Design and implementation of car rental system

Fadi Fayez Abdel-Jaber
DESIGN AND IMPLEMENTATION OF CAR RENTAL SYSTEM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Fadi Fayez Abdel-Jaber
December 2001
DESIGN AND IMPLEMENTATION OF CAR RENTAL SYSTEM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Fadi Fayez Abdel-Jaber
December 2001

Approved by:

[Signatures]

Yasha Karant, Chair, Computer Science
George M. Georgiou
Kerstin Voigt
© 2001 Fadi Fayez Abdel-Jaber
ABSTRACT

When someone wants to rent a car, the customer will usually think twice about the company from which they want to rent. The decision will be based on factors such as good rates, quality and customer service. The service the company representative offers the client should be fast, clear and accurate. This goal cannot be achieved without an informative system that will enable the customer representative to answer the various questions the client might have. The system in any car rental office should provide functionality’s to the end users to perform a particular transaction, whether this transaction is renting a car, receiving a car or querying about a client. This is indeed crucial to the car rental business since the whole renting process will depend on the accuracy of the implemented system. Therefore, the more informative the system is, the more customer satisfaction the company will obtain.
ACKNOWLEDGMENTS

I would first of all like to thank God for giving me hope and health to finish this project. Also, I would like to thank the faculty of the Computer Science department for giving me the opportunity to pursue my Master’s degree at California State University, San Bernardino. Extreme thanks to my advisor Dr. Yasha Karant, my friend Paul Amaya and the two-committee members, Dr. George Georgiou and Dr. Kerstin Voigt for all their support and help throughout the project design and implementation. Finally, profound thanks to my wife who has always supported me and given me hope in life.
# TABLE OF CONTENTS

ABSTRACT ......................................................... iii
ACKNOWLEDGMENTS ........................................ iv
LIST OF TABLES ............................................... viii
LIST OF FIGURES ............................................. xi

CHAPTER ONE: INTRODUCTION

1.1 Introduction ........................................... 1
1.2 Motivation .............................................. 2
1.3 Scope .................................................. 3
1.4 How This System is Different from Other Rental Systems .......... 4
1.5 Organization of Chapters ............................... 5

CHAPTER TWO: SYSTEM DEFINITION AND CONCEPTUAL DESIGN ................. 7

2.1 Database Application Life Cycle ....................... 8
2.2 System Definition .................................... 9
2.3 Conceptual Design .................................... 11
   2.3.1 Conceptual Design Using Enhanced Entity Relation Model .......... 11
   2.3.2 Entities of the System ................................ 12
   2.3.3 Relationships of the System .......................... 14
   2.3.4 Constraints of the System ............................ 18
   2.3.5 Class Diagram of the System ........................... 20

CHAPTER THREE: DATABASE DESIGN

3.1 Goals of Database Design .............................. 22
3.2 Logical and Physical Database Design ............... 23
3.2.1 Dependencies of the System Attributes ................................................. 24
3.2.2 Normalization ................................................................................. 27
3.2.3 Normalized Enhanced Entity Relation Model ........................................ 40

CHAPTER FOUR: CHOICE OF DATABASE MANAGEMENT SYSTEM

4.1 Oracle Developer 6i Key Features ..................................................... 41
4.2 Project Builder .................................................................................. 43
4.3 Form Builder .................................................................................... 44
4.4 Report Builder .................................................................................. 44

CHAPTER FIVE: PROJECT FORMS AND REPORTS DESIGN AND IMPLEMENTATION

5.1 System Forms Navigation Diagram .................................................... 45
5.2 System Forms and Reports Design and Functionality ....................... 46
   5.2.1 Login Window .............................................................................. 46
   5.2.2 Welcome Window ......................................................................... 46
   5.2.3 Main Menu Window ..................................................................... 47
   5.2.4 Customer Choice Window .............................................................. 48
   5.2.5 Car Window ................................................................................ 56
   5.2.6 Rental Branches Window ............................................................... 57
   5.2.7 Rent Information Choice Window .................................................... 59
   5.2.8 New User Account Window ........................................................... 63
   5.2.9 Contract Sub Window ..................................................................... 65
5.3 Menu Module ..................................................................................... 69
   5.3.1 File Menu Module ....................................................................... 69
   5.3.2 Edit Menu Module ....................................................................... 70
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.3 Forms Menu Module</td>
<td>70</td>
</tr>
<tr>
<td>5.3.4 Exit Menu Model</td>
<td>71</td>
</tr>
<tr>
<td>5.3.5 Window Menu Module</td>
<td>71</td>
</tr>
<tr>
<td>5.4 Database Application Life Cycle after Design Process</td>
<td>71</td>
</tr>
<tr>
<td>5.4.1 Loading Data Conversion</td>
<td>71</td>
</tr>
<tr>
<td>5.4.2 Application Conversion</td>
<td>71</td>
</tr>
<tr>
<td>5.4.3 Testing and Validation</td>
<td>71</td>
</tr>
<tr>
<td>5.4.4 Operation</td>
<td>72</td>
</tr>
<tr>
<td>5.4.5 Monitoring and Maintenance</td>
<td>72</td>
</tr>
<tr>
<td>CHAPTER SIX: SECURITY AND TESTING</td>
<td></td>
</tr>
<tr>
<td>6.1 Levels of Security</td>
<td>73</td>
</tr>
<tr>
<td>6.2 Constraints</td>
<td>74</td>
</tr>
<tr>
<td>6.3 Assumptions and Dependencies</td>
<td>75</td>
</tr>
<tr>
<td>6.4 Loading Testing Table</td>
<td>75</td>
</tr>
<tr>
<td>6.5 Transaction Testing Table</td>
<td>76</td>
</tr>
<tr>
<td>6.6 Conclusion</td>
<td>76</td>
</tr>
<tr>
<td>APPENDIX A: SOME STRUCTURED QUERY LANGUAGE CODES OF THE SYSTEM</td>
<td>78</td>
</tr>
<tr>
<td>APPENDIX B: IMPLEMENTATION OF SYSTEM TABLES</td>
<td>92</td>
</tr>
<tr>
<td>APPENDIX C: SOME COMPUTER TERMS DEFINITION</td>
<td>97</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>99</td>
</tr>
</tbody>
</table>
**LIST OF TABLES**

<table>
<thead>
<tr>
<th>Table 3.1</th>
<th>Individual Customer Table Attributes</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3.2</td>
<td>Corporate Customer Table Attributes</td>
<td>25</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>Car Table Attributes</td>
<td>25</td>
</tr>
<tr>
<td>Table 3.4</td>
<td>Job Table Attributes</td>
<td>25</td>
</tr>
<tr>
<td>Table 3.5</td>
<td>License Table Attributes</td>
<td>26</td>
</tr>
<tr>
<td>Table 3.6</td>
<td>Rent Master Table Attributes</td>
<td>26</td>
</tr>
<tr>
<td>Table 3.7</td>
<td>Some of Rent Detail Table Attributes</td>
<td>26</td>
</tr>
<tr>
<td>Table 3.8</td>
<td>Rental Branches Table Attributes</td>
<td>27</td>
</tr>
<tr>
<td>Table 3.9</td>
<td>Rent Type Table Attributes</td>
<td>27</td>
</tr>
<tr>
<td>Table 3.10</td>
<td>Insurance Type Table Attributes</td>
<td>27</td>
</tr>
<tr>
<td>Table 3.11</td>
<td>Rental Fee Table Attributes</td>
<td>27</td>
</tr>
<tr>
<td>Table 3.12</td>
<td>Copy of Individual Customer Table Attributes</td>
<td>28</td>
</tr>
<tr>
<td>Table 3.13</td>
<td>Customer Output1 Table</td>
<td>29</td>
</tr>
<tr>
<td>Table 3.14</td>
<td>Customer Type Table Attributes</td>
<td>29</td>
</tr>
<tr>
<td>Table 3.15</td>
<td>Customer-Join-Customer-Type Table Attributes</td>
<td>30</td>
</tr>
<tr>
<td>Table 3.16</td>
<td>Customer Type Output1 Table Attributes</td>
<td>30</td>
</tr>
<tr>
<td>Table 3.17</td>
<td>Customer Output2 Table Attributes</td>
<td>30</td>
</tr>
<tr>
<td>Table 3.18</td>
<td>Customer Zip Code Table Attributes</td>
<td>31</td>
</tr>
<tr>
<td>Table 3.19</td>
<td>Customer-Join-Customer-Type Output1 Table Attributes</td>
<td>31</td>
</tr>
<tr>
<td>Table 3.20</td>
<td>Customer-Type Output1 Table Attributes</td>
<td>31</td>
</tr>
</tbody>
</table>
Table 3.21. Customer Final Output Table
Attributes ...................................... 31

Table 3.22. Corporate Customer Table Attributes .......... 32

Table 3.23. Corporate Customer Output1 Table
Attributes ........................................ 33

Table 3.24. Corporate Customer-Zip Code Table
Attributes ........................................ 33

Table 3.25. Corporate Customer Output2 Table
Attributes ........................................ 33

Table 3.26. Rental Fee Table Attributes ..................... 33

Table 3.27. Car Output1 Table Attributes .................... 34

Table 3.28. Car Flag (Status) Table Attributes ............. 34

Table 3.29. Job Table Attributes ........................ 34

Table 3.30. Job Output1 Table Attributes ................... 35

Table 3.31. Job-Address Table Attributes ................... 35

Table 3.32. Customer-Job Table Attributes ................... 35

Table 3.33. Copy of License Table Attributes ............... 36

Table 3.34. Copy of Rent Master Table Attributes .......... 36

Table 3.35. Rent Master Output1 Table Attributes .......... 37

Table 3.36. Copy of Rent Detail Table Attributes .......... 37

Table 3.37. Copy of Rent Type Output1 Table
Attributes ......................................... 37

Table 3.38. Copy of Rent Type Output2 Table
Attributes ......................................... 37

Table 3.39. Copy of Rental Branches Table
Attributes ......................................... 38

Table 3.40. Rental Branches Output1 Table
Attributes ......................................... 38
Table 3.41. Rental Branches Join Phone Table
Attributes ........................................ 38

Table 3.42. Rental Branches Output2 Table
Attributes ........................................ 38

Table 3.43 Rental Branches-Zip Code Table
Attributes ........................................ 39
LIST OF FIGURES

Figure 2.1. Enhanced Entity Relation Model of the System ........................................ 13
Figure 2.2. Primary Key Representation .......................... 19
Figure 2.3. Class Diagram of the System ......................... 20
Figure 3.1. Normalized Entity Relation Model .......... 40
Figure 5.1. Forms Navigation Diagram ...................... 45
Figure 5.2. Login Window ........................................ 46
Figure 5.3. Welcome Window ................................... 47
Figure 5.4. Main Menu Window ................................. 48
Figure 5.5. Individual Customer Choice Window ........ 49
Figure 5.6. Zip Code Tab Page Window ................. 52
Figure 5.7. Customer Tab Page Window ................. 53
Figure 5.8. Job and License Tab page Window .... 54
Figure 5.9. Company Tab Page Window ................ 56
Figure 5.10. Car Window ........................................ 57
Figure 5.11. Rental Branches Window ..................... 58
Figure 5.12. Rent Choice Window ......................... 59
Figure 5.13. Customer Choice Window .................. 60
Figure 5.14. Individual Customer Rent a Car Window .......................... 61
Figure 5.15. Corporate Customer Rent A Car Window .... 62
Figure 5.16. Receiving a Car Window ..................... 63
Figure 5.17. New User Account Window ............... 65
Figure 5.18. Contract Choice Window ................. 66
Figure 5.19. Corporate Parameter Window ............. 66
Figure 5.20. Corporate Contract Report Window ........... 67
Figure 5.21. Individual Parameter Contract Window ....... 68
Figure 5.22. Individual Contract Report Window .......... 69
Figure 6.1. Loading Records Table ......................... 76
Figure 6.2. Transactions Time Table ......................... 76
CHAPTER ONE
INTRODUCTION

1.1 Introduction

Day by day the car rental process becomes increasingly complicated due to the needs and requirements of various consumers as well as the new marketing strategies to attract new clients. Such strategies can include advertisement of discounts to clients and promotional campaigns such as free cellular phones when renting a car. Therefore, any car rental system ought to have the capacity to incorporate all the varying elements into an informative business tool that stresses customer service.

This project consists of many forms and reports that been designed to be easy to use and understandable to the end users. The end users for this project could be anyone that can read and write English. Also he/she should have the basic training on how to use personal computers. The forms have been designed using Oracle (Developer 6i) under MICROSOFT Windows platform. Each field or item in any form has a hint property to guide the end users to their suitable choice. These hints will pop up when the cursor
is in any desired item. The reports for this project have been designed using Oracle (Report Builder 6i).

There are goals of implementing this project. One of these goals is to deliver a graphical user interface system to the end users since most market systems are character-based systems. Another goal is to deliver a secured system that will protect client's information from unauthorized users.

1.2 Motivation

The motivation for this project came from a series of unmet needs that most of the car rental systems in the market are not addressing. One of them is the need for a graphical user interface system. Most of the current application systems are character-based only, such as Pioneer system [10], the end users will face difficulties because these systems are usually not user-friendly and require more time and effort to accomplish specific tasks.

Another motivation is the implementation of a database application system with a short response time in terms of transactions. Furthermore, the application should include a good security system that will prevent unauthorized users from accessing into the client's information database. Some of the current car rental
systems have many problems in regards to security and response time such as EasyRent system [8] which most transactions takes more time than usual to be performed. Common transactions such as delete, update or query take a long time to be performed, forcing end users and customers to waste much time. This system has been designed for future enhancements.

1.3 Scope

This system will provide the following functionality

1. Increase customer service efficiency because of the short response time that a transaction may last.

2. Present the end user a with graphical user interface system that will make the rental process more flexible.

3. Give the end user the ability to retrieve records from the database in the short time possible.

4. Provide good security on information.

5. Provide the end users with an informative and quick navigation system through the forms and the reports.
1.4 How This System is Different from Other Rental Systems

1. The ability to handle huge numbers of records that other systems could not handle. For example, EasyRent [8] system could not handle large numbers of records because the database language (MS Access) that the system uses cannot handle large numbers of records.


3. Windows-based system, providing flexibility and ease of use to the end users. End users will face some difficulties learning character-based systems since most of these systems platforms are not user-friendly. An example of that Pioneer system [10] which is UNIX based system.

4. The front end and the engine (back end) of this system are designed and implemented using only Oracle. This approach differs from traditional market applications in which different front end and back end (engine) are used, such as EasyRent [8] system which uses Microsoft (MS Access) as a back end and front end user interface.

5. High speed of transaction and response time.
6. Easy to enhance and to Re-engineer.

1.5 Organization of Chapters

Chapter Two introduces the first two steps of database application life cycle, which are the conceptual design and the system definitions. The first part of the chapter talks about system definitions and how the database designers will determine the requirements of the system and collect the data. The second part of chapter 2 explains in detail the conceptual design process using Enhanced Entity Relation model (EER). It mentions most of the conceptual design process components such as entities, attributes, relationships and constraints.

The logical and physical design of any database system is one of the most important roles in the database application life cycle. Chapter Three will begin by describing the goals of the application design process. Then, it will explain in detail the logical and physical design of this system. The explanation of the design process will include the functional dependencies of the system attributes, normalization of the tables, the physical design, the implementation of the systems tables and other components.
The choice of Database Management System (DBMS) is another step of the database application life cycle. Chapter Four explains the factors in choosing the suitable DBMS. This chapter will also introduce the definitions of Oracle Developer6i, Report Builder and Project Builder which all of them are important tools that have been used in this project.

Chapter Five describes the actual design of the project forms and reports. This chapter explains in detail the functionality of each form and report of this system. It shows the user how to navigate throughout the project forms, beginning from the welcome window and ending with the exit option in the main menu form.

Chapter Six describes the levels of user categories and the security for the system. It also describes the limitations, constraints and further enhancements for this system.
CHAPTER TWO
SYSTEM DEFINITION AND
CONCEPTUAL DESIGN

The first step of the application life cycle is the requirement collection and analysis of any system. During this step, the database designers interview prospective database users to understand and document their requirements. Once all requirements have been collected and analyzed, the next step is to create a conceptual design for the database, using a high-level conceptual design model.

The conceptual schema is a concise description of data requirements of the users and includes detailed descriptions of entity types, relationships and constraints; these are expressed using the concepts provided by the high-level data model. Since these concepts do not include implementation details, they are usually easier to understand and can be used to communicate with non-technical users. The high level conceptual schema can also be used as a reference to ensure that all user data requirements are met and that the requirements do not include conflicts.
2.1 Database Application
Life Cycle

1. System Definition: The scope of the database system, its users, and its applications are defined. The interfaces for various categories of users, the response time constraints, storage and processing needs are identified.

2. Conceptual Design.

3. Database Design: At the end of this phase, complete logical and physical design of the database system on the chosen DBMS is ready.

4. Database Implementation: This comprises the process of specifying the conceptual, external, and internal database definitions, creating empty database files, and implementing the software applications.

5. Loading or Data Conversion: The database is populated either by loading the data directly or by converting existing files into the database system format.

6. Application Conversion: Any software applications from previous systems are converted to the new system.
7. Testing and Validation: The new system is tested and validated.

8. Operation: The database system and its applications are put into operation.

9. Monitoring and Maintenance: During the operational phase, the system is constantly monitored and maintained.

2.2 System Definition

The main goal of this project is to deliver the most flexible and easy-to-use car rental system to the end users, in order to make the rental process fast, easy and accurate for users as well as for the clients. There are four user categories in this system. These categories may change depending on the type of operation and management.

1. Database Administrators user account: This account has full permission to all forms, tables and reports in the system. Also, this type of user account has full permission to perform all kind of transactions, such as delete, insert, update and query.

2. Managers and Supervisors user account: This type of user account has access to most of the forms in the system except to 'Create User Account
Form'. Also this user account has full permission to perform all kind of transactions, such as delete, insert, update and query.

3. Data Entry users account: This type of user account has access to most of the forms except to 'Create User Account Form' and to 'Rental Branches Form'. This user account has limited access to perform transactions such as delete, insert, update and query. Limited access means, a restriction to perform specific transactions.

4. Receptionists and Temporary Employees user account: This type of user account is limited from performing anything except query only transactions in all the forms. Just like the Data Entry users, this account has limited access to perform transactions such as delete, insert, update and query. Also this user account has limited access to some forms and reports such as 'Rental Branches Form' and to 'Create User Account Form'.

The response time of any system will depend on many factors. The first factor is the interface between the database storage (tables, views, sequences) and the front-end application (forms, reports). In this system,
the response time to retrieve information from the back end tables to the front-end application (forms, reports) is fast, leaving no time constraints. Another factor that will effect the response time is the type of transaction to be performed by the end user. The response time in this system is minimal for all kinds of transactions. Another factor that will effect the response time is the network traffic in the system. The more elevated network traffic is the longer the response time.

2.3 Conceptual Design

The conceptual design is a graph representation of the entities, relationships and constraints of the database applications. This step will occur after the system definition step finishes. During or after the conceptual design, the basic data model operations can be used to specify the high-level user operations identified during the functional analysis. This also serves to confirm that the conceptual schema meets all the identified functional requirements.

2.3.1 Conceptual Design Using Enhanced Entity Relation Model

The Enhanced Entity Relation Model (EER) is a popular high-level conceptual data model. This model and its variations are frequently used for the conceptual design
of database applications, and many database design tools employ its concepts. The EER describes data as entities, relationships, and attributes. The definitions of entities, attributes and relationships will be explained later in this chapter. The EER for this system is represented in figure 2.1.

2.3.2 Entities of the System

The basic object that EER represents is an entity, which is something in the real world with an independent existence. An entity may be an object with a physical existence, such as a particular house, tree, animal and person or it may be an object with conceptual existence, such as university, company, name or job. Each entity has attributes describing its particular properties. For example a customer name, an address, telephone, age and others may describe a customer entity.

Some of the system entities as described below:

**Customer Entity:** This entity has many attributes such as social security number, customer name, date of birth, address, phone number and others.

**Customer Type Entity:** This entity has two attributes, which are customer type and social security number.
Figure 2.1. Enhanced Entity Relation Model of the System
Car Entity: This entity has many attributes such as plate number, make, model, year, color, status and others.

Rental Branches Entity: This entity has many attributes such as branch number, location, zip code, State and others.

There are other entities in the system described in figure 2.1 such as job, license, insurance type, and rent master.

2.3.3 Relationships of the System

1. (1:1): - one-to-one relationship is not a commonly used relationship between tables. The solution of such relationship is obtained by referencing one of the tables to the other by placing its primary key in the other table. It happens that this system has only one relation (one-to-one) as described below:

   a) Customer and license: A customer could have only one license and one license per customer. To solve this relationship, table license will have social security number (SSN) as
foreign key referencing (SSN) in table customer.

2. \(1:N\): One-to-Many relationship is the most common relationship between tables in relational database. The best way to solve such a relationship is by placing the primary key of the \(1\) table in the \(N\) table. Thus the copy of the primary key of the \(1\) table will be a foreign key in the \(N\) table.

The Car Rental system has a lot of those One-to-Many relationships as described below:

a) Customer (person, corporate), rent master: One customer could have one or more rent transactions. On the other hand, one transaction could not include more than one customer. Therefore one customer could rent one or more cars, but one car cannot be rented to more than one customer.

b) Rental branches, rent master: More than one rental transaction could occur in a rental location, but a specific rental transaction could not
occur in multiple or more rental locations.

c) Car, rent detail: One rental transaction could include one or more cars, but one car could not be in two rental transactions. In other words, it is impossible to rent a particular car more than one time at the same moment to different customers.

d) Rent detail, rent type: A rent type (daily, weekly, and monthly) could be given to more than one car. In other words, a particular car in rent transaction could have only one rent type.

e) Insurance type, rent detail: A particular car in rent transaction could use only one insurance type at a time. In other words, a particular insurance (8.99/day) could be given to one or more cars.

f) Car, car status: A car status (rented, idle, in-garage) could be given to more than one car. Therefore, a car
could have only one status, but one or more cars could have the same status.
g) Rent master, rent detail: One rent could have many details. In other words, one rent master could have more than one rental car in its detailed information.

3. (N:M) - A Many-to-Many relationship is another common relationship between tables. Removing the two primary keys of the tables that have (N,M) and placing them in new table can solve this type of relationship. The attributes of the new table will be the primary key for the new table. Those attributes will reference to the original two tables. This system has some (N:M) relationships as follow:
a) Customer, customer type: One customer could have one or more types, such a customer type is senior, garage referenced or insurance referenced, etc. Also, a particular type could be given to one or more customers. An assumption made for this particular
relation schema (customer, customer type) is to choose the maximum discount for the customer from his/her types. Each customer has one or more types. Each type has a discount (ex: senior, 5%). Therefore when renting a car, a customer may have more than one type, and each type has a specific discount along with it; consequently the system will choose the maximum discount among the types for the customer.

b) Car, rental fee: One or more cars could have the same rental fee. Also, one car could have one or more rental fees, depending on the season.

c) Customer, job: One or more customers could have the same job with the same company or one customer could have one or more jobs.

2.3.4 Constraints of the System

Relationship types usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set. These
constraints are determined by the actual situation that the relationships represent. There are four types of constraint. The first type of constraint is a primary key, which has been represented in figure 2.2 by an attribute that has a line beneath it. The primary key is the attribute that has distinct values for each individual entity in the collection. For example social security number is a primary key in entity customer because it's impossible to find two persons with the same Social Security Number.

Figure 2.2. Primary Key Representation

The second type of constraint is a foreign key, which will be used to reference tables to each other and has the same definition of the primary key. The third type of constraint is a check key. Check key will be used to validate the value of an attribute between ranges of values. An example of that is the attribute 'date of birth' in customer form. This attribute will check whether the customer age is between 25 and 68 years. The last type of constraint is not null. This type of constraint will prevent an attribute from having the value of null and
will enforce the end user to give a valid value to that attribute.

2.3.5 Class Diagram of the System

Figure 2.3. Class Diagram of the System
2.3.5.1 Class Diagram Components. Class Diagram is a graphical representation that describes objects and their relationships. Figure 2.3 represents the class diagram of the Car Rental System. The diagram contains the classes and their attributes. Also it contains the associations between classes and attributes and the candidate keys of each class in the system.

A class is a description of group of objects with similar properties (object attributes). An example of a class is car, customer, rent master and others. An object is an instance or occurrence of a class. An example of an object is type, license number, social security number and others. There are relationships between classes and such a relationship called association. An association is a description of group of links with common structure and common semantics. Just like in relationships in EER, an association has different multiplicity, Such as exactly one or many or (zero or one) and others.
CHAPTER THREE
DATABASE DESIGN

3.1 Goals of Database Design

The goals of database design are multiple:

♦ Satisfy the information content requirements of the specified users and applications.
♦ Provide a natural and easy-to-understand structuring of the information.
♦ Support processing requirements and any performance objectives such as response time, processing time and storage space.

Four phases of the database design process can be identified:

1. Requirement collection and analysis.
2. Conceptual database design.
3. Logical database design.
4. Physical database design.

The requirement collection and conceptual design have already been explained in detail in the last chapter. The next phase of database design is to create the logical design for the data model of that selected Database Management System by mapping those schemas produced in
phase 2 (conceptual design). The mapping can proceed in two stages:

1. System-independent mapping.
2. Tailoring the schemas to specific DBMS.

The result of this phase should be Data Definition Language (DDL) statements in the language of the chosen DBMS that specify the conceptual and external level schemas of the database system. Nevertheless, if the DDL statements include some physical design parameters, complete DDL specifications must wait until after the physical database design phase is completed.

3.2 Logical and Physical Database Design

Logical design is the actual implementation of the database, using a commercial DBMS. Most current commercial DBMS use an implementation data model, such as the relational or the object database models; the conceptual schema is transformed from the high-level data model into the implementation data model. The result from this logical design is a database schema in the implementation database model of the DBMS.

The physical database design is the process of choosing specific storage structures and accessing paths for the database files to achieve good performance for the
various database applications. Once a specific DBMS is chosen, the physical database design process is restricted to choosing the most appropriate structures for the database files from among the options offered by DBMS.

3.2.1 Dependencies of the System Attributes

A Functional Dependency is a property of the relation schema R, not of a particular legal relation state (extension) r of R. Hence, a Functional Dependency cannot be inferred automatically from a given relation extension r but must be defined explicitly by someone who knows the semantics of the attributes of R. Thus basically a Functional Dependency occurs when an attribute is depending on other attribute in a relation. The system attributes have the following functional Dependencies:

Table 3.1. Individual Customer Table Attributes

<table>
<thead>
<tr>
<th>SSN</th>
<th>Fname</th>
<th>Ml</th>
<th>Lname</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip code</th>
<th>DOB</th>
<th>Phone</th>
<th>Type</th>
</tr>
</thead>
</table>

SSN —→ Fname
SSN —→ Lname
SSN —→ Ml
SSN —→ Address
SSN —→ City
SSN —→ State
SSN —→ Zip code
SSN —→ DOB
SSN —→ Phone
SSN —→ Type
Zip code —→ City
Zip code —→ State
Table 3.2. Corporate Customer Table Attributes

<table>
<thead>
<tr>
<th>Co No</th>
<th>Co name</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip code</th>
<th>Contact Person</th>
<th>Contact Person Phone</th>
</tr>
</thead>
</table>

- CompanyNo ➝ Co name
- CompanyNo ➝ Address
- CompanyNo ➝ City
- CompanyNo ➝ State
- CompanyNo ➝ Zip code
- CompanyNo ➝ Contact Person
- CompanyNo ➝ Contact Person Phone
- Zip code ➝ City
- Zip code ➝ Co name

Table 3.3. Car Table Attributes

<table>
<thead>
<tr>
<th>Plate No</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Summary</th>
<th>Flag</th>
<th>Rental Fee</th>
</tr>
</thead>
</table>

- PlateNo ➝ Make
- PlateNo ➝ Model
- PlateNo ➝ Year
- PlateNo ➝ Color
- PlateNo ➝ Summary
- PlateNo ➝ Flag
- PlateNo ➝ Rental Fee

Table 3.4. Job Table Attributes

<table>
<thead>
<tr>
<th>Co name</th>
<th>Job Title</th>
<th>Phone No</th>
<th>Co Address: Street No, City, State, Zip code</th>
<th>SSN</th>
</tr>
</thead>
</table>

- Co name ➝ Job Title
- Co name ➝ Phone No
- Co name ➝ Co Address
Table 3.5. License Table Attributes

<table>
<thead>
<tr>
<th>LicenseNo</th>
<th>Type</th>
<th>Issue State</th>
<th>Expiration Date</th>
<th>SSN</th>
</tr>
</thead>
</table>

LicenseNo $\rightarrow$ Type
LicenseNo $\rightarrow$ Issue State
LicenseNo $\rightarrow$ Expiration Date

Table 3.6. Rent Master Table Attributes

<table>
<thead>
<tr>
<th>RentNo</th>
<th>SSN</th>
<th>Rent Date</th>
<th>Card No</th>
<th>Card Type</th>
<th>Card Expiration</th>
<th>PlateNo</th>
<th>Rent Branch No</th>
</tr>
</thead>
</table>

RentNo $\rightarrow$ SSN
RentNo $\rightarrow$ Rent Date
RentNo $\rightarrow$ Card No
RentNo $\rightarrow$ Card Type
RentNo $\rightarrow$ Card Expiration
RentNo $\rightarrow$ Rental Branch No.

Table 3.7. Some of Rent Detail Table Attributes

<table>
<thead>
<tr>
<th>RentNo</th>
<th>PlateNo</th>
<th>BranchNo</th>
<th>MileOut</th>
<th>MileIn</th>
<th>GasOut</th>
<th>GasIn</th>
<th>Ins. Type</th>
<th>Rent Type</th>
<th>Mgr Disc.</th>
<th>Other Charges</th>
<th>Total</th>
</tr>
</thead>
</table>

RentNo $\rightarrow$ PlateNo
RentNo $\rightarrow$ Rental Branch No.
RentNo $\rightarrow$ Mile Out
RentNo $\rightarrow$ Mile In
RentNo $\rightarrow$ Gas Out
RentNo $\rightarrow$ Gas Price/Gallon
RentNo $\rightarrow$ Gas In
RentNo $\rightarrow$ Insurance Type
RentNo $\rightarrow$ Rent Type
RentNo $\rightarrow$ Mgr Discount
RentNo $\rightarrow$ Other Charges
RentNo $\rightarrow$ Total
RentNo $\rightarrow$ Total Paid
RentNo $\rightarrow$ Balance
PlateNo $\rightarrow$ Rent_no
PlateNo $\rightarrow$ Rental Branch No.
PlateNo $\rightarrow$ Mile Out
PlateNo $\rightarrow$ Mile In
PlateNo $\rightarrow$ Gas Out
PlateNo $\rightarrow$ Gas Price/Gallon
PlateNo $\rightarrow$ Gas In
PlateNo $\rightarrow$ Insurance Type
PlateNo $\rightarrow$ Rent Type
PlateNo $\rightarrow$ Mgr Discount
PlateNo $\rightarrow$ Other Charges
PlateNo $\rightarrow$ Total
PlateNo $\rightarrow$ Total Paid
### Table 3.8. Rental Branches Table Attributes

<table>
<thead>
<tr>
<th>BranchNo</th>
<th>BName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip code</th>
<th>Tax</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>BranchNo</td>
<td>BName</td>
<td>Address</td>
<td>City</td>
<td>State</td>
<td>Zip code</td>
<td>Tax</td>
<td>Phone</td>
</tr>
<tr>
<td>BranchNo</td>
<td>Address</td>
<td>City</td>
<td>State</td>
<td>Zip code</td>
<td>Tax</td>
<td>Phone</td>
<td></td>
</tr>
<tr>
<td>BranchNo</td>
<td>City</td>
<td>State</td>
<td>Zip code</td>
<td>Tax</td>
<td>Phone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BranchNo</td>
<td>State</td>
<td>Zip code</td>
<td>Tax</td>
<td>Phone</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BranchNo</td>
<td>Zip code</td>
<td>Tax</td>
<td>Phone</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BranchNo</td>
<td>Tax</td>
<td>Phone</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BranchNo</td>
<td>Phone</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zip code</td>
<td>City</td>
<td>State</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zip code</td>
<td>City</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zip code</td>
<td>State</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 3.9. Rent Type Table Attributes

<table>
<thead>
<tr>
<th>Rent Type</th>
<th>Rent Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rent Type</td>
<td>Rent Description</td>
</tr>
</tbody>
</table>

### Table 3.10. Insurance Type Table Attributes

<table>
<thead>
<tr>
<th>Ins.Type</th>
<th>Ins Daily Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ins.Type</td>
<td>Ins Daily Price</td>
</tr>
</tbody>
</table>

### Table 3.11. Rental Fee Table Attributes

<table>
<thead>
<tr>
<th>Plate No</th>
<th>Rental Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plate No</td>
<td>Rental Fee</td>
</tr>
<tr>
<td>Rental Fee</td>
<td>Plate No</td>
</tr>
</tbody>
</table>

#### 3.2.2 Normalization

Normalization is the process of taking relation schema through a number of steps to clarify whether or not that schema satisfies a certain normal form. There are five normal forms (one through five) and an advanced third
normal form called Boyce-Code normal form. All normal forms are based on the Functional Dependencies among the attributes of a relation except the fourth and the fifth normal forms which they depend on multi-valued dependencies and join dependencies respectively.

First Normal Form was defined to disallow multi-valued attributes, composite attributes, and their combinations. Any relation is in Second Normal Form if every non-key attribute is fully functionally dependent on the key attributes. A relation is in Third Normal Form if there is no transitive relation among attributes. In other words, an attribute should not depend on any non-key attribute.

Usually a database designer or analyst will derive to third normal form to simplify the design. Also, normalization depends on the complexity of the relations in the database application. Two reasons for applying the normalization process on a relational database are to reduce the redundancy of data and to attempt to achieve the most reliable format of tables and relationships.

3.2.2.1 Relation (1): Customer.

Table 3.12. Copy of Individual Customer Table Attributes

<table>
<thead>
<tr>
<th>SSN</th>
<th>Fname</th>
<th>MI</th>
<th>LName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip code</th>
<th>DOB</th>
<th>Phone</th>
<th>Type</th>
</tr>
</thead>
</table>

28
In this system, there are two kinds of customers: individual customers and corporate customers. The individual customer table is shown in table 3.12.

Table 3.12 is not in 1NF because of the multi-value attribute (customer type). A customer could have more than one type. One customer could be a senior customer and could be a golden club member, or one customer could be a senior and be referred by an insurance company to the car rental branch. Considering this, the above relation schema is not in 1NF because of the multi-value field (customer type). The solution to the above relationship will be to remove the multi-value field along with the primary key (SSN) and place them in a new table (table 3.14). The new relation will be as follows:

Table 3.13. Customer Output1 Table

<table>
<thead>
<tr>
<th>SSN</th>
<th>Fname</th>
<th>Mi</th>
<th>LName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip code</th>
<th>Dob</th>
<th>Phone</th>
</tr>
</thead>
</table>

Table 3.14. Customer Type Table Attributes

<table>
<thead>
<tr>
<th>Cust Type</th>
<th>SSN</th>
<th>Cust Description</th>
</tr>
</thead>
</table>

By reading the new relation schema table 3.14, there is still a relationship problem (Many-to-Many) between customer and customer type. In other words, more than one customer could have the same type and a customer could
have more than one type. This problem is not a normalization-related problem. It is a relationship-related issue, and it is good to mention it and solve it in this section. To solve this relationship, there is a need to create a new table that will contain a copy of the primary keys from both tables (customer, customer type). It will be called Customer-Join-Customer-Type. The new relation will be as follows:

Table 3.15. Customer-Join-Customer-Type Table Attributes

<table>
<thead>
<tr>
<th>Cust Type</th>
<th>SSN</th>
</tr>
</thead>
</table>

Table 3.16. Customer Type Output1 Table Attributes

<table>
<thead>
<tr>
<th>Cust Type</th>
<th>Cust_Description</th>
</tr>
</thead>
</table>

Table 3.17. Customer Output2 Table Attributes

<table>
<thead>
<tr>
<th>SSN</th>
<th>Fname</th>
<th>Mi</th>
<th>LName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip code</th>
<th>Dob</th>
<th>Phone</th>
</tr>
</thead>
</table>

The above relation (table 3.17) is in 2ndNF. All the non-key attributes are functionally dependent on the primary key (SSN).

Table 3.17 is not in 3rdNF because of the transitive relations between zip code, city, state as follow:

SSN → zip code and zip code → city, state
Zip code is neither a candidate key nor part of the primary key, and there are two attributes that depend on it, which are city and state. The solution of the above relation will be by removing the violated attributes (zip code, city and state) and place them in a new table as shown in table 3.18. It will be called customer-zip code table. There will also a copy of zip code in table customer for referencing between the two tables (customer, customer zip code). The new relation will look like:

Table 3.18. Customer Zip Code Table Attributes

<table>
<thead>
<tr>
<th>Zip code</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
</table>

Table 3.19. Customer-Join-Customer-Type Output1 Table Attributes

<table>
<thead>
<tr>
<th>Cust Type</th>
<th>SSN</th>
</tr>
</thead>
</table>

Table 3.20. Customer-Type Output1 Table Attributes

<table>
<thead>
<tr>
<th>Cust Type</th>
<th>Cust_Description</th>
</tr>
</thead>
</table>

Table 3.21. Customer Final Output Table Attributes

<table>
<thead>
<tr>
<th>SSN</th>
<th>F.Name</th>
<th>MI</th>
<th>L.Name</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip code</th>
<th>Dob</th>
<th>Phone</th>
</tr>
</thead>
</table>
3.2.2.2 Relation (2): Company.

Table 3.22. Corporate Customer Table Attributes

<table>
<thead>
<tr>
<th>Company No.</th>
<th>Name</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip code</th>
<th>Contact Person</th>
<th>Contacted Person</th>
<th>Phone</th>
</tr>
</thead>
</table>

The attribute Company number in the relation schema (table 3.22) is a counter that is referenced to a sequence created in SQL*Plus. The end user will not be asked to insert a company number. Instead, the system will automatically generate a sequence number for a company in each inserted transaction.

The relation schema (table 3.22) is in 1NF because there are no multi-value attributes and table 3.22 is in 2ndNF because all non-key attributes are fully functionally depends on the primary key (company number).

Table 3.22 is not in 3NF because of the transitive relation between zip code, city, and states.

Co. no \(\rightarrow\) zip code and \(\rightarrow\) city, state

Zip code is neither a candidate key nor part of the primary key, and there are two attributes that depend on it, which are city and state. The solution of the above relation (table 3.22) will be by removing the violated attributes (zip code, city, and state) and place them in new table (table 3-22). There will also be a copy of zip
code in table company for referencing purposes. The new relation will look like:

Table 3.23. Corporate Customer Output1 Table Attributes

<table>
<thead>
<tr>
<th>Company.No.</th>
<th>Name</th>
<th>Address</th>
<th>Zip code</th>
<th>Contact Person</th>
<th>Contacted Person</th>
<th>Phone</th>
</tr>
</thead>
</table>

Table 3.24. Corporate Customer-Zip Code Table Attributes

<table>
<thead>
<tr>
<th>Zip code</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
</table>

3.2.2.3 Relation (3): Car.

Table 3.25. Corporate Customer Output2 Table Attributes

<table>
<thead>
<tr>
<th>Plate No</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Summary</th>
<th>Rental Fee</th>
</tr>
</thead>
</table>

The above relation (table 3.25) is not in 1NF. The reason for that is a car may have more than one rental fee, depending on the season. Therefore, a copy of the primary key (Plate No) along with the rental fee attribute will be placed in new table (table 3.26) as follows:

Table 3.26. Rental Fee Table Attributes

<table>
<thead>
<tr>
<th>Plate No</th>
<th>Rental Fee</th>
<th>Season</th>
</tr>
</thead>
</table>

33
Table 3.27. Car Output Table Attributes

<table>
<thead>
<tr>
<th>Plate No</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Flag</th>
<th>Summary</th>
</tr>
</thead>
</table>

The above relation (table 3.27) is in 2ndNF. All non-key attributes are fully functionally dependent on the primary key (plate no.). Also the above relation is in 3rdNF because there are no transitive relations between the attributes.

Table 3.28. Car Flag (Status) Table Attributes

<table>
<thead>
<tr>
<th>Flag No</th>
<th>Description</th>
</tr>
</thead>
</table>

The purpose for flag number in the above relation (table 3.28) is to determine whether the status of the car is (‘0’ idle ‘1’ rented, ‘2’ in the garage). Since there are no multi-value attributes, the above relation is in 1NF. The above relation is in 2ndNF since no partial dependency appears. Also, the above relation is in 3rdNF because there is no transitive relation among attributes.

3.2.2.4 Relation (4): Job.

Table 3.29. Job Table Attributes

<table>
<thead>
<tr>
<th>CoName</th>
<th>Job Title</th>
<th>Phone No</th>
<th>Co Address: Street No, City, State, Zip code</th>
</tr>
</thead>
</table>
The above relation (table 3.29) is not in 1NF because of the composite attribute (address). Therefore, the above relation (table 3.29) needs to separate into two tables as shown below:

Table 3.30. Job Output Table Attributes

<table>
<thead>
<tr>
<th>CoName</th>
<th>Job Title</th>
<th>Phone No</th>
</tr>
</thead>
</table>

Table 3.31. Job-Address Table Attributes

<table>
<thead>
<tr>
<th>CoName</th>
<th>Address</th>
</tr>
</thead>
</table>

It is assumed that every customer will have only one home phone number in our system. All non-keys attributes are fully functionally dependent on the primary key (co. name). Consequently, the above relation schema is in 2ndNF. There is no transitive dependency on the above relation, so it is also in 3rdNF.

3.2.2.5 Relation (5): Customer-Job.

Table 3.32. Customer-Job Table Attributes

<table>
<thead>
<tr>
<th>SSN</th>
<th>CoName</th>
</tr>
</thead>
</table>

The attribute social security number in table 3.32 references to table customer and the attribute company
name references to table job. The above relation (table 3.32) is in 1NF, 2ndNF and 3rdNF.

3.2.2.6 Relation (6): License.

Table 3.33. Copy of License Table Attributes

<table>
<thead>
<tr>
<th>LicenseNo</th>
<th>Type</th>
<th>Issue State</th>
<th>Expiration Date</th>
<th>SSN</th>
</tr>
</thead>
</table>

The above relation (table 3.33) is in 1NF because there are no multi-value attributes. There is no partial dependency in the above relation schema. In other words, all non-keys attributes are fully functionally dependent on the primary key (co. name). Consequently, the above relation schema (table 3.33) is in 2ndNF. There is no transitive dependency on the above relation, so it is also in 3rdNF.

3.2.2.7 Relation (7): Rent Master.

Table 3.34. Copy of Rent Master Table Attributes

<table>
<thead>
<tr>
<th>RentNo</th>
<th>SSN</th>
<th>Rent Date</th>
<th>CardNo</th>
<th>Card Type</th>
<th>Card Expiration</th>
<th>PlateNo</th>
<th>Rent Branch No</th>
</tr>
</thead>
</table>

The above relation (table 3.34) is in 1NF because there are no multi-value or composite attributes in it. Table 3.34 is in 2ndNF because there is no partial dependency. All the non-key attributes are fully functionally dependent on the primary key (rent_no).
Also, Table 3.34 is in 3rdNF. There is a One-to-Many relationship in Table 3.34; there could be more than one car in a rental transaction. That will lead to a master-detail relationship as follows:

Table 3.35. Rent Master Output Table Attributes

<table>
<thead>
<tr>
<th>RentNo</th>
<th>SSN</th>
<th>RentDate</th>
<th>CardNo</th>
<th>Card Type</th>
<th>Card Expiration</th>
<th>RentBranch</th>
</tr>
</thead>
</table>

Table 3.36. Copy of Rent Detail Table Attributes

<table>
<thead>
<tr>
<th>RentNo</th>
<th>PlateNo</th>
<th>Distance</th>
<th>MileOut</th>
<th>MileIn</th>
<th>GasOut</th>
<th>GasPrice</th>
<th>GasIns</th>
<th>RentType</th>
<th>Discount</th>
<th>ToH</th>
<th>Balance</th>
</tr>
</thead>
</table>

In the above two relations, one rent could have one or more rent detail information. By constructing the above relations from the relation schema (Table 3.36), the one-to-many relationship between rent master and rent detail will be eliminated.

3.2.2.8 Relation (8): Rent Type - Insurance Type

Table 3.37. Copy of Rent Type Output 1 Table Attributes

<table>
<thead>
<tr>
<th>RentType</th>
<th>RentDescription</th>
</tr>
</thead>
</table>

Table 3.38. Copy of Rent Type Output 2 Table Attributes

<table>
<thead>
<tr>
<th>InsType</th>
<th>InsDailyPrice</th>
</tr>
</thead>
</table>

In the above two relations, one rent could have one or more rent detail information. By constructing the above relations from the relation schema (Table 3.36), the one-to-many relationship between rent master and rent detail will be eliminated.

Table 3.35. Rent Master Output Table Attributes

<table>
<thead>
<tr>
<th>RentNo</th>
<th>PlateNo</th>
<th>Distance</th>
<th>MileOut</th>
<th>MileIn</th>
<th>GasOut</th>
<th>GasPrice</th>
<th>GasIns</th>
<th>RentType</th>
<th>Discount</th>
<th>ToH</th>
<th>Balance</th>
</tr>
</thead>
</table>

Table 3.36. Copy of Rent Detail Table Attributes

<table>
<thead>
<tr>
<th>RentNo</th>
<th>PlateNo</th>
<th>Distance</th>
<th>MileOut</th>
<th>MileIn</th>
<th>GasOut</th>
<th>GasPrice</th>
<th>GasIns</th>
<th>RentType</th>
<th>Discount</th>
<th>ToH</th>
<th>Balance</th>
</tr>
</thead>
</table>

Also, Table 3.34 is in 3rdNF. There is a One-to-Many relationship in Table 3.34; there could be more than one vehicle in a rental transaction. That will lead to a master-detail relationship as follows:
The above two tables in relation (8) are in 1NF, 2ndNF and 3rdNF.

3.2.2.9 Relation (9): Rental Branches.

Table 3.39. Copy of Rental Branches Table Attributes

<table>
<thead>
<tr>
<th>BranchNo</th>
<th>Bname</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip code</th>
<th>Tax</th>
<th>Phone</th>
</tr>
</thead>
</table>

The above relation (table 3.39) is not in 1NF because of the multi-value attribute phone number. A branch could have more than one phone number therefore relation (9) needs to be normalized.

Table 3.40. Rental Branches Output1 Table Attributes

<table>
<thead>
<tr>
<th>BranchNo</th>
<th>Bname</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip code</th>
<th>Tax</th>
</tr>
</thead>
</table>

Table 3.41. Rental Branches Join Phone Table Attributes

<table>
<thead>
<tr>
<th>BranchNo</th>
<th>BranchPhone</th>
</tr>
</thead>
</table>

The above relation (table 3.41) is in 2ndNF but not in 3rdNF because of the transitive dependencies between Zip code, city, and state. Table 3.41 will be normalized to 3rdNF as follows:

Table 3.42. Rental Branches Output2 Table Attributes

<table>
<thead>
<tr>
<th>BranchNo</th>
<th>BName</th>
<th>Address</th>
<th>Zip code</th>
</tr>
</thead>
</table>
Table 3.43 Rental Branches-Zip Code Table Attributes

<table>
<thead>
<tr>
<th>Zip Code</th>
<th>City</th>
<th>State</th>
<th>Tax</th>
</tr>
</thead>
</table>
3.2.3 Normalized Enhanced Entity Relation Model

Figure 3.1. Normalized Entity Relation Model
CHAPTER FOUR
CHOICE OF DATABASE
MANAGEMENT SYSTEM

The choice of a DBMS is governed by a number of factors, some technical, others economic, and still others concerned with the politics of the organization. The technical factors are concerned with the suitability of the DBMS for the task at hand. Issues to consider here are the type of DBMS (relational, object-relational, object, other), the storage structures, the user and programmer interfaces available, the types of high-level query languages, the economic and organizational factors that affect the choice of DBMS, and others. The following costs must be considered:

1. Software Acquisition Cost.
3. Hardware Acquisition Cost.
4. Database creation and Conversion Cost.
5. Training Cost and others.

4.1 Oracle Developer 6i Key Features

1. Comprehensive GUI Support: Oracle Developer supports the native features of Microsoft
Windows 95 and Windows NT 4.0, and it provides portability to Macintosh and character-mode production environments.

2. Distributed Applications: The tools provide local client server and web support with multiple database connections per application. In addition to Oracle 7 and Oracle 8, the Oracle applications can access SQL databases through open database connectivity (ODBC).

3. Productive and comprehensive tools: Oracle Developer provides the same design facilities across tools for common functions such as the Layout Editor and the Object Navigator.

4. Application Partitioning: Individual PL/SQL program units can be placed on the database server, or in the client-side application, whichever is most suitable in each case. Objects between modules can move and copy the database server by using convenient drag-and-drop techniques.

5. Flexible Source Control: Definitions of your application modules can be stored in flat files or in Oracle database. Version control can be performed on these modules and produce
documentation by using Oracle Developer facilities.

6. Extended Scalability: Applications from single users to tens of thousands can be scaled with no changes to the application, since scalability is inherent in the multithread architecture of the product. There is support for server functionality, such as array DML, databases cursors, bind variables and result sets.

7. Object Orientation: Oracle Developer offers an inheritance model that facilitates the inheritance of attributes and code from one object to another and from one application to another, through subleasing and object libraries.

4.2 Project Builder

This component of release 6 helps in the entire project life cycle. The navigator-style interface provides easy access to all project files and to the action associated with them. The customizable launcher enables you to launch any application directly, such as Form Builder, Report Builder or Microsoft Word. The user can
package his/her application and then deploy it on other
machines by using utilities of his/her own choosing.

4.3 Form Builder

This development interface helps you to build
sophisticated, interactive applications and to pull down
menus. Form Builder can present information through
textual item, GUI objects and bitmapped images. Users can
perform database transactions by interacting with these
objects. Wizards help to build simple, standard modules
very quickly. One can then use the powerful features of
Form Builder to enhance the appearance and functionality
of the application.

4.4 Report Builder

This development interface helps you build both
simple and extremely complex production-quality reports.
The wizard offers a variety of styles. Templates help one
to develop professional and standard reports quickly and
easily.
CHAPTER FIVE

PROJECT FORMS AND REPORTS

DESIGN AND IMPLEMENTATION

PROJECT FORMS AND REPORTS

CHAPTER FIVE
5.2 System Forms and Reports
   Design and Functionality

5.2.1 Login Window

This window (figure 5.2) will ask the user to enter his/her user name and password to login to the system. The user must enter the correct user name and password. After three failed attempts the system will force the end user to exit out of the system.

![Login Window Diagram]

Figure 5.2. Login Window

5.2.2 Welcome Window

This welcome window (figure 5.3) contains a greeting message. It has a date field, which has the current date and user field that contains the current user of the system. This window also has a text that consists of university name, department of computer science, student name, advisor and committee names. This window has a timer.
that will last for 10 seconds. After the timer expires, the main menu window will pop up automatically.

![Welcome Window](image)

Figure 5.3. Welcome Window

### 5.2.3 Main Menu Window

The main menu (figure 5.4) will have seven push-button options. These options are customer, car, rent, rental branches, contract, administrator, and exit. When clicking on any option, another window or menu will pop up. At the top of the main menu there is a horizontal tool bar, which contains the current date and user name fields. There is a custom menu at the top of the screen of most windows in the system.
5.2.4 Customer Choice Window

When the user chooses the customer option from the main menu window, a sub menu will pop up (figure 5.5). This window will ask the end user to choose the customer type (corporate or individual). The two customer type options are represented in this window by radio buttons. There is a push button in this window that enables the user to go to the main menu window.
5.2.4.1 Individual Customer Window. This customer menu has three tab pages, the first one contains the zip code information (figure 5.6), the second one contains customer information (figure 5.7) and the last page contains job and license information (figure 5.8). The user can navigate through the windows by clicking on the tab page title. In most system windows, there are horizontal and vertical toolbars. The horizontal toolbar is located at the top of the screen and consists of two fields (date, user name).

The vertical toolbar is located on the left side of each window in the system and has seven push buttons options. In the customer window, the vertical toolbar has seven options. Since all items in this system have a default property of no insert and update, the 'Enter
Query' option will enable all items in the specified forms to be updateable items.

In other words, the user cannot insert or update a record in this system by default. The option of 'Enter query' will make all items (fields) in the customer window in insert and query modes, so the user will enter any value that he/she want to query about. After the user enters the value in the proper field, he/she will click the second option 'Execute Query'. This option will bring the proper record that the user asked to retrieve from the database.

The third option of the vertical toolbar is 'Insert Record'. This option is used to insert new records to the system. By clicking 'Insert Record', the cursor will go to the first field and all the fields in the window will be in insert mode. Then, the user can enter the information to each field. The forth option in the vertical toolbar is 'Update Record'. This option is used to update any particular field in the database. The fifth option in the vertical tool bar is 'Delete Record'.

First, the user must retrieve the record that he/she wants to delete. Then by clicking 'Delete Record', the system will delete that record. In the 'Update & Delete Record' options, the user has the option to save the
record that has been updated or deleted by clicking the sixth option of the vertical toolbar 'Save'.

The last option in the vertical toolbar is 'Return'. If the user clicks this option, the system will take him/her to the main menu window. If the user exits by mistake without saving the information that he/she entered, the system automatically asks <Do you want to save the information that has been entered? >. After this, the user will respond by clicking < (Yes) or (No) >.

5.2.4.1.1 Zip Code Tab Page. This is the first tab page of the individual customer window (figure 5.6). This page contains all information of the zip code, city and state. The entire fields in this page are by default in no insert mode, update mode and delete mode. The user will choose the mode, depending on the kind of the transaction he/she intends to perform.
5.2.4.1.2 Customer Tab Page. This page (figure 5.7) contains all the information about the customers (SSN, name, address, phone, etc).

There is a check constraint on the date of birth field in this page. It has been assumed that, the customer age in this system will be between 25 and 68 years. Therefore, if the user enters any date of birth out of this range, the system will not accept it and a message will pop up saying 'The customer age must be between 25 and 68 years', the cursor will freeze in the field until the user enters a valid date of birth.
5.2.4.1.3 Job and License Tab Page. This page (figure 5.8) contains the customer's job and license information.

Figure 5.7. Customer Tab Page Window

<table>
<thead>
<tr>
<th>Zip code</th>
<th>Customer</th>
<th>Job &amp; License</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zip code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Home Phone</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>Cust.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>▼</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▼</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▼</td>
</tr>
</tbody>
</table>
Figure 5.8. Job and License Tab page Window

5.2.4.2 Corporate Customer Window. This window consists of two tab pages, one is 'Zip code ' and the other is 'Company'. These windows also have horizontal and vertical toolbars. Both of these toolbars contain the same options and fields all over the system windows. Basically, the users can query about record, insert a record, delete a record and update a record.

5.2.4.2.1 Zip Code Tab Page. This is the first tab page for both individual customer and corporate customer windows as shown in (figure 5.6). The zip code tab page in
both individual and corporate customers is referenced to the same back end tables in the database.

This page contains all information about the zip code, city and state of each customer. The entire fields in this page are by default in no insert mode, update mode and delete mode. The user will choose the mode, depending on the kind of the transaction he/she intends to perform.

5.2.4.2.2 Company Tab Page. This page (figure 5.9) contains all the information about the companies that rented cars in the system. Such company information includes company name, address, phone number, contact person, etc. There is a master detail relationship between the zip code page and company page. That means one or more companies could have the same zip code.
5.2.5 Car Window

The second option of the main menu is the car window (figure 5.10). This window will have all the information about cars in the system, such as plate number, model, make, year, color, etc. This window also has horizontal and vertical toolbars. Both of these toolbars contain the same options all over the system windows. The user can inquire about records, insert, delete and update records. There is a field in this window called 'Flag'. This field aims to represent the status of the car and has the following values: '0' ----> which means that the car is
available for rent; '1'----> means that the car is already rented; and '2'----> means that the car is in a garage for repair purposes.

![Car Window](image)

**Figure 5.10. Car Window**

### 5.2.6 Rental Branches Window

This is a secured window (figure 5.11), in which the only users who have access to this window are the Database Administrators, the Managers and any users that have the proper privileges from the DBA. If the user does not have
the privileges for this window, the push button on the 
main menu window will be turned off. Otherwise this button 
will be turned on. To access this window, the user has to 
have the sufficient privileges from the DBA.

This window contains information about rental 
branches for the car rental company such as branch number, 
location, city, etc. This window also has a horizontal and 
a vertical toolbars. Both of these toolbars contain the 
same options in all the system windows.

Figure 5.11. Rental Branches Window
5.2.7 Rent Information Choice Window

The user can access this window (figure 5.12) from the main menu window by clicking on the 'Rent' push button. This window has two radio buttons, one is 'Rent a car' and the other is 'Receive a car'. The purpose of this window is to ask the user what is the rent transaction to perform (renting a car or receiving a car).

This window has a date and a user field and both of them are located at the top of the window in the horizontal toolbar. The plate number field in this window is a list of value field, which means that when the cursor is in plate number field, the user will press 'F9' to list all the cars in the system that have 'Flag' equal to zero (available). The list of value consists of the car plate number, make, model, year and color.

![Rent Information Choice Window](image)

Figure 5.12. Rent Choice Window
5.2.7.1 Customer Choice Window. When the user chooses the customer options from the main menu window, a submenu will pop up (figure 5.13). This window will ask the end user to choose the customer type (corporate or individual). The two options of the customer type are represented in this window by radio buttons. There is a push button in this window that enables the user to go to the main menu window, which is the 'Go Main Menu'. By clicking on that button, the main menu window will pop up.

![Customer Choice Window](image)

Figure 5.13. Customer Choice Window

5.2.6.1.1 Individual Customer Rent A Car Window. This window (figure 5.14) has information from multiple tables, such as customer information, car information, and credit card information. It has all the information that the user needs have in order to perform a rent transaction. This window has horizontal and vertical toolbars.
Figure 5.14. Individual Customer Rent a Car Window

5.2.7.1.2. Corporate Customer Rent a Car Window.

This window (Figure 5.15) has information from multiple tables, such as company information, car information, and credit card information. This window has horizontal and vertical toolbars.
5.2.7.2 Receiving Car Window. This window (figure 5.16) has all the information regarding a rent transaction that the user needs when a client returns the rented car. This window differs from the rent a car window (figure 5.14) because there are extra fields on this window such as (mile_in, gas_in, tax, total, and others). There are two list items in this window. The first one is insurance type. The users are limited to four kinds of insurance types. The insurance list includes (8.99, 11.99, 20.25, 'No Insurance'). The other list item is rent type. Also
the users are limited to three rent types which are ('Daily', 'Weekly', 'Monthly').

![Figure 5.16. Receiving a Car Window](image)

5.2.8 New User Account Window

This window (figure 5.17) can be accessed from the 'Administrator' button in the main menu window. This window is a secured window and only the Database
Administrator has access to it. If such user does not have the sufficient privileges to this window, the 'Administrator' push button will be turn off in the main menu window. Otherwise it will be turn on.

The purpose of this window is to create new user account for the system. The window contains three fields: user name, password and re enter password. The password and re enter password fields must have the same value in order to create a new user in the system. If this is not the case, the system will show the following message ('Re enter User name and password again'). There are two push buttons in this window. One is 'Create' to create the user and the other 'Return' to return to the main menu window. Basically, the DBA will enter the information for the new user and click 'Create' to perform the transaction.
5.2.9 Contract Sub Window

The user can access this window (figure 5.18) from the main menu window by clicking on the 'Contract' push button. This window has two radio buttons options, one is corporate contract and the other is personal or individual contract. Therefore, the purpose for this window is only to ask the user the contract type. This window has a date and a user field and both of them are located on the top of the window in the horizontal toolbar.
5.2.9.1 Corporate Parameter Contract Window. This window (figure 5.19) can be accessed through the contract sub menu. This window has been designed using Report Builder. When the user enters this window, he/she will be asked to choose the company name from the list of values. After this, the end user will click ‘Enter’ or ‘Run’ to run the report.
5.2.9.1.1 Corporate Contract Report. This report (figure 5.20) contains information from multiple tables such as rent, car, corporate customer, etc. This report will be accessed from the corporate parameter window.

![Personal Legal Contract](image)

Figure 5.20. Corporate Contract Report Window

5.2.9.2 Individual Parameter Contract Window. This window (figure 5.21) can be accessed through the contract sub menu. This window has been designed using Report Builder. When the user enters this window, he/she will be asked to choose the company name from the list of values.
After this, the end user will click 'Enter' or 'Run' to run the report.

Figure 5.21. Individual Parameter Contract Window

5.2.9.2.1 Individual Contract Report. This report (figure 5.22) contains information from multiple tables such as rent, car, corporate customer tables. This report will be accessed from the corporate parameter window.
## Personal Legal Contract

### Customer Information
- SSN
- Name
- Address
- City
- St.
- Zip code
- Phone

### Car and License Information
- Plate No.
- Make
- Model
- Year
- Color
- Fee
- License No
- State
- Lice. Type
- Vin #

### Rent Information
- Rent No.
- Br. No
- address
- Branch
- City
- State
- Zip code
- Mile out
- Gas out
- Gas price
- Ins. Type
- Rent type
- period
- Ins. Price
- Card No
- Card type
- Card expire

### Customer Signature

---

**Figure 5.22. Individual Contract Report Window**

### 5.3 Menu Module

The system has a menu module that appears at the top in all of its windows. This menu module consists of six menu options, which are (File, Edit, Forms, Help, Quit, and Window). Each of these menu models has components as follows:

#### 5.3.1 File Menu Module

This menu module has five components. The first option is 'Save' to save any transaction, 'Clear' to clear...
fields, 'Page Setup' to change the page setup properties, 'Print' to print the current page or form, and 'Exit' to exit the system.

5.3.2 Edit Menu Module

This menu module has three components. The first one is 'Cut' to cut a specific value. Second is 'Paste' to paste a specific value, and 'Copy' to copy a specific value.

5.3.3 Forms Menu Module

This is an easy access menu module to the end users to allow them to navigate quickly throughout the system windows. This window makes the navigation between the forms and reports easy and quick. This menu module has six options. The first option is the 'Welcome Page', to invoke the welcome page window. Second option is 'Main Menu', to invoke the main menu window. Third option is 'Customer Menu', to invoke the customer window. Forth option is 'Rent' and has a submenu of two values, the first one is 'Rent a Car' and the second one is 'Receive a Car' and both of them invoke the rent a car window and receive a car windows respectively. The menu module's fifth option is 'Car', to invoke car window, and the last option is 'Rental Branches', to invoke rental branches window.
5.3.4 Exit Menu Model

The exit menu module purpose is to quit from the system.

5.3.5 Window Menu Module

This menu module has three components and they are, 'New Window', 'Split', and 'Arranges All'.

5.4 Database Application Life Cycle after Design Process

5.4.1 Loading Data Conversion

The data will be loaded through *.SQL data files. There are other techniques for data loading, such as using text files or SQL scripting and others.

5.4.2 Application Conversion

This system does not depend on a prior existing or old system. This system is not an enhancement of an existing system. Therefore it does not have an application conversion.

5.4.3 Testing and Validation

The testing process will include all the forms and reports of the system that includes inserting, updating, querying and deleting records from each form and report. Also the test will include the security and the privileges that have been assigned by the DBA to each end user. This
phase of application life cycle will be the last phase before placing the system for operation.

5.4.4 Operation

After the testing and validation of each component in this master project, the system will be ready for operation.

5.4.5 Monitoring and Maintenance

The final phase of any application life cycle is to support and monitor the system. This task requires computer experts to perform it.
CHAPTER SIX
SECURITY AND TESTING

One of the main issues in designing any system is the data and forms security. The system is using Oracle built-in security features to achieve the system security. Oracle provides good security features on the data, such as roles, privileges and DBA tools. Thus, third party cannot have access to any information in the system without having a valid user name and a password. After the login process is completed, the system security will automatically be turned on and each user will see the proper privileges assigned to them by the DBA.

6.1 Levels of Security

This system has two levels of security:

1. Forms Level: Which user has access to which form.

2. Transaction Level: How many transactions (delete, insert, update, and query only) have been granted to each user.

The system has the following types of user groups:

1. Database Administrator user accounts: This account has full permission to all forms, tables and reports in the system. Also this type of
user account has full permission to do any kind of transactions, such as delete, insert, update and query.

2. Managers and Supervisors users accounts: This type of account has access to most of the forms in the system except to 'Create User Account Form'. Also, this type of user account has full permission to do any kind of transactions, such as delete, insert, and update and query any record in any form.

3. Data Entry Users and other employees: This type of user account has access to most of the forms except to 'Create User Account Form' and 'Rental Branches Form'. Also, this user has limited permission to do any kind of transactions, such as delete, insert, update and query.

4. Receptionist and Temporary Employees: This type of user account is restricted to all kinds of transactions except to query only and exit options.

6.2 Constraints

One of the issues to stress is the need to obtain the proper licenses of the software that the system will run
on. Oracle Corporation will sell the license based on the number of terminals that will operate the system. Needless to say, the more terminals the system uses, the more licenses the company needs and the more expensive the system will become.

6.3 Assumptions and Dependencies

The system will be easy to upgrade and enhance. The engineer or designer will be able to take the existing design and make the proper enhancements by applying the Reverse Engineering process to meet the new requirements.

6.4 Loading Testing Table

The system performance has been compared to the performance of two of other systems, which are EasyRent system [8] and Hertz system [9] when loading 400,000 records. The results format is represented in HH: MM: SS as shown below:
6.5 Transaction Testing Table

The results below are represented in milliseconds and have the following format SS: MS

<table>
<thead>
<tr>
<th>Transaction</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hertz</td>
</tr>
<tr>
<td>Retrieve</td>
<td>00:260</td>
</tr>
<tr>
<td>Insert</td>
<td>00:406</td>
</tr>
<tr>
<td>Update</td>
<td>00:515</td>
</tr>
</tbody>
</table>

6.6 Conclusion

This system is informative and has been designed to be user-friendly for both clients and end users. The system design is easy to upgrade and in order to meet future requirements. All forms and reports of the system have been designed using Oracle 8i (Developer 6i) under Graphical User Interface platform (MS Windows).
The Choice Of the DBMS commercial language (Oracle) came from a series of factors, such as the build in security system that Oracle provide, the property pallet that Oracle form provides for each item, block and form that will save a lot of time of the coding process, and many other factors. There were some difficulties during the design and implementation of the system. One of the difficulties was scheduling interviews with the end users and asking them about problems that they face with current rental systems. Another one is the cost of learning Developer 6i language and the cost of the training on Oracle forms for the end users.
APPENDIX A

SOME STRUCTURED QUERY LANGUAGE

CODES OF THE SYSTEM
Security Form
when_new_form_instance //When the form open
:top_can.cur_user := user; //Assign item user to the current user
:top_can.cur_date := sysdate; //Assign item date to the current date
set_window_property(forms_mdi_window,window_state,maximize); // Maximize the current window
set_window_property('dept_win',window_state,maximize);
when_button_pressed (Item level)
if :con.pwl <> :con.pw2 //comparision between item pw1 and pw2
then
message ('reenter user name and password again');
else
begin
forms_ddl('create user'||''||':con.username'||''||'identified by'||''||':con.pwl');
//invoke build in procedure form_dll
forms_ddl('grant connect,resource to'||''||':con.username'); // invoke build
forms_ddl('grant create session to'||''||':con.username'); // in procedure form_dll
if not form_success then
message('can not create user name ');
else
message('user name been created');
end if;
end;
end if;
Welcome form
when_new_form_instance trigger (Form level)
set_window_property('welcome_page',window_state,maximize); // Maximize the current window
set_window_property(forms_mdi_window,title,'rental car system');
declare
tl_id timer; //declaring a timer variable
begin
tl_id := create_timer('tl',10000,no_repeat); //using create_timer
//build in procedure timer variable
//timer will last 10 second before expires
:welcome.cur_date:=sysdate;
:welcome.cur_user:=user;
end;
when_timer_expired (Form level)
new_form('c:\project\main_menu.fmx'); //Invoking new form called main menu when timer expires
About form
When_new_form_instance(Form level)
set_window_property(forms_mdi_window,title,'rental car system'); // Maximize the current window
set_window_property('about',window_state,maximize);
declare
tl_id timer;

begin
    tl_id :=create_timer('tl1',10000,no_repeat);
    procedure
    :welcome.cur_date:=sysdate;
    :welcome.cur_user:=user;
end;

When_timer_expired (New form)
new_form('c:\project\main_menu.fmx'); //Invoking new form called main menu when timer expires

Car form
When_new_form_instance(form level)
:top_can.cur_user := user;
:top_can.cur_date :=sysdate;
begin
    declare
        cursor insert_role is select granted_role from //creating new cursor to insert
        user_role_privs where granted_role like 'r_';
        x varchar2(20); //defining the variable x as variable character
    begin
        open insert_role; //opening the cursor
        insert_role
            loop //Loop
                fetch insert_role into x; //fetching the cursor insert_role
                exit when insert_role%notfound; //terminate condition when insert_role
                is empty
                    if x='r1' then //if statement to compare x to each value of
                        insert_role
                            set_item_property('control.save',enabled,property_true);
                            //setting some of the items in the vertical tool
                            set_item_property('control.insert_record',enabled,property_true);
                            set_item_property('control.return',enabled,property_true);
                            set_item_property('control.enter_query',enabled,property_true);
                            set_item_property('control.exe_query',enabled,property_true);
                    end if;
                    if x='r2' then
                        set_item_property('control.save',enabled,property_true);
                        //setting all items in the vertical tool bar to be enabled
                        set_item_property('control.insert_record',enabled,property_true);
                        set_item_property('control.update_record',enabled,property_true);
                        set_item_property('control.delete_record',enabled,property_true);
                        set_item_property('control.return',enabled,property_true);
                        set_item_property('control.enter_query',enabled,property_true);
                end if;
        end loop;
    end;
end;
set_item_property('control.exe_query',enabled,property_true);
end if;
if x='r3' then
    set_item_property('control.return',enabled,property_true);
    //setting some of the items in the vertical tool bar to be enabled
    set_item_property('control.enter_query',enabled,property_true);
    set_item_property('control.exe_query',enabled,property_true);
end if;
end loop;
end;
set_window_property(forms_mdi_window,window_state,maximize);
// Maximize the current window
set_window_property('dept_win',window_state,maximize);
On_clear_detail trigger(block level) //trigger will be fired when detail block is clear
    //This trigger will be fired in all forms //that has master details relationships
-- begin default relation program section
--
begin
clear_all_master_details;
//invoke procedure to clear all master detail relationships
end;
--
-- end default relation program section
--
on_populate_details trigger(block level)
--
-- begin default relation declare section
--
declare
    recstat  VARCHAR2(20) := :system.record_status;  //defining variable recstat to be of
    startitm VARCHAR2(61) := :system.cursor_item;  //defining variable startitm to be of
    rel_id   relation;
--
-- end default relation declares section
--
--
-- begin default relation program section
--
begin
    if ( recstat = 'new' or recstat = 'insert' ) then  //comparison
        return;
    end if;
end;
-- begin car_price detail program section

if ( (:car.plate_no is not null) ) then
  //comparison
  rel_id := find_relation('car.car_car_price'); //assigning variable rel_id by calling procedure find_relation
  query_master_details(rel_id, 'car_price');
end if;

-- end car_price detail program section

if ( :system.cursor_item <> startitm ) then
  go_item(startitm); //build in function check_package_failure;
end if;
end;

-- end default relation program section

on_check_detail_master trigger

-- begin default relation declare section

declare
dummy_define char(1);

-- begin car_price detail declare section

cursor car_price_cur is //defining new cursor
    select 1 from car_price c //SQL statement
    where c.plate_no = :car.plate_no;

-- end car_price detail declare section

begin

-- begin car_price detail program section

open car_price_cur; //opening cursor

82
fetch car_price_cur into dummy_define; //fetching values in variable dummy_define
if ( car_price_cur%found ) then //checking if there is any value in the master table
  message('cannot delete master record when matching detail records exist.');
  close car_price_cur; // closing the cursor
end if;
car_price_cur
  raise form_trigger_failure;
end if;
close car_price_cur;
--
-- end car_price detail program section
--
end;
--
-- end default relation program section
--

Key_next_item(irem level)
go_item('car.summery'); //build in function

Vertical Toolbar Codes //This toolbar are in most of the system forms

When_botttn_pressed('enter query' ,item level) //Trigger will be fired when the item pressed
set_item_property('car.plate_no',queryable,property_true); //all items in car for will be in query mode
set_item_property('car.make',queryable,property_true);
set_item_property('car.model',queryable,property_true);
set_item_property('car.year',queryable,property_true);
set_item_property('car.colore',queryable,property_true);
set_item_property('car.flag',queryable,property_true);
set_item_property('car.summery',queryable,property_true);
set_item_property('car_price.daily_price',queryable,property_true);
enter_query;
When_botttn_pressed('execute query' ,item level) //Trigger will be fired when the item pressed
execute_query; //build in ket to retrieve all records
set_item_property('car.plate_no',queryable,property_false); //all items in car for will be in not in update mode
set_item_property('car.make',queryable,property_false);
set_item_property('car.model',queryable,property_false);
set_item_property('car.year',queryable,property_false);
set_item_property('car.colore',queryable,property_false);
set_item_property('car.flag', queryable, property_false);
set_item_property('car.summery', queryable, property_false);
set_item_property('car_price.daily_price', queryable, property_false);
set_item_property('car.plate_no', update_allowed, property_false);
set_item_property('car.make', update_allowed, property_false);
set_item_property('car.model', update_allowed, property_false);
set_item_property('car.year', update_allowed, property_false);
set_item_property('car.colore', update_allowed, property_false);
set_item_property('car.flag', update_allowed, property_false);
set_item_property('car.summery', update_allowed, property_false);
set_item_property('car_price.daily_price', update_allowed, property_false);

When_button_pressed('insert record', item level)
set_item_property('car.plate_no', insert_allowed, property_true);  // all items in car
for
will be in query mode
set_item_property('car.make', insert_allowed, property_true);
set_item_property('car.model', insert_allowed, property_true);
set_item_property('car.year', insert_allowed, property_true);
set_item_property('car.colore', insert_allowed, property_true);
set_item_property('car.flag', insert_allowed, property_true);
set_item_property('car.summery', insert_allowed, property_true);
set_item_property('car_price.daily_price', insert_allowed, property_true);
enter_query;
// form will be in enter query mode (clear all records)
set_item_property('control.save', enabled, property_true);
set_item_property('control.return', enabled, property_true);
set_item_property('control.insert_record', enabled, property_false);
When_button_pressed('update record', item level)
set_item_property('car.plate_no', update_allowed, property_true);
set_item_property('car.make', update_allowed, property_true);
set_item_property('car.model', update_allowed, property_true);
set_item_property('car.year', update_allowed, property_true);
set_item_property('car.colore', update_allowed, property_true);
set_item_property('car.flag', update_allowed, property_true);
set_item_property('car.summery', update_allowed, property_true);
set_item_property('car_price.daily_price', update_allowed, property_true);
When_button_pressed('delete record', item level)
set_item_property('car.plate_no', update_allowed, property_true);  // all items in car
for
will be in update mode
set_item_property('car.make', update_allowed, property_true);
set_item_property('car.model', update_allowed, property_true);
set_item_property('car.year', update_allowed, property_true);
set_item_property('car.colore', update_allowed, property_true);
set_item_property('car.flag', update_allowed, property_true);
set_item_property('car.summery', update_allowed, property_true);
set_item_property('car_price.daily_price', update_allowed, property_true);
go_block('car_price');
//build in word to go to the specified block
delete_record; //delete all records in the form
go_block('cat'); //build in word to go to the specified block
delete_record;
set_item_property('control.save',enabled,property_true);
set_item_property('control.return',enabled,property_true);
set_item_property('control.delete_record',enabled,property_true);

When_btnn_pressed('save',item level)
commit_form; //save all items in the current form
set_item_property('control.save',enabled,property_false);
//define the same cursor insert_role to return everything as default
declare
cursor insert_role is select granted_role from
user_role_privals where granted_role like '%r_';
x varchar2(20); //declaring x as character variable
begin
open insert_role; //opening cursor insert_role
loop
fetch insert_role into x; //fetch from insert role to x
exit when insert_role%notfound;
if x='r1' then //comparison
set_item_property('control.save',enabled,property_true);
set_item_property('control.insert_record',enabled,property_true);
set_item_property('control.return',enabled,property_true);
set_item_property('control.enter_query',enabled,property_true);
set_item_property('control.exe_query',enabled,property_true);
end if; //comparison
if x='r2' then
set_item_property('control.save',enabled,property_true);
set_item_property('control.insert_record',enabled,property_true);
set_item_property('control.update_record',enabled,property_true);
set_item_property('control.delete_record',enabled,property_true);
set_item_property('control.return',enabled,property_true);
set_item_property('control.enter_query',enabled,property_true);
set_item_property('control.exe_query',enabled,property_true);
end if; //comparison
if x='r3' then
set_item_property('control.return',enabled,property_true);
set_item_property('control.enter_query',enabled,property_true);
set_item_property('control.exe_query',enabled,property_true);
end if;
end loop;
end;
When_bottn_pressed('return',item level)
new_form('c:\project\main_menu.fmx'); //invoking the main form window

Customer form
When_tab_page_changed(form level)  //Trigger will be fired when clicking on
    the tab page
    declare
c  varchar2(20);  //defining variable x as character
    begin
c := :system.tab_new_page;  //assigning x to be system tab
    variable type
    if c='zipcode'  //comparison if x is in zip code
    then go_block('zipcode_city');  //if so then go to the zip code
    window
    elsif c='customer' then  //comparison
        go_block('customer');
    elsif x='job' then
        go_block('job');
    elsif x='company' then  //comparison
        go_block('company');
eend if;
end;
When_new_form_instance(form level)
:top_can.cur_user := user;  //assigning user variable to
    current user
:top_can.cur_date :=sysdate;  //assigning date variable to current date
set_window_property(forms_mdi_window,window_state,maximize);  //maximize the
    current
    window
set_window_property('sub_menu',window_state,maximize);
    declare
rg recordgroup;  //declaring a record group
    er number;  //declaring er as number
    begin
rg:=create_group_from_query('gl','select descrip,to_char(cust_type) ,
    //creating gl as record group and filling it with description
from cust_type');
er:=populate_group(rg);  //populate record group rg
populate_list('c_type',rg);  //populate list c_type
end;

When_mouse_click(item level)  //Trigger will be fired when clicking on
    the mouse
    if :cust = 1  //comparison
    then go_block('zipcode_city');  //build in function that will take you to
    specific block
    set_tab_page_property('customer.company',visible,property_false);
    //making page corporate customer not visible
end if;
if :cust = 2
then
  go_block('zipcode_city');
  set_tab_page_property('customer.customer',visible,property_false); //making page individual customer not visible
  set_tab_page_property('customer.job',visible,property_false); //making page job not visible
  Pre_insert trigger(block level) //Trigger will be fired before inserting any value
  select max(co_no)+ 1 into :co_no from company; //SQL statement that will take the max. number //from table company and add one to it;
End if;
Pre_query(block level) //Trigger will be fired before querying about any value 
:cust_l_name:=:cust_l_name||"{}"; //assigning cust_l_name to last name with any character
Key_next_item(dob item level) //Trigger will be fired when pressing Enter key
if round(((sysdate - :dob)/365.25), 0)between 25 and 68 //checking if the customer age between 25 and 68 years
go_item('home_phone');
else
  message ('the customer age must be between 25 and 68 years'); //message box if the above if statement failed
  message ('the customer age must be between 25 and 68 years'); //message box
end if;
On_populate_details(block level) //Trigger will be fired when populating detail block
-- begin default relation declare section
--
declarerelstat varchar2(20) := :system.record_status; //defining two character variables
startitm varchar2(61) := :system.cursor_item;
rel_id relation; //defining a relationship variable
--
-- end default relation declare section
--
-- begin default relation program section
--
begin
  if ( recstat = 'new' or recstat = 'insert' ) then //comparison
return;
end if;

-- begin cust_join_type detail program section
--
if ( (:cust_type.cust_type is not null) ) then //comparison
  rel_id := find_relation('cust_type.cust_type_cust_join_type');

Main Menu form
When_button_pressed(item level)
new_form('c:\project\rent.fmx'); //invoking rent form
new_form('c:\project\customer.fmx'); //invoking individual customer form
new_form('c:\project\security.fmx'); //invoking security form
new_form('c:\project\car.fmx'); //invoking car form
new_form('c:\project\dept_insert.fmx'); //invoking rental branches form
exit_form; //build in function to exit out of the forms

Rent a Car form
When_tab_page_changed(form level)
declare
  x varchar2(20); //declaring x as character variable
begin
  x := :system.tab_new_page; //assigning x to be a tab page type
  if x='zipcode'
    then go_block('zipcode_city'); //take cursor to block zip code
  elsif x='customer' then
    go_block('customer'); //take cursor to block customer
  elsif x='job' then
    go_block('job'); //take cursor to block job
  end if;
end;

When_mouse_click(item level)
if :rent = 1 //comparison
  then
go_block('rent_master'); //take cursor to block rent master
elsif :rent = 2
  then new_form('c:\project\rent_recieve.fmx'); //invoking receive a car form
end if;
Post_insert(block level) //Trigger will be fired after inserting process completed
update car
set flag=1
where plate_no = :plate_no;

Post_change(item level)
declare
f_name varchar2(20);
end;
begin
select cust_f_name,cust_m_i,cust_l_name into f_name,m_name,l_name
from customer
where ssn=:ssn;
end;

When_validate_item(item level)
-- begin default

When_validate_item(item level)
-- begin default enforce data integrity constraint fk_rentno22 section

declare
cursor primary_cur is select 'x' from scott.department
where dept_no = :rent_master.dept_no;
primary_dummy char(1);
begin
if ( :rent_master.dept_no is not null ) then
   open primary_cur; //opening primary_cur
   fetch primary_cur into primary_dummy;
   if ( not primary_cur%found ) then //when there is no value left
      message('foreign key value does not currently exist in the primary key table.'); //message box
      close primary_cur; //close primary_cur cursor
      raise form_trigger_failure;
   end if;
end if;
close primary_cur;
end;

--
-- end default enforces data integrity constraint fk_rentno22 section

Post_change(item level)
select make,model,year,colore into :make,:model,:year,:colore
from car
where car.plate_no = :plate_no
Post_change(item level)
select ins_daily_price into :ins_daily_price
from ins_type
where ins_type=:ins_type;
Post_change(item level)
select rent_description into :rent_description
from rent_type
where rent_type= :rent_type;

Receive a car form
Post_update(block level)
update car
set flag=0
where plate_no = :plate_no' ;
Post_change(item level)
select daily_price into :daily_price.
from car_price
where plate_no = :plate_no;
Post_change(item level)

if :gas_out_gallon - :gas_in_gallon >0
//comparison
then
select ((gas_out_gallon - :gas_in_gallon) * gas_price_gallon) into :gas_charge
//SQL statement(calculated item)
from rent_detail where plate_no = :plate_no ;
else
:gas_charge := 0;
end if;
Post_change(item level)
select ins_daily_price into :ins_daily_price
from ins_type
where ins_type=:ins_type;
if :rent_type=1
//comparison
then :ins_charge := rent_period*ins_daily_price ;
elsif :rent_type=2
//comparison
then :ins_charge := rent_period*ins_daily_price*7 ;
elsif :rent_type=3
//comparison
then :ins_charge:=rent_period*:ins_daily_price*30 ;
end if;

Post_change(item level)
select rent_description into :rent_description
from rent_type
where rent_type= :rent_type;

Key_next_item(item level)
select (:rent_charge )*(nvl(max(discount),0)/100) into :cust_disc from cust_type a,cust_join_type b
where a.cust_type = b.c_type
and b.ssn = :ssn;
go_field('discount');

Key_next_item(item level)
:sub_tot:= (:rent_charge + nvl(ins_charge,0) + nvl(other Charg,0) +
nvl(gas_charge,0) )-
((:cust_disc)+(discount));
:tax_amount := (:taxl/100) * :sub_tot;
:total := :sub_tot + :tax_amount ;
go_item('tot_paid');

Key_next_item(item level)
:balance := :total - :tot_paid; //SQL statement

go_field('balanc');
APPENDIX B

IMPLEMENTATION OF SYSTEM TABLES
Car Flag Table

```sql
CREATE TABLE Car_Flag
(
    Flag_no NUMBER (1),
    Flag_description VARCHAR2(30),
    CONSTRAINT pk_car_flag PRIMARY KEY (Flag_no)
);
```

Car Table

```sql
CREATE TABLE Car
(
    Plate_no VARCHAR2(10),
    Make VARCHAR2(15) NOT NULL,
    Model VARCHAR2(15) NOT NULL,
    Year NUMBER (4) NOT NULL,
    Color VARCHAR2(15),
    Flag NUMBER (1),
    Summary VARCHAR2(50),
    CONSTRAINT pk_plate_no PRIMARY KEY (Plate_no),
    CONSTRAINT fk_flag_no FOREIGN KEY (Flag_no) REFERENCES car_flag(Flag_no)
);
```

Rental Fee Table

```sql
CREATE TABLE Rental_Fee
(
    Plate_no VARCHAR2(10),
    Rental_fee NUMBER (5,2) NOT NULL,
    CONSTRAINT pk_plate_price PRIMARY KEY (Plate_no, Rental_fee),
    CONSTRAINT fk_plate12_no FOREIGN KEY (Plate_no) REFERENCES car (Plate_no)
);
```

Rental Branches Zip code Table

```sql
CREATE TABLE Rental_Branches_Zip_code
(
    Zip_code NUMBER (5),
    City VARCHAR2(25) NOT NULL,
    State VARCHAR2(2) NOT NULL,
    Tax NUMBER (4,4) NOT NULL,
    CONSTRAINT pk_zip_code PRIMARY KEY (Zip_code)
);
```

Rental Branches Table

```sql
CREATE TABLE Rental_Branches
(
    Branch_no NUMBER (5),
    Branch_name VARCHAR2(20),
    Branch_address VARCHAR2(30),
    Zip_code NUMBER (5),
    Branch_phone_no NUMBER (10),
    CONSTRAINT pk_dept_no3 PRIMARY KEY (Branch_no),
    CONSTRAINT fk_d_code FOREIGN KEY (Zip_code) REFERENCES Rental_Branches_Zip_code (Zip_code)
);
```

Customer Zip code Table

```sql
CREATE TABLE Customer_Zip_code
(
    Zip_code NUMBER (5),
    City VARCHAR2(25) NOT NULL,
    State VARCHAR2(2) NOT NULL,
    CONSTRAINT pk_zip_code12 PRIMARY KEY (zip_code)
);
```

Individual Customer Table

```sql
CREATE TABLE Customer
(
    SSN NUMBER (9),
    CONSTRAINT pk_customer PRIMARY KEY (SSN)
);
```
Cust_f_name varchar2 (15) not null,
Cust_m_i varchar2 (1),
Cust_l_name varchar2 (20) not null,
Cust_address varchar2 (30) not null,
Zip code number (5) not null,
DOB date not null,
Home phone number (10) not null,
Constraint pk_ssn primary key (SSN),
Constraint fk_zip_code12 foreign key (Zip code) references Customer_Zip code (Zip code)).
Customer Type Table
Create table Customer_Type ( 'Cust_type number (1),
Description varchar2 (20) not null,
Discount number (4,4),
Constraint pk_cust_type11 primary key (Cust_type));
Customer Join Type Table
Create table Customer_Join_Type(ssn number(9),
Cust_type number (1),
Constraint pk_cust_join_type primary key (SSN,Cust_type),
Constraint fk_cust_type33 foreign key (Cust_type) references Cust_type(cust_type),
Constraint fk_cust_ssn33 foreign key (SSN) references customer (SSN)).
Job Table
Create table Job (Co_name varchar2 (30),
Job_title varchar2 (20),
Job_address varchar2 (30),
Job_phone number (10),
Constraint pk_c_name primary key(C_name,SSN)).
Customer Job Table
Create table Customer_Job (SSN number (9),
Co_name varchar2 (30),
Constraint pk_ssn_co_name primary key (SSN,Co_name),
Constraint fk_ssn144 foreign key (SSN) references customer (SSN),
Constraint fk_co_name foreign key (Co_name) references job (Co_name)).
Corporate Customer Table
Create table Companyr( Co_no number (10),
Co_name varchar2 (30) not null,
Co_address varchar2 (40),
Co_phone_no number (10) not null,
Contact_person varchar2 (30) not null,
Zip code number (5) not null,
Constraint pk_co_no primary key (Co_no),
Constraint fk_zip_code12 foreign key (Zip code) references Customer_Zip code (Zip code)).
License Table
Create table License (}
Lic_no varchar2 (14),
Lic_type varchar2 (10) not null,
State_issued varchar2 (2) not null,
Expiration_date date not null,
SSN number (9),
Constraint pk_lic_no primary key (Lic_no),
Constraint fk_ss_no155 foreign key (SSN) references customer (SSN)).
Rent Type Table
    Create table Rent_Type(
        Rent_type number (1),
        Rent_limit number (3),
    Constraint pk_rent_type primary key (Rent_type)).
Insurance Type
    Create table Insurance_type {
        Ins_type number (1),
        Ins_daily_price number (3,2),
    Constraint pk_ins_type primary key (Ins_type)).
Rent Master Table
    Create table Rent_Master {
        Rent_no number (30),
        SSN number (9),
        Dept_no number (5),
        Card_no number (16),
        Rent_date date not null,
        Card_expiration date not null,
    Constraint pk_rent_no primary key (Rent_no),
    Constraint fk_ssnl foreign key (SSN) references customer (SSN),
    Constraint fk_co_nol foreign key (Co_no) references customer (Co_no),
    Constraint fk_dept_no990 foreign key (Dept_no) references rental branches (Branch_no),
    Constraint fk_cardll_no foreign key (Card_no) references credit_type(Card_no)).
Rent Detail Table
    Create table Rent_Detail {
        Rent_no number (30),
        Plate_no varchar2 (10),
        Return_dept_no number (5) not null,
        Mile_out number (8) not null,
        Mile_in number (8),
        Gas_out_gallon number (2,2) not null,
        Gas_price_gallon number (2,2) not null,
        Gas_in_gallon number (2,2),
        Ins_type number (1),
        Rent_type number (1),
        MGR_discount number (5,2),
        Other_charges number (7,2),
        Total number (7,2) not null,
        Total_paid number (7,2) not null,
        Balance number (7,2) not null,
Constraint pk_rent_no primary key (Rent_no, Plate_no),
Constraint fk_ins_type foreign key (Ins_type) references ins_type(Ins_type),
Constraint fk_rent_type foreign key (Rent_type) references rent_type(Rent_type),
Constraint fk_rent_type foreign key (Rent_type) references rent_type(Rent_type),
Constraint fk_platell_no foreign key (Rent_no) references rent_master(Rent_no)).
APPENDIX C

SOME COMPUTER TERMS DEFINITION
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: 1</td>
<td>One to One Relationship</td>
</tr>
<tr>
<td>1: N</td>
<td>One to Many Relationships</td>
</tr>
<tr>
<td>N: M</td>
<td>Many to Many Relationships</td>
</tr>
<tr>
<td>DBMS</td>
<td>Database Management System</td>
</tr>
<tr>
<td>DDL</td>
<td>Data Definition Language</td>
</tr>
<tr>
<td>EER</td>
<td>Enhanced Entity Relation</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>ODBC</td>
<td>Open Database Connection</td>
</tr>
<tr>
<td>OS</td>
<td>Operating system</td>
</tr>
<tr>
<td>PL/SQL</td>
<td>Procedural Language / Structured Query Language</td>
</tr>
<tr>
<td>PK</td>
<td>Primary Key</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
</tbody>
</table>
REFERENCES


