds As Long
Dim iMinutes As Integer
Dim sngAmount As Single
Dim sData As String

datStartTime = Time

Randomize
Payment = False

'// District
lD_ID = Int(Rnd() * 10) + 1

'// Customer
'If Int(Rnd() * 10) + 1 > 6 Then
    '// Uses ID
'   lC_ID = NURand(1023, 1, 3000)
'Else
    '// Query Last Name
    lC_ID = 0
    Do While lC_ID = 0
        j = NURand(255, 0, 999)
        sLastName = GenerateLastNameStr(Str(j))
        ThesisEnv.CustomerQueryLast sLastName, W_ID, lD_ID
        With ThesisEnv.rsCustomerQueryLast
            If .RecordCount > 0 Then
                i = .RecordCount / 2
                .Move i
                lC_ID = !C_ID
            End If
            .Close
        End With
    Loop
'endif

'// Amount
sngAmount = Int(Rnd() * 500000) / 100 + 1

'// Begin Transaction
ThesisEnv.ThesisConn.BeginTrans
bTransaction = True


PrintResult "Payment", 33, , True
PrintResult "Date: " & Format(Time, "dd-mm-yyyy hh:mm:ss"), , ,
True
PrintResult "", , , True

'// Open Warehouse
ThesisEnv.WarehouseQuery2 W_ID
PrintResult "Warehouse: " & Format(W_ID, "0000")

'// Get Next Order ID
ThesisEnv.DistrictQuery2 W_ID, lD_ID
PrintResult "District: " & Format(lD_ID, "00"), 25, , True

PrintResult ThesisEnv.rsWarehouseQuery2!W_STREET_1, , 20
PrintResult ThesisEnv.rsDistrictQuery2!D_STREET_1, 20, 20, True
PrintResult ThesisEnv.rsWarehouseQuery2!W_STREET_2, , 20
PrintResult ThesisEnv.rsDistrictQuery2!D_STREET_2, 20, 20, True
PrintResult ThesisEnv.rsWarehouseQuery2!W_CITY, , 20
PrintResult ThesisEnv.rsWarehouseQuery2!W_STATE, 1
PrintResult Left(ThesisEnv.rsWarehouseQuery2!W_ZIP, 5) & "-" &
Right(ThesisEnv.rsWarehouseQuery2!W_ZIP, 4), 1
PrintResult ThesisEnv.rsDistrictQuery2!D_CITY, 6, 20
PrintResult ThesisEnv.rsDistrictQuery2!D_STATE, 1
PrintResult Left(ThesisEnv.rsDistrictQuery2!D_ZIP, 5) & "-" &
Right(ThesisEnv.rsDistrictQuery2!D_ZIP, 3), 1, , True
PrintResult "", , , True

'// Open Customer
ThesisEnv.CustomerQuery2 lC_ID, W_ID, lD_ID
sData = ThesisEnv.rsCustomerQuery2!C_DATA

'// Check Customer Credit
If ThesisEnv.rsCustomerQuery2!C_CREDIT = "BC" Then
    sAux = "Entry: " & lC_ID & ", " & lD_ID & "; " & W_ID & "; "
& lD_ID & "; " & W_ID & "; " & sngAmount & "."
    If Len(sData) - Len(sAux) > 500 Then
        ThesisEnv.rsCustomerQuery2!C_DATA = sAux & Left(sData,
500 - Len(sAux))
        sData = sAux & Left(sData, 500 - Len(sAux))
    Else
        ThesisEnv.rsCustomerQuery2!C_DATA = sAux & sData
        sData = sAux & sData
    End If
    ThesisEnv.rsCustomerQuery2.Update
End If

PrintResult "Customer:  " & Format(lC_ID, "0000"), , , True
PrintResult "Name:    "
PrintResult ThesisEnv.rsCustomerQuery2!C_FIRST, , 16
PrintResult ThesisEnv.rsCustomerQuery2!C_MIDDLE, 1
PrintResult ThesisEnv.rsCustomerQuery2!C_LAST, 1, 16
PrintResult "Since:  " &
Format(ThesisEnv.rsCustomerQuery2!C_SINCE, "dd-mm-yyyy"), 5, ,
True
PrintResult ThesisEnv.rsCustomerQuery2!C_STREET_1, 8, 20
PrintResult "Credit: " & ThesisEnv.rsCustomerQuery2!C_CREDIT, 21,
, True
PrintResult ThesisEnv.rsCustomerQuery2!C_STREET_2, 8, 20
```

```
PrintResult "%Disc:  " &
Format(ThesisEnv.rsCustomerQuery2!C_DISCOUNT * 100, "00.00"), 21,
, True
PrintResult ThesisEnv.rsCustomerQuery2!C_CITY, 8, 20
PrintResult ThesisEnv.rsCustomerQuery2!C_STATE, 1
PrintResult Left(ThesisEnv.rsCustomerQuery2!C_ZIP, 5) & "-" &
Right(ThesisEnv.rsCustomerQuery2!C_ZIP, 3), 1
PrintResult "Phone:  " & Left(ThesisEnv.rsCustomerQuery2!C_PHONE,
6) & "-" & Mid(ThesisEnv.rsCustomerQuery2!C_PHONE, 7, 3) & "-" &
Mid(ThesisEnv.rsCustomerQuery2!C_PHONE, 10, 3) & "-" &
Right(ThesisEnv.rsCustomerQuery2!C_PHONE, 4), 8, , True
PrintResult "", , , True

PrintResult "Amount Paid:"
PrintResult Format(sngAmount, "$0000.00"), 10
PrintResult "New Cust-Balance:", 6
PrintResult "$" & Format(ThesisEnv.rsCustomerQuery2!C_BALANCE -
sngAmount, "000000000.00"), 1, , True
PrintResult "Credit Limit:"
PrintResult "$" & Format(ThesisEnv.rsCustomerQuery2!C_CREDIT_LIM,
"000000000.00"), 4, , True
PrintResult "", , , True
If ThesisEnv.rsCustomerQuery2!C_CREDIT = "BC" Then
    PrintResult "Cust-Data:"
    PrintResult Left(sData, 50), 1, , True
    PrintResult Mid(sData, 51, 50), 11, , True
    PrintResult Mid(sData, 101, 50), 11, , True
    PrintResult Mid(sData, 151, 50), 11, , True
    PrintResult "", , , True
End If

With ThesisEnv.rsHistoryInput
    .Open
    .AddNew
    !H_C_ID = lC_ID
    !H_C_D_ID = lD_ID
    !H_C_W_ID = W_ID
    !H_D_ID = lD_ID
    !H_W_ID = W_ID
    !H_DATE = Date
    !H_AMOUNT = sngAmount
    !H_DATA = ThesisEnv.rsWarehouseQuery2!W_NAME & "    " &
ThesisEnv.rsDistrictQuery2!D_NAME
    .Update
End With

lSeconds = DateDiff("s", datStartTime, Time)
iMinutes = lSeconds / 60
lSeconds = lSeconds - iMinutes * 60
PrintResult "Time Elapsed: " & Format(iMinutes, "00") & ":" &
Format(lSeconds, "00")

bTransaction = False
ThesisEnv.ThesisConn.CommitTrans

Payment = True
'MsgBox lD_ID & " " & lC_ID

PEnd:
On Error Resume Next
With ThesisEnv
.rsCustomerQuery2.Close
.rsHistoryInput.Close
.rsDistrictQuery2.Close
.rsWarehouseQuery2.Close
End With
Exit Function

PError:
If bTransaction Then
    bTransaction = False
    ThesisEnv.ThesisConn.RollbackTrans
End If
MsgBox Error, vbCritical
Resume 'PEnd

End Function

Private Function ClientTransaction(W_ID As Long) As Boolean
On Error GoTo CTError
Dim bTransaction As Boolean
Dim iOrder_Cnt As Integer
Dim lD_ID As Long
Dim lC_ID As Long
Dim lO_ID As Long
Dim lI_ID As Long
Dim i As Integer
Dim j As Integer
Dim bBg As Boolean
Dim iQuantity As Integer
Dim sSql As String
Dim sngTotal As Single
Dim SUPP_W_ID As Integer

Randomize
ClientTransaction = False

'// Number of lines  [5..15]
'//iOrder_Cnt = Int(Rnd() * 11) + 5

iOrder_Cnt = 8

'// District
lD_ID = Int(Rnd() * 10) + 1

'// Customer
lC_ID = NURand(1023, 1, 3000)

'// Fix SQL Statement for Stock
sSql = "Select S_QUANTITY, S_DATA, S_ORDER_CNT, S_YTD, S_DIST_" &
Format(lD_ID, "00") & " as S_DIST"
sSql = sSql & " from Stock where S_I_ID=? and S_W_ID=?"
ThesisEnv.Commands(STOCK_QUERY_NUMBER).CommandText = sSql

'// Begin Transaction
ThesisEnv.ThesisConn.BeginTrans
bTransaction = True

PrintResult "New Order", 33, , True

'// Open Warehouse
ThesisEnv.WarehouseQuery W_ID
PrintResult "Warehouse: " & Format(W_ID, "0000")

'// OpenCustomer
ThesisEnv.CustomerQuery lC_ID, W_ID, lD_ID

'// Get Next Order ID
ThesisEnv.DistrictQuery lD_ID, W_ID
lO_ID = ThesisEnv.rsDistrictQuery!D_NEXT_O_ID

PrintResult "District: " & Format(lD_ID, "00"), 3
PrintResult "Date: " & Format(Time, "dd-mm-yyyy hh:mm:ss"), 23, ,
True
PrintResult "Customer:  " & Format(lC_ID, "0000")
PrintResult "Name: " & Left(ThesisEnv.rsCustomerQuery!C_LAST,
16), 3, 21
PrintResult "Credit: " & ThesisEnv.rsCustomerQuery!C_CREDIT, 3
PrintResult "%Disc: " &
Format(ThesisEnv.rsCustomerQuery!C_DISCOUNT * 100, "00.00"), 3, ,
True
PrintResult "Order Number: " & Format(lO_ID, "00000000")
PrintResult "Number of Lines: " & Format(iOrder_Cnt, "00"), 2
PrintResult "W_tax: " & Format(ThesisEnv.rsWarehouseQuery!W_TAX *
100, "00.00"), 7
PrintResult "D_tax: " & Format(ThesisEnv.rsDistrictQuery!D_TAX *
100, "00.00"), 3, , True
PrintResult "", , , True
PrintResult "Supp_W", 1
PrintResult "Item_Id", 2
PrintResult "Item_Name", 2
PrintResult "Qty", 16
PrintResult "Stock", 2
PrintResult "B/G", 2
PrintResult "Price", 2
PrintResult "Amount", 4, , True

'// Insert District_Order
With ThesisEnv.rsOrderInput
    .Open
    .AddNew
    !o_id = lO_ID
    !O_C_ID = lC_ID
    !O_D_ID = lD_ID
    !O_W_ID = W_ID
    !o_entry_date = Format(Date, "mm/dd/yyyy")
    !O_ALL_LOCAL = 1
    !O_OL_CNT = 0
    .Update
End With

With ThesisEnv.rsOrderLineInput
.Open
sngTotal = 0
For i = 1 To iOrder_Cnt
    '// Find the item
    lI_ID = NURand(8191, 1, 100000)
    '// Quantity
    iQuantity = Int(Rnd() * 10) + 1

    '// Open Item
    ThesisEnv.ItemQuery lI_ID

    '// Find the Supplier
    SUPP_W_ID = Int(Rnd() * 2) + 1

    '// Open Stock
    ThesisEnv.StockQuery lI_ID, SUPP_W_ID

    '// Update STOCK table
    If ThesisEnv.rsStockQuery!S_QUANTITY >= iQuantity + 10 Then
        ThesisEnv.rsStockQuery!S_QUANTITY =
ThesisEnv.rsStockQuery!S_QUANTITY - iQuantity
    Else
        ThesisEnv.rsStockQuery!S_QUANTITY =
ThesisEnv.rsStockQuery!S_QUANTITY + 91
    End If
    ThesisEnv.rsStockQuery!S_YTD = ThesisEnv.rsStockQuery!S_YTD +
iQuantity
    ThesisEnv.rsStockQuery!S_ORDER_CNT =
ThesisEnv.rsStockQuery!S_ORDER_CNT + 1
    ThesisEnv.rsStockQuery.Update

    '// Check for B/G
    bBg = False
    If InStr(ThesisEnv.rsItemQuery!I_DATA, "ORIGINAL") > 0 Then
        If InStr(ThesisEnv.rsStockQuery!S_DATA, "ORIGINAL") > 0
Then
            bBg = True
        End If
    End If

    '// Insert Order_Line
    .AddNew
    !OL_O_ID = lO_ID
    !OL_D_ID = lD_ID
    !OL_W_ID = W_ID
    !OL_NUMBER = i
    !OL_I_ID = lI_ID
    !OL_SUPPLY_W_ID = SUPP_W_ID
    !OL_AMOUNT = iQuantity * ThesisEnv.rsItemQuery!I_PRICE
    !OL_QUANTITY = iQuantity
    !OL_DIST_INFO = ThesisEnv.rsStockQuery!S_DIST
    .Update
```

```
    sngTotal = sngTotal + iQuantity *
ThesisEnv.rsItemQuery!I_PRICE

    PrintResult Format(SUPP_W_ID, "0000"), 2
    PrintResult Format(lI_ID, "000000"), 3
    PrintResult Left(ThesisEnv.rsItemQuery!I_NAME, 23), 3, 23
    PrintResult Format(iQuantity, "00"), 2
    PrintResult Format(ThesisEnv.rsStockQuery!S_QUANTITY, "000"),
4
    PrintResult IIf(bBg, "B", "G"), 4
    PrintResult Format(ThesisEnv.rsItemQuery!I_PRICE, "$000.00"),
3
    PrintResult Format(ThesisEnv.rsItemQuery!I_PRICE * iQuantity,
"$000.00"), 2, , True


    '// Close Item and Stock
    ThesisEnv.rsItemQuery.Close
    ThesisEnv.rsStockQuery.Close
Next
PrintResult "Total:  " & Format(sngTotal, "$0000.00"), 11, , True
PrintResult "Execution Status: Ok"
End With

bTransaction = False
ThesisEnv.ThesisConn.CommitTrans

ClientTransaction = True
'MsgBox lD_ID & " " & lC_ID

CTEnd:
On Error Resume Next
With ThesisEnv
.rsWarehouseQuery.Close
.rsDistrictQuery.Close
.rsCustomerQuery.Close
.rsOrderLineInput.Close
.rsOrderInput.Close
End With
Exit Function

CTError:
If bTransaction Then
    bTransaction = False
    ThesisEnv.ThesisConn.RollbackTrans
End If
MsgBox Error, vbCritical
Resume CTEnd
End Function

Private Sub cmdTransaction_Click(index As Integer)
Dim bResult As Boolean
Dim W_ID As Long

W_ID = 2

txtResult = ""
Screen.MousePointer = vbHourglass
Select Case index
    Case 0
        bResult = ClientTransaction(W_ID)
    Case 1
        bResult = Payment(W_ID)
    Case 2
        bResult = Order_Status(W_ID)
    Case 3
        bResult = Delivery(W_ID)
    Case 4
        bResult = Stock_Level(W_ID)
End Select
Screen.MousePointer = vbNormal

End Sub
Public Function ExecuteTransaction(index As Integer)
    cmdTransaction_Click index
End Function
Private Function GenerateLastNameStr(sCode As String) As String
Dim iIndex As Long
Dim sAux As String

sCode = Trim(sCode)
If Len(sCode) < 3 Then
    sCode = Space(3 - Len(sCode)) & sCode
End If
iIndex = Val(Right(sCode, 1))
sAux = LNSyllables(iIndex)
If Len(sCode) = 2 Then
    iIndex = Val(Left(sCode, 1))
    sAux = LNSyllables(iIndex) & sAux
ElseIf Len(sCode) > 2 Then
    iIndex = Val(Mid(sCode, 2, 1))
    sAux = LNSyllables(iIndex) & sAux
    iIndex = Val(Left(sCode, 1))
    sAux = LNSyllables(iIndex) & sAux
End If
GenerateLastNameStr = sAux

End Function
Private Sub Form_Load()
LNSyllables(0) = "BAR"
LNSyllables(1) = "OUGHT"
LNSyllables(2) = "ABLE"
LNSyllables(3) = "PRI"
LNSyllables(4) = "PRES"
LNSyllables(5) = "ESE"
LNSyllables(6) = "ANTI"
LNSyllables(7) = "CALLY"
LNSyllables(8) = "ATION"
LNSyllables(9) = "EING"
End Sub
```

3. N-Tier Data Objects

```
Option Explicit

'// Customer

'local siable(s) to hold property value(s)
Private mlId As Long 'local copy
'local siable(s) to hold property value(s)
Private mDistrict As District 'local copy
'local siable(s) to hold property value(s)
Private msFirst As String 'local copy
Private msLast As String 'local copy
Private msMiddle As String 'local copy
Private msStreet1 As String 'local copy
Private msStreet2 As String 'local copy
Private msCity As String 'local copy
Private msState As String 'local copy
Private msZIP As String 'local copy
Private msPhone As String 'local copy
'local variable(s) to hold property value(s)
Private mdatSince As Date 'local copy
Private msCredit As String 'local copy
Private msngLimit As Single 'local copy
Private msngDiscount As Single 'local copy
Private msngBalance As Single 'local copy
Private msngYTD_Payment As Single 'local copy
Private miPayment_CNT As Integer 'local copy
Private miDelivery_CNT As Integer 'local copy
Private msData As String 'local copy

Public Property Let DATA(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.DATA = 5
    msData = vData
End Property

Public Property Get DATA() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.DATA
    DATA = msData
End Property

Public Property Let Delivery_CNT(ByVal vData As Integer)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Delivery_CNT = 5
    miDelivery_CNT = vData
End Property

Public Property Get Delivery_CNT() As Integer
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Delivery_CNT
    Delivery_CNT = miDelivery_CNT
End Property

Public Property Let Payment_CNT(ByVal vData As Integer)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Payment_CNT = 5
    miPayment_CNT = vData
End Property

Public Property Get Payment_CNT() As Integer
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Payment_CNT
    Payment_CNT = miPayment_CNT
End Property

Public Property Let YTD_Payment(ByVal vData As Single)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.YTD_Payment = 5
    msngYTD_Payment = vData
End Property

Public Property Get YTD_Payment() As Single
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.YTD_Payment
    YTD_Payment = msngYTD_Payment
End Property

Public Property Let Balance(ByVal vData As Single)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Balance = 5
    msngBalance = vData
End Property

Public Property Get Balance() As Single
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Balance
    Balance = msngBalance
End Property

Public Property Let Discount(ByVal vData As Single)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Discount = 5
    msngDiscount = vData
End Property

Public Property Get Discount() As Single
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Discount
    Discount = msngDiscount
End Property

Public Property Let Limit(ByVal vData As Single)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Limit = 5
    msngLimit = vData
End Property

Public Property Get Limit() As Single
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Limit
    Limit = msngLimit
End Property

Public Property Let Credit(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Credit = 5
    msCredit = vData
End Property

Public Property Get Credit() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Credit
    Credit = msCredit
End Property

Public Property Let Since(ByVal vData As Date)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Since = 5
    mdatSince = vData
End Property

Public Property Get Since() As Date
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Since
    Since = mdatSince
End Property

Public Property Let Phone(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Phone = 5
    msPhone = vData
End Property

Public Property Get Phone() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Phone
    Phone = msPhone
End Property

Public Property Let ZIP(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.ZIP = 5
    msZIP = vData
End Property

Public Property Get ZIP() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.ZIP
    ZIP = msZIP
End Property

Public Property Let State(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.State = 5
    msState = vData
End Property

Public Property Get State() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.State
    State = msState
End Property

Public Property Let City(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.City = 5
    msCity = vData
End Property

Public Property Get City() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.City
```

```
    City = msCity
End Property

Public Property Let Street2(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Street2 = 5
    msStreet2 = vData
End Property

Public Property Get Street2() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Street2
    Street2 = msStreet2
End Property

Public Property Let Street1(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Street1 = 5
    msStreet1 = vData
End Property

Public Property Get Street1() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Street1
    Street1 = msStreet1
End Property

Public Property Let Middle(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Middle = 5
    msMiddle = vData
End Property

Public Property Get Middle() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Middle
    Middle = msMiddle
End Property

Public Property Let Last(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Last = 5
    msLast = vData
End Property

Public Property Get Last() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Last
    Last = msLast
End Property

Public Property Let First(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.First = 5
    msFirst = vData
End Property

Public Property Get First() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.First
    First = msFirst
End Property

Public Property Get District() As District
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.District
    Set District = mDistrict
End Property

Public Property Let Id(ByVal vData As Long)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Id = 5
    mlId = vData
End Property

Public Property Get Id() As Long
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Id
    Id = mlId
End Property

Public Function OpenWithLast(LastName As String, Dist As
ThesisDO.District, cn As ADODB.Connection) As Boolean
Dim dbcmdCustomer As New ADODB.Command
Dim rstCustomer As ADODB.Recordset

Set mDistrict = Dist
If mDistrict Is Nothing Then Exit Function

With dbcmdCustomer
.CommandText = "Select * from Customer where C_LAST = ? and
C_D_ID= " & Dist.Id & " and C_W_ID=" & Dist.Warehouse.Id
Set .ActiveConnection = cn
.Parameters(0).Value = LastName

Set rstCustomer = .Execute
End With

With rstCustomer
If Not .EOF Then
    If .RecordCount > 0 Then
        .Move .RecordCount / 2
    End If
    mlId = !C_ID
    msFirst = !C_FIRST
    msMiddle = !C_MIDDLE
    msLast = !C_LAST
    msStreet1 = !C_STREET_1
    msStreet2 = !C_STREET_2
    msCity = !C_CITY
    msZIP = !C_ZIP
    msState = !C_STATE
    msPhone = !C_PHONE
    mdatSince = !C_SINCE
    msCredit = !C_CREDIT
    msngLimit = !C_CREDIT_LIM
    msngDiscount = !C_DISCOUNT
    msngBalance = !C_BALANCE
    msngYTD_Payment = !C_YTD_PAYMENT
    miPayment_CNT = !C_PAYMENT_CNT
    miDelivery_CNT = !C_DELIVERY_CNT
    msData = !C_DATA
    OpenWithLast = True
End If
End With

End Function

Public Function OpenWith(C_ID As Long, Dist As ThesisDO.District,
cn As ADODB.Connection) As Boolean
Dim rstCustomer As New ADODB.Recordset


Set mDistrict = Dist
If mDistrict Is Nothing Then Exit Function

With rstCustomer
.Open "Select * from Customer where C_ID = " & C_ID & " and
C_D_ID= " & Dist.Id & " and C_W_ID=" & Dist.Warehouse.Id, cn,
adOpenForwardOnly, adLockReadOnly
If Not .EOF Then
    mlId = !C_ID
    msFirst = !C_FIRST
    msMiddle = !C_MIDDLE
    msLast = !C_LAST
    msStreet1 = !C_STREET_1
    msStreet2 = !C_STREET_2
    msCity = !C_CITY
    msZIP = !C_ZIP
    msState = !C_STATE
    msPhone = !C_PHONE
    mdatSince = !C_SINCE
    msCredit = !C_CREDIT
    msngLimit = !C_CREDIT_LIM
    msngDiscount = !C_DISCOUNT
    msngBalance = !C_BALANCE
    msngYTD_Payment = !C_YTD_PAYMENT
    miPayment_CNT = !C_PAYMENT_CNT
    miDelivery_CNT = !C_DELIVERY_CNT
    msData = !C_DATA
End If
End With
OpenWith = True
End Function

Public Function Save(cn As ADODB.Connection) As Boolean
On Error GoTo SErrors
Dim rstCustomer As New ADODB.Recordset

If mDistrict Is Nothing Then Exit Function

With rstCustomer
10: .Open "Select * from Customer where C_ID = " & mlId & " and
C_D_ID= " & mDistrict.Id & " and C_W_ID=" &
mDistrict.Warehouse.Id, cn, adOpenDynamic, adLockPessimistic
If .EOF Then Exit Function
!C_DATA = msData
!C_BALANCE = msngBalance
!C_DELIVERY_CNT = miDelivery_CNT
20: .Update
End With
Save = True
SFim:
Exit Function

SErrors:
MsgBox Erl & ":" & Err.Number & "-" & Err.Description, vbCritical
Resume SFim
End Function


Option Explicit

'// District

Private mlId As Long 'local copy
'local variable(s) to hold property value(s)
Private mWarehouse As Warehouse 'local copy
Private msngTax As Single 'local copy
'local variable(s) to hold property value(s)
Private mlNextOrderId As Long 'local copy
Private msStreet1 As String 'local copy
Private msStreet2 As String 'local copy
Private msCity As String 'local copy
Private msState As String 'local copy
Private msZIP As String 'local copy
Private msngYTD As Single
'local variable(s) to hold property value(s)
Private msName As String 'local copy

Public Property Get YTD() As Single
    YTD = msngYTD
End Property

Public Property Let YTD(sngVal As Single)
```

```
    mWarehouse.YTD = mWarehouse.YTD - msngYTD + sngVal
    msngYTD = sngVal
End Property

Public Property Let Name(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Name = 5
    msName = vData
End Property

Public Property Get Name() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Name
    Name = msName
End Property

Public Property Let ZIP(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.ZIP = 5
    msZIP = vData
End Property

Public Property Get ZIP() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.ZIP
    ZIP = msZIP
End Property

Public Property Let State(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.State = 5
    msState = vData
End Property

Public Property Get State() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.State
    State = msState
End Property

Public Property Let City(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.City = 5
    msCity = vData
End Property

Public Property Get City() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.City
    City = msCity
End Property

Public Property Let Street2(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Street2 = 5
    msStreet2 = vData
End Property

Public Property Get Street2() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Street2
    Street2 = msStreet2
End Property

Public Property Let Street1(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Street1 = 5
    msStreet1 = vData
End Property

Public Property Get Street1() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Street1
    Street1 = msStreet1
End Property

Public Property Let NextOrderId(ByVal vData As Long)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.NextOrderId = 5
    mlNextOrderId = vData
End Property

Public Property Get NextOrderId() As Long
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.NextOrderId
    NextOrderId = mlNextOrderId
End Property

Public Property Let Tax(ByVal vData As Single)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Tax = 5
    msngTax = vData
End Property

Public Property Get Tax() As Single
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Tax
    Tax = msngTax
End Property

Public Property Set Warehouse(w As Warehouse)
    Set mWarehouse = w
End Property

Public Property Get Warehouse() As Warehouse
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Warehouse
    Set Warehouse = mWarehouse
End Property

Public Property Let Id(ByVal vData As Long)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Id = 5
    mlId = vData
End Property

Public Property Get Id() As Long
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Id
    Id = mlId
End Property

Public Function OpenWith(D_ID As Long, War As ThesisDO.Warehouse,
cn As ADODB.Connection) As Variant
Dim rstDistrict As New ADODB.Recordset

If D_ID <= 0 Then Exit Function
Set mWarehouse = War

With rstDistrict
.Open "Select * from District where D_ID=" & D_ID & " and D_W_ID
= " & War.Id, cn, adOpenForwardOnly, adLockReadOnly
If Not .EOF Then
    mlId = D_ID
    msStreet1 = !D_STREET_1
    msStreet2 = !D_STREET_2
    msCity = !D_CITY
    msZIP = !D_ZIP
    msState = !D_STATE
    msName = !D_NAME
    msngTax = !D_TAX
    msngYTD = !D_YTD
    mlNextOrderId = !D_NEXT_O_ID
    OpenWith = True
End If
.Close
End With
End Function

Public Function Save(cn As ADODB.Connection) As Boolean
Dim rstDistrict As New ADODB.Recordset

With rstDistrict
.Open "Select * from District where D_ID=" & mlId & " and D_W_ID
= " & mWarehouse.Id, cn, adOpenDynamic, adLockPessimistic
If Not .EOF Then
    !D_NEXT_O_ID = mlNextOrderId
    !D_YTD = msngYTD
    .Update
End If
.Close
End With
Save = True
End Function

Option Explicit

'// History

'local variable(s) to hold property value(s)
Private mCustomer As Customer 'local copy
Private mDistrict As District 'local copy
Private mdatDate As Date 'local copy
Private msngAmount As Single 'local copy
Private msData As String 'local copy

Public Function Save(cn As ADODB.Connection) As Boolean
Dim rstHistory As New ADODB.Recordset

If mCustomer Is Nothing Then Exit Function
If mDistrict Is Nothing Then Exit Function

With rstHistory
.Open "Select * from History where H_C_ID = 0", cn,
adOpenDynamic, adLockPessimistic
.AddNew
!H_C_ID = mCustomer.Id
!H_C_D_ID = mCustomer.District.Id
!H_C_W_ID = mCustomer.District.Warehouse.Id
!H_D_ID = mDistrict.Id
!H_W_ID = mDistrict.Warehouse.Id
!H_DATE = mdatDate
!H_AMOUNT = msngAmount
!H_DATA = msData
.Update
End With

Save = True
End Function

Public Property Let DATA(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.DATA = 5
    msData = vData
End Property

Public Property Get DATA() As String
```

```
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.DATA
    DATA = msData
End Property

Public Property Let Amount(ByVal vData As Single)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Amount = 5
    msngAmount = vData
End Property

Public Property Get Amount() As Single
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Amount
    Amount = msngAmount
    If Not mDistrict Is Nothing Then
        mDistrict.YTD = mDistrict.YTD + msngAmount
    End If
    If Not mCustomer Is Nothing Then
        mCustomer.Delivery_CNT = mCustomer.Delivery_CNT + 1
        mCustomer.YTD_Payment = mCustomer.YTD_Payment +
msngAmount
        mCustomer.Balance = mCustomer.Balance - msngAmount
    End If
End Property

Public Property Let EntryDate(ByVal vData As Date)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.EntryDate = 5
    mdatDate = vData
End Property

Public Property Get EntryDate() As Date
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.EntryDate
    EntryDate = mdatDate
End Property

Public Property Set District(Dist As ThesisDO.District)
    Set mDistrict = Dist
    '// Set District YTD
    If msngAmount > 0 Then
        mDistrict.YTD = mDistrict.YTD + msngAmount
    End If
End Property

Public Property Get District() As District
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.District
    Set District = mDistrict
End Property

Public Property Set Customer(ByVal vData As Customer)
'used when assigning an Object to the property, on the left side
of a Set statement.
'Syntax: Set x.Customer = Form1
    Set mCustomer = vData
    If msngAmount > 0 Then
        mCustomer.Delivery_CNT = mCustomer.Delivery_CNT + 1
        mCustomer.YTD_Payment = mCustomer.YTD_Payment +
msngAmount
        mCustomer.Balance = mCustomer.Balance - msngAmount
    End If
End Property

Public Property Get Customer() As Customer
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Customer
    Set Customer = mCustomer
End Property

Private Sub Class_Initialize()
    Set mDistrict = Nothing
    Set mCustomer = Nothing
    msngAmount = 0
End Sub

Option Explicit

'// Item

'local variable(s) to hold property value(s)
Private mlId As Long 'local copy
Private msName As String 'local copy
Private msngPrice As Single 'local copy
Private msData As String 'local copy
'local variable(s) to hold property value(s)

Public Function OpenWith(I_ID As Long, cn As ADODB.Connection)
Dim rstItem As New ADODB.Recordset

With rstItem
.Open "Select * from Item where I_ID = " & I_ID, cn,
adOpenForwardOnly, adLockReadOnly
If Not rstItem.EOF Then
    mlId = !I_ID
    msName = !I_Name
    msngPrice = !I_Price
    msData = !I_DATA
End If
End With
OpenWith = True
End Function

Public Property Let DATA(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Data = 5
    msData = vData
End Property

Public Property Get DATA() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Data
    DATA = msData
End Property

Public Property Let Price(ByVal vData As Single)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Price = 5
    msngPrice = vData
End Property

Public Property Get Price() As Single
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Price
    Price = msngPrice
End Property

Public Property Let Name(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.ItemName = 5
    msName = vData
End Property

Public Property Get Name() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.ItemName
    Name = msName
End Property

Public Property Let Id(ByVal vData As Long)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Id = 5
    mlId = vData
End Property

Public Property Get Id() As Long
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Id
    Id = mlId
End Property


Option Explicit

'// New Order

'local variable(s) to hold property value(s)
Private mOrder As Order 'local copy

Public Function Save(cn As ADODB.Connection) As Boolean
Dim rstNewOrder As New ADODB.Recordset

If mOrder Is Nothing Then Exit Function

With rstNewOrder
    .Open "Select * from New_Order where NO_O_ID = 0", cn,
adOpenDynamic, adLockPessimistic
    .AddNew
    !NO_O_ID = mOrder.Id
    !NO_D_ID = mOrder.Customer.District.Id
    !NO_W_ID = mOrder.Customer.District.Warehouse.Id
    .Update
    .Close
End With
Set rstNewOrder = Nothing
Save = True
End Function

Public Property Set Order(ByVal vData As Order)
'used when assigning an Object to the property, on the left side
of a Set statement.
'Syntax: Set x.Order = Form1
    Set mOrder = vData
End Property


Public Property Get Order() As Order
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Order
    Set Order = mOrder
End Property


Option Explicit

'// Order

'local variable to hold collection
Private mCol As Collection
Private mlId As Long 'local copy
'local variable(s) to hold property value(s)
Private mCustomer As Customer 'local copy
'local variable(s) to hold property value(s)
Private mdatEntryDate As Date 'local copy
Private mlCarrierId As Long 'local copy
Private miOL_CNT As Integer 'local copy
Private mbAllLocal As Boolean 'local copy
'local variable(s) to hold property value(s)
Private msngTotal As Single 'local copy

Public Property Let Total(ByVal vData As Single)
```

```
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Total = 5
    msngTotal = vData
End Property

Public Property Get Total() As Single
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Total
    Total = msngTotal
End Property

Public Property Let AllLocal(ByVal vData As Boolean)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.AllLocal = 5
    mbAllLocal = vData
End Property

Public Property Get AllLocal() As Boolean
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.AllLocal
    AllLocal = mbAllLocal
End Property

Public Property Let OL_CNT(ByVal vData As Integer)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.OL_CNT = 5
    miOL_CNT = vData
End Property

Public Property Get OL_CNT() As Integer
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.OL_CNT
    OL_CNT = miOL_CNT
End Property

Public Property Let CarrierID(ByVal vData As Long)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.CarrierID = 5
    mlCarrierId = vData
End Property

Public Property Get CarrierID() As Long
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.CarrierID
    CarrierID = mlCarrierId
End Property

Public Property Let EntryDate(ByVal vData As Date)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.EntryDate = 5
    mdatEntryDate = vData
End Property

Public Property Get EntryDate() As Date
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.EntryDate
    EntryDate = mdatEntryDate
End Property

Public Property Set Customer(ByVal vData As Customer)
'used when assigning an Object to the property, on the left side
of a Set statement.
'Syntax: Set x.Customer = Form1
    Set mCustomer = vData
    '// Get the Id
    If mlId = 0 Then
        mlId = vData.District.NextOrderId
        vData.District.NextOrderId = mlId + 1
    End If
End Property

Public Property Get Customer() As Customer
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Customer
    Set Customer = mCustomer
End Property

Public Property Let Id(ByVal vData As Long)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Id = 5
    mlId = vData
End Property

Public Property Get Id() As Long
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Id
    Id = mlId
End Property

Public Function OpenOLs(cn As ADODB.Connection, Optional
WithItems As Boolean = True, Optional WithStockItems As Boolean =
False) As Boolean
Dim rstOrderLine As New ADODB.Recordset
Dim oOL As ThesisDO.OrderLine
Dim oItem As ThesisDO.Item
Dim oStockItem As ThesisDO.StockItem

If mlId = 0 Then Exit Function
If mCustomer Is Nothing Then Exit Function
If mCustomer.District Is Nothing Then Exit Function

With rstOrderLine
    '// Open Order
    .Open "Select * from Order_Line where OL_O_ID = " & mlId & "
and OL_D_ID = " & mCustomer.District.Id & " and OL_W_ID=" &
mCustomer.District.Warehouse.Id, cn, adOpenForwardOnly,
adLockReadOnly
    Do While Not .EOF
        Set oOL = CreateObject("ThesisDO.OrderLine")
        oOL.Id = !OL_O_ID
        oOL.Number = !OL_NUMBER
        oOL.SupplyWId = !OL_SUPPLY_W_ID
        oOL.Total = !OL_AMOUNT
        oOL.Quantity = !OL_QUANTITY
        If WithItems Then
            Set oItem = CreateObject("ThesisDO.Item")
            oItem.OpenWith !OL_I_ID, cn
            Set oOL.Item = oItem
        End If
        If WithStockItems And WithItems Then
            Set oStockItem = CreateObject("ThesisDO.StockItem")
            oStockItem.OpenWith oItem,
mCustomer.District.Warehouse, cn
            Set oOL.StockItem = oStockItem
        End If
        Add oOL
        Set oOL = Nothing
        Set oItem = Nothing
        Set oStockItem = Nothing
        .MoveNext
    Loop
End With

Set rstOrderLine = Nothing
OpenOLs = True


End Function


Public Function Add(objNewMember As ThesisDO.OrderLine, Optional
sKey As String) As OrderLine

    Set objNewMember.Order = Me
    objNewMember.Number = mCol.Count + 1
    msngTotal = msngTotal + objNewMember.Total

    If Len(sKey) = 0 Then
        mCol.Add objNewMember
    Else
        mCol.Add objNewMember, sKey
    End If

    'Set the dependent values
    miOL_CNT = miOL_CNT + 1
    mCustomer.Balance = mCustomer.Balance + objNewMember.Total

    'return the object created
    Set Add = objNewMember

End Function

Public Property Get Item(vntIndexKey As Variant) As OrderLine
    'used when referencing an element in the collection
    'vntIndexKey contains either the Index or Key to the
collection,
    'this is why it is declared as a Variant
    'Syntax: Set foo = x.Item(xyz) or Set foo = x.Item(5)
  Set Item = mCol(vntIndexKey)
End Property

Public Property Get Count() As Long
    'used when retrieving the number of elements in the
    'collection. Syntax: Debug.Print x.Count
    Count = mCol.Count
End Property

Public Sub Remove(vntIndexKey As Variant)
    'used when removing an element from the collection
    'vntIndexKey contains either the Index or Key, which is why
    'it is declared as a Variant
    'Syntax: x.Remove(xyz)
    mCol.Remove vntIndexKey
End Sub

Public Property Get NewEnum() As IUnknown
    'this property allows you to enumerate
    'this collection with the For...Each syntax
    Set NewEnum = mCol.[_NewEnum]
End Property

Private Sub Class_Initialize()
    'creates the collection when this class is created
    Set mCol = New Collection
    mlId = 0
End Sub

Private Sub Class_Terminate()
    'destroys collection when this class is terminated
    Set mCol = Nothing
End Sub

Public Function OpenWith(Id As Long, Cust As ThesisDO.Customer,
cn As ADODB.Connection) As Variant
Dim rstOrder As New ADODB.Recordset

If Cust Is Nothing Then Exit Function

Set mCustomer = Cust
With rstOrder
    '// Open Order
    .Open "Select O_CARRIER_ID, O_ENTRY_DATE from District_Order
where O_ID = " & Id & " and O_D_ID = " & Customer.District.Id & "
and O_W_ID=" & Customer.District.Warehouse.Id, cn,
adOpenForwardOnly, adLockReadOnly
```

```
    If .EOF Then Exit Function
    mlId = Id
    If Not IsNull(!O_CARRIER_ID) Then mlCarrierId = !O_CARRIER_ID
    mdatEntryDate = !O_ENTRY_DATE
    .Close
End With

Set rstOrder = Nothing
OpenWith = True

End Function

Public Function OpenWithD(Id As Long, Dist As ThesisDO.District,
cn As ADODB.Connection) As Variant
Dim rstOrder As New ADODB.Recordset
Dim Cust As ThesisDO.Customer

If Dist Is Nothing Then Exit Function

With rstOrder
    '// Open Order
    .Open "Select O_CARRIER_ID, O_ENTRY_DATE, O_C_ID from
District_Order where O_ID = " & Id & " and O_D_ID = " & Dist.Id &
" and O_W_ID=" & Dist.Warehouse.Id, cn, adOpenForwardOnly,
adLockReadOnly
    If .EOF Then Exit Function
    mlId = Id
    Set Cust = CreateObject("ThesisDO.Customer")
    If Not Cust.OpenWith(!O_C_ID, Dist, cn) Then Exit Function
    Set mCustomer = Cust
    If Not IsNull(!O_CARRIER_ID) Then mlCarrierId = !O_CARRIER_ID
    mdatEntryDate = !O_ENTRY_DATE
    If Not OpenOLs(cn, False) Then Exit Function
End With

Set rstOrder = Nothing
OpenWithD = True

End Function

Public Function Save(cn As ADODB.Connection) As Boolean
Dim rstOrder As New ADODB.Recordset

If mCustomer Is Nothing Then Exit Function

With rstOrder
    .Open "Select * from District_Order where O_ID = " & mlId & "
and O_D_ID = " & mCustomer.District.Id & " and O_W_ID=" &
mCustomer.District.Warehouse.Id, cn, adOpenDynamic,
adLockPessimistic
    If .EOF Then
        .AddNew
        !O_ID = mlId
        !O_C_ID = mCustomer.Id
        !O_D_ID = mCustomer.District.Id
        !O_W_ID = mCustomer.District.Warehouse.Id
    End If
    !O_ENTRY_DATE = mdatEntryDate
    !O_ALL_LOCAL = 1
    !O_OL_CNT = mCol.Count
    If mlCarrierId > 0 Then !O_CARRIER_ID = mlCarrierId
    .Update
End With

Set rstOrder = Nothing
Save = True
End Function

Option Explicit

'// Order Line

'local variable(s) to hold property value(s)
Private mItem As Item 'local copy
'local variable(s) to hold property value(s)
Private miQuantity As Integer 'local copy
'local variable(s) to hold property value(s)
Private mStockItem As StockItem 'local copy
Private mbBG As Boolean 'local copy
'local variable(s) to hold property value(s)
Private mOrder As Order 'local copy
'local variable(s) to hold property value(s)
Private miNumber As Integer 'local copy
'local variable(s) to hold property value(s)
Private msngTotal As Single 'local copy
'local variable(s) to hold property value(s)
Private mdatDeliveryDate As Date 'local copy
Private mlSupplyWId As Long 'local copy
Private mlId As Long

Public Property Let Id(ByVal vData As Long)
    mlId = vData
End Property

Public Property Get Id() As Long
    Id = mlId
End Property

Public Property Let SupplyWId(ByVal vData As Long)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.SupplyWId = 5
    mlSupplyWId = vData
End Property

Public Property Get SupplyWId() As Long
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.SupplyWId
    SupplyWId = mlSupplyWId
End Property

Public Property Let DeliveryDate(ByVal vData As Date)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.DeliveryDate = 5
    mdatDeliveryDate = vData
End Property

Public Property Get DeliveryDate() As Date
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.DeliveryDate
    DeliveryDate = mdatDeliveryDate
End Property

Public Property Let Total(ByVal vData As Single)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Total = 5
    msngTotal = vData
End Property

Public Property Get Total() As Single
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Total
    Total = msngTotal
End Property

Public Property Let Number(ByVal vData As Integer)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Number = 5
    miNumber = vData
End Property

Public Property Get Number() As Integer
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Number
    Number = miNumber
End Property

Public Property Set Order(ByVal vData As Order)
'used when assigning an Object to the property, on the left side
of a Set statement.
'Syntax: Set x.Order = Form1
    Set mOrder = vData
End Property

Public Property Get Order() As Order
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Order
    Set Order = mOrder
End Property

Private Sub CheckBG()
'// Check for B/G

mbBG = False
If mItem Is Nothing Then Exit Sub
If mStockItem Is Nothing Then Exit Sub

If InStr(mItem.DATA, "ORIGINAL") > 0 Then
    If InStr(mStockItem.DATA, "ORIGINAL") > 0 Then
        mbBG = True
    End If
End If

End Sub

Public Function Save(cn As ADODB.Connection) As Boolean
Dim rstOrderLine As New ADODB.Recordset

If mStockItem Is Nothing Then Exit Function
If mItem Is Nothing Then Exit Function
If mOrder Is Nothing Then Exit Function

With rstOrderLine
    .Open "Select * from Order_Line where OL_O_ID = " & mOrder.Id
& " and OL_D_ID=" & mOrder.Customer.District.Id & " and OL_W_ID="
& mOrder.Customer.District.Warehouse.Id & " and OL_NUMBER=" &
miNumber, cn, adOpenDynamic, adLockPessimistic
    If .EOF Then
        .AddNew
    End If

    '// Update Order Line
    !OL_O_ID = mOrder.Id
    !OL_D_ID = mOrder.Customer.District.Id
    !OL_W_ID = mOrder.Customer.District.Warehouse.Id
    !OL_NUMBER = miNumber
    !OL_I_ID = mItem.Id
    !OL_SUPPLY_W_ID = mOrder.Customer.District.Warehouse.Id
    !OL_AMOUNT = msngTotal
    !OL_QUANTITY = miQuantity
    !OL_DIST_INFO =
mStockItem.DistInfo(mOrder.Customer.District.Id)
    If mdatDeliveryDate > 0 Then
        !OL_DELIVERY_DATE = mdatDeliveryDate
    End If
    .Update
    '// Save Stock
    Save = mStockItem.Save(cn)

End With

Set rstOrderLine = Nothing
Save = True

End Function

Public Property Get BG() As Boolean
'used when retrieving value of a property, on the right side of
an assignment.
```

```
'Syntax: Debug.Print X.BG
    BG = mbBG
End Property

Public Property Set StockItem(ByVal vData As StockItem)
'used when assigning an Object to the property, on the left side
of a Set statement.
'Syntax: Set x.StockItem = Form1
    Set mStockItem = vData
    CheckBG
End Property

Public Property Get StockItem() As StockItem
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.StockItem
    Set StockItem = mStockItem
End Property

Public Property Let Quantity(ByVal vData As Integer)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Quantity = 5
    miQuantity = vData
    If mItem Is Nothing Then Exit Property
    msngTotal = mItem.Price * miQuantity
End Property

Public Property Get Quantity() As Integer
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Quantity
    Quantity = miQuantity
End Property

Public Property Set Item(ByVal vData As Item)
'used when assigning an Object to the property, on the left side
of a Set statement.
'Syntax: Set x.Item = Form1
    Set mItem = vData
    CheckBG
    msngTotal = mItem.Price * miQuantity
End Property

Public Property Get Item() As Item
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Item
    Set Item = mItem
End Property


Private Sub Class_Initialize()
    mdatDeliveryDate = 0
End Sub


Option Explicit

'// Stock Item

'local variable(s) to hold property value(s)
Private msData As String 'local copy
Private miQuantity As Integer 'local copy
Private mWarehouse As Warehouse 'local copy
Private msngYTD As Single 'local copy
Private miOrder_CNT As Integer 'local copy
Private mItem As Item 'local copy
Private msDist01 As String
Private msDist02 As String
Private msDist03 As String
Private msDist04 As String
Private msDist05 As String
Private msDist06 As String
Private msDist07 As String
Private msDist08 As String
Private msDist09 As String
Private msDist10 As String

Public Property Get DistInfo(District As Integer) As String
    Select Case District
        Case 1: DistInfo = msDist01
        Case 2: DistInfo = msDist02
        Case 3: DistInfo = msDist03
        Case 4: DistInfo = msDist04
        Case 5: DistInfo = msDist05
        Case 6: DistInfo = msDist06
        Case 7: DistInfo = msDist07
        Case 8: DistInfo = msDist08
        Case 9: DistInfo = msDist09
        Case 10: DistInfo = msDist10
    End Select
End Property

Public Property Set Item(ByVal vData As Item)
'used when assigning an Object to the property, on the left side
of a Set statement.
'Syntax: Set x.Item = Form1
    Set mItem = vData
End Property

Public Property Get Item() As Item
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Item
    Set Item = mItem
End Property

Public Property Let Order_CNT(ByVal vData As Integer)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Order_CNT = 5
    miOrder_CNT = vData
End Property

Public Property Get Order_CNT() As Integer
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Order_CNT
    Order_CNT = miOrder_CNT
End Property

Public Property Let YTD(ByVal vData As Single)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.YTD = 5
    msngYTD = vData
End Property

Public Property Get YTD() As Single
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.YTD
    YTD = msngYTD
End Property

Public Property Get Warehouse() As District
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.District
    Set Warehouse = mWarehouse
End Property

Public Property Let Quantity(ByVal vData As Integer)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Quantity = 5
    miQuantity = vData
End Property

Public Property Get Quantity() As Integer
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Quantity
    Quantity = miQuantity
End Property

Public Property Let DATA(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Data = 5
    msData = vData
End Property

Public Property Get DATA() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Data
    DATA = msData
End Property

Public Function Save(cn As ADODB.Connection) As Boolean
Dim rstStock As New ADODB.Recordset

If mWarehouse Is Nothing Then Exit Function

With rstStock
    .Open "Select S_QUANTITY, S_YTD, S_ORDER_CNT from Stock
where S_I_ID = " & mItem.Id & " and S_W_ID=" & mWarehouse.Id, cn,
adOpenDynamic, adLockPessimistic
    If .EOF Then Exit Function
    !S_QUANTITY = miQuantity
    !S_YTD = msngYTD
    !S_ORDER_CNT = miOrder_CNT
    .Update
End With

Set rstStock = Nothing
Save = True

End Function
Public Function OpenWith(Item As ThesisDO.Item, Warehouse As
ThesisDO.Warehouse, cn As ADODB.Connection) As Variant
Dim rstStock As New ADODB.Recordset


Set mWarehouse = Warehouse
If mWarehouse Is Nothing Then Exit Function
If Item Is Nothing Then Exit Function
Set mItem = Item

With rstStock
.Open "Select * from Stock where S_I_ID = " & Item.Id & " and
S_W_ID=" & mWarehouse.Id, cn, adOpenForwardOnly, adLockReadOnly
If Not .EOF Then
    miQuantity = !S_QUANTITY
    msDist01 = !S_DIST_01
    msDist02 = !S_DIST_02
    msDist03 = !S_DIST_03
    msDist04 = !S_DIST_04
    msDist05 = !S_DIST_05
    msDist06 = !S_DIST_06
    msDist07 = !S_DIST_07
    msDist08 = !S_DIST_08
    msDist09 = !S_DIST_09
    msDist10 = !S_DIST_10
    msngYTD = !S_YTD
    miOrder_CNT = !S_ORDER_CNT
    msData = !S_DATA
End If
End With
OpenWith = True
End Function

Option Explicit

'// Warehouse
```

```
Private mlId As Long 'local copy
'local variable(s) to hold property value(s)
Private msngTax As Single 'local copy
'local variable(s) to hold property value(s)
Private msStreet1 As String 'local copy
Private msStreet2 As String 'local copy
Private msCity As String 'local copy
Private msState As String 'local copy
Private msZIP As String 'local copy
Private msName As String
Private msngYTD As Single

Public Property Get YTD() As Single
    YTD = msngYTD
End Property

Public Property Let YTD(sngVal As Single)
    msngYTD = sngVal
End Property

Public Property Let Name(s As String)
    msName = s
End Property

Public Property Get Name() As String
    Name = msName
End Property

Public Property Let ZIP(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.ZIP = 5
    msZIP = vData
End Property

Public Property Get ZIP() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.ZIP
    ZIP = msZIP
End Property

Public Property Let State(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.State = 5
    msState = vData
End Property

Public Property Get State() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.State
    State = msState
End Property

Public Property Let City(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.City = 5
    msCity = vData
End Property

Public Property Get City() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.City
    City = msCity
End Property

Public Property Let Street2(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Street2 = 5
    msStreet2 = vData
End Property

Public Property Get Street2() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Street2
    Street2 = msStreet2
End Property

Public Property Let Street1(ByVal vData As String)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Street1 = 5
    msStreet1 = vData
End Property

Public Property Get Street1() As String
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Street1
    Street1 = msStreet1
End Property

Public Property Let Tax(ByVal vData As Single)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Tax = 5
    msngTax = vData
End Property

Public Property Get Tax() As Single
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Tax
    Tax = msngTax
End Property

Public Property Let Id(ByVal vData As Long)
'used when assigning a value to the property, on the left side of
an assignment.
'Syntax: X.Id = 5
    mlId = vData
End Property

Public Property Get Id() As Long
'used when retrieving value of a property, on the right side of
an assignment.
'Syntax: Debug.Print X.Id
    Id = mlId
End Property

Public Function OpenWith(W_ID As Long, cn As ADODB.Connection) As
Variant
Dim rstWarehouse As New ADODB.Recordset

If W_ID <= 0 Then Exit Function

With rstWarehouse
.Open "Select * from Warehouse where W_ID = " & W_ID, cn,
adOpenForwardOnly, adLockReadOnly
If Not .EOF Then
    mlId = W_ID
    msStreet1 = !W_STREET_1
    msStreet2 = !W_STREET_2
    msCity = !W_CITY
    msZIP = !W_ZIP
    msState = !W_STATE
    msngTax = !W_TAX
    msName = !W_NAME
    msngYTD = !W_YTD
    OpenWith = True
End If
End With
End Function

Public Function Save(cn As ADODB.Connection) As Boolean
Dim rstWarehouse As New ADODB.Recordset

With rstWarehouse
.Open "Select * from Warehouse where W_ID = " & mlId, cn,
adOpenDynamic, adLockPessimistic
If Not .EOF Then
    !W_YTD = msngYTD
    .Update
End If
.Close
End With
Save = True
End Function
```

4. N-tier Business Objects

```
Option Explicit

'// Transaction

Dim mcnThesis As ADODB.Connection
Dim msResult As String
Dim mDelivery As New ThesisQP.DeliveryRcv

Public Function IsOk() As Boolean
    IsOk = True
End Function

Public Function GetStockLevel(ByVal W_ID As Long, ByVal D_ID As
Long, ByVal Threshold As Integer, sResult As String) As Boolean
On Error GoTo GSError
Dim bTransaction As Boolean
Dim rstStockLevel As New ADODB.Recordset
Dim sSql As String
Dim iStockLevel As Integer
Dim oWarehouse As ThesisDO.Warehouse
Dim oDistrict As ThesisDO.District

Set oWarehouse = CreateObject("ThesisDO.Warehouse")
Set oDistrict = CreateObject("ThesisDO.District")

mcnThesis.Open

'// Warehouse
oWarehouse.Id = W_ID

'// District
Set oDistrict.Warehouse = oWarehouse
oDistrict.OpenWith D_ID, oWarehouse, mcnThesis

sSql = "SELECT COUNT(*) as Low_Stock FROM Order_Line INNER JOIN
Stock ON "
sSql = sSql & "Order_Line.OL_I_ID = Stock.S_I_ID And
Order_Line.OL_SUPPLY_W_ID = "
sSql = sSql & "Stock.S_W_ID WHERE OL_D_ID =" & oDistrict.Id & "
AND OL_W_ID = "
sSql = sSql & oDistrict.Warehouse.Id & " AND OL_O_ID > " &
oDistrict.NextOrderId - 21
sSql = sSql & " AND " & "S_QUANTITY < " & Threshold

With rstStockLevel
.Open sSql, mcnThesis, adOpenForwardOnly, adLockReadOnly
iStockLevel = !Low_Stock
End With
mcnThesis.Close

msResult = ""
PrintResult "Stock-Level", 32, , True
PrintResult "Warehouse: " & Format(W_ID, "0000")
PrintResult "District: " & Format(oDistrict.Id, "00"), 3, , True
PrintResult "", , , True
PrintResult "Stock Level Threshold: " & Format(Threshold, "00"),
, , True
PrintResult "", , , True
PrintResult "Low Stock: " & Format(iStockLevel, "00"), , , True
PrintResult "", , , True

sResult = msResult
GetStockLevel = True

GSFim:
Exit Function

GSAbort:
mcnThesis.Close
GoTo GSFim

GSError:
MsgBox Error, vbCritical
Resume GSFim

End Function

Public Function GetMaxOrder(ByVal W_ID As Long, ByVal D_ID As
Long, ByVal C_LAST As String, sResult As String) As Boolean
On Error GoTo GMError
Dim bTransaction As Boolean
Dim oWarehouse As ThesisDO.Warehouse
Dim oDistrict As ThesisDO.District
Dim oCustomer As ThesisDO.Customer
Dim oMaxOrder As ThesisDO.Order
Dim oItem As ThesisDO.Item
Dim oOrderLine As ThesisDO.OrderLine
Dim rstOrder As New ADODB.Recordset
Dim lMaxId As Long

Set oWarehouse = CreateObject("ThesisDO.Warehouse")
Set oDistrict = CreateObject("ThesisDO.District")
Set oCustomer = CreateObject("ThesisDO.Customer")
Set oMaxOrder = CreateObject("ThesisDO.Order")

mcnThesis.Open

'// Warehouse
oWarehouse.Id = W_ID

'// District
Set oDistrict.Warehouse = oWarehouse
oDistrict.Id = D_ID

'// Customer
oCustomer.OpenWithLast C_LAST, oDistrict, mcnThesis

With rstOrder
    '// Get Maximum Order
    .Open "Select Max(O_ID) as MaxOID from District_Order where
O_D_ID = " & oCustomer.District.Id & " and O_W_ID=" &
oCustomer.District.Warehouse.Id, mcnThesis, adOpenForwardOnly,
adLockReadOnly
    If .EOF Then GoTo GMAbort
    lMaxId = !MaxOID
End With

'// Open Order
If Not oMaxOrder.OpenWith(lMaxId, oCustomer, mcnThesis) Then GoTo
GMAbort

'// Open Order Lines
oMaxOrder.OpenOLs mcnThesis
mcnThesis.Close

'// Print Results
msResult = ""
PrintResult "Order-Status", 33, , True
PrintResult "Warehouse: " & Format(W_ID, "0000")
PrintResult "District: " & Format(oDistrict.Id, "00"), 3, , True
PrintResult "Customer:  " & Format(oCustomer.Id, "0000")
With oCustomer
PrintResult .First, 3, 16
PrintResult .Middle, 1
PrintResult .Last, 1, 16, True
PrintResult "Cust-Balance: "
PrintResult "$" & Format(.Balance, "000000000.00"), , , True
PrintResult "", , , True
End With
With oMaxOrder
PrintResult "Order-Number: " & Format(.Id, "00000000")
PrintResult "Entry-Date: " & Format(.EntryDate, "dd-mm-yyyy
hh:mm:ss"), 3
PrintResult "Carrier-Number: ", 2
If Not IsNull(.CarrierID) Then
     PrintResult Format(.CarrierID, "00"), , , True
Else
    PrintResult "NULL", , , True
End If
PrintResult "Supp_W"
PrintResult "Item_Id", 7
PrintResult "Qty", 4
PrintResult "Amount", 5
PrintResult "Delivery-Date", 6, , True
End With
For Each oOrderLine In oMaxOrder
    With oOrderLine
    PrintResult Format(.SupplyWId, "0000"), 1
    PrintResult Format(.Item.Id, "000000"), 8
    PrintResult Format(.Quantity, "00"), 5
    PrintResult Format(.Total, "$00000.00"), 5
    If .DeliveryDate = 0 Then
        PrintResult "NULL", 6, , True
    Else
        PrintResult Format(.DeliveryDate, "dd-mm-yyyy"), 6, ,
True
    End If
    End With
Next

sResult = msResult
GetMaxOrder = True

GMFim:
Exit Function

GMAbort:
mcnThesis.Close
GoTo GMFim

GMError:
MsgBox Error, vbCritical
Resume GMFim

End Function

Public Function Payment(ByVal W_ID As Long, ByVal D_ID As Long,
ByVal C_LAST As String, ByVal Amount As Single, sResult As
String) As Boolean
On Error GoTo PAError
Dim bTransaction As Boolean
Dim sAux As String
Dim oWarehouse As ThesisDO.Warehouse
Dim oDistrict As ThesisDO.District
Dim oCustomer As ThesisDO.Customer
Dim oHistory As ThesisDO.History

Payment = False

Set oWarehouse = CreateObject("ThesisDO.Warehouse")
Set oDistrict = CreateObject("ThesisDO.District")
Set oCustomer = CreateObject("ThesisDO.Customer")
Set oHistory = CreateObject("ThesisDO.History")

mcnThesis.Open
mcnThesis.BeginTrans
bTransaction = True

'// Warehouse
oWarehouse.OpenWith W_ID, mcnThesis

'// District
```

```
oDistrict.OpenWith D_ID, oWarehouse, mcnThesis

'// Customer
oCustomer.OpenWithLast C_LAST, oDistrict, mcnThesis

With oCustomer
'// Check Customer Credit and update data if necessary
If .Credit = "BC" Then
    sAux = "Entry: " & .Id & ", " & .District.Id & "; " &
.District.Warehouse.Id
    sAux = sAux & "; " & .District.Id & "; " &
.District.Warehouse.Id & "; "
    sAux = sAux & Amount & "."
    If Len(.Data) - Len(sAux) > 500 Then
        .Data = sAux & Left(.Data, 500 - Len(sAux))
    Else
        .Data = sAux & .Data
    End If
    If Not oCustomer.Save(mcnThesis) Then GoTo PAAbort
End If
End With

'// Save History
oHistory.Amount = Amount
oHistory.EntryDate = Date
Set oHistory.Customer = oCustomer
Set oHistory.District = oCustomer.District
oHistory.Data = oCustomer.District.Warehouse.Name & "    " &
oCustomer.District.Name
If Not oHistory.Save(mcnThesis) Then GoTo PAAbort

'// Save Warehouse
If Not oWarehouse.Save(mcnThesis) Then GoTo PAAbort

'// Save District
If Not oDistrict.Save(mcnThesis) Then GoTo PAAbort

'// Save Customer
If Not oCustomer.Save(mcnThesis) Then GoTo PAAbort

mcnThesis.CommitTrans
bTransaction = False
mcnThesis.Close

msResult = ""
PrintResult "Payment", 33, , True
PrintResult "Date: " & Format(Time, "dd-mm-yyyy hh:mm:ss"), , ,
True
PrintResult "", , , True
PrintResult "Warehouse: " & Format(W_ID, "0000")
PrintResult "District: " & Format(oDistrict.Id, "00"), 25, , True
PrintResult oWarehouse.Street1, , 20
PrintResult oDistrict.Street1, 20, 20, True
PrintResult oWarehouse.Street2, , 20
PrintResult oDistrict.Street2, 20, 20, True
PrintResult oWarehouse.City, , 20
PrintResult oWarehouse.State, 1
PrintResult Left(oWarehouse.ZIP, 5) & "-" & Right(oWarehouse.ZIP,
4), 1
PrintResult oDistrict.City, 6, 20
PrintResult oDistrict.State, 1
PrintResult Left(oDistrict.ZIP, 5) & "-" & Right(oDistrict.ZIP,
3), 1, , True
PrintResult "", , , True
PrintResult "Customer:  " & Format(oCustomer.Id, "0000"), , ,
True
PrintResult "Name:   "
With oCustomer
PrintResult .First, , 16
PrintResult .Middle, 1
PrintResult .Last, 1, 16
PrintResult "Since:  " & Format(.Since, "dd-mm-yyyy"), 5, , True
PrintResult .Street1, 8, 20
PrintResult "Credit: " & .Credit, 21, , True
PrintResult .Street2, 8, 20
PrintResult "%Disc:  " & Format(.Discount * 100, "00.00"), 21, ,
True
PrintResult .City, 8, 20
PrintResult .State, 1
PrintResult Left(.ZIP, 5) & "-" & Right(.ZIP, 3), 1
PrintResult "Phone:  " & Left(.Phone, 6) & "-" & Mid(.Phone, 7,
3) & "-" & Mid(.Phone, 10, 3) & "-" & Right(.Phone, 4), 8, , True
PrintResult "", , , True
PrintResult "Amount Paid:"
PrintResult Format(Amount, "$0000.00"), 10
PrintResult "New Cust-Balance:", 6
PrintResult "$" & Format(.Balance - Amount, "000000000.00"), 1, ,
True
PrintResult "Credit Limit:"
PrintResult "$" & Format(.Limit, "000000000.00"), 4, , True
PrintResult "", , , True

If .Credit = "BC" Then
    PrintResult "Cust-Data:"
    PrintResult Left(.Data, 50), 1, , True
    PrintResult Mid(.Data, 51, 50), 11, , True
    PrintResult Mid(.Data, 101, 50), 11, , True
    PrintResult Mid(.Data, 151, 50), 11, , True
    PrintResult "", , , True
End If
End With

sResult = msResult
Payment = True

PAFim:
Exit Function

PAAbort:
If bTransaction Then
    bTransaction = False
    mcnThesis.RollbackTrans
End If
mcnThesis.Close

GoTo PAFim

PAError:
If bTransaction Then
    bTransaction = False
    mcnThesis.RollbackTrans
End If
MsgBox Error, vbCritical
Resume 'PAFim

End Function

Public Function NewOrder(ByVal W_ID As Long, ByVal D_ID As Long,
ByVal C_ID As Long, ByVal OrderCnt As Integer, ByVal Items As
Variant, ByVal Qtys As Variant, sResult As String) As Boolean
On Error GoTo NOError

Dim bTransaction As Boolean
Dim i As Integer
Dim Item() As Long
Dim Qty() As Long

Dim oWarehouse As ThesisDO.Warehouse
Dim oDistrict As ThesisDO.District
Dim oCustomer As ThesisDO.Customer
Dim oStock As ThesisDO.StockItem
Dim oNewOrder As ThesisDO.NewOrder
Dim oOrder As ThesisDO.Order
Dim oItem As ThesisDO.Item
Dim oOrderLine As ThesisDO.OrderLine

NewOrder = False

Set oWarehouse = CreateObject("ThesisDO.Warehouse")
Set oDistrict = CreateObject("ThesisDO.District")
Set oCustomer = CreateObject("ThesisDO.Customer")
Set oStock = CreateObject("ThesisDO.StockItem")
Set oNewOrder = CreateObject("ThesisDO.NewOrder")
Set oOrder = CreateObject("ThesisDO.Order")

mcnThesis.Open
mcnThesis.BeginTrans
bTransaction = True

'// Warehouse
oWarehouse.OpenWith W_ID, mcnThesis

'// District
oDistrict.OpenWith D_ID, oWarehouse, mcnThesis

'// Customer
oCustomer.OpenWith C_ID, oDistrict, mcnThesis

'// Order
Set oOrder.Customer = oCustomer

'// Order Lines
ReDim Item(1 To OrderCnt)
ReDim Qty(1 To OrderCnt)
Item = Items
Qty = Qtys

'// Generate Results
msResult = ""
PrintResult "New Order", 33, , True
PrintResult "Warehouse: " & Format(W_ID, "0000")
PrintResult "District: " & Format(D_ID, "00"), 3
PrintResult "Date: " & Format(Time, "dd-mm-yyyy hh:mm:ss"), 23, ,
True
PrintResult "Customer:  " & Format(oCustomer.Id, "0000")
PrintResult "Name: " & Left(oCustomer.Last, 16), 3, 21
PrintResult "Credit: " & oCustomer.Credit, 3
PrintResult "%Disc: " & Format(oCustomer.Discount * 100,
"00.00"), 3, , True
PrintResult "Order Number: " & Format(oOrder.Id, "00000000")
PrintResult "Number of Lines: " & Format(OrderCnt, "00"), 2
PrintResult "W_tax: " & Format(oWarehouse.Tax * 100, "00.00"), 7
PrintResult "D_tax: " & Format(oDistrict.Tax * 100, "00.00"), 3,
, True
PrintResult "", , , True
PrintResult "Supp_W", 1
PrintResult "Item_Id", 2
PrintResult "Item_Name", 2
PrintResult "Qty", 16
PrintResult "Stock", 2
PrintResult "B/G", 2
PrintResult "Price", 2
PrintResult "Amount", 4, , True

For i = 1 To OrderCnt
    Set oItem = CreateObject("ThesisDO.Item")
    oItem.OpenWith Item(i), mcnThesis

    '// Order Line
    Set oOrderLine = CreateObject("ThesisDO.OrderLine")
    With oOrderLine
    Set .Item = oItem
    .Quantity = Qty(i)

    '// Update Stock
    oStock.OpenWith .Item, oOrder.Customer.District.Warehouse,
mcnThesis
    Set .StockItem = oStock
    If oStock.Quantity > .Quantity + 10 Then
        oStock.Quantity = oStock.Quantity - .Quantity
    Else
        oStock.Quantity = oStock.Quantity + 91
    End If
    oStock.YTD = oStock.YTD + .Quantity
    oStock.Order_CNT = oStock.Order_CNT + 1

    oOrder.Add oOrderLine

    '// Results
```

```
    PrintResult Format(W_ID, "0000"), 2
    PrintResult Format(.Item.Id, "000000"), 3
    PrintResult Left(.Item.Name, 23), 3, 23
    PrintResult Format(.Quantity, "00"), 2
    PrintResult Format(.StockItem.Quantity, "000"), 4
    PrintResult IIf(.BG, "B", "G"), 4
    PrintResult Format(.Item.Price, "$000.00"), 3
    PrintResult Format(.Item.Price * .Quantity, "$000.00"), 2, ,
True

    Set oOrderLine = Nothing
    Set oItem = Nothing
    End With
Next

With oOrder
'// Save District
If Not .Customer.District.Save(mcnThesis) Then GoTo NOAbort
    .EntryDate = Format(Date, "mm/dd/yyyy")
    .AllLocal = True
'// Save Customer
    If Not .Customer.Save(mcnThesis) Then GoTo NOAbort
'// Save order
If Not .Save(mcnThesis) Then GoTo NOAbort
'// Save New Order
Set oNewOrder.Order = oOrder
    If Not oNewOrder.Save(mcnThesis) Then GoTo NOAbort
End With

For i = 1 To OrderCnt
    '// Update OrderLine
    If Not oOrder(i).Save(mcnThesis) Then GoTo NOAbort
Next

PrintResult "Execution Status: Ok"
PrintResult "Total:  " & Format(oOrder.Total, "$0000.00"), 11, ,
True

60: mcnThesis.CommitTrans
bTransaction = False
mcnThesis.Close
NewOrder = True

sResult = msResult

NOFim:
Exit Function

NOAbort:
If bTransaction Then
    bTransaction = False
    mcnThesis.RollbackTrans
End If
GoTo NOFim

NOError:
If bTransaction Then
    bTransaction = False
    mcnThesis.RollbackTrans
End If
MsgBox Erl & ":" & Err.Number & " - " & Err.Description,
vbCritical
Resume NOFim

End Function

Private Sub Class_Initialize()
Dim sConnection As String
Set mcnThesis = CreateObject("ADODB.Connection")
sConnection = "Provider=SQLOLEDB.1;Integrated
Security=SSPI;Persist Security Info=False; "
sConnection = sConnection & "User ID=sa;Initial
Catalog=Thesis;Data Source=abcnt09a;"
sConnection = sConnection & "Locale Identifier=1046;Connect
Timeout=15;Use Procedure for Prepare=1;"
sConnection = sConnection & "Auto Translate=True;Packet
Size=4096;Workstation ID=ALEXANDRENT"
mcnThesis.ConnectionString = sConnection
mDelivery.Start
End Sub
Private Sub PrintResult(sText, Optional iSpaces As Integer = 0,
Optional SizeToFit As Integer = 0, Optional bLineFeed As Boolean
= False)
Static bOldLine As Boolean

    If Not bOldLine Then
        iSpaces = iSpaces + 1
        bOldLine = True
    End If
    msResult = msResult & Space(iSpaces)
    msResult = msResult & sText
    If SizeToFit > 0 Then
        If Len(sText) < SizeToFit Then
            msResult = msResult & Space(SizeToFit - Len(sText))
        End If
    End If
    If bLineFeed Then
        msResult = msResult & vbCrLf
        bOldLine = False
    End If
End Sub

Option Explicit

'// Gen Rand

Public Function Random(x As Long, y As Long) As Long
Randomize
Random = Int(Rnd() * (y - x)) + x
End Function

Public Function NURand(A As Long, x As Long, y As Long) As Long
Dim C As Long
C = A / 2
NURand = (((Random(0, A) Or Random(x, y)) + C) Mod (y - x + 1)) +
x
End Function

Public Function GenerateStr(iLen As Long) As String
Dim i As Long
Dim sAux As String
Dim cAux As String

For i = 1 To iLen
    cAux = Chr(Int(58 * Rnd) + 32)
    sAux = sAux & cAux
Next
GenerateStr = sAux
End Function

Option Explicit

'// LastNameGen

Private LNSyllables(0 To 9) As String

Private Sub Class_Initialize()
LNSyllables(0) = "BAR"
LNSyllables(1) = "OUGHT"
LNSyllables(2) = "ABLE"
LNSyllables(3) = "PRI"
LNSyllables(4) = "PRES"
LNSyllables(5) = "ESE"
LNSyllables(6) = "ANTI"
LNSyllables(7) = "CALLY"
LNSyllables(8) = "ATION"
LNSyllables(9) = "EING"
End Sub

Public Function GenerateLastNameStr(sCode As String) As String
Dim iIndex As Long
Dim sAux As String

sCode = Trim(sCode)
If Len(sCode) < 3 Then
    sCode = Space(3 - Len(sCode)) & sCode
End If
iIndex = Val(Right(sCode, 1))
sAux = LNSyllables(iIndex)
If Len(sCode) = 2 Then
    iIndex = Val(Left(sCode, 1))
    sAux = LNSyllables(iIndex) & sAux
ElseIf Len(sCode) > 2 Then
    iIndex = Val(Mid(sCode, 2, 1))
    sAux = LNSyllables(iIndex) & sAux
    iIndex = Val(Left(sCode, 1))
    sAux = LNSyllables(iIndex) & sAux
End If
GenerateLastNameStr = sAux

End Function

Option Explicit

'// DeliveryRcv

Private WithEvents EventThesis As MSMQEvent
Dim mQueue As MSMQQueue
Dim mcnThesis As ADODB.Connection

Public Function Start() As Boolean
    Dim MQInfo As New MSMQQueueInfo
    Dim sConnection As String

    Set mcnThesis = CreateObject("ADODB.Connection")
    sConnection = "Provider=SQLOLEDB.1;Integrated
Security=SSPI;Persist Security Info=False; "
    sConnection = sConnection & "User ID=sa;Initial
Catalog=Thesis;Data Source=abcnt09a;"
    sConnection = sConnection & "Locale Identifier=1046;Connect
Timeout=15;Use Procedure for Prepare=1;"
    sConnection = sConnection & "Auto Translate=True;Packet
Size=4096;Workstation ID=ALEXANDRENT"
    mcnThesis.ConnectionString = sConnection

    Set EventThesis = New MSMQEvent
    MQInfo.PathName = "abcnt06b\thesis"
    Set mQueue = MQInfo.Open(MQ_RECEIVE_ACCESS, MQ_DENY_NONE)
    mQueue.EnableNotification EventThesis

End Function

Private Sub EventThesis_Arrived(ByVal Queue As Object, ByVal
Cursor As Long)
On Error GoTo ArrError

    Dim msgResp As MSMQMessage
    Dim W_ID As Long
    Dim CARRIER_ID As Long
    Dim D_ID As Long
    Dim O_ID As Long
    Dim rstNewOrder As New ADODB.Recordset
    Dim oOrder As ThesisDO.Order
    Dim oWarehouse As ThesisDO.Warehouse
    Dim oDistrict As ThesisDO.District
    Dim oOL As ThesisDO.OrderLine
    Dim fTransaction As Boolean

    mcnThesis.Open
    Set msgResp = mQueue.Receive
    W_ID = Val(Left(msgResp.Body, 2))
    CARRIER_ID = Val(Right(msgResp.Body, 2))
    For D_ID = 1 To 10
        With rstNewOrder
        O_ID = 0
```

```
        .Open "Select Min(NO_O_ID) as O_ID from New_Order where
NO_W_ID = " & W_ID & " and NO_D_ID = " & D_ID, mcnThesis,
adOpenForwardOnly, adLockReadOnly
        If Not .EOF Then
            If Not IsNull(!O_ID) Then O_ID = !O_ID
        End If
        .Close
        End With
        If O_ID > 0 Then
            mcnThesis.BeginTrans
            fTransaction = True

            Set oOrder = CreateObject("ThesisDO.Order")
            Set oDistrict = CreateObject("ThesisDO.District")
            Set oWarehouse = CreateObject("ThesisDO.Warehouse")

            '// Set Warehouse
            oWarehouse.Id = W_ID

            '// Set District
            oDistrict.Id = D_ID
            Set oDistrict.Warehouse = oWarehouse

            '// Open Order
            If Not oOrder.OpenWithD(O_ID, oDistrict, mcnThesis)
Then GoTo ArrFail

            '// Update order
            oOrder.CarrierID = CARRIER_ID
            oOrder.Save mcnThesis

            '// Update Customer
            oOrder.Customer.Balance = oOrder.Customer.Balance -
oOrder.Total
            oOrder.Customer.Delivery_CNT =
oOrder.Customer.Delivery_CNT + 1
            oOrder.Customer.Save mcnThesis

            For Each oOL In oOrder
                '// Update Order Line
                oOL.DeliveryDate = Date
                oOL.Save mcnThesis
            Next

            '// Delete New Order
            mcnThesis.Execute "Delete New_Order where NO_O_ID = "
& O_ID & " and NO_D_ID =" & D_ID & " and NO_W_ID = " & W_ID

            mcnThesis.CommitTrans
            fTransaction = False.
        End If

        Set oOrder = Nothing
        Set oDistrict = Nothing
        Set oWarehouse = Nothing
    Next

ArrFim:
    mcnThesis.Close
    mQueue.EnableNotification EventThesis
    Exit Sub

ArrFail:
    If fTransaction Then mcnThesis.RollbackTrans
    GoTo ArrFim

ArrError:
    MsgBox Err.Description, vbCritical
    Resume ArrFail

End Sub
```

5. N-tier Business Objects (MTS)

```
Option Explicit

'// Transactions (MTS)

Dim msResult As String
Dim mDelivery As ThesisQPMTS.DeliveryRcv
Dim miCounter As Integer
Dim miClients As Integer

Private Enum MyErrors
    GeneralError = vbObjectError
    NewOrderErr
End Enum

Public Property Get Counter() As Integer
    Counter = miCounter
End Property

Public Property Get Clients() As Integer
    Clients = miClients
End Property

Public Function IsOk() As Boolean
    IsOk = True
End Function

Public Function GetStockLevel(ByVal W_ID As Long, ByVal D_ID As
Long, ByVal Threshold As Integer, sResult As String) As Boolean
On Error GoTo GSError
Dim bTransaction As Boolean
Dim oWarehouse As ThesisDO.Warehouse
Dim oDistrict As ThesisDO.District

Set oWarehouse = CreateObject("ThesisDO.Warehouse")
Set oDistrict = CreateObject("ThesisDO.District")

'// Warehouse
oWarehouse.Id = W_ID

'// District
Set oDistrict.Warehouse = oWarehouse
oDistrict.OpenWith D_ID, oWarehouse

msResult = ""
PrintResult "Stock-Level", 32, , True
PrintResult "Warehouse: " & Format(W_ID, "0000")
PrintResult "District: " & Format(oDistrict.Id, "00"), 3, , True
PrintResult "", , , True
PrintResult "Stock Level Threshold: " & Format(Threshold, "00"),
, , True
PrintResult "", , , True
PrintResult "Low Stock: " & Format(oDistrict.LowStock(Threshold),
"00"), , , True
PrintResult "", , , True

sResult = msResult
GetStockLevel = True

GSFim:
Exit Function

GSAbort:
'mcnThesis.Close
GoTo GSFim

GSError:
MsgBox Error, vbCritical
Resume GSFim

End Function

Public Function GetMaxOrder(ByVal W_ID As Long, ByVal D_ID As
Long, ByVal C_LAST As String, sResult As String) As Boolean
On Error GoTo GMError
Dim bTransaction As Boolean
Dim oWarehouse As ThesisDO.Warehouse
Dim oDistrict As ThesisDO.District
Dim oCustomer As ThesisDO.Customer
Dim oMaxOrder As ThesisDO.Order
Dim oItem As ThesisDO.Item
Dim oOrderLine As ThesisDO.OrderLine
Dim rstOrder As New ADODB.Recordset
Dim lMaxId As Long

Set oWarehouse = CreateObject("ThesisDO.Warehouse")
Set oDistrict = CreateObject("ThesisDO.District")
Set oCustomer = CreateObject("ThesisDO.Customer")
Set oMaxOrder = CreateObject("ThesisDO.Order")

'mcnThesis.Open

'// Warehouse
oWarehouse.Id = W_ID

'// District
Set oDistrict.Warehouse = oWarehouse
oDistrict.Id = D_ID

'// Customer
oCustomer.OpenWithLast C_LAST, oDistrict

'// Open Order
If Not oMaxOrder.OpenWith(0, oCustomer) Then GoTo GMAbort

'// Open Order Lines
oMaxOrder.OpenOLs
'mcnThesis.Close

'// Print Results
msResult = ""
PrintResult "Order-Status", 33, , True
PrintResult "Warehouse: " & Format(W_ID, "0000")
PrintResult "District: " & Format(oDistrict.Id, "00"), 3, , True
PrintResult "Customer:  " & Format(oCustomer.Id, "0000")
With oCustomer
PrintResult .First, 3, 16
PrintResult .Middle, 1
PrintResult .Last, 1, 16, True
PrintResult "Cust-Balance: "
PrintResult "$" & Format(.Balance, "000000000.00"), , , True
PrintResult "", , , True
End With
With oMaxOrder
PrintResult "Order-Number: " & Format(.Id, "00000000")
PrintResult "Entry-Date: " & Format(.EntryDate, "dd-mm-yyyy
hh:mm:ss"), 3
PrintResult "Carrier-Number: ", 2
If Not IsNull(.CarrierID) Then
    PrintResult Format(.CarrierID, "00"), , , True
Else
    PrintResult "NULL", , , True
End If
PrintResult "Supp_W"
PrintResult "Item_Id", 7
PrintResult "Qty", 4
PrintResult "Amount", 5
PrintResult "Delivery-Date", 6, , True
End With
For Each oOrderLine In oMaxOrder
    With oOrderLine
    PrintResult Format(.SupplyWId, "0000"), 1
    PrintResult Format(.Item.Id, "000000"), 8
    PrintResult Format(.Quantity, "00"), 5
    PrintResult Format(.Total, "$00000.00"), 5
    If .DeliveryDate = 0 Then
        PrintResult "NULL", 6, , True
    Else
        PrintResult Format(.DeliveryDate, "dd-mm-yyyy"), 6, ,
True
    End If
    End With
Next

sResult = msResult
GetMaxOrder = True

GMFim:
Exit Function

GMAbort:
'mcnThesis.Close
GoTo GMFim

GMError:
MsgBox Error, vbCritical
Resume GMFim

End Function

Public Function Payment(ByVal W_ID As Long, ByVal D_ID As Long,
ByVal C_LAST As String, ByVal Amount As Single, sResult As
String) As Boolean
On Error GoTo PAError
Dim bTransaction As Boolean
Dim sAux As String
Dim oWarehouse As ThesisDO.Warehouse
Dim oDistrict As ThesisDO.District
Dim oCustomer As ThesisDO.Customer
Dim oHistory As ThesisDO.History
Dim oContext As MTxAS.ObjectContext

Payment = False

Set oContext = GetObjectContext()
If oContext Is Nothing Then
   MsgBox "oops"
End If

Set oWarehouse = CreateObject("ThesisDO.Warehouse")
Set oDistrict = CreateObject("ThesisDO.District")
Set oCustomer = CreateObject("ThesisDO.Customer")
Set oHistory = CreateObject("ThesisDO.History")

'mcnThesis.Open
'mcnThesis.BeginTrans
bTransaction = True

'// Warehouse
oWarehouse.OpenWith W_ID

'// District
oDistrict.OpenWith D_ID, oWarehouse

'// Customer
oCustomer.OpenWithLast C_LAST, oDistrict

With oCustomer
'// Check Customer Credit and update data if necessary
If .Credit = "BC" Then
```

```
    sAux = "Entry: " & .Id & ", " & .District.Id & "; " &
.District.Warehouse.Id
    sAux = sAux & "; " & .District.Id & "; " &
.District.Warehouse.Id & "; "
    sAux = sAux & Amount & "."
    If Len(.Data) - Len(sAux) > 500 Then
        .Data = sAux & Left(.Data, 500 - Len(sAux))
    Else
        .Data = sAux & .Data
    End If
    If Not oCustomer.Save Then GoTo PAAbort
End If
End With

'// Save History
oHistory.Amount = Amount
oHistory.EntryDate = Date
Set oHistory.Customer = oCustomer
Set oHistory.District = oCustomer.District
oHistory.Data = oCustomer.District.Warehouse.Name & "    " &
oCustomer.District.Name
If Not oHistory.Save Then GoTo PAAbort

'// Save Warehouse
If Not oWarehouse.Save Then GoTo PAAbort

'// Save District
If Not oDistrict.Save Then GoTo PAAbort

'// Save Customer
If Not oCustomer.Save Then GoTo PAAbort

'mcnThesis.CommitTrans
bTransaction = False
'mcnThesis.Close
oContext.SetComplete

msResult = ""
PrintResult "Payment", 33, , True
PrintResult "Date: " & Format(Time, "dd-mm-yyyy hh:mm:ss"), , ,
True
PrintResult "", , , True
PrintResult "Warehouse: " & Format(W_ID, "0000")
PrintResult "District: " & Format(oDistrict.Id, "00"), 25, , True
PrintResult oWarehouse.Street1, , 20
PrintResult oDistrict.Street1, 20, 20, True
PrintResult oWarehouse.Street2, , 20
PrintResult oDistrict.Street2, 20, 20, True
PrintResult oWarehouse.City, , 20
PrintResult oWarehouse.State, 1
PrintResult Left(oWarehouse.ZIP, 5) & "-" & Right(oWarehouse.ZIP,
4), 1
PrintResult oDistrict.City, 6, 20
PrintResult oDistrict.State, 1
PrintResult Left(oDistrict.ZIP, 5) & "-" & Right(oDistrict.ZIP,
3), 1, , True
PrintResult "", , , True
PrintResult "Customer:  " & Format(oCustomer.Id, "0000"), , ,
True
PrintResult "Name:   "
With oCustomer
PrintResult .First, , 16
PrintResult .Middle, 1
PrintResult .Last, 1, 16
PrintResult "Since:  " & Format(.Since, "dd-mm-yyyy"), 5, , True
PrintResult .Street1, 8, 20
PrintResult "Credit: " & .Credit, 21, , True
PrintResult .Street2, 8, 20
PrintResult "%Disc:  " & Format(.Discount * 100, "00.00"), 21, ,
True
PrintResult .City, 8, 20
PrintResult .State, 1
PrintResult Left(.ZIP, 5) & "-" & Right(.ZIP, 3), 1
PrintResult "Phone:  " & Left(.Phone, 6) & "-" & Mid(.Phone, 7,
3) & "-" & Mid(.Phone, 10, 3) & "-" & Right(.Phone, 4), 8, , True
PrintResult "", , , True
PrintResult "Amount Paid:"
PrintResult Format(Amount, "$0000.00"), 10
PrintResult "New Cust-Balance:", 6
PrintResult "$" & Format(.Balance - Amount, "000000000.00"), 1, ,
True
PrintResult "Credit Limit:"
PrintResult "$" & Format(.Limit, "000000000.00"), 4, , True
PrintResult "", , , True

If .Credit = "BC" Then
    PrintResult "Cust-Data:"
    PrintResult Left(.Data, 50), 1, , True
    PrintResult Mid(.Data, 51, 50), 11, , True
    PrintResult Mid(.Data, 101, 50), 11, , True
    PrintResult Mid(.Data, 151, 50), 11, , True
    PrintResult "", , , True
End If
End With

sResult = msResult
Payment = True

PAFim:
Exit Function

PAAbort:
If bTransaction Then
    bTransaction = False
    'mcnThesis.RollbackTrans
End If
'mcnThesis.Close
GoTo PAFim

PAError:
If oContext.IsInTransaction Then
    oContext.SetAbort
End If
If bTransaction Then
    bTransaction = False
    'mcnThesis.RollbackTrans
End If
MsgBox Error, vbCritical
Resume PAFim

End Function

Public Function NewOrder(ByVal W_ID As Long, ByVal D_ID As Long,
ByVal C_ID As Long, ByVal OrderCnt As Integer, ByVal Items As
Variant, ByVal Qtys As Variant, sResult As String, Optional ByVal
SUPP_W_ID As Long = 1) As Boolean
On Error GoTo NOError

Dim bTransaction As Boolean
Dim i As Integer
Dim Item() As Long
Dim Qty() As Long
Dim SUPP_W_ID2 As Long

Dim oContext As MTxAS.ObjectContext
Dim oWarehouse As ThesisDO.Warehouse
Dim oSuppWar As ThesisDO.Warehouse
Dim oDistrict As ThesisDO.District
Dim oCustomer As ThesisDO.Customer
Dim oStock As ThesisDO.StockItem
Dim oNewOrder As ThesisDO.NewOrder
Dim oOrder As ThesisDO.Order
Dim oItem As ThesisDO.Item
Dim oOrderLine As ThesisDO.OrderLine

NewOrder = False
Set oContext = GetObjectContext()
If oContext Is Nothing Then
    MsgBox "oops"
End If
'oContext.SetComplete
'Exit Function
'With oContext
Set oWarehouse = New ThesisDO.Warehouse
Set oDistrict = New ThesisDO.District
Set oCustomer = New ThesisDO.Customer
Set oStock = New ThesisDO.StockItem
Set oNewOrder = New ThesisDO.NewOrder
Set oOrder = New ThesisDO.Order
'End With

'mcnThesis.Open
'mcnThesis.BeginTrans
'bTransaction = True

'// Warehouse
oWarehouse.OpenWith W_ID

'// District
oDistrict.OpenWith D_ID, oWarehouse

'// Customer
oCustomer.OpenWith C_ID, oDistrict

'// Order
Set oOrder.Customer = oCustomer

'// Order Lines
ReDim Item(1 To OrderCnt)
ReDim Qty(1 To OrderCnt)
Item = Items
Qty = Qtys

'// Generate Results
msResult = ""
PrintResult "New Order", 33, , True
PrintResult "Warehouse: " & Format(W_ID, "0000")
PrintResult "District: " & Format(D_ID, "00"), 3
PrintResult "Date: " & Format(Time, "dd-mm-yyyy hh:mm:ss"), 23, ,
True
PrintResult "Customer:  " & Format(oCustomer.Id, "0000")
PrintResult "Name: " & Left(oCustomer.Last, 16), 3, 21
PrintResult "Credit: " & oCustomer.Credit, 3
PrintResult "%Disc: " & Format(oCustomer.Discount * 100,
"00.00"), 3, , True
PrintResult "Order Number: " & Format(oOrder.Id, "00000000")
PrintResult "Number of Lines: " & Format(OrderCnt, "00"), 2
PrintResult "W_tax: " & Format(oWarehouse.Tax * 100, "00.00"), 7
PrintResult "D_tax: " & Format(oDistrict.Tax * 100, "00.00"), 3,
, True
PrintResult "", , , True
PrintResult "Supp_W", 1
PrintResult "Item_Id", 2
PrintResult "Item_Name", 2
PrintResult "Qty", 16
PrintResult "Stock", 2
PrintResult "B/G", 2
PrintResult "Price", 2
PrintResult "Amount", 4, , True

For i = 1 To OrderCnt
    Set oItem = New ThesisDO.Item
    Set oSuppWar = New ThesisDO.Warehouse

    oItem.OpenWith Item(i)
    '// Open the correct Warehouse
    SUPP_W_ID2 = Int(Rnd() * 2) + 1

    oSuppWar.Id = SUPP_W_ID2

    '// Order Line
    Set oOrderLine = New ThesisDO.OrderLine
    With oOrderLine
    Set .Item = oItem
    .Quantity = Qty(i)

    '// Update Stock
    oStock.OpenWith .Item, oSuppWar
```

```
    Set .StockItem = oStock
    If oStock.Quantity > .Quantity + 10 Then
        oStock.Quantity = oStock.Quantity - .Quantity
    Else
        oStock.Quantity = oStock.Quantity + 91
    End If
    oStock.YTD = oStock.YTD + .Quantity
    oStock.Order_CNT = oStock.Order_CNT + 1

    oOrder.Add oOrderLine

    '// Results
    PrintResult Format(SUPP_W_ID2, "0000"), 2
    PrintResult Format(.Item.Id, "000000"), 3
    PrintResult Left(.Item.Name, 23), 3, 23
    PrintResult Format(.Quantity, "00"), 2
    PrintResult Format(.StockItem.Quantity, "000"), 4
    PrintResult IIf(.BG, "B", "G"), 4
    PrintResult Format(.Item.Price, "$000.00"), 3
    PrintResult Format(.Item.Price * .Quantity, "$000.00"), 2, ,
True

    Set oOrderLine = Nothing
    Set oItem = Nothing
    End With
Next

With oOrder
'// Save District
If Not .Customer.District.Save Then GoTo NOAbort
    .EntryDate = Format(Date, "mm/dd/yyyy")
    .AllLocal = True
'// Save Customer
    If Not .Customer.Save Then GoTo NOAbort
'// Save order
If Not .Save Then GoTo NOAbort
'// Save New Order
Set oNewOrder.Order = oOrder
    If Not oNewOrder.Save Then GoTo NOAbort
End With

For i = 1 To OrderCnt
    '// Update OrderLine
    If Not oOrder(i).Save Then GoTo NOAbort
Next

PrintResult "Execution Status: Ok"
PrintResult "Total:  " & Format(oOrder.Total, "$0000.00"), 11, ,
True

' i = 5 / 0 ' => To fail transaction

'60: mcnThesis.CommitTrans
bTransaction = False
'oContext.SetComplete
'mcnThesis.Close
NewOrder = True

sResult = msResult
miCounter = oWarehouse.Counter
miClients = oWarehouse.Clients

NOFim:
Set oContext = Nothing

Exit Function

NOAbort:

If bTransaction Then
    bTransaction = False
    'mcnThesis.RollbackTrans
End If
GoTo NOFim

NOError:
If oContext.IsInTransaction Then
    oContext.SetAbort
End If
If bTransaction Then
    bTransaction = False
    'mcnThesis.RollbackTrans
End If
Err.Raise NewOrderErr, "ThesisBOMTS - NewOrder", Erl & ":" &
Err.Number & " - " & Err.Description, vbCritical
Resume NOFim

End Function

Private Sub Class_Initialize()
'Dim sConnection As String
'Set mcnThesis = CreateObject("ADODB.Connection")
'sConnection = "Provider=SQLOLEDB.1;Integrated
Security=SSPI;Persist Security Info=False; "
'sConnection = sConnection & "User ID=sa;Initial
Catalog=Thesis;Data Source=Helen;"
'sConnection = sConnection & "Locale Identifier=1046;Connect
Timeout=15;Use Procedure for Prepare=1;"
'sConnection = sConnection & "Auto Translate=True;Packet
Size=4096;Workstation ID=ALEXANDRENT"
'mcnThesis.ConnectionString = sConnection
'Set mDelivery = CreateObject("ThesisQPMTS.DeliveryRcv")
'mDelivery.Start
End Sub
Private Sub PrintResult(sText, Optional iSpaces As Integer = 0,
Optional SizeToFit As Integer = 0, Optional bLineFeed As Boolean
= False)
Static bOldLine As Boolean

    If Not bOldLine Then
        iSpaces = iSpaces + 1
        bOldLine = True
    End If
    msResult = msResult & Space(iSpaces)
    msResult = msResult & sText
    If SizeToFit > 0 Then
        If Len(sText) < SizeToFit Then
            msResult = msResult & Space(SizeToFit - Len(sText))
        End If
    End If
    If bLineFeed Then
        msResult = msResult & vbCrLf
        bOldLine = False
    End If
End Sub
```

6. N-tier Front End

```
Option Explicit

'// frmTransactions

Dim moTrans As ThesisBOMTS.Transactions
Dim moTrans As ThesisBO3.Transactions
Dim moRand As ThesisBO2.GenRand
Dim moLastName As ThesisBO2.LastNameGen
Dim mQueue As MSMQQueue


Private Sub PrintResult(sText, Optional iSpaces As Integer = 0,
Optional SizeToFit As Integer = 0, Optional bLineFeed As Boolean
= False)
Static bOldLine As Boolean

    If Not bOldLine Then
        iSpaces = iSpaces + 1
        bOldLine = True
    End If
    txtResult = txtResult & Space(iSpaces)
    txtResult = txtResult & sText
    If SizeToFit > 0 Then
        If Len(sText) < SizeToFit Then
            txtResult = txtResult & Space(SizeToFit - Len(sText))
        End If
    End If
    If bLineFeed Then
        txtResult = txtResult & vbCrLf
        bOldLine = False
    End If
End Sub

Private Function Stock_Level(W_ID As Long) As Boolean
On Error GoTo SError
Dim D_ID As Long
Dim iMinThreshold As Integer
Dim bTransaction As Boolean
Dim sResult As String

Randomize
Stock_Level = False

iMinThreshold = Int(Rnd * 11) + 10

'// District
D_ID = Int(Rnd() * 10) + 1

'// Check Stock
If Not moTrans.GetStockLevel(W_ID, D_ID, iMinThreshold, sResult)
Then
    MsgBox "Transaction Failed!", vbCritical
    Exit Function
End If

txtResult = sResult
Stock_Level = True

SEnd:
On Error Resume Next
Exit Function

SError:
MsgBox Err.Description & Err.HelpFile & Err.HelpContext,
vbCritical
Resume SEnd

End Function

Private Function Delivery(W_ID As Long) As Boolean
On Error GoTo DError
Dim bTransaction As Boolean
Dim lCarrier_ID As Long
Dim oMsg As New MSMQMessage

Randomize
Delivery = False

lCarrier_ID = Int(Rnd() * 10) + 1

oMsg.Label = "Delivery Message"
oMsg.Body = Format(W_ID, "00") & ":" & Format(lCarrier_ID, "00")
oMsg.SEnd mQueue

PrintResult "Order-Status", 35, , True
PrintResult "Warehouse: " & Format(W_ID, "0000"), , , True
PrintResult "", , , True
PrintResult "Carrier Number: " & Format(lCarrier_ID, "00"), , ,
True
PrintResult "", , , True
PrintResult "Execution Statuos: Delivery has been queued.", , ,
True
PrintResult "", , , True
Delivery = True

DEnd:
On Error Resume Next
Exit Function

DError:
MsgBox Error, vbCritical
Resume DEnd

End Function

Private Function Order_Status(W_ID As Long) As Boolean

On Error GoTo OSError
Dim D_ID As Long
Dim C_LAST As String
Dim sResult As String

Randomize
Order_Status = False

'// District
D_ID = Int(Rnd() * 10) + 1

'// Customer
C_LAST = moLastName.GenerateLastNameStr(Str(moRand.NURand(255, 0,
999)))

If Not moTrans.GetMaxOrder(W_ID, D_ID, C_LAST, sResult) Then
    MsgBox "Transaction Failed!", vbCritical
    Exit Function
End If

txtResult = sResult
Order_Status = True

OSEnd:
Exit Function

OSError:
MsgBox Error, vbCritical
Resume OSEnd

End Function

Private Function Payment(W_ID As Long) As Boolean
On Error GoTo PError
Dim D_ID As Long
Dim C_LAST As String
Dim sData As String
Dim sngAmount As Single
Dim sResult As String

Randomize
Payment = False

'// District
D_ID = Int(Rnd() * 10) + 1

'// Customer
C_LAST = moLastName.GenerateLastNameStr(Str(moRand.NURand(255, 0,
999)))

'// Payment Transaction
sngAmount = Int(Rnd() * 500000) / 100 + 1
Payment = moTrans.Payment(W_ID, D_ID, C_LAST, sngAmount, sResult)
If Not Payment Then
    MsgBox "Transaction Failed!", vbCritical
    Exit Function
End If

txtResult = sResult

Payment = True
PEnd:
Exit Function

PError:
MsgBox Error, vbCritical
Resume PEnd

End Function


Private Function ClientTransaction(W_ID As Long) As Boolean
On Error GoTo CTError
Dim iOrder_Cnt As Integer
Dim D_ID As Long
Dim C_ID As Long
Dim O_ID As Long
Dim Item() As Long
Dim Qty() As Long
Dim Items As Variant
Dim Qtys As Variant
Dim i As Integer
Dim sResult As String

Randomize
ClientTransaction = False

'// Number of lines  [5..15]
iOrder_Cnt = 8 'Int(Rnd() * 11) + 5

'// District
D_ID = Int(Rnd() * 10) + 1

'// Customer
C_ID = moRand.NURand(1023, 1, 3000)

'// Order Lines
ReDim Item(1 To iOrder_Cnt) As Long
ReDim Qty(1 To iOrder_Cnt) As Long
For i = 1 To iOrder_Cnt
    '// Item
    Item(i) = moRand.NURand(1023, 1, 3000)
    Qty(i) = Int(Rnd() * 10) + 1
Next

'// Execute Transaction
```

```
Items = Item
Qtys = Qty
'Set moContext = GetObjectContext()
'Set moTrans = CreateObject("ThesisBOMTS.Transactions")
ClientTransaction = moTrans.NewOrder(W_ID, D_ID, C_ID,
iOrder_Cnt, Items, Qtys, sResult)
'MsgBox moTrans.Counter
'MsgBox moTrans.Clients
'Set moTrans = Nothing

If ClientTransaction Then

    '// Print Results
    txtResult = sResult
Else
    MsgBox "Transaction Failed!", vbCritical
End If

CTEnd:
Exit Function

CTError:
MsgBox Error, vbCritical
Resume CTEnd
End Function

Private Sub cmdTransaction_Click(index As Integer)
Dim bResult As Boolean
Dim W_ID As Long

W_ID = 2

txtResult = ""
'MsgBox TypeName(moTrans)
'Exit Sub
Screen.MousePointer = vbHourglass
Select Case index
    Case 0
        bResult = ClientTransaction(W_ID)
    Case 1
        bResult = Payment(W_ID)
    Case 2
        bResult = Order_Status(W_ID)
    Case 3
        bResult = Delivery(W_ID)
    Case 4
        bResult = Stock_Level(W_ID)
End Select
Screen.MousePointer = vbNormal

End Sub

Public Function ExecuteTransaction(index As Integer)
    cmdTransaction_Click index
End Function

Private Sub Form_Load()
    Dim MQInfo As New MSMQQueueInfo
    MQInfo.PathName = "abcnt06b\thesis"
    Set mQueue = MQInfo.Open(MQ_SEND_ACCESS, MQ_DENY_NONE)
    Set moRand = New ThesisBO2.GenRand
    Set moLastName = New ThesisBO2.LastNameGen
    ' Set moTrans = CreateObject("ThesisBOMTS.Transactions")
    Set moTrans = CreateObject("ThesisBO3.Transactions")
'    If Not moTrans.IsOk Then MsgBox "Error", vbCritical
End Sub
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	15 Jun 1999	Mater's Thesis

4. TITLE AND SUBTITLE
ANALYSIS OF N-TIER ARCHITECTURE APPLIED TO DISTRIBUTED-DATABASE SYSTEMS

5. FUNDING NUMBERS

6. AUTHOR(S)
Alexandre Gomes Valente, 1st Lt., Brazilian Air Force

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
2950 P Street, Bldg 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCE/ENG/99J-04

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Brazilian Ministry of Aeronautics
Esplanada dos Ministerios
Brasilia - Distrito Federal
Brasil

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Dr. Gary B. Lamont
COMM: (937) 255-3636 x4718 DSN: 785-3636 x4718

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

N-tier architecture has been more commonly used as a methodology for developing large database applications. This work evaluates the use of this ar-chitecture instead of the classical Client/Server architecture in developing corpo-rate applications based on distributed databases. The comparison between ar-chitectures is performed using applications that execute transactions similar to those defined in the Transaction Process Council Type C benchmark (TPC-C). The environment used for development and testing was the AFIT Bimodal Cluster (ABC) - an heterogeneous cluster of PCs, running Microsoft Windows NT 4.0 OS. The comparative experimental analysis demonstrated that the N-tier architecture allows more efficient bandwidth utilization between client and server machines, with similar performance. Results led to conclusion that the N-tier architecture is better suited than the Client/Server for use in corporate sys-tems interconnected by low-bandwidth Wide-Area-Networks (WANs), such as the Internet.

14. SUBJECT TERMS	15. NUMBER OF PAGES
N-TIER ARCHITECTURE, DISTRIBUTED DATABASES, DISTRIBUTED OBJECTS, DCOM, CLUSTER OF PCS, WINDOWS NT, MTS, MSMQ	170
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL