A.D.S.

System Design

on behalf of

Eurosun Inc.

CSC340 - UTM
Professor: Arnold Rosenbloom
TA: C. Jull
April 16, 2001

Nelson Arruda
Eric De Paolis
Michael Scrivo

Table of Contents

1. Introduction

In designing a good Information System, it is necessary to review the requirements outlined during the requirements analysis phase. It is a well known fact that many errors in designing an Information System occur in determining the requirements. It is also a well known fact that fixing errors in the requirements and design stage are much more cost effective than fixing them during the implementation phase. Thus, the main goal of this study is to consider architectural alternatives along with hardware and software as proposed by the requirements analysis. In the process of doing this, the requirements will be refined and built upon as needed, and a viable, carefully planned out system design will emerge.

2. The Current System

2.1. Description:

The current system in place is based on a two-tier client server model. There are twelve systems in total, which include two main servers: one application server and one Internet/Domain server. As mentioned above, the heart of the system, which is integral to business operations, is the proprietary Holiday Booking Program. This program was created using GW Basic in 1987 and took two years to develop. Note that for the purposes of this study, we will ignore the Internet portion of the global architecture since it is not a factor in designing this particular system. However, we will consider security issues pertaining to the fact that the system may be accessible from the Internet.

The application server is a Windows based system that requires legacy support to run GW Basic programs. The Holiday Booking Program along with its proprietary database are both stored and run from this server. Each employee, using a Windows based client system, has access to this program and uses it to make bookings and store client information. The program stores information such as flight departures, flight arrivals, hotel accommodations and car rentals. Monetary information is also stored so that invoices and vouchers can be generated by Management.

2.2. Problems:

The problems with the current system stem from using a program based on an old DOS programming language: GW Basic. First and foremost, the Holiday Booking program does not have a coherent software architecture. As a result, it is not clear how the underlying system operates. All that is known about its architecture is that it operates only from a single machine where the program itself and all its related data are stored. Because each client directly executes the program from the server, 100% of the processing occurs on the server. Furthermore, the program suffers from instabilities with newer Operating Systems which cause the server to crash very often. Some other issues are a lack of features required by Eurosun and a poor security implementation. See the Feasibility Study for a more detailed analysis of the problems with the system.

2.3. Problem Scope and Solution:

To solve the problems mentioned above, it was determined that the best alternative would be to create a whole new software package to replace the current GW Basic Holiday Booking program. There was no justification in trying to patch and add features to the Holiday Booking program (as noted in the Feasibility Study). Amongst the various alternatives presented, it was decided that custom software based on industry standards would be developed to replace the current system.

3. Software Architecture

The software architecture defines the components (or subsystems) of the software system and how they use each other's functionality and data.

3.1. Alternatives:

As determined in the requirements analysis (see Appendix N: Requirements Analysis: Section 1.2) a DBMS (Database Management System) is best suited to handle the data management needs of Eurosun Inc. A DMBS is the best choice since Eurosun requires a centralized database which multiple users can use simultaneously. With that said, the following architectures will be considered as possible solutions:

3.1.1. Repository Based Architecture:

A repository architecture consists of a central data structure (often a database) and a collection of independent components which operate on the central data structure. The repository architecture is geared towards environments which have many other different systems using the centralized data for different jobs (see Appendix A1: Repository Based Architecture).

3.1.2. Client Server Architecture:

A client server architecture consists of components which are *service consumers* (clients) and *service providers* (servers). All communication between clients and servers is accomplished through messages. There are three protocols for exchanging messages:

a) Remote Procedure Call (RPC)
The client invokes a remotely located procedure. This procedure is then executed and the results are sent back to the client.

b) Remote Data Access (RDA)
The client invokes a remotely located procedure. Here the invoked procedure is a database query and the response is often a large set of data (usually in the form of a table).

c) Queued Message Processing
Here requests are queued on the server and processed whenever possible.

Once the type of message passing is chosen, a specific client server model must be chosen. The two types which will be considered for this application are the two-tier and three-tier models:

3.1.2.1. Two-Tier Client Server Architecture:

In a two-tier system, the service consumers interact directly with the service provider; there is no middle processing layer (see Appendix A2: Two-Tier Client Server Architecture). In this particular application, the client would contain the necessary code to query the database. The DBMS tier would contain all the data and business logic to carry out the queries and perform integrity checks.

3.1.2.2. Three-Tier Client Server Architecture:

In a three-tier client server architecture, there is a processing layer between the service consumer and the service provider (see Appendix A3: Three-Tier Client Server Architecture). The user can only communicate with the processing tier and the server can only communicate with the processing tier. Hence, the processing tier contains most of the business components and logic and is the most critical part of the system. The advantage of using the three-tier model is that the processing tier is separate from the data tier which means that modifications to the processing tier are more easily accomplished with as little changes as possible to the data tier. Furthermore, upgrades to the data tier are simpler as well, since there is no integrated business code to worry about. The processing and data tiers may reside on the same server to save space and money, or they may be on separate servers.

3.2.1. Comparison between Two-Tier and Three-Tier Architectures:

3.2.1.1. Criteria:

a) Development Time:
How long will it take to develop in relation to the other alternatives?
(Short, Same, Long)

b) Component Upgradeability:
How easy will it be to upgrade a component in the system?
(Easy, Difficult)

c) Maintenance Costs:
How much will it cost to fix bugs and maintain the system?
(Low, Medium, High)

3.2.1.2. Results:

a) Development Time:
Developing the application using a three-tier model will take slightly longer than using a two-tier model. The three-tier model requires that more care be placed on specifying communication protocols between the three tiers and precise module specifications. The two-tier model only requires using the communication model provided by the DBMS and places less emphasis on modularization.

b) Component Upgradeability:
As stated in the previous section, three-tier applications put more emphasis on modularization. Thus components can easily be upgraded so long as they still conform to specification. Furthermore, if in the future the data tier is required to be upgraded or changed, a three-tier model facilitates a swap with greater ease.

c) Maintenance Costs:
Because a three-tier model provides a higher-level of modularization and abstraction, maintenance is easier to perform since there is a clear boundary of where each stage of processing occurs and it is easier to pinpoint the location where maintenance is to be performed. Furthermore, the business rules and processing can easily be tweaked without affecting the data tier since they are not tightly integrated with the DBMS. Simple independent modules are easier to maintain since other parts of the system rarely have to be modified.

Model	Development Time	Component Upgradeability	Maintenance Costs
2-tier	Short	Difficult	Medium
3-tier	Long	Easy	Low

Table 1: Summary - Client Server Model Comparison

3.2. Recommendation:

A repository based solution does not make much sense for this system. Repository based systems are usually used in situations where the input from one of the departments is used by another department. In this application, all data must be available to all users simultaneously, thus the client server architecture is the most viable solution. Next, it is necessary to choose a client server model. From the comparison given above, it was determined that a three-tier model is the best solution for the system. Thus the system will use a three-tier architecture based on the RDA protocol.

4. Global Architecture Design

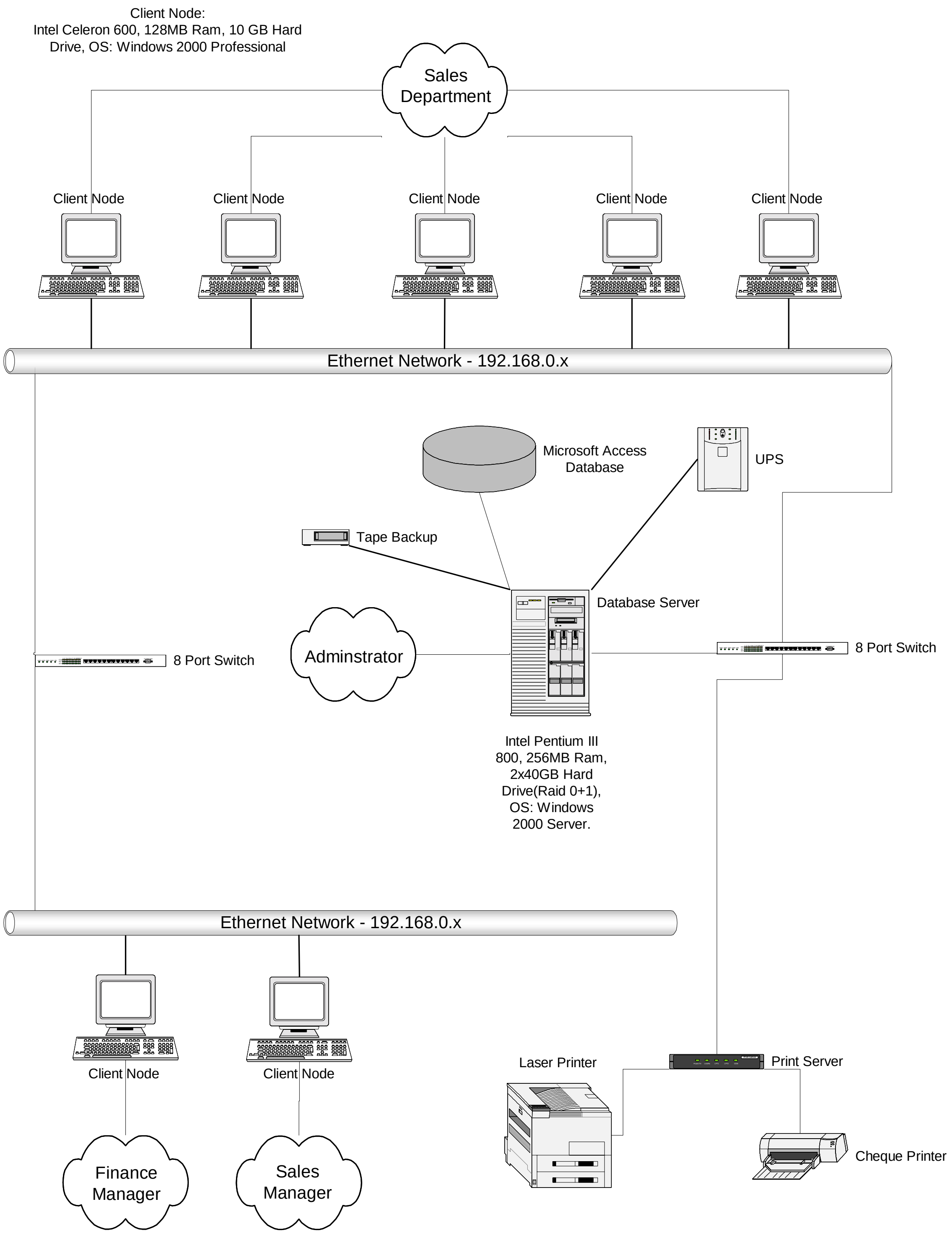

Client Node:
Intel Celeron 600, 128MB Ram, 10 GB Hard Drive, OS: Windows 2000 Professional
Sales Department
Client Node
Client Node
Client Node
Client Node
Client Node
Ethernet Network - 192.168.0.x
Microsoft Access Database
UPS
Tape Backup
Database Server
Adminstrator
8 Port Switch
8 Port Switch
Intel Pentium III 800, 256MB Ram, 2x40GB Hard Drive(Raid 0+1), OS: Windows 2000 Server.
Ethernet Network - 192.168.0.x
Client Node
Client Node
Finance Manager
Sales Manager
Laser Printer
Print Server
Cheque Printer

Figure 1: Global Architecture (with existing and new equipment)

4.1. Global Architecture Overview:

The above diagram provides a physical representation of the global architecture at Eurosun Inc. The actual distribution of machines and network structure has only been modified slightly from their original layout. In particular, the 10Mbps Ethernet network will be replaced with a high performance 100Mbps Ethernet network with switches instead of hubs. Also, the existing client machines will be upgraded to meet the specifications outlined in the Requirements Analysis (see Appendix N: Requirements Analysis: Section 2.7). The laser and cheque printers will be connected to a print server which will allow the printers to be independent of any one machine. This reduces overhead and allows any machine to print to these printers without having to go through any other systems. The server will be connected to a UPS system and a Tape Backup system as per requirements 2.3.2 and 2.6.3.

It should be noted that although each user has a specific client machine assigned to him, any user can login at any terminal. Security privileges will be assigned to the user's ID rather than a specific access location.

4.2. Distribution Issues:

This deals with how the system on a high level, handles inputs and outputs. The various possibilities are as follows:

Batch Mode: Process a batch of inputs/outputs together.

On-Line Mode: Process inputs/outputs as they become available.

Remote Batch: Data are input on-line on several machines, then fed in batch mode to a centralized database.

From the functional requirements analysis, (see Appendix N: Requirements Analysis: Section 1.2), it was determined that all data exchanged with the database must be available to all users instantaneously, thus I/O will be handled using an On-Line protocol.

4.3. Process Cycles:

Because the system uses an online mode for handling input and output, there are no periodical batch jobs for the system to process. Therefore all processes need to be run as soon as possible with respect to the order that processes are requested.

4.4. Data Distribution:

The number of users expected to use the system at present and in the near future is not expected to be over twenty-five. Thus, all data will be stored in one central location on the Database Server. A centralized database is better suited than a distributed database due to the much lower cost, easier implementation and lower maintenance requirements. The benefit of using a distributed database is the ability to allocate heavy loads amongst many systems. However, since the requests to the database are expected to be relatively low, the added performance does not outweigh the larger cost. Furthermore, a centralized database meets the performance requirements specified in the Requirements Analysis (see Appendix Requirements Analysis: Section 1.2)

4.5. Process Distribution:

There is no need for a distributed set of services since it is not likely that the maximum number of users will ever warrant the need for such a system. Distributed processes are reserved for cases in which the database and/or program are expected to be used by a large number of users, simultaneously or in cases where heavy processing is required (such as a 3D rendering farm).

5. Selections

5.1. Hardware Classes:

It is necessary to first select a class of hardware systems which will form the basis of the hardware platform. The various possibilities are:

- Mainframes
- Commercial Minicomputers
- Microcomputers
- Embedded Systems

Mainframes and Commercial Minicomputers are for much larger operations with either a very large number of users or large amounts of processing. Today, microcomputers, specifically PC's, have become powerful enough and inexpensive enough to handle all the needs of small and medium sized businesses with low maintenance and easy upgrade paths. Embedded systems, such as integrated mobile communication devices, are not yet required by Eurosun Inc. and will not be considered at this time.

Using standard PC components gives us a very open standard to work with. PC components are manufactured by a very wide array of companies. The server that will be selected will be from a large manufacturer such as Dell or IBM, so the level of openness is slightly reduced since support will come directly from the manufactures. The client systems will be completely open and when a component fails, finding and replacing it with the same or better component will pose no problems at all.

5.2. Hardware Selection:

5.2.1. Database Server Selection Criteria:

The server is the backbone of the entire system. It must be very reliable and powerful enough to handle the current load and future expansion. The server should have a modularized design, preferably with hot-swappable components so that if a component such as a hard drive or power supply fails, it can be replaced with no downtime. An off-the-shelf server such as those from Dell, Compaq and IBM is preferred over building a server simply because of the better and faster support available from these companies. These high-grade servers are usually very reliable as well.

Things to consider when choosing a server are (in order of decreasing importance): reliability, support, warranty, price, performance, features, and availability.

5.2.2. Database Server Recommendation:

All the systems were configured with relatively the same components and therefore have very similar prices (see Appendix B1: Server Comparison). As a consequence, the most important categories for making the final choice are reliability and support. Since we have no prior experience with any of these servers, we researched opinions of credible people on the Internet and contacted people who have had experience with these servers. The overwhelming response was that the Dell PowerEdge Servers were the favourites. Dell PowerEdge servers are heralded as very reliable and in the rare cases when they failed, support was top notch.

5.2.3. Client/Workstation Node Recommendation:

The purpose of each client machine will be to allow the users of the system to have access to the data stored on the server. Small amounts of processing will take place on the client nodes, so relatively little power is needed. Each client will be outfitted with an Intel Celeron 600, 128 MB of RAM, and a 10GB hard drive as determined by the non-functional requirements in Appendix N. The systems will be built by a local computer shop or brand-name systems will be purchased. This decision will be left to Eurosun Inc. as either option fits the requirements and is only a matter of preference. See Appendix B2: Client System Configuration, for a breakdown of the required components along with estimated pricing.

5.3. Software:

5.3.1. Database Server Operating System:

The Operating System (OS) is one of the most important parts of the system as a whole. The OS bridges the communication between the hardware and the applications running on top of the OS. It also handles the bulk of the network communications in conjunction with the network hardware (switches and routers). It is very important that the OS provides excellent reliability and robust networking. Eurosun Inc. has recently purchased a Windows 2000 Server + 10 CAL license which they wish to make use of. Fortunately, Windows 2000 Server is a very reliable OS with excellent networking capabilities and should fit well into the system.

5.3.2. Client/Workstation Systems:

As stated above, Eurosun Inc. has a 10 CAL license for Windows 2000 Server, so each client system will be installed with Windows 2000 Professional. Since each system has 128MB of RAM, Windows 2000 Professional will perform very well on the client systems with a high degree of stability. Windows 9x should not be used on any of the client systems. The 9x series provides a very poor degree of stability and does not support true domain connectivity.

5.3.3. Database:

5.3.3.1. Overview:

The database market has become a very large one in the past ten years. There are several widely used and supported DBMS' on the market from companies like IBM, Interbase, Oracle, Sybase and Microsoft. In the past few years, we have even seen the advent of free databases such as MySQL and PostgreSQL. These two packages are gaining in popularity, especially for web driven applications. However, these free DBMS' are not mature enough to consider for this system.

5.3.3.1.1. IBM DB2:

IBM's DB2 is a highly scalable and reliable system backed by many years of development and refinement by IBM's superb engineers. It is available on wide variety of platforms including: Windows, Linux, Solaris, HP-UX, NUMA-Q, AIX, OS/2 and even handheld systems such as Windows CE and the Palm platform. Having a database that works on multiple platforms is good for future expandability. If in the future, Eurosun would like to run its server on a different platform, this is a possibility. Besides being available on multiple platforms, DB2 also has a wide range of utilities and applications built for it.

DB2 is an Acid Compliant Database system (see Appendix D1: Acid Compliant Database). ACID compliant databases are essential in situations where transactions are crucial to business operations such as E-commerce sites. DB2 was ranked first in Service and Responsiveness, Scalability and Industry Expertise. DB2 is not the most cost-effective solution; it fits somewhere between MS SQL Server on the inexpensive end and Oracle on the expensive end.

5.3.3.1.2. Oracle:

Oracle is considered to superior to most databases. It is available on many platforms and has a suite of applications designed to work with it to customize the database to the company's exact requirements. Oracle was ranked first in Features and Innovation and ranked high in all other categories. Performance and Scalability are considered to be amongst the best in the industry. Unfortunately, Oracle is notorious for being difficult to setup and administer and requires a highly trained staff to tune it and keep it running well. Like DB2, Oracle is an ACID compliant database. Lastly, Oracle is the most expensive of all database solutions.

5.3.3.1.3. Microsoft SQL Server:

Microsoft is somewhat of a newcomer to the database market, because of this, their market share is significantly lower than IBM's or Oracle's. The most recent versions of SQL Server have caught up with DB2 and Oracle in terms of performance and features but still lag behind in a few areas. SQL server was ranked first in Programming Expertise and Pricing and Value. MS SQL Server builds on Microsoft's expertise in developing programming applications which are easy to use. MS SQL Server integrates tightly with Microsoft's popular Visual Studio development package. It is also an ACID compliant database, but unlike Oracle and DB2, it is currently only available on the Windows platform. Lastly, MS SQL Server can easily replace a smaller MS Access database because of the way Microsoft has designed the ADO (Access Data Objects) API (Application Programming Interface) to work seamlessly between Access and SQL Server by providing a level of abstraction.

5.3.3.1.4. Microsoft Access:

Microsoft Access is the cheapest of the database solutions (besides the free ones). Eurosun Inc. already has Microsoft Office 2000 licenses for each user, so using Access as a database comes free. Access is not an ACID compliant database, nor does it have nearly as many features as any of the other database solutions. However, Access is practical for a small number of users and for companies on a low budget. Furthermore, an Access application can be created using Microsoft's ADO API which provides a good level abstraction. Using ADO, Eurosun can upgrade the server to MS SQL Server by simply importing the Access database. ADO provides enough abstraction so that only a very minimal amount of code needs to be changed in the software to perform this upgrade.

5.3.3.2. Recommendation:

The most important factor in choosing a database for this application is the price. Database products can be extremely expensive, usually much more expensive than hardware. With that said, Oracle or DB2 would be overkill for this application and would not make sense budget-wise. Microsoft SQL Server is the best solution for Eurosun Inc., however, it is still quite expensive and not needed with less than ten employees using the system concurrently. We recommend that Eurosun use a Microsoft Access database where transactions are performed using the ADO API. This will allow Eurosun Inc. to upgrade easily to MS SQL Server in the future if the situation warrants it.

5.3.4. Programming Language:

5.3.4.1. Overview:

Since the chosen Operating System for both the Server and the Clients is Microsoft Windows 2000, and since access to the ADO API is required by the chosen Access database, there is little choice in the programming environment that can be used. The choices are as follows:

5.3.4.1.1. Microsoft Visual Basic:

Microsoft Visual Basic is the de facto standard is rapid application development. It is an event-driven system which is very easy to program since it provides a high level of abstraction from the Win32 API (the Application Programming Interface to the Windows Family of Operating Systems). Visual Basic also provides decent levels of performance and has relatively good customization capabilities. Visual Basic 6 and later versions support ADO.

5.3.4.1.2. Microsoft Visual J++:

Like Visual Basic, Microsoft Visual J++ provides a good rapid application development environment, although with a bit more control than Visual Basic. One disadvantage of J++ is that Microsoft has discontinued it in favour of their C# language. The other disadvantage of J++ 6 is its poor performance compared to C++. Visual J++ 6 supports ADO.

5.3.4.1.3. Microsoft Visual C++:

Microsoft Visual C++ is the most widely used C++ programming environment. Visual C++ provides the highest levels of performance and customization. However, Visual C++ requires deep knowledge of the Win32 API and development usually takes much longer. Visual C++ 6 and later versions support ADO.

5.3.4.1.4. Microsoft Visual Basic & Visual C++:

This option involves using Visual Basic to create the interface and any rudimentary functions which do not require high performance. The data or middle tier will then be created using Visual C++. This tier involves making the ADO/SQL calls and performing heavy I/O and/or calculations.

5.3.4.2. Recommendation:

We recommend that the application be created using a mix of Microsoft Visual Basic and Visual C++. This option provides the best of both worlds: rapid application development and high performance. Furthermore, it allows for much better modularization and clearly defined tiers which two programmers (or groups) can work on independently.

5.4. Network

5.4.1. Type:

10/100Mbps Ethernet (IEEE 802.3u) is the standard LAN implementation for almost all businesses today. Ethernet hardware and wiring is very inexpensive and provides very good performance over moderate distances. Also, a large number of Ethernet devices are supported on many platforms. Eurosun Inc. already has a 10Mbps Ethernet network which will be replaced by an upgraded 10/100Mbps network. The following section will outline which devices and wiring will be used:

5.4.2. Network Interface Cards:

There are a wide variety of network cards available. Feature-wise, they are all relatively the same. Prices for 10/100 PCI Ethernet cards range from $20 to $100. The more expensive variants usually provide slightly higher transfer rates and higher reliability. The clients will not be overstressing the network cards, so a lower-end card will suffice. Each client will be equipped with a D-Link DFE-530+TX network card. This card costs approximately $35. These cards provide good reliability and a lifetime warranty.

5.4.3. Connectivity:

5.4.3.1. Overview:

The clients and server must all be connected together in some way. There are three classes of hardware which accomplish this task: a workgroup hub, a workgroup switch, or a router.

5.4.3.1.1. Workgroup Hub:

A workgroup hub provides the most basic level of performance and routing. Hubs operate based on the source and destination addresses of the computers involved in the communication. Furthermore, only one message can be sent through a hub at a time, thus network throughput is relatively low. Occasional lag, when performing queries to the server, will be noticed when multiple people are using the system.

5.4.3.1.2. Workgroup Switch:

A workgroup switch provides much higher levels of performance than a hub. A switch transfers data based on the MAC addresses of each network card. The Ethernet standard requires each network card manufactured to have a unique MAC address. Switches are able to transfer data between more than two nodes at a time and have a peak network throughput of 200Mbps.

5.4.3.1.3. Router:

Routers provide the highest level of network performance and manageability. Routers operate in the same manner as switches but essentially are a computer in themselves (sometimes they are actually entire computers systems running a UNIX based OS). Regular hardware routers have a small operating system built in that allows for precise filtering rules, which directs certain traffic to certain nodes and stops unwanted traffic.

5.4.3.2. Recommendation:

The system is heavily network based, so high network performance and reliability is essential. Having relatively high-powered systems and a slow network provide an unbalanced and wasteful environment. Because of this requirement, workgroup switches will be used. A router is not necessary since there is no special routing that must be done. Furthermore, routers require extra maintenance which is not desirable. We specifically recommend two D-Link DSS-8+ 8-port switches. These switches will allow for future expansion and come at a very affordable price ($150 each).

5.4.4. Network Wiring:

The network will be wired using Category 5e (CAT5e) Ethernet copper wire (EIA/TIA 568 100-ohm STP). The interface connectors will be RJ-45 in the straight-through configuration. All wires will be less than one hundred feet in length, although the Ethernet specification states that CAT5 wiring can be used in lengths of up to 100 Metres.

6. Input/Output Procedures:

In all I/O procedures, the client application is responsible for making sure basic inputs are in valid formats and are acceptable inputs (i.e. a variable must be greater than 0). The processing tier is then responsible for making additional integrity checks to ensure the data has not been corrupted and the transaction is allowed to take place. The first level of constraint validation (on the client) takes a good portion of the load off the processing tier since the data has already been verified to be in the proper format.

The system will be designed using a modularized methodology and will be proportioned to fit nicely into the three-tier client server model. Each module will represent a basic department within Eurosun Inc. These modules will facilitate upgrading and allow for easy maintenance of the system as stated in the system architecture section.

The following system modules were derived from the data input and output requirements defined in the requirements analysis. (See Appendix N: Functional Requirements. We will refrain from further referencing of the appendix in the following sections to avoid clutter.)

6.1. System Modules:

6.1.1. Administration Module:

The Administration Module captures the I/O interaction between the Sales Manager and the system. This module will be built so that only the Sales Manager class will have privileges to access this sensitive information. See the security section below for further security details.

6.1.1.1. Add Employee:

An existing Sales Manager enters the new employee's name, address, phone number, and any other personal information that may be required. Furthermore, the Sales Manager is also required to indicate the new employee's title. The system then gathers this new information, encrypts it, and places it into the Employee table in the database.

6.1.1.2. Add Supplier:

An existing Sales Manager inserts a new supplier into the database. The supplier can be a flight, a hotel, or a car rental agency. Each supplier is assigned a supplier ID which is automatically generated by the system. Furthermore, a supplier name, contact name, and the suppliers address are required to complete the transaction. All this information is encrypted and stored in the Supplier table.

6.1.1.3. Add Flight:

An existing Sales Manager inserts the new flight information into the database. This information contains the flight ID, departure time, departure date, as well as the number of seats available on the plane. The system then makes sure the airline associated with the flight exists in the Supplier table. If the airline does not exist, the system will not allow the transaction to succeed, and will notify the Sales Manager. If the airline does exist, then the system creates a record for the new flight in the FlightOfferedBy table, and the flight is inserted in the Flight table. The same procedure is followed when adding a new hotel season and car rental agency.

6.1.2. Financial/Reports Module:

The Financial/Reports Module captures the I/O interaction between the Financial Manager and the system. This module deals with financial matters at Eurosun Inc.

6.1.2.1. Pay Supplier:

An existing Financial Manager accesses the Supplier table in the database to acquire the supplier's outstanding amount. Then the Financial Manager enters the amount to be paid to the supplier and the system generates a corresponding cheque. Furthermore, the cheque number and the amount paid are stored in the Pays table for future reference. Finally, the current account is then updated in the suppliers field.

6.1.2.2. View Financial Reports:

This I/O procedure is simply a request of data from the system. If the Financial Manager wishes to view any financial reports, the system accesses the Pays table. Within this table, the system has the ability to access and display the amount paid to a certain supplier. Note that the Financial Manager also has the ability to print the report if desired.

6.1.3. Booking Module:

The Booking Module captures the I/O interactions between the Sale Representatives and the system. This module is by far the most extensive and highly used. It is within this module that customer input is handled by the database

6.1.3.1. Create New Booking:

The Sales Representative creates a new booking which is stored in the Booking table. The booking is assigned a booking ID. The customers who are involved with the booking are added to CustomerWithBooking table.

6.1.3.2. Cancel Booking:

The Sales Representative simply indicates the booking that is marked for cancellation by accessing the booking ID via the Manages table. The system then takes the required steps to adjust all other quantities in tables associated the particular booking (see Appendix K: Collaboration Diagram – Creating a New Booking for a collaboration diagram that corresponds to this operation). Once the adjustment is done, the booking is removed.

6.1.3.3. Add Flight to Booking:

The Sales Representative queries the system for a particular booking ID. If this ID does not exist the transaction will fail. If the booking ID exists, the system accesses the corresponding booking through the Manages table. The system then acquires the flight ID from the Flight table, and inserts the flight into the booking by adding the flight ID and booking ID to the FlightWithBooking table.

6.1.3.4. Add Hotel Reservation to Booking:

The Sales Representative queries the system for a particular booking ID. If this ID does not exist the transaction will fail. If the booking ID exists the system accesses the corresponding booking through the Manages table. The system then acquires the information about the requested hotel from the Hotel Season table. Then a new hotel reservation is created in the Hotel Reservation table, and room ID and the booking ID are stored in the ReservationWithBooking table.

6.1.3.5. Add Car Rental to Booking:

This procedure is similar to the one above.

6.1.3.6. Add Customer Account:

The Sales Representative queries the system to see if the customer currently exists in the system. It the customer does exist the transaction is cancelled, otherwise the customer's personal information is entered into the Customer table and the system automatically assigns the customer a unique ID.

6.1.3.7. Look Up Customer Account:

The Sales Representative queries the system to obtain the customer information. The system searches the Customer table to find a customer that matches the query. If a customer is found, the system returns the information of the customer to the Sales Representative. If a customer is not found, the transaction is cancelled and the Sales Representative is alerted of the missing customer.

6.2. Security

All passwords will be encrypted and strict security rules will be implemented since the network is connected to the Internet. The database will be placed in a non-shared folder which can only be accessed by the DBMS using a user ID and password. Employee, supplier, and customer information will be encrypted along with passwords. In the unlikely event that a 'hacker' manages to get access to the database, this will ensure that sensitive data will be very difficult to extract.

7. Database Design:

7.1. E-R Diagram:

The preliminary E-R diagram can be viewed in Appendix F1: Initial E-R Diagram.
The optimized E-R diagram can be viewed in Appendix F2: Optimized E-R Diagram.

The following is a detailed design and construction of the database. The task is to capture the information found in the revised Class diagram (see Appendix H: Revised Class Diagram), and translate it into a corresponding E-R diagram. The initial E-R diagram is then analyzed for partitioning, generalizations and redundancies, resulting in an optimized E-R diagram. This final E-R diagram corresponds to the database schema below. This schema represents the actual tables, along with the attributes and their primary identifiers which make up the physical database.

7.1.1. E-R Diagram Optimization Overview:

The logical design steps required to transform the E-R Diagram to a corresponding database schema are as follows:

a) Analysis of Redundancies
In order to determine whether it is beneficial to maintain or remove redundancies in our relational database, we compared the cost of each operation associated with the redundant information. In Appendix E, you will find the performance analysis. In this analysis we determined that the absence of redundant information in our E-R Diagram improved overall performance.

b) Removing Generalizations
The relational database model does not support generalizations found within E-R diagrams. The most suitable way to remove generalizations from the initial E-R diagram was to delete the child entities and add a Type attribute to the parent. This could easily be done since there was no real distinction between child entities and their parent.

7.1.2. Database Schema:

The following schema corresponds to the optimized E-R diagram and represents the entities and relationships which will form the tables in the Eurosun Inc. database (see Appendix F3: E-R Data Dictionary for further details). The schema complies with 1NF standards (see Appendix D2: Relational Schema using Normal Forms) , where no relation contains any multiple valued attributes. Compliance with any higher forms of normalization would cause a decrease in performance .

7.1.2.1. Entities:

Employee (EmployeeID, Name, Phone, Address, Type)

Customer (CustID, Name, Phone, Type)

- CustID: The unique ID that each customer is assigned.
- Name: Contains the name of the customer.
- Phone: Contains the contact phone number of the customer.
- Type: One of: Vacationer or Travel Agent.

Booking (BookingID, TotalCost)

- BookingID: The unique ID that each individual booking is assigned.
- TotalCost: Contains the current cost of the booking to be billed to the customer.

Flight (FlightID, NumSeatsAvail, DepartCity, DepartDate, DepartTime, ArrivalCity)

- FlightID: The unique ID that each flight is assigned.
- NumSeatsAvail: Contains the number of seats currently available for sale on the flight.
- DepartCity: Contains the name of the city from which the flight departs.
- DepartDate: Contains the date on which the flight will depart.
- DepartTime: Contains the time of day on which the flight will depart.
- ArrivalCity: Contains the name of the city from which the flight will arrive.

Supplier (SupplierID, Name, AmountOwed, Phone, Address, City, Country, ContactName, Type)

- SupplierID: The unique ID that each supplier is assigned.
- Name: Contains the name of the supplier.
- Phone: Contains the business phone number of the supplier.
- Address: Contains the address of the supplier.
- City: Contains the city in which the supplier is located.
- Country: Contains the country in which the supplier is located.
- AmountOwed: Contains the current amount owed to the supplier.
- Type: One of: Airline, a Hotel, or a Car Rental Agency.

HotelReservation (RoomID, RoomType, StartDate, EndDate)

- RoomID: The unique ID that each Hotel Reservation is assigned.
- RoomType: Contains a description of the type of room.

- StartDate: Contains the date on which the vacationer is to arrive at the hotel.
- EndDate: Contains the date on which the vacationer is to depart from the hotel.

RentalCar (CarID, CarType, StartDate, EndDate)

- CarID: The unique ID that each Rental Car is assigned.
- CarType: Contains a description of the type of Car.
- StartDate: Contains the date on which the vacationer is to obtain the car.
- EndDate: Contains the date on which the vacationer is to return the car.

7.1.2.2. Relationships:

Reports (DateRange, DateStamp, Type, Supplier, Employee).

- DateRange: Contains the date range (start date and end date) of this report.
- DateStamp: Contains the date this report was prepared.
- Type: The template used by this report.
- Supplier: Contains the ID of the supplier that this report is about.
- Employee: Contains the ID of the employee who prepared this report.

Pays (DateStamp, ChequeNumber, AmountPaid, Supplier, Employee).

- DateStamp: Contains the date this cheque was prepared.
- ChequeNumer: Contains the cheque number of the cheque prepared.
- AmountPaid: Contains the amount paid by cheque.
- Supplier: Contains the ID of the supplier that is being paid.
- Employee: Contains the ID of the employee who prepared the cheque.

Enters (DateStamp, Supplier, Employee).

- DateStamp: Contains the date the supplier was entered in the database.
- Supplier: Contains the ID of the supplier entered.
- Employee: Contains the ID of the employee who entered the supplier.

FlightsOfferedBy (Supplier, Flight).

- Supplier: Contains the ID of the supplier that offers this flight.
- Flight: Contains the ID of the flight offered.

HotelSeason (Rating, RoomRate, SeasonStartDate, SeasonEndDate, Supplier, HotelReservation).

- Rating: Contains the star rating of the hotel.
- RoomRate: Contains the price rate for the rooms offered.
- SeasonStartDate: Contains the date on which the season begins.
- SeasonEndDate: Contains the date on which the season ends.
- Supplier: Contains the ID of the hotel, which is a supplier.
- HotelReservation: Contains the ID of the reservation for this hotel.

CarRentalSeason (CarRate, SeasonStartDate, SeasonEndDate, Supplier, RentalCar).

• CarRate:	Contains the price rate for the rental car offered.
• SeasonStartDate:	Contains the date on which the season begins.
• SeasonEndDate:	Contains the date on which the season ends.
• Supplier:	Contains the ID of the car rental agency, which is a supplier.
• RentalCar:	Contains the ID of the rental car.

Manages (Employee, Booking).

• Employee:	Contains the ID of the employee managing the booking.
• Booking:	Contains the ID of the booking being managed.

FlightWithBooking (Flight, Booking).

• Flight:	Contains the ID of the flight that is included in the booking.
• Booking:	Contains the ID of a booking that contains this flight.

ReservationWithBooking (Booking, HotelReservation).

• Booking:	Contains the ID of a booking that contains this hotel reservation.
• HotelReservation:	Contains the ID of the hotel reservation included in this booking.

RentalCarWithBooking (Booking, RentalCar).

• Booking:	Contains the ID of a booking that contains this rental car.
• RentalCar:	Contains the ID of the rental car included in this booking.

CustomerWithBooking (Customer, Booking).

• Customer:	Contains the ID of the customer associated with this booking.
• Booking:	Contains the ID of a booking that contains this customer.

8. Interface Design

After developing a well designed system infrastructure, it is necessary to develop a user interface which harnesses the capabilities and speed of the infrastructure. A poorly designed interface can lead to low productivity and user frustration even though the underlying system may have been well designed. Because of this, it is imperative that great emphasis be put into designing an interface which takes full advantage of the system and makes it as intuitive as possible for users to carry out their tasks.

8.1. Designing the Interface:

There will be a client application which will be the only interface to the system. All classes of users (including management) will use this program to access the processing tier of the system. Having a single client application supporting all functionality of the system with user class privileges will be easiest to maintain since there is only one code base. However, security may be an issue if a malicious employee decides to try to gain access to another higher user class to gain additional privileges. Though this situation is unlikely, care must be taken to place the appropriate safeguards into the application to prevent this. Also, the system should log all activities by all users in case such an event occurs.

The client interface will be designed using the guidelines in Appendix G: Characteristics of a Good Interface. The prototype client interface that has been created adheres closely to these guidelines. See Appendix H for screenshots of the prototype. Instead of highlighting interface characteristics here, they have been placed below each screenshot to facilitate easier reading.

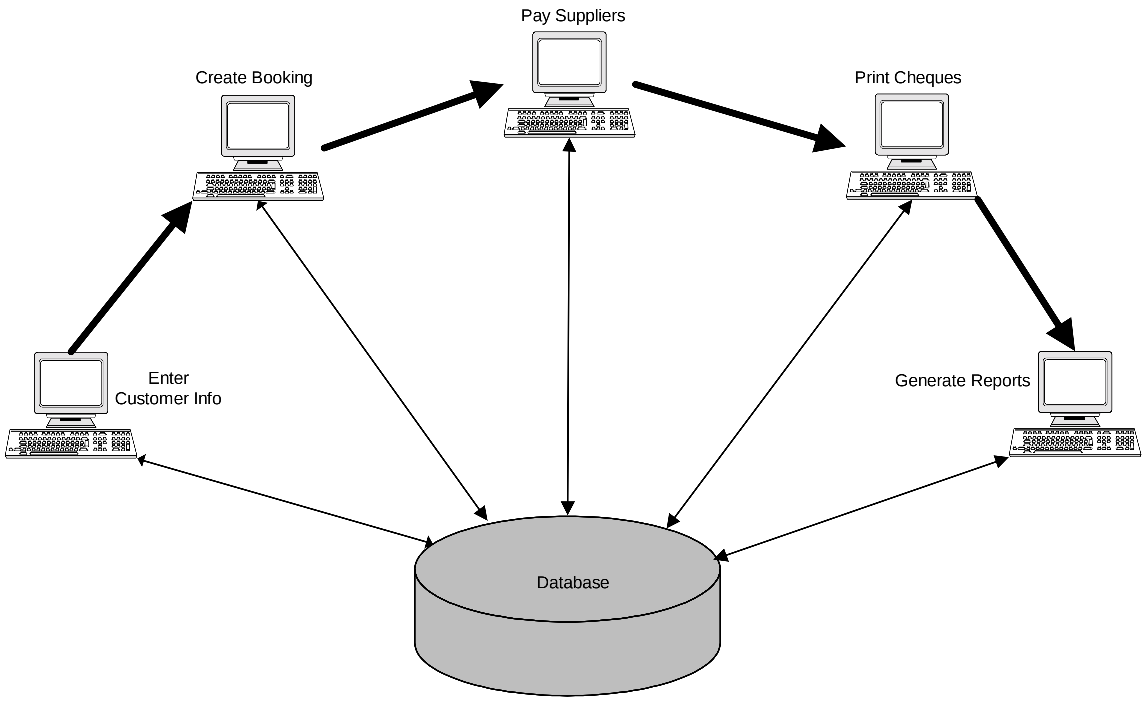
9. Conclusion

The above system recommendations are a solid basis for the implementation of the system. Each of the recommendations provided was carefully chosen so as to satisfy the requirements determined in the requirements analysis. Beyond the requirements, recommendations were chosen based on price, interoperability and degree of upgradeability. Once completed, this system will remedy all of the problems Eurosun Inc. had with their old software system. It will also provide them with more features and plenty of performance to satisfy their requirements well into the future.

Appendices

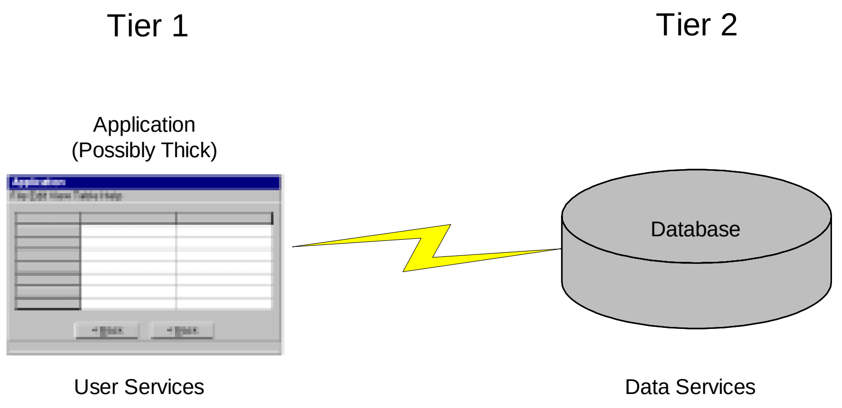
Appendix A1: Repository Based Software Architecture

The following is a representation of a repository based system applied to this Eurosun Inc. Typically one process should be completed before another one begins. Although this is generally going to be the case in this application, operations will not always work this way, so a repository system is not well suited to this system.

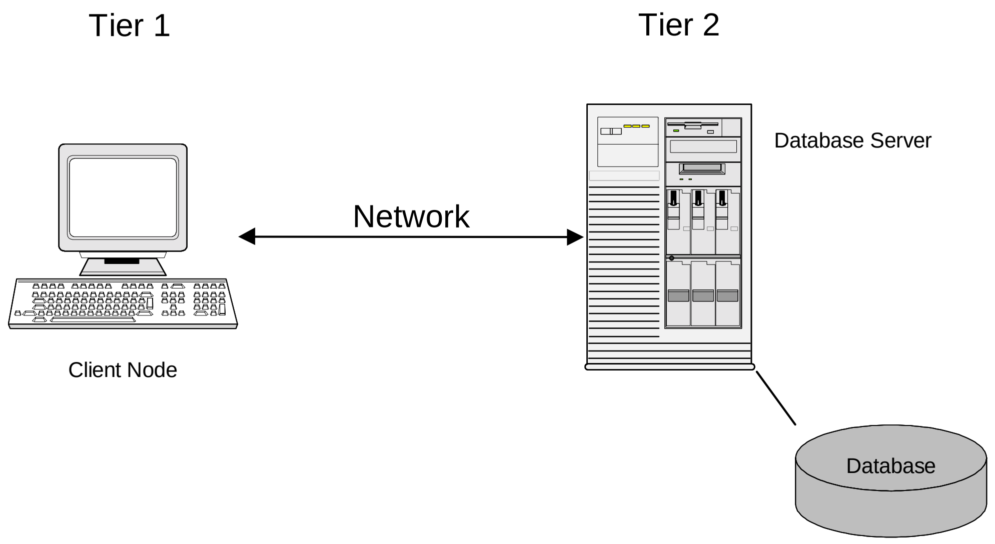


Appendix A2: Two-Tier Client Server Architecture

Logical Representation

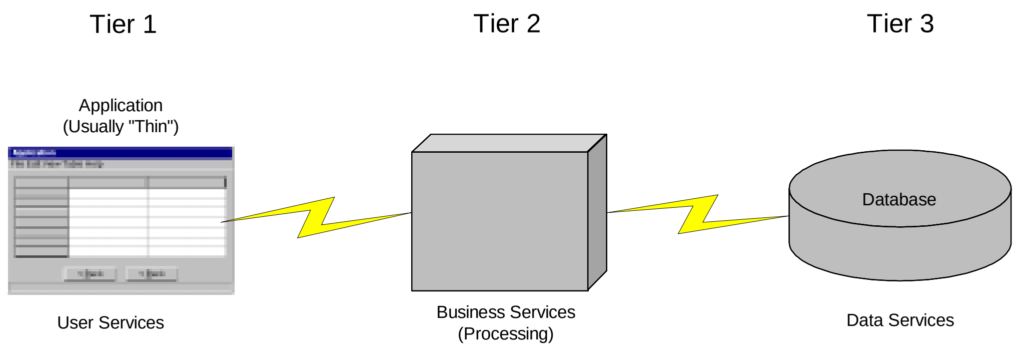


Physical Representation

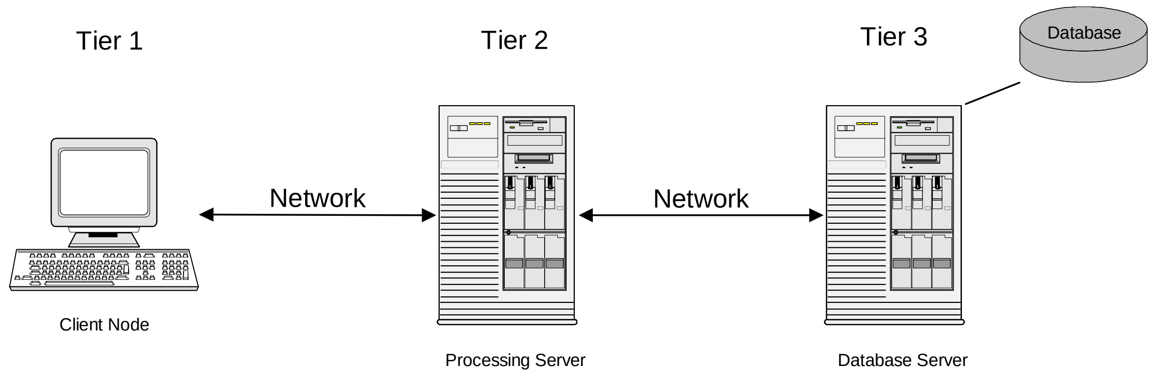


Appendix A3: Three-Tier Client Server Architecture

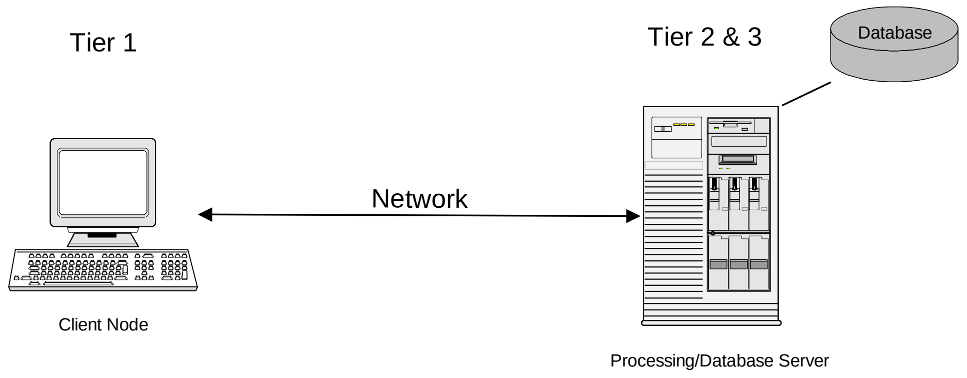
Logical Representation



Physical Representation - Tier 2 & 3 Separate



Physical Representation - Tier 2 & 3 Together



Appendix B1: Server Comparison

Database Server Comparison			
	Dell Power 1400 (Customized)	**Compaq ProLiant ML330**	**IBM Netfinity 5100**
Form Factor	Medium Tower	Medium Tower	Medium Tower
Processor	Single Intel Pentium III 800MHz (133MHz Bus)	Single Intel Pentium III 800MHz (133MHz Bus)	Single Intel Pentium III 866MHz (133MHz Bus)
Hard Drives	2 x 9GB, 7200RPM, Ultra160 SCSI	2 x 9GB, 7200RPM, Ultra2 SCSI	2 x 9GB, 7200RPM, Ultra160 SCSI
Hard Drive Configuration	C3, Add-In RAID 1	RAID 1	RAID 1
RAM	256MB SDRAM (1x256) 133MHz	256MB SDRAM (2x64, 1x128) 133MHz	256MB SDRAM (2x64) ECC 133MHz
Controller Card	Perc2-DC, 64MB, 2-Internal Channels	Smart Array 431 RAID Controller	ServerRAID-4L Ultra160
Network Cards	On-Board NIC + Intel Pro 100 Plus NIC	Compaq 10/100 TX UTP	Integrated 10/100 Ethernet Controller
Warranty	3 Year Next Business Day On-Site Service	Compaq 3 Year Limited Warranty	Not Mentioned
Support	DirectLine, 3 Res/Expire 3 Years	9am - 5pm 4 hour on-site coverage-3 years	Not Mentioned
Operating System	None	None	None
Tape Backup	Seagate 20GB Int SCSI	12/24 GB DAT Tape Drive.	Seagate 20GB Int SCSI
Uninterruptible Power Supply	UPS 700VA, 120 Standalone	Compaq UPS T700	Smart-UPS 1000
Price	**$4,562.00**	**$4,152.00**	**$5,016.00**

Table 1: Server Comparison

Appendix B2: Client System Configuration

Category	Chosen Component	
CPU	Intel Celeron 600MHz	
Case	Medium Tower ATX	
Motherboard	ASUS CUV4X-E	
Memory	128MB PC133 SDRAM	
Hard Drive	10GB Fujitsu UltraDMA/66 5400RPM	
CD-ROM Drive	Creative Labs 52X IDE	
Sound Card	Ensoniq PCI	
Video Card	ATI Xpert 98 8MB AGP	
Monitor	KDS 17" SVGA 1280x1024 Max	
Network Card	D-Link DFE-530+TX	
Total		**$ 934.90 Each**

Table 2: Client Systems Configuration

Appendix C: Database Comparison

	Oracle	IBM DB2	MS SQL Server	Microsoft Access
Service & Responsiveness	Very Good	Excellent	Good	Good
Features & Innovation	Excellent	Very Good	Good	Poor
Industry Expertise	Very Good	Excellent	Good	Excellent
Pricing & Value	Poor	Fair	Good	Excellent
Programming Expertise	Good	Very Good	Excellent	Good
Reliability	Very Good	Excellent	Good	Poor
Scalable And VLDBs	Excellent	Excellent	Very Good	Poor

Table 3: Database Comparison

References:

Whiting, Rick. "Database Grudge Match" InformationWeek.com. 4 December 2000. <http://www.informationweek.com/815/database.htm>.

Appendix D1: ACID Compliant Database

Atomicity

Results of a transaction's execution are either all committed or all rolled back. All changes take effect, or none do. That means, for Joe User's money transfer, that both his savings and checking balances are adjusted or neither are.

Consistency

The database is transformed from one valid state to another valid state. This defines a transaction as legal only if it obeys user-defined integrity constraints. Illegal transactions aren't allowed and, if an integrity constraint can't be satisfied then the transaction is rolled back. For example, suppose that you define a rule that, after a transfer of more than $10,000 out of the country, a row is added to an audit table so that you can prepare a legally required report for the IRS. Perhaps for performance reasons that audit table is stored on a separate disk from the rest of the database. If the audit table's disk is off-line and can't be written, the transaction is aborted.

Isolation

The results of a transaction are invisible to other transactions until the transaction is complete. For example, if you are running an accounting report at the same time that Joe is transferring money, the accounting report program will either see the balances before Joe transferred the money or after, but never the intermediate state where checking has been credited but savings not yet debited.

Durability

Once committed (completed), the results of a transaction are permanent and survive future system and media failures. If the airline reservation system computer gives you seat 22A and crashes a millisecond later, it won't have forgotten that you are sitting in 22A and also give it to someone else. Furthermore, if a programmer spills coffee into a disk drive, it will be possible to install a new disk and recover the transactions up to the coffee spill, showing that you had seat 22A.

References:

Greenspun, Phillp. Phillip and Alex's Guide to Web Publishing. Chapter 12 <http://www.arsdigita.com/books/panda/index.html>.

Appendix D2: Relational Schema using Normal Forms

Normal forms are used as criteria to judge whether a schema is “better” than another. In the general case, a schema with a higher NF rating is considered to be “better” than those schemas with a lower rating. Normalization helps reduce the number of redundancies and anomalies in the relational schema.

The standard normalization forms are:

a) First Normal Form (1NF):
A relational schema is in this form if it does not include any multiple valued attributes or composite attributes.

b) Second Normal Form (2NF):
A relational schema is in this form if it is in 1NF and all non-key attributes depend on all elements of its key, rather than a subset.

c) Third Normal Form (3NF):
A relational schema is in this form if it is in 2NF and none of its non-key attributes depends on any other non-key attribute.

Appendix E1: Database Cost Model

Concept	Type	Volume
Sales Rep.	E	5
Sales Manager	E	1
Financial Manager	E	1
Supplier	E	75
Flight	E	3500
Hotel Reservation	E	3000
Rental Car	E	1250
Customer	E	5030
Hotel Season	R	300
Flight Offered By	R	3500
Car Rental Season	R	300
Booking	R	1500
Entered To DB	R	4175
Pay Supplier	R	2000

Table 1: Volumes

Operation	Type	Frequency
Add flight to database	I	50/month
Add flight to booking	I	20/day
Print Report	I	2/month
Pay Supplier	I	15/month
Add Hotel to database	I	5/month
Add Car Agency to database	I	5/month
Add Airline to database	I	5/month
Add Hotel Season	I	120/year
Add Car Rental Season	I	120/year
Create New Booking	I	10/day
Cancel Booking	I	2/day
Add Hotel Reservation to Booking	I	20/day
Add Car Rental to Booking	I	20/day
Look Up Customer	I	25/day

Table 2: Operations

Accesses:

Concept	Type	Accesses	Type
Sales Manager	Entity	1	R
Entered to DB	Relationship	1	W
Supplier	Entity	1	R
Flight Offered By	Relationship	1	W
Flight	Entity	1	W

Table 3: Add Flight to Database

Concept	Type	Accesses	Type
Sales Rep	Entity	1	R
Booking	Relationship	1	RW
Flight	Entity	1	RW
Customer	Entity	4 (avg)	RW

Table 4: Add Flight to Booking

Concept	Type	Accesses	Type
Financial Manager	Entity	1	R
Report	Relationship	1	W
Supplier	Entity	75	R

Table 5: Print Report

Concept	Type	Accesses	Type
Financial Manager	Entity	1	R
Pay Supplier	Relationship	1	W
Supplier	Entity	1	RW

Table 6: Pay Supplier

Concept	Type	Accesses	Type
Sales Manager	Entity	1	R
Entered to DB	Relationship	1	W
Supplier	Entity	1	W

Table 7: Add Hotel to Database

Concept	Type	Accesses	Type
Sales Manager	Entity	1	R
Entered to DB	Relationship	1	W
Supplier	Entity	1	W

Table 8: Add Car Rental Agency to Database

Concept	Type	Accesses	Type
Sales Manager	Entity	1	R
Entered to DB	Relationship	1	W
Supplier	Entity	1	W

Table 9: Add Airline to Database

Concept	Type	Accesses	Type
Sales Manager	Entity	1	R
Entered to DB	Relationship	1	W
Supplier	Entity	1	R
Hotel Season	Relationship	1	W

Table 10: Add Hotel Season to Database

Concept	Type	Accesses	Type
Sales Manager	Entity	1	R
Entered to DB	Relationship	1	W
Supplier	Entity	1	R
Car Rental Season	Relationship	1	W

Table 11: Add Car Rental Season to Database

Concept	Type	Accesses	Type
Sales Rep	Entity	1	R
Booking	Relationship	1	W
Customer	Entity	4 (avg)	W

Table 12: Create New Booking

Concept	Type	Accesses	Type
Sales Rep.	Entity	1	R
Booking	Relationship	1	RW
Hotel Reservation	Entity	2 (avg)	RW
Hotel Season	Relationship	1	RW
Rental Car	Entity	2 (avg)	RW
Car Rental Season	Relationship	1	RW
Flight	Entity	2 (avg)	RW
Customer	Entity	4 (avg)	RW

Table 13: Cancel Booking

Concept	Type	Accesses	Type
Sales Rep.	Entity	1	R
Booking	Relationship	1	RW
Hotel Season	Relationship	1	R
Customer	Entity	4 (avg)	RW
Hotel Reservation	Entity	1	W

Table 14: Add Hotel Reservation to Booking

Add Car Rental to Booking

Concept	Type	Accesses	Type
Sales Rep.	Entity	1	R
Booking	Relationship	1	RW
Car Rental Season	Relationship	1	R
Customer	Entity	4 (avg)	RW
Car Rental	Entity	1	W

Table 15: Add Car Rental to Booking

Concept	Type	Accesses	Type
Sales Rep	Entity	1	R
Booking	Relationship	1	R
Customer	Entity	1	R

Table 16: Look Up Customer Account

Appendix E2: Cases of Redundancy

Presence of redundancy: There is an AmountOwed attribute.

Absence of redundancy: There is no AmountOwed attribute.

Operation 1: Look up Customer account.
Operation 2: Create New Booking.
Operation 3: Add Flight to Booking
Operation 4: Add Hotel to Booking
Operation 5: Add Car Rental to Booking

Assume Read(R) has cost of 1, Write(W) has cost of 2, and ReadWrite(RW) has cost of 3

Concept	Type	Accesses	Type
Sales Rep	Entity	1	R
Booking	Relationship	1	R
Customer	Entity	1	R

Table 17: Look Up Customer Account (Presence of Redundancy)

Concept	Type	Accesses	Type
Sales Rep	Entity	1	R
Booking	Relationship	1	W
Customer	Entity	4 (avg)	W

Table 18: Create New Booking (Presence of Redundancy)

Add Car Rental to Booking (Presence of redundancy)

Concept	Type	Accesses	Type
Sales Rep.	Entity	1	R
Booking	Relationship	1	RW
Car Rental Season	Relationship	1	R
Customer	Entity	4 (avg)	RW
Car Rental	Entity	1	W

Table 19: Add Car Rental to Booking (Presence of Redundancy)

Concept	Type	Accesses	Type
Sales Rep.	Entity	1	R
Booking	Relationship	1	RW

Hotel Season	Relationship	1	R
Customer	Entity	4 (avg)	RW
Hotel Reservation	Entity	1	W

Table 20: Add Hotel Reservation to Booking (Presence of Redundancy)

Concept	Type	Accesses	Type
Sales Rep	Entity	1	R
Booking	Relationship	1	RW
Flight	Entity	1	RW
Customer	Entity	4 (avg)	RW

Table 21: Add Flight to Booking (Presence of Redundancy)

Concept	Type	Accesses	Type
Sales Rep	Entity	1	R
Booking	Relationship	5 (avg)	R
Customer	Entity	1	R

Table 22: Look Up Customer Account (Absence of Redundancy)

Concept	Type	Accesses	Type
Sales Rep	Entity	1	R
Booking	Relationship	1	W
Customer	Entity	4 (avg)	W

Table 23: Create New Booking (Absence of Redundancy)

Concept	Type	Accesses	Type
Sales Rep.	Entity	1	R
Booking	Relationship	1	RW
Car Rental Season	Relationship	1	R
Car Rental	Entity	1	W

Table 24: Add Car Rental to Booking (Absence of Redundancy)

Concept	Type	Accesses	Type
Sales Rep.	Entity	1	R
Booking	Relationship	1	RW
Hotel Season	Relationship	1	R
Hotel Reservation	Entity	1	W

Table 25: Add Hotel Reservation to Booking (Absence of Redundancy)

Concept	Type	Accesses	Type
Sales Rep	Entity	1	R
Booking	Relationship	1	RW
Flight	Entity	1	RW

Table 26: Add Flight to Booking (Absence of Redundancy)

Cost Comparison:

Cost of Operation 1 “Look Up Customer”:
Presence of Redundancy: (3R + 0W) $\times$ 25/day = 75 accesses per day
Absence of Redundancy: (7R+ 0W) $\times$ 25/day = 175 accesses per day

Cost of Operation 2 “Create New Booking”:
Presence of Redundancy: (1R + 5W) $\times$ 10/day = 110 accesses per day
Absence of Redundancy: (1R + 5W) $\times$ 10/day = 110 accesses per day

Cost of Operation 3 “Add Car Rental to Booking”:
Presence of Redundancy: (2R + 1W + 5RW) $\times$ 20/day = 380 accesses per day
Absence of Redundancy: (2R+ 1W + 1RW) $\times$ 20/day = 140 accesses per day

Cost of Operation 4 “Add Hotel Reservation to Booking”:
Presence of Redundancy: (2R + 1W + 5RW) $\times$ 20/day = 380 accesses per day
Absence of Redundancy: (2R+ 1W + 1RW) $\times$ 20/day = 140 accesses per day

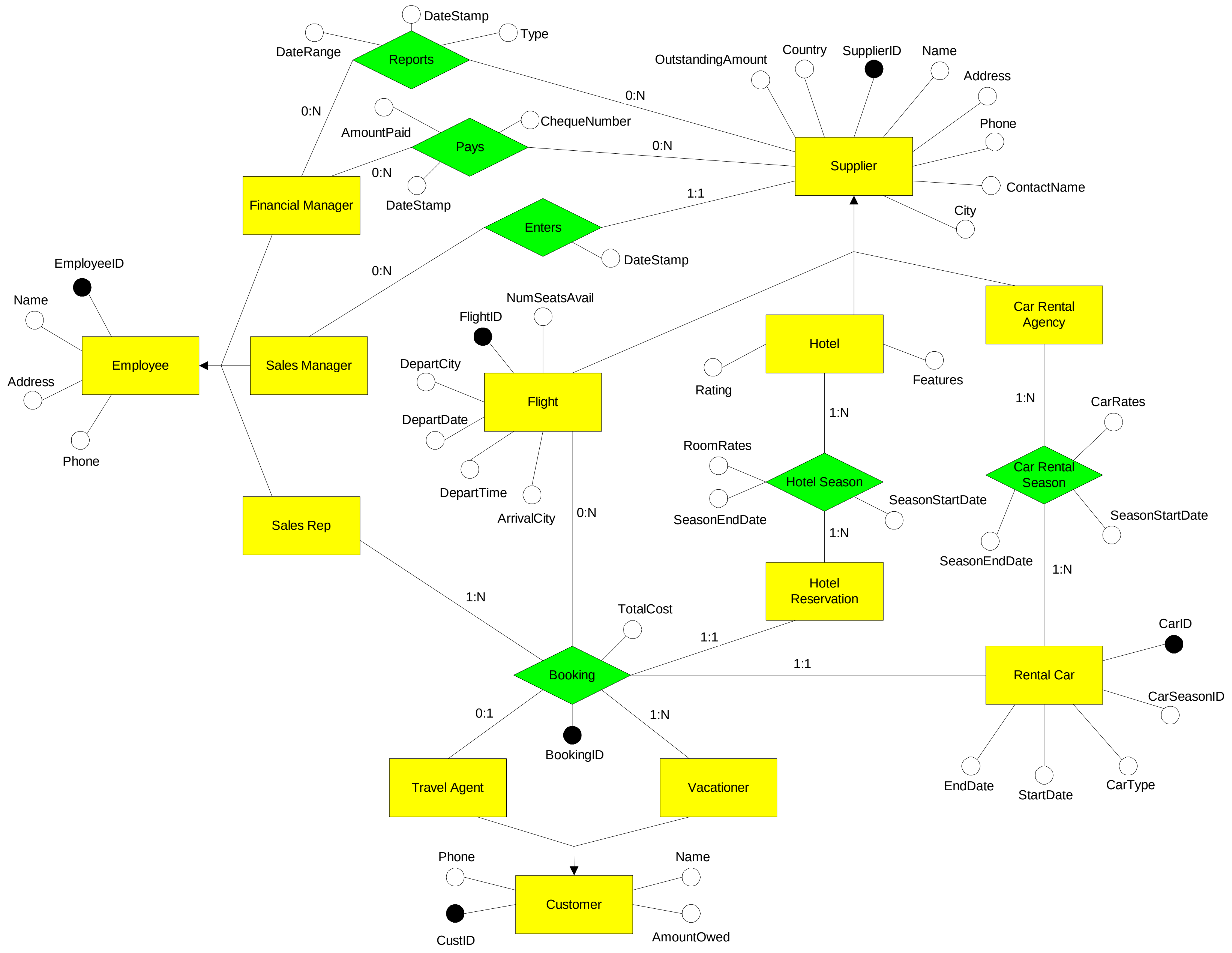
Cost of Operation 5 “Add flight to booking”:
Presence of Redundancy: (0R + 1W + 6RW) $\times$ 20/day = 400 accesses per day
Absence of Redundancy: (0R+ 1W + 2RW) $\times$ 20/day = 160 accesses per day

Total Cost (Presence of Redundancy) = 1345 accesses per day
Total Cost (Absence of Redundancy) = 725 accesses per day

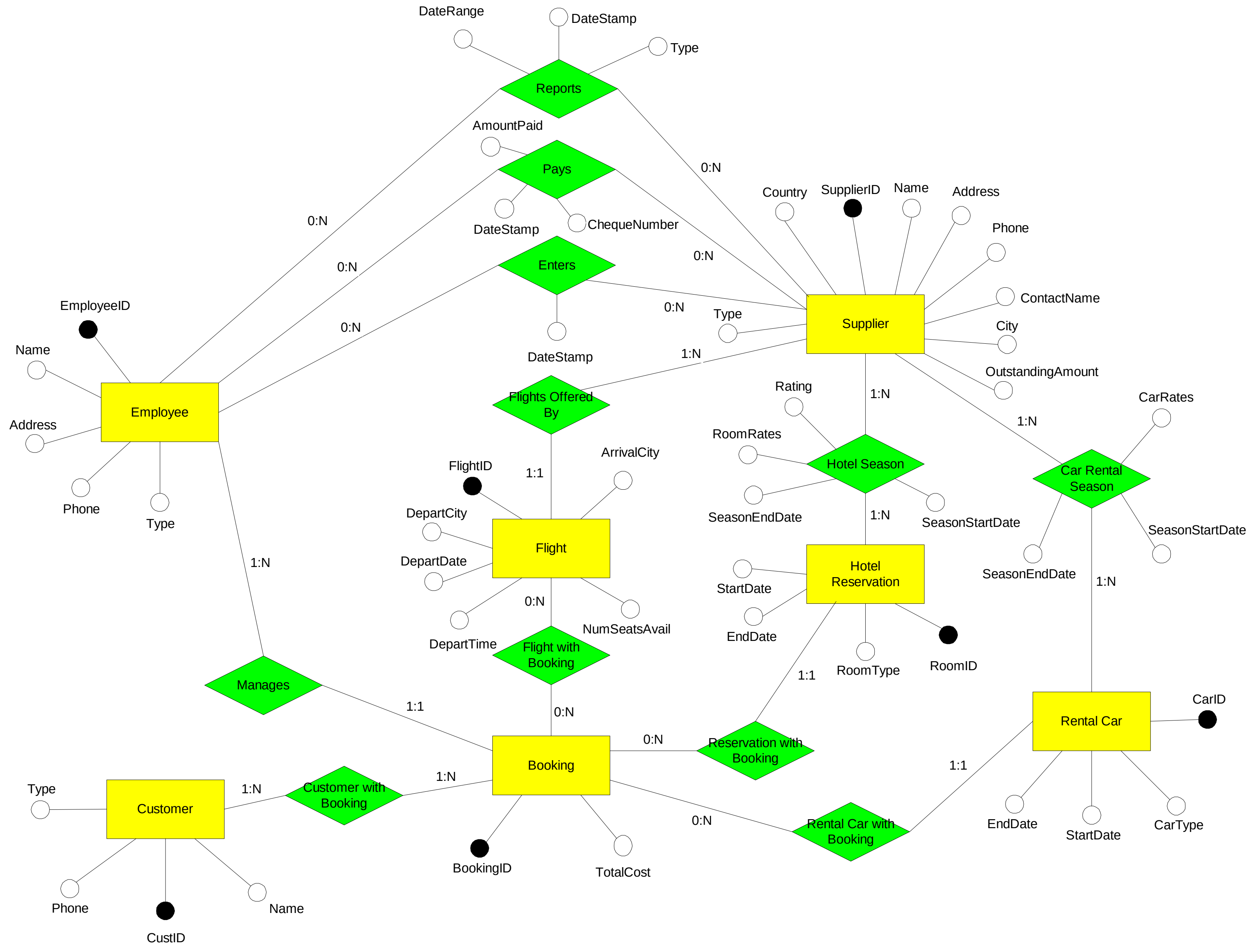
Results:

As evident in the above data, the absence of redundancy improves performance of the database.

Appendix F1: Initial E-R Diagram



Appendix F2: Optimized E-R Diagram (Tables in Database)



Appendix F3: E-R Data Dictionary

Entities

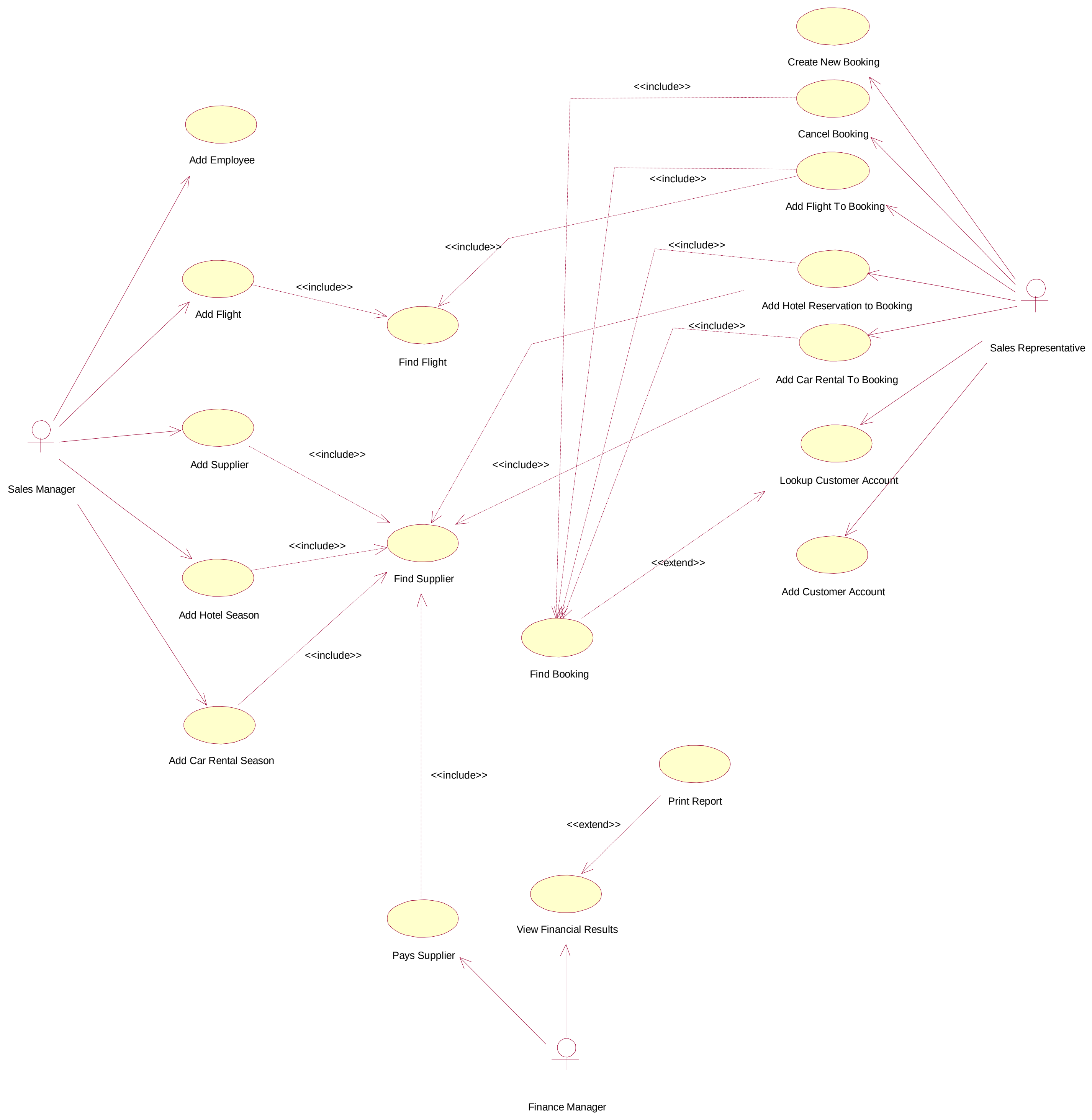
Entity	Description	Attributes	Identifier
Employee	Employee contains information pertaining to each employee at Eurosun Holiday's.	EmployeeID, Name, Phone, Address, Type	EmployeeID
Customer	Customer contains information pertaining to each customer of Eurosun Inc.	CustID, Name, Phone, Type	CustID
Booking	Each customer of Eurosun Inc. is associated with a certain number of bookings. The Booking is used to acquire all the information for each customer's vacation.	BookingID, TotalCost	BookingID
Flight	Flight contains information about each flight that Eurosun offers or has offered to their customers in the past.	FlightID, NumSeatsAvail, DepartCity, DepartDate, DepartTime, ArrivalCity	FlightID
Supplier	Supplier contains the information of each supplier of Eurosun Inc. This information may be used to contact suppliers.	SupplierID, Name, Phone, Address, City, Country, AmountOwed, Type	SupplierID
Hotel Reservation	Hotel Reservation contains information for each hotel reservation that a vacationer of Eurosun Inc. makes.	RoomID, RoomType, StartDate, EndDate	RoomID
Rental Car	Rental Car contains information for each car rental that a vacationer of Eurosun obtains.	CarID, CarType, StartDate, EndDate	CarID

Relationships

Relationship	Description	Entities Involved	Attributes
Reports	Reports contains information about each report made by an employee about a certain supplier.	Supplier(0,N), Employee (0, N)	DateRange, DateStamp, Type
Pays	Pays contains records about each payment that Eurosun has issued to its suppliers.	Supplier(0, N), Employee (0, N)	DateStamp, ChequeNumber, AmountPaid
Enters	Enters keeps track of which employees enter which suppliers in the database.	Supplier(0, N), Employee (0, N)	DateStamp
Flights Offered By	FlightsOfferedBy contains information about which flights certain suppliers offer.	Supplier(0, N), Flight(0, N)	
Hotel Season	HotelSeason contains information about which hotel reservations are within certain seasons for a specific supplier.	Supplier(1, N), Hotel Reservation(1, N)	Rating, RoomRate, SeasonStartDate, SeasonEndDate

Car Rental Season	CarRentalSeason contains information about which cars are available for rent in a specific season for a specific supplier.	Supplier(1, N), Rental Car (1, N)	CarRate, SeasonStartDate, SeasonEndDate
Manages	Manages contains information about which employees are handling certain bookings.	Employee(1, N), Booking (1, 1)	
Flight With Booking	FlightWithBooking contains information about which flights are included in certain bookings.	Flight(0, N), Booking(0, N)	
Reservation With Booking	ReservationWithBooking contains information about which reservations are included in certain bookings.	Hotel Reservation(1,1), Booking(0, N)	
Rental Car With Booking	RentalCarWithBooking contains information about which rental cars are included in certain bookings.	Rental Car(1, 1), Booking (0, N)	
Customer With Booking	CustomerWithBooking table contains information about which customers are included in certain bookings.	Customer(1, N), Booking (1, N)	

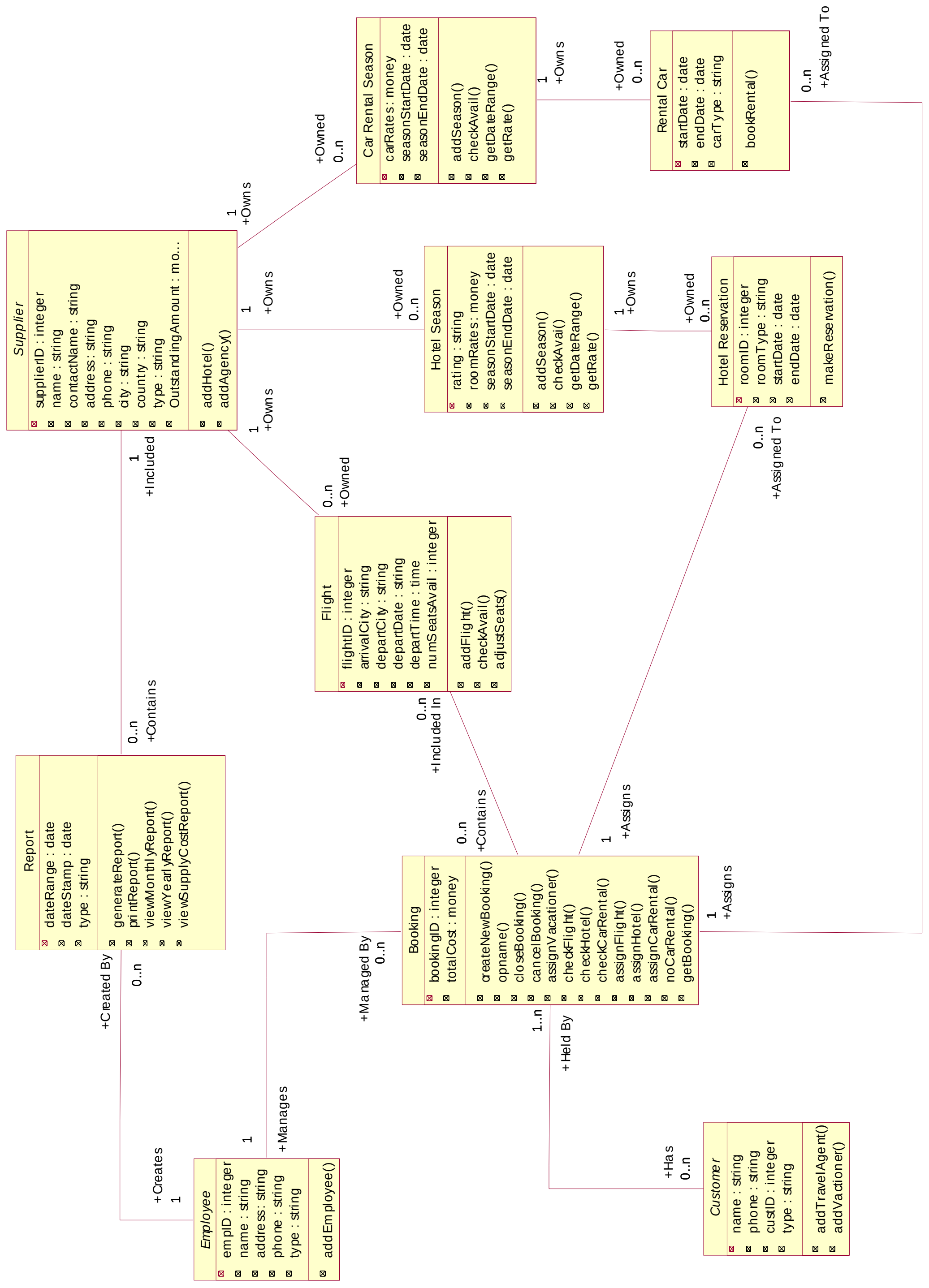
Appendix G1: Revised Use Case Diagram



Appendix G2: Use Case Diagram Data Dictionary

Use Case	Description
Add Employee	Sales Manager adds a new employee to the system, and sets there status to active
Add Flight	Sales Manager adds a new flight to the system. The flight is added by specifying a unique flight ID.
Add Supplier	Sales Manager adds a new supplier to the database. The supplier may be an actual Hotel, Airline or Car Rental Agency.
Add Hotel Season	Because this is a hotel season, we already how the hotel is in the system or we would not get this information. The Manager then searches for the associated hotel, and proceeds to enter the new season. The system searches for a matching hotel season, doesn't find one, in which case, the hotel season is added.
Add Car Rental Season	Sales Manager initiates procedure to add a Car Rental Agency. The Manager then proceeds to search for an existing record. The system performs this search, doesn't find one, in which case the Manager enters the information and the car rental agency is added.
Pay Supplier	Manager initiates the procedure to print a cheque. The system generates the cheque provided that funds are available. A transaction record is stored and funds are adjusted.
View Financial Results	Finance Manager initiates the procedure to view a report. He chooses which type of report to view and has the option to print or close it.
Print Report	Prints any report specified by the Financial manager.
Create New Booking	Sales Representative initiates procedure to create a new booking. Information is exchanged between Sales Rep and customer and the Sales Rep finds any of the flights, hotels and car rentals that match the customers requests. If each request is met and the customer confirms, the booking is created, otherwise it is cancelled
Cancel Booking	Sales Representative initiates the procedure to cancel a booking, by searching using some field. The system finds the booking and removes it, replacing all resources that were assigned to it.
Add Flight To Booking	Sales Representative initiates the procedure to add a flight to the booking. He searches for the flight. The system performs the search, finds the flight, in which case it is added to the booking and the appropriate seats are deducted from the flight.
Add Hotel Reservation To Booking	Sales Representative initiates the procedure to add a hotel reservation to the booking. He searches for the hotel and date range requested. The system performs the search and determines that there are rooms available, in which case it is added to the booking.
Add Car Rental To Booking	Sales Representative initiates the procedure to add a car rental to the booking. He searches for available rentals. The system performs the search, finds available rentals, in which case the Sales Representative adds a car rental to the booking.
Look Up Customer Account	Sales Representative initiates procedure to find customer account. He enters in information to search for. The system performs the search, and if found, allows the Sales Representative to edit the record.
Add Customer Account	New customer or travel agent calls or walks in. Sales Representative initiates procedure to add a new customer. The Sales Representative enters in their information, and if the system does not find a match, the customer is added as either a travel agent or vacationer.

Appendix H1: Revised Class Diagram

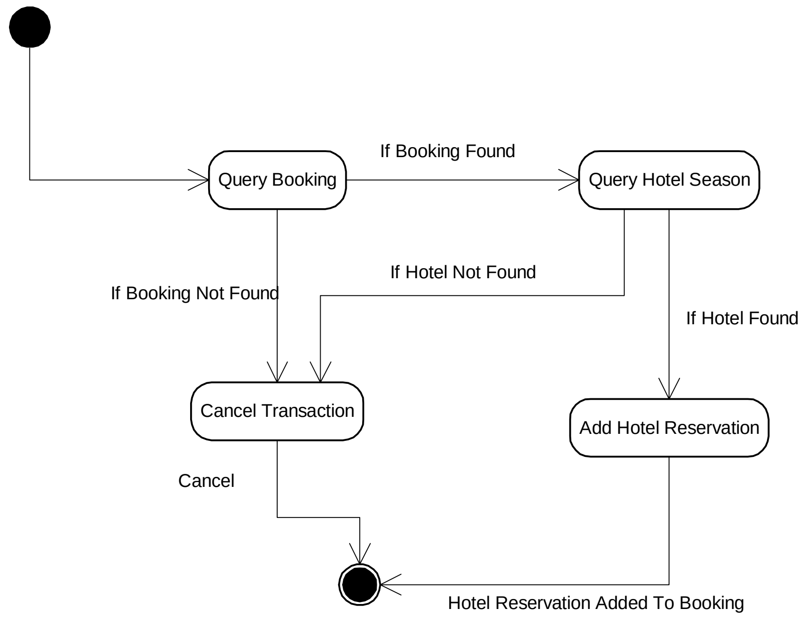


Appendix H2: Class Diagram Data Dictionary

Class	Description	Attributes	Operations
Employee	This class contains all personal information for each employee of Eurosun Inc. Each employee has their own unique employee ID that corresponds to what access rights they have on the system. All employees also have a password for the new system.	- empID - name - address - phone - type (Financial Manager, Sales Manager, Sales Rep)	- addEmployee() : adds an employee to the database.
Customer	This class contains all personal information for each customer of Eurosun Inc.	- name - phone - custID - type (Vacationer, Travel Agent)	- addTravelAgent() : adds a travel agent to the database. - addVacationer() : adds a vacationer to the database.
Report	This class is responsible for assisting the Finance Manager. It creates many different reports for the major areas of the system.	- dateRange - dateStamp - type (Sales, Supplies, Costs)	- generateReport() : generates a variety of financial reports base on data in the database. - printReport( ): prints any report generated by the system. - viewMonthlyReport() : a report generated on a monthly basis that captures recent sales figures. - viewYearlyReport() : a report generated on a yearly basis that captures recent sales figures. - viewSupplyCostReport(): a report that captures all the supply costs of business.
Booking	This class represents the bookings for Eurosun Inc. Each booking has all the information about a customer's vacation. This includes the vacationers, hotel reservations, rental cars, and flights. Each booking is given a unique identification number for tracking purposes.	- bookingID - totalCost	- reateNewBooking() : creates a new booking in the database. - getBooking() : opens an existing booking in the database. - closeBooking() : close an open booking after changes have been made to it. - cancelBooking() : removes a booking from the database. - checkFlight() : checks whether a specific flight is available, and determines how many seats are on the flight. - checkHotel() : checks whether there are any vacancies at a specific hotel. -checkCarRental() : checks whether there a specific location that offer rental car service. - assignVacationer() : assigns a vacationer to a booking,. - assignFlight() : assigns a flight to a booking, if it exists. - assignHotel() : assigns a hotel to a booking, if it exists. - assignCarRental() : assigns a rental car to a booking, if it exists. - noCarRental() : sets no rental car status on a booking.
Supplier	This class contains general information about each supplier that provides goods and services to Eurosun Inc.	- supplierID - name - address - phone - type (Airline, Hotel, Car Rental	- addHotel() : adds a new hotel to the database. - addAgency() : adds a new car rental agency to the database. - addAirline() : adds a new airline to the database

		Agency) - outAmount	
Flight	This class extends the Supplier class and holds information that is specific to flights such as arrival city, departure city, times, dates, and seat prices.	- flightID - arrivalCity - departCity - departDate - departTime - numSeatsAvail	- addFlight() : adds the specific flight to the database. The flight must be offered by one of the airlines known by the database.. - adjustSeats() : adjusts the number of seats on a particular flight. - checkAvail() : checks if a particular flight is available.
HotelSeason	This class keeps track of hotels prices over certain periods of time (seasons). It includes information about room rates, availability and dates.	- rating - roomRates - seasonStartDate - seasonEndDate	- addSeason() : adds a hotel season to the database. - getDateRange() : determines the number of days that the hotel room must be booked for. - getRate() : determines the proper room rate depending on the current season. - checkAvail() : checks if a particular hotel has vacant rooms.
Hotel Reservation	This class keeps track of a vacationer's stay at a hotel and the total price of the reservation.	- roomID - roomType (one, two, etc. bedroom, honeymoon) - startDate - endDate	MakeReservation() : assigns a hotel reservation to a booking
CarRental Agency	This class extends the Supplier class and holds information that is specific to Car Rental Agencies such as category of car.	- category (airport, touring, business) - carRates - seasonStartDate - seasonEndDate	- addSeason() : adds a car rental season to the database. - getDateRange() : determines the number of days that the rental car must be booked for. - getRate() : determines the proper car rate depending on the current season. - checkAvail() : checks if a particular car agency has available rental cars.
RentalCar	This class keeps track of a vacationer's car rental, if they choose to rent one. It has the total price of the rental and the date range in which they have rented the car.	- startDate - endDate - carType (economy, mid-size, sports, luxury)	BoorRental() : assigns a rental car to a booking.

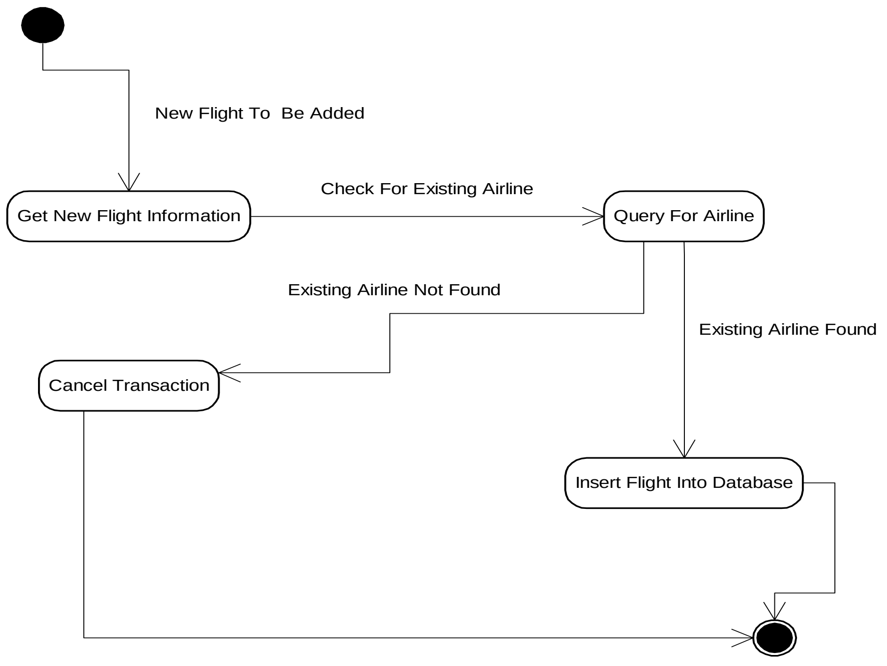
Appendix I1: State Diagram - Add Hotel Reservation To Booking



The above state diagram captures all the states that occur in the system when adding a hotel reservation to a booking.

The initial state begins with the Sales Representative being asked to add a particular hotel reservation to a booking. The next state, Query Booking, is entered when the Sales Representative first queries the system to check if such a booking exists. If the booking does not exist, the transaction is cancelled and the end state is reached. If the booking does exist, we enter a new state, Query Hotel Season, that queries the system on a particular hotel season. This state must check if the requested hotel exists. If the hotel does not exist, the transaction is cancelled and we reach the end state. If the hotel does exist, we enter a new state, Add Hotel Reservation, and add the hotel reservation to the booking.

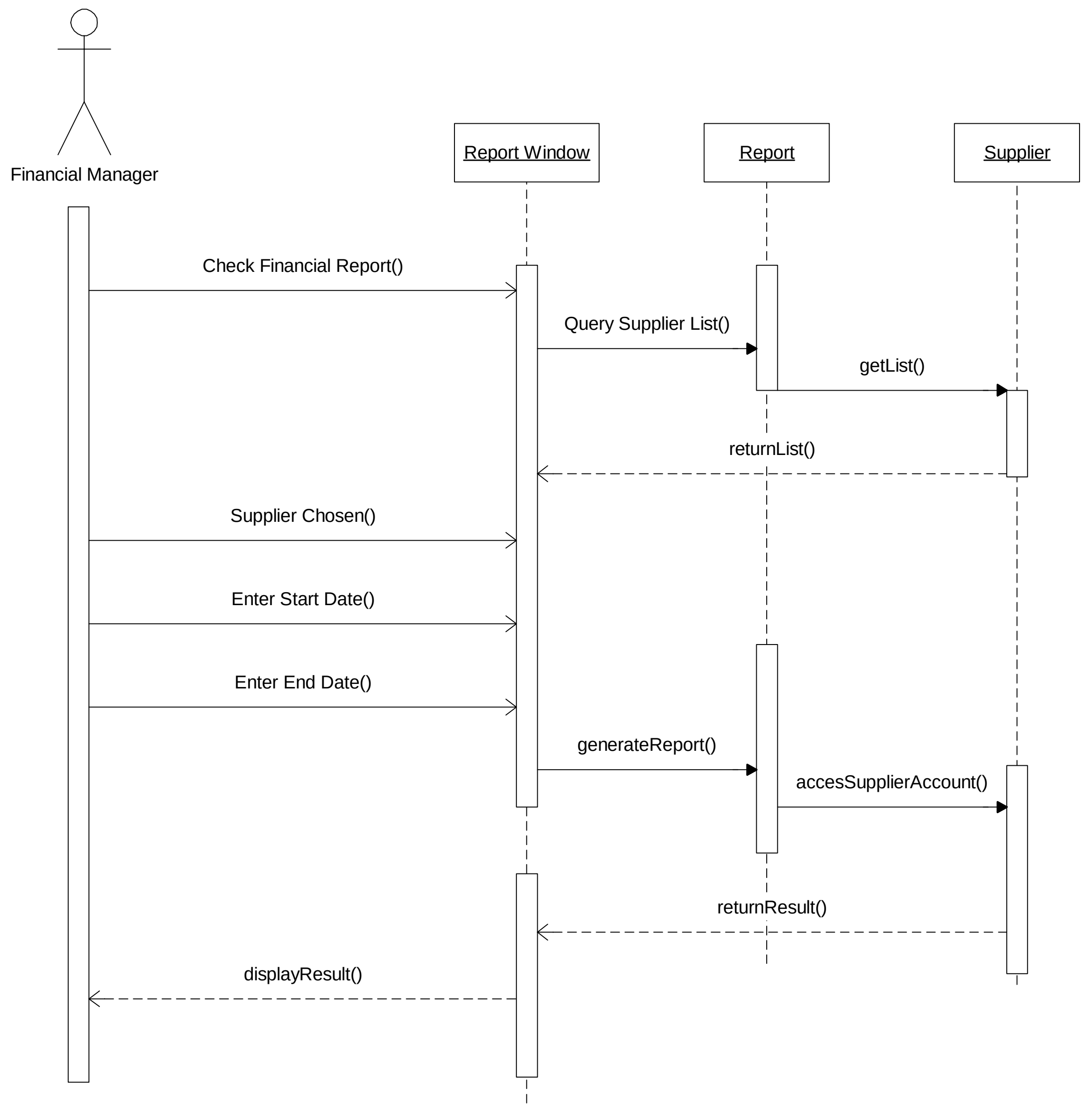
Appendix I2: State Diagram – Add Flight to Database

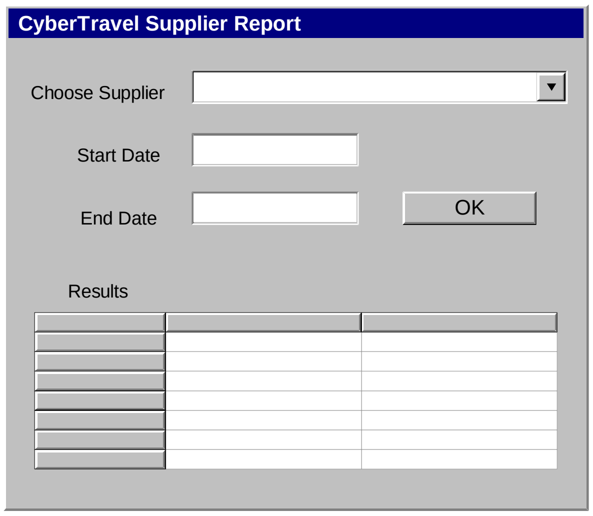

New Flight To Be Added
Get New Flight Information
Check For Existing Airline
Query For Airline
Existing Airline Not Found
Existing Airline Found
Cancel Transaction
Insert Flight Into Database

The above state diagram models all the states that occur in the system when adding a flight to a database.

The initial state begins with the Sales Manager adding a flight to the database. The next state, Get New Flight Information, is entered and the system is asked to check the given flight information. Once the information is read, we enter the Query For Airline state which queries the database for the airline. If the airline does not exist, the transaction is cancelled and the final state is reached. If the airline does exist, we enter the final state, Insert Flight Into Database, which allows the Sales Manager to insert the new flight information into the database.

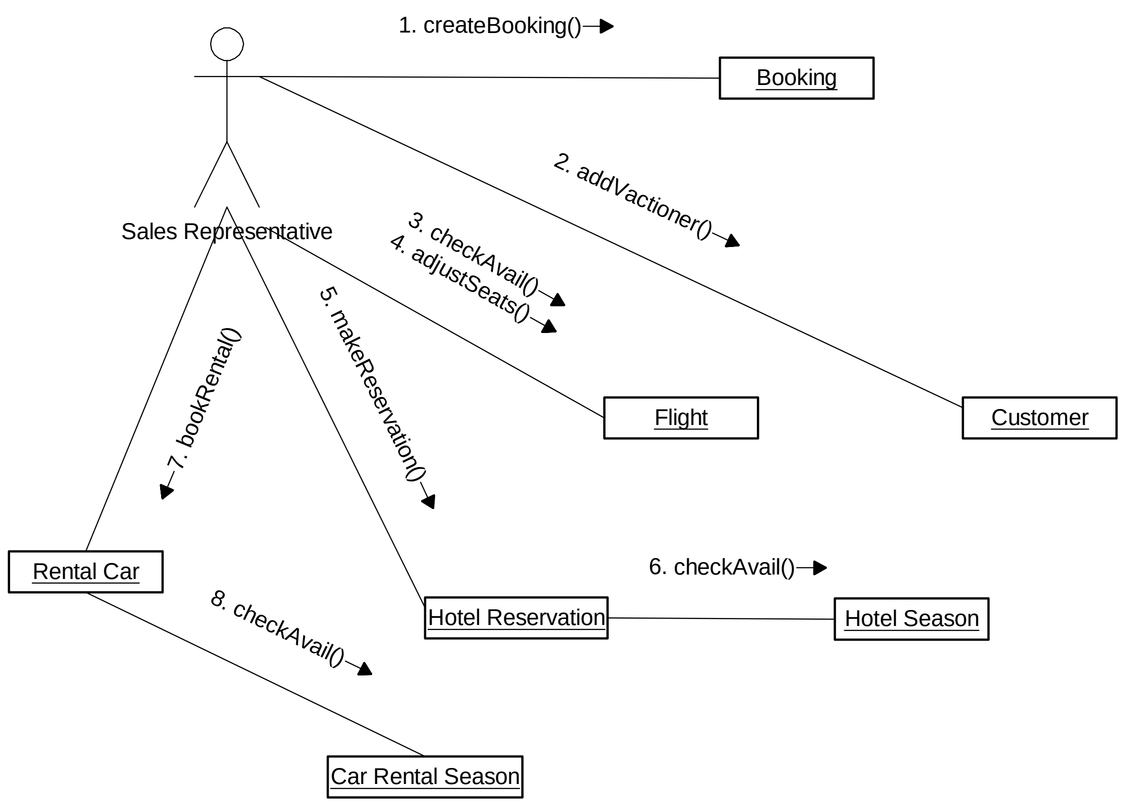
Appendix J: Sequence Diagram – Generate Supplier Report





The above diagram shows the interaction between the Financial manager and the database system. This sequence diagram is similar to the sequence diagram in the Requirements Analysis. However, this diagram implements the application interface that the Financial Manager will use to generate a supplier report.

Appendix K: Collaboration Diagram – Creating a New Booking



The above collaboration diagram models how the Sales Representative actor uses the Create New Booking use case. The diagram shows the interaction between the Sales Representative and the system components.

A Sales Representative takes the customer request and proceeds to create a new booking in the system. The Sales Representative asks for a list of vacationers that will be attending the trip. The list of vacationers is then assigned to the booking. The Sales Representative then searches for the requested flight(s). If the required flights are available, the flights are assigned to the booking and the availability is adjusted.

Next the Sales Representative checks for Hotel availability. If the required number of hotel rooms is available, the Sales Representative assigns the Hotel reservations to the booking.

If the customer requires a rental car, the Sales Representative checks the local Car Rental Agencies for available vehicles. If a rental car is available, then we assign it to a booking.

The system has completed the required steps to create a new booking.

Appendix L: Characteristics of a Good Interface

a) Affordances:
This refers to the level in which the functions provided by the interface are self-explanatory and easy to locate. For example, is a function hidden deep within a menu?

b) Mapping:
This refers to how well the interface relates an action performed by the system to what the user expects the system to do. For example, does the image on a button sufficiently describe the function it performs?

c) Feedback:
This refers to the level in feedback the interface provides to user’s actions. For example, if a user clicks a button, they are provided with a dialog box which asks them whether or not they would like to continue.

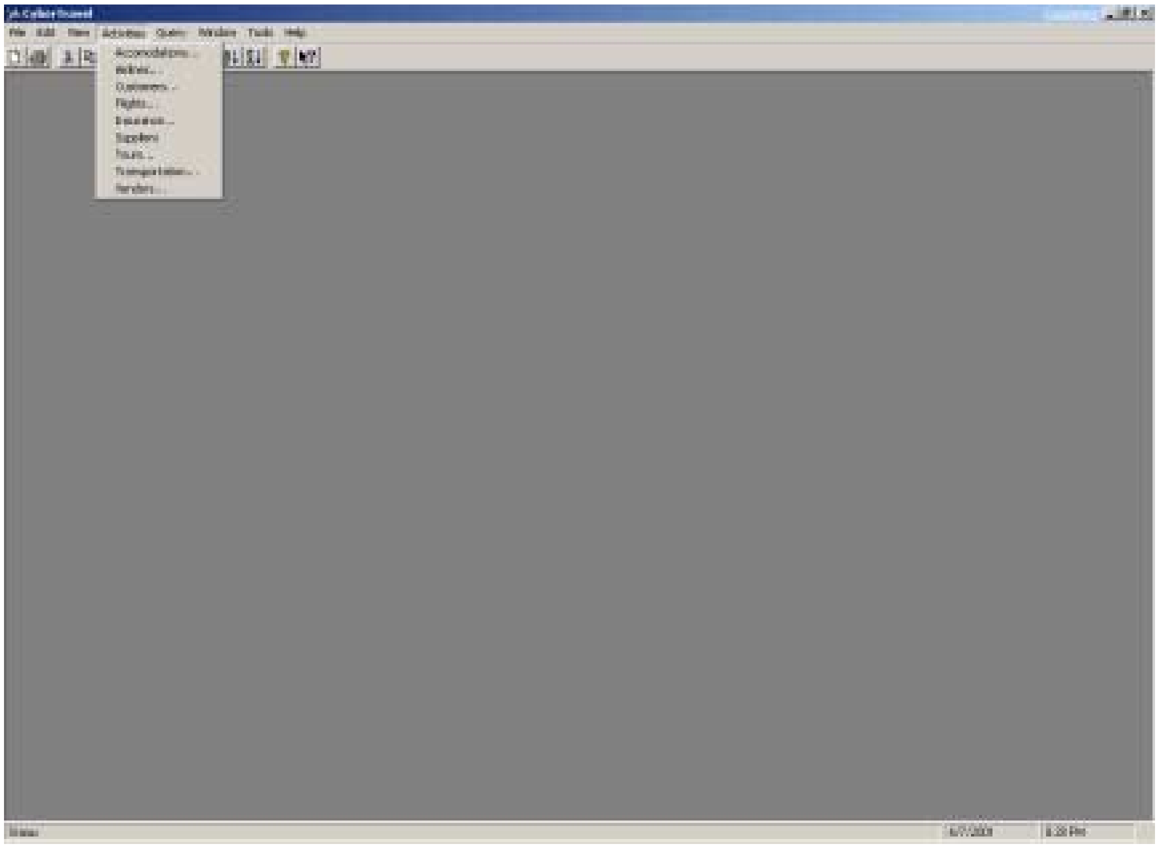
d) Mental Model:
This refers to the level in which the interface can relate with the user’s understanding of what is going on in the system when they perform an action. For example, the send button in an email application provides a good mental model of what happens when the button is clicked.

e) Forcing Functions:
This refers to the level in which the interface hides functionality until that function is allowed to be performed. For example, if the interface contains a list with no data, then the delete action should be greyed out to let the user know they cannot delete anything.

f) Automatic Learning:
This refers to the level in which the interface facilitates in the user’s learning its functions quickly. To accomplish this, the interface should remain consistent and remember the user’s settings.

Appendix M1: User Interface – Main Program

The following screenshot displays what the program looks like when it is started for the first time.



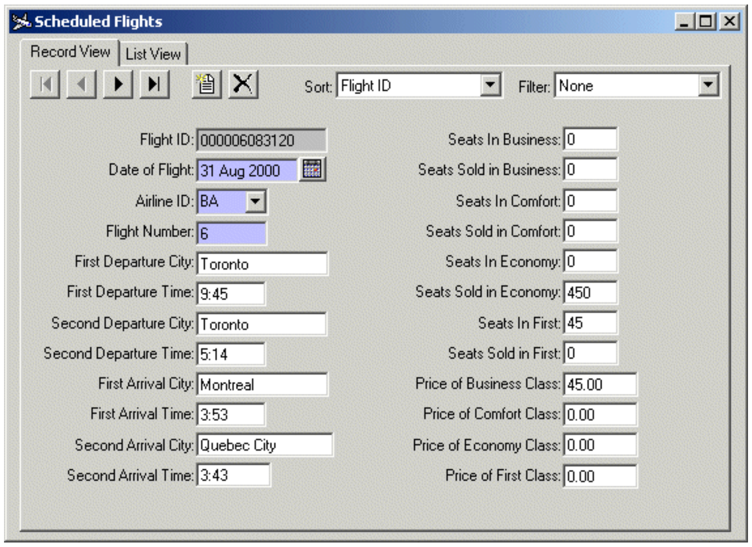
Automatic Learning:
When the program is started for the first time, no windows are shown, however any windows that are still on screen when closing the program, will appear in the exact same location on the next loading of the program, providing a type of bookmarking function that allows the user to quickly pick up where they left off.

Affordances:
The main interface has very little clutter which makes it easy to find functions, also the application uses icons and menus standard to many Windows based programs.

Mapping:
The menus are appropriately labelled to help the user find the action they want to perform quickly. If the user wants to perform a fast search, they use the Query menu. If they want to manage the flights table, they go to Activities>Flights.

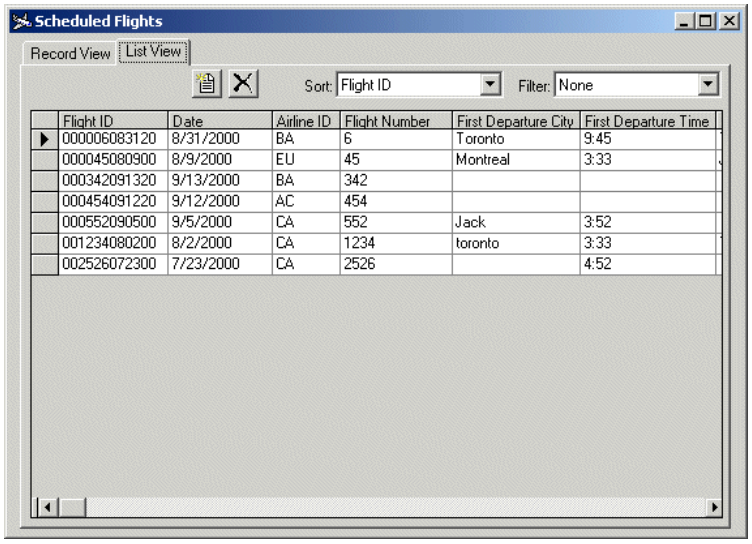
Appendix M2: User Interface – Flights Window

The following screenshot demonstrates the interface to the flights table, accessible through the Activities>Flights menu.



Sort: Flight ID
Filter: None

Flight ID:	000006083120
Date of Flight:	31 Aug 2000
Airline ID:	BA
Flight Number:	6
First Departure City:	Toronto
First Departure Time:	9:45
Second Departure City:	Toronto
Second Departure Time:	5:14
First Arrival City:	Montreal
First Arrival Time:	3:53
Second Arrival City:	Quebec City
Second Arrival Time:	3:43
Seats In Business:	0
Seats Sold in Business:	0
Seats In Comfort:	0
Seats Sold in Comfort:	0
Seats In Economy:	0
Seats Sold in Economy:	450
Seats In First:	45
Seats Sold in First:	0
Price of Business Class:	45.00
Price of Comfort Class:	0.00
Price of Economy Class:	0.00
Price of First Class:	0.00



Sort: Flight ID
Filter: None

Flight ID	Date	Airline ID	Flight Number	First Departure City	First Departure Time
000006083120	8/31/2000	BA	6	Toronto	9:45
000045080900	8/9/2000	EU	45	Montreal	3:33
000342091320	9/13/2000	BA	342		
000454091220	9/12/2000	AC	454		
000552090500	9/5/2000	CA	552	Jack	3:52
001234080200	8/2/2000	CA	1234	toronto	3:33
002526072300	7/23/2000	CA	2526		4:52

Affordances:
The form uses standard windows components, which most users are very comfortable with using.

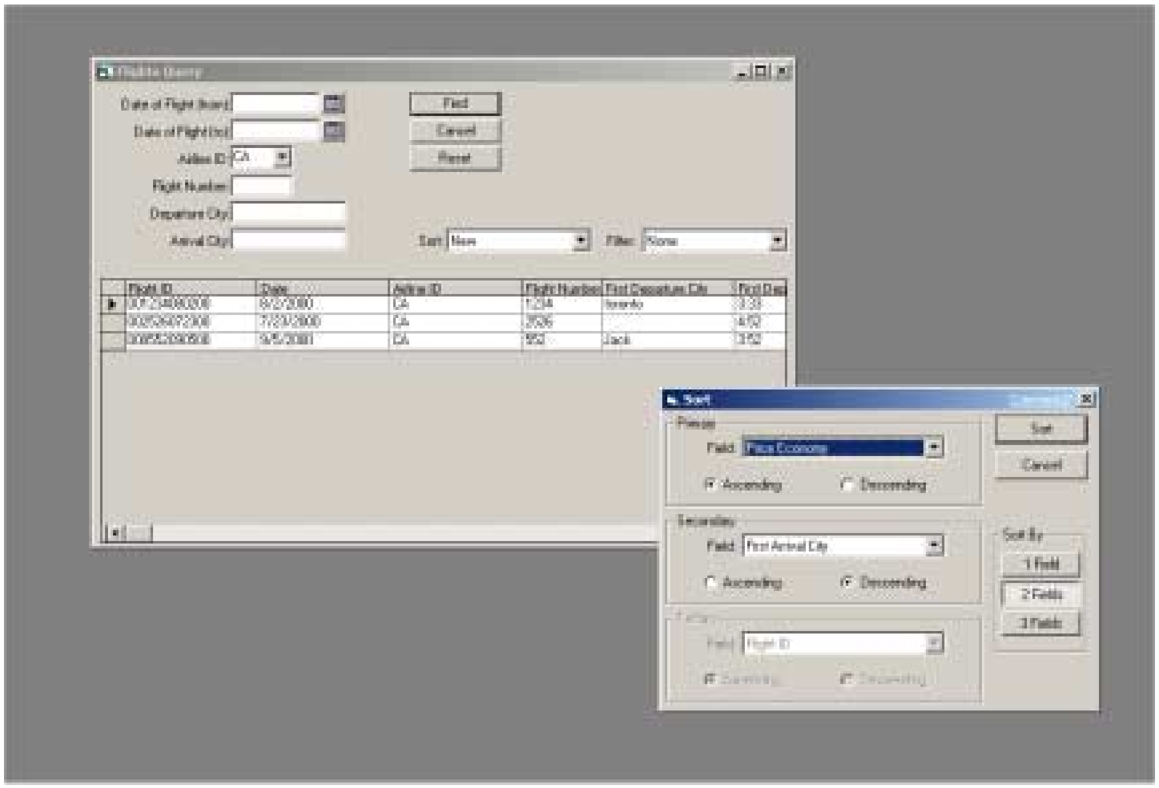
Mapping:
All the buttons have clear and simple icons which explain exactly what they do. The navigation buttons have arrows, the insert button has a picture of a record, the delete has an X. Most people, through their experiences with other programs and devices such as VCR's have a good understanding of what these buttons do. For those users that don't, they can simply move their mouse over the button to get a description of what it does.

Forcing Functions:
Depending on which record the user is currently looking at, the navigation buttons in the top left hand corner are active or inactive. For example, when the user is on the first record the first two buttons (which go to previous records) are disabled.

Mental Model:
This form provides two views of the data. Some users prefer to view the data in tables, while others prefer to only see one record at a time. The form layout provides excellent mental models of what the system is doing for different types of users.

Appendix M3: User Interface – Querying Database/Sorting/Filtering

The following screenshot demonstrates the querying abilities of the system. It allows for querying by any field. Once the results are returned, filters may be applied to any field to narrow down the results. Furthermore, the data can be sorted by any field (including multiple fields).



Date of Flight (from):
Date of Flight (to):
Airline ID: CA
Flight Number:
Departure City:
Arrival City:
Sort: None
Filter: None

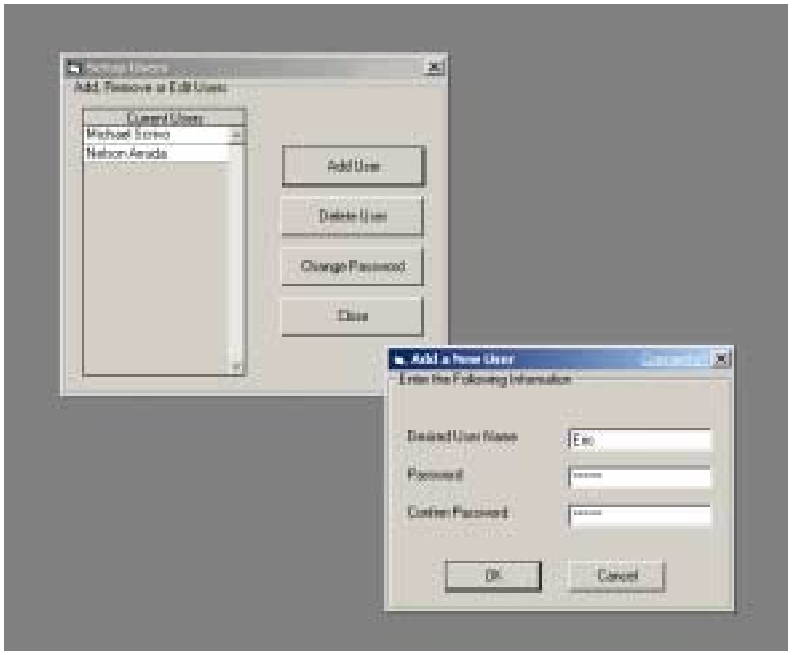
Flight ID	Date	Airline ID	Flight Number	First Departure City	First Dep
[illegible]	8/2/2000	CA	1234	toronto	3:33
[illegible]	7/23/2000	CA	2526		4:52
[illegible]	9/5/2001	CA	552	Jack	3:52

Primary: Field: Price Economy; [✓ Ascending] Descending
Secondary: Field: First Arrival City; Ascending [✓ Descending]
Tertiary: Field: Flight ID; Ascending Descending

Forcing Functions:
The sorting fields in the Sort dialog are greyed out until the user chooses the number of fields they want to sort by on the right hand side. Only one of these buttons can be clicked at any time.

Appendix M4: Administration Tools

The following screenshot demonstrates tools available to management/administration to allow adding, editing and removing of users of the system. The final version of the tools will allow administrators to pick and choose which components of the system certain classes of employees are allowed to access.



Michael Scrivo
Nelson Arruda

Desired User Name: Eric
Password: ******
Confirm Password: ******

Appendix N: Requirements Analysis

The following is a condensed version of the Requirements Analysis which only includes the revised Functional and Non-Functional Requirements.

1. Functional Requirements:

In what follows, we will outline the functional requirements of the new system. Functional requirements describe what processing the system must provide, for example how inputs are handled and what output is provided. The functional requirements have been broken down to Input, Data Management, and Output requirements.

1.1. Input:

1.1.1. New Supplier Information:

The software must allow the Sales Manager to insert flight, hotel, hotel season, car rental agency, and car rental season information. They must also have the ability to update this information when necessary. The system must store standard information for each of these items. This information was obtained by looking at company standards and document standards. The results can be found in Class Diagram.

1.1.2. New Customers:

The software must allow the Sales Representatives to add information about new customers. Customers include vacationers and travel agents. Customers will contact the Sales Representatives either in person or by phone.

1.1.3. Create Bookings:

The software must allow the Sales Representatives to create new bookings for both new and existing customers. The bookings must be able to accommodate groups and special tours. Input will come from customer phone calls/faxes, in person or through a form on the company web site.

1.1.4. Financial Management:

The software must allow the Financial Manager to view and print monthly and yearly reports about various aspects of the system. They must also be able to mange commissions for travel agents and pay for resources used from suppliers.

1.2. Data Management:

To manage all data in the system, the software will employ the use of a commercial DMBS (Database Management System). Commercial DBMS' are widely used and proven technologies and will allow development to progress at a much quicker rate, since almost all I/O is taken care of by the DBMS.

1.2.1. Bookings:

The system must be able to track information about individual bookings. The information must include the vacationers, flight information, hotel reservations, etc.

1.2.2. Cost of Goods Sold and Inventory:

The system must keep track of the costs of items sold, and what is available in inventory to sell.

1.2.3. Monitor Sales for Time Periods:

The system must track sales information so that management can use this information for future decisions.

1.2.4. Customer and Travel Agencies:

The system must be able to keep track of all customer information, and customer feedback. Information about Travel Agencies also must be kept to be able to contact them about sales and promotions.

1.2.5. Transactions:

Records of all transactions must be stored in a database for an indefinite amount of time. Old records should not be deleted under any circumstance. Storage and performance issues will be discussed in the non-function requirements.

1.3. Output:

1.3.1. Reports:

The software must allow the Finance Manager to be able to generate monthly and yearly reports regarding figures from sales, supplies, and costs.

1.3.2. Cheques:

The software must allow the Finance Manager to generate, review, and print cheques. These cheques need to be sent to suppliers, and travel agents who are owed commission.

2. Non-Functional Requirements:

Non-Functional requirements describe aspects of the system that are concerned with how well it supports the functional requirements. It is broken down into the following categories: Interface, Performance Operating, Life Cycle, Economic and Platform Requirements.

2.1. Interface:

To aid in a quick transition period and to allow for as little training as possible, the software must have a well designed, user friendly, GUI interface developed for the Windows family of Operating Systems.

2.2. Performance:

2.2.1. Reliability:

The system, comprising of the software, hardware, and network components must have a 99% uptime. The business relies completely on this system, so it must be running continuously. The system shall exhibit a Mean Time To Repair (MTTF) of no more than one hour. During the testing phase of our software development we will use the technique of bebugging. A number of seeded bugs will be placed in the code so to help use identify any other bugs within the software. This approach has its roots in the Monte Carlo statistical analysis techniques for random events. The final software will have no more than twenty bugs per thousand lines of code.

2.2.2. Usability:

To reduce the time that it takes to enter and retrieve information, there should be a minimal amount of screens required to perform a specific task. A well-designed interface will facilitate this requirement.

2.2.3. Resources:

Although resources are cheap and plentiful in this day and age, resource usage should still be kept as low as possible (i.e. Memory & Disk Space). Putting constraints on the amount of resources used helps develop a more bug-free and robust system.

2.2.4. Efficiency:

Will measure the level at which a software system uses scare resources.
Capacity: The company network will be able to handle at least 25 simultaneous connections. The software and database backend will be able to handle at least 25 simultaneous users.
Degradation of Service: When the network receives more than 25 simultaneous connections, the system will continue to run with degraded performance.

2.2.5. Security:

Only allow management to add or adjust information about flights, car rentals, and hotels. Furthermore, only management will have access to financial records concerning sales and costs of operations. Most importantly the Finance Manager will have exclusive access to matters concerning payments.

2.3. Operating:

2.3.1. Maintenance:

Maintenance (if required) must be done after regular hours and on weekends.

2.3.2. Backup:

The system must perform a nightly tape backup of the entire database. To save space, incremental daily backups can be used, but a full backup must be performed at least once per week. The system shall have an Uninterruptible Power Supply that provides no less than 45 minutes of uptime for the server and clients in the event of a power outage.

2.3.3. Restart Requirements:

In the event of an operating system or database error, the server will disregard all current uncompleted transactions and will restart itself. The server will be back online within five minutes. In the event of a prolonged power outage, the server will start itself upon the restoration of electricity.

2.3.4. Environmental Conditions:

The system must be installed in a location with good ventilation, sub 23° Celsius temperatures, and low dust levels to minimize hardware failure.

2.4. Life Cycle:

2.4.1. Quality of Design:

The development software tools must be standardized, widely used and actively supported. The system will be designed in such a way that each component is modularized. If functionality is needed in the future, a module can be easily written to plug into the existing system with minimal modification and interruption to business. The program shall provide a life span of no less than ten years. Portability is an additional benefit but is not required in this case. It is only required that the client front-end runs optimally on the Windows platform. It is likely that the backend can be written to run on multiple platforms such as Windows and Unix. If this fits into the development schedule and will not hinder functionality or performance, this should be done.

2.4.2. Limits on Development:

Development must take no longer than one year. This includes the period of building the system to delivering and installing the system.

2.5. Economic:

Development costs of the system shall not exceed $85,000, and maintenance costs must remain below $5000 annually.

2.6. Server Platform:

2.6.1. Operating System:

The server will run Windows 2000 Server as its Operating System. The company already has a 10-Client license for this OS, and it is the most secure and reliable release of the Windows Server family to date, so it is a good choice.

2.6.2. Memory:

Memory is incredibly cheap at present, so there is no sense in being conservative. Thus, the server shall be equipped with 512MB, which will satisfy all the memory requirements of Windows 2000 Server and the RDBMS overhead. However, the system board chosen should support more than 1GB for future expandability.

2.6.3. Storage:

The server shall have 40GB of storage spread across two high-speed 40GB hard disk drives setup in a RAID 0+1 configuration. An inexpensive IDE RAID configuration allows the data to be mirrored across two 40GB drives, so that if one fails, it can easily be replaced without any disruption to service or data loss.

The server will also have a large Tape Backup to perform nightly backups just in case both hard drives should fail.

2.6.4. CPU:

The server shall use an 800MHz Pentium III CPU. Databases rely much more heavily on the Input/Output component of the system, so an 800MHz CPU is more than enough to handle the load. The system board should support dual processors, so that a second processor can be added in the future for expandability.

2.6.5. Peripherals:

The server will require two printers. One laser printer is required to print invoices, itineraries, and reports. A second specialized printer is required to print cheques. Fortunately, the company already possesses these printers for these tasks.

2.6.6. Network:

A network upgrade would have been required to supply the necessary bandwidth for a multi-user database application. Conveniently, the company has a modern network in place that was recently upgraded. It is based on a 100Mbps Ethernet network and is connected by high-speed switches.

2.7. Client Systems:

2.7.1. Operating System:

Each client will run Windows 2000 Professional since the company owns a 10-Client license when they purchased Windows 2000 Server. Windows 2000 Professional is a robust and very stable operating system, so it is ideal for this setting.

2.7.2. Memory:

Again, since memory is currently available for such a low cost, each machine shall be equipped with no less than 128MB. Windows 2000 Professional does not run well with any less, but runs very well with 128MB.

2.7.3. Storage:

Each client shall have at least 10GB hard disk drives. The storage space is only required for the operating system and installing any applications. 10GB are the smallest drives available for purchase and are very cheap.

2.7.4. CPU:

Each client shall use at least a Pentium II/Celeron class CPU of 500MHz or higher.

2.7.5. Network:

Each client requires a PCI 10/100Mbps network card (NIC) to connect to the company Ethernet network.