A Mathematical Modelling for Manpower Planning

By

Memet ULUDAĞ

A Dissertation Submitted to the
Graduate School in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

Department: Computer Engineering
Major: Computer Software

İzmir Institute of Technology
İzmir, Turkey

June, 2001

We approve the thesis of **Memet ULUDAĞ**

Date of Signature

[illegible]

Assoc. Prof. Ahmet H. KOLTUKSUZ
Supervisor
Department of Computer Engineering — 27.06.2001

[illegible]

Prof. Dr. Sıtkı AYTAÇ
Department of Computer Engineering — 27.06.2001

[illegible]

Asst. Prof. Murat GÜNAYDIN
Department of Architecture — 27.06.2001

[illegible]

Prof. Dr. Sıtkı AYTAÇ
Head of Department — 27.06.2001

ACKNOWLEDGEMENT

I would like to express my many thanks to my advisor *Assoc. Prof. Ahmet H. Koltuksuz, Ph.D.* for his support and encouragament, making it possible for me to write this thesis.

I like thank to Helen Sherlock, who was helping me on accessing many resourses and references in Dublin, much required by me during research and writing phases of this thesis.

Many friends and family members, giving me support and patience. I like to thank them for everything.

ABSTRACT

Information systems are one of the most important tools for organizations to conduct business today. The dependency of organizational functions to information systems makes them critical for the people using them. So becomes the impact of failure or success of information systems more significant in organizations. The initial task for information systems designers is to appreciate this real life situation and understand various aspects of information systems and their evolution over the past decades.

Information system design is a detailed process which has to be planned and implemented with great care not only for, but also with the business experts and users. Designers and users should be aware of the problems, requirements within the organizational context. In this thesis, we will describe and discuss various information systems in organizations. Different information system types will be given.

Database management systems (or databases in organizations) are the essential parts of information systems. A “good” information system is surely backed up with a “good” database behind it. Designing a “good” database systems is the critical part of the process. Following structural and well defined methods on database design is something we need in today in organizations. We will present, further in this thesis, various database management systems and database types. Relational model is the most widely used database management system today. We will study various relational database model concepts which will be a base for our management information system database design. We will follow various design and refinement methodologies to end-up with a well documented and refined relational database model. During our design we will utilize a sophisticated database design tool. We will use the database design tool Sybase PowerDesigner. Our design will be modeling a man-power planning database..

ÖZ

Bilgi sistemleri, günümüz organizasyonlarında, en önemli iş araçlarından biri haline gelmiştir. Organizasyonel fonksiyonların, bilgi sistemlerine olan bağımlılıkları, bilgi sistemlerini kullanıcıları için oldukça kritik hale getirmiştir. Bu nedenle de, bilgi sistemlerinin başarısı veya başarısızlığı, organizasyonlar için daha da önemli hale gelmiştir. Tasarımcılar için ilk iş, bu gerçek yaşam senaryosunu algılamak ve bilgi sistemlerinin değisik yönlerini, gelişimleri ıle birlikte kavramaktır.

Bilgi sistemi tasarımı oldukça detaylı ve cok dikkatli planlanması gereken bir işlemdir. Bu sadece kullanıcıları düşünerek değil, aynı zamanda kullanıcıların da tasarım çabasına katılımını sağlayarak yapılmalıdır. Bu tez çalışmasında, değişik bilgi sistemleri yapıları tanımlanmış ve detayları verilmiştir.

Veri tabanı yönetim sistemleri, bilgi sistemlerinin vazgeçilmez parçalarıdır. İyi bir bilgi sistemi, mutlaka iyi bir veri tabanı sistemiyle desteklenmelidir. Genel olarak, iyi bir veri tabanı tasarımı yapmak, işin en önemli kısmıdır. Günümüzde organizasyonlarda ihtiyaç duyulan, yapısal ve iyi tanımlanmış veri tabanı tasarım metodlarıdır. Tezin ilerleyen kısmında, değisik veri tabanı ve veri tabanı yönetim sistemleri verilmiştir. İlişkisel (Relational) model, günümüzde en yaygın kullanılan verı tabanı modelidir. Bu modelin değişik kavramları, bizce gerçeklestirilecek tasarım çalısmasına temel olması açısından, bu çalışmada detaylandırılmıştır. Burada, tasarım ve geliştirme (refinement) metodlarını izlenerek, iyi dökümanlanmış ve geliştirilmiş (arıtılmış) ilişkisel bir veri tabanı modeline ulaşılmıştır. Bu tasarım sürecinde, gelişmiş bir veri tabanı tasarım paketi Sybase PowerDesigner kullanılmıştır.Veri tabanı tasarımımız, insan gücü planlamasına yönelik bir model ortaya çıkarmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

IZMIR YUKSEK TEKNOLOJİ ENSTİTÜSÜ
REKTÖRLÜĞÜ
Kütüphane ve Dokümantasyon Daire Bşk.

Chapter 1

INTRODUCTION

1.1 Motivation

Computers, and computer based systems have evolved dramatically, since they were first introduced in the 1940's as the products of scientific research and development in to date "high-tech" laboratories of Universities and research labs. Popular availability of computing power lead to the evolutionary development of information systems and databases used by the organizations. Not only the nature of the organizations and business practices, but also the understanding of information systems have somehow changed over the past decades.

Availability of more powerful and economically feasible computing technology has brought new tools and utilities for, not only computer people, but also for business experts. Thus, various information system architectures have been developed along with the different database management systems. Computer usage - at least in the western world - has become a part of daily businesses and personal lives of people in one way or the other.

New roles within the organizations have been defined, creating more "gray" than "black and white" job descriptions and expertise areas. Furthermore, the successful utilization of computing power, specially information systems, have become strategic aspects for the organizations.

Information systems are costly to purchase – or develop – deploy and maintain.[1] Given the amount of expertise, time and effort put into information systems, the expectancy of "success", effectiveness and user satisfaction is at a very high level. These factors in fact make the task of information system design a technical and social combination of efforts. Technical because, building a technically strong infrastructure is always a prerequisite for development. And social because, the existence of the information systems in the organizations are pretty much alive and evolving in nature; where anything from raw data to subjective viewpoints are thrown into a pot and melted together to perform daily and decision making tasks in the organizations. User orientation and awareness of business objectives on one side of the table, and technological abilities / limitations on the other. From these are emerging the final products : The information systems.

The study and implementation of information systems undoubtedly requires strong theoretical knowledge and technical capabilities. But given the nature of them, they also strongly require a practical understanding and experience gained over the years and different exercises carried out in real life. We cannot deny the existence of pro or contra technology feelings of business expert within the organizations. People in their

[1] Munshi J., A Framework for MIS Effectiveness, Working Paper, For presentation to the Aca
Business Administration, Athens, International Conference, July 1996, p.1

own experiences either made a great use of technology, or had struggled to perform their very basic business operations by using "badly" designed and implemented information systems. We believe the question of a "good" information system, at least in the minds of business people and users, is a difficult question. But we also believe that understanding the overall impacts of the information systems to people and to the organization is a good starting point to find answers. And as we have mentioned before, a strong technical ability and understanding is the main weapon that system analyst and designers, technical architects and in general all developers need to have to tackle the problems of information systems.

Databases, as a big entity within the information systems area, are occupying an important role in the overall picture. Not only because they are the source of data, but also because they are sophisticated platforms that require a specific understanding and expertise. Their quality, effectiveness, design characteristics directly effect the overall quality and characteristics of an information system using them. We cannot design and implement an "good" information system without a "good" design and implementation of its database background. One might argue that this rule is valid for any entity of information systems, for example the user applications, but we strongly suspect that databases do play a bigger role since they hold and control the important entity, business data, after all. A badly designed front-end system may be replaced with better versions but a badly designed and implemented database will take more effort and cost to become a better one. What is more, sometimes it is not even possible to achieve this, given the lessons we have learned from the old big legacy systems, which are to important to throw away and to big and badly designed to make more effective and useful.

This thesis was possible by experiencing the real life cases along with the theoretical studies during the last years. The study has been a mixture of "real-life" and academic research. This thesis is aimed to give an overall view of developments in the information system area. Different architectures and types of information systems are introduced before going into the details of database design and modeling. We aimed to distinguish the different information system types from each other. Our categorization is a more functional categorization. One can categorize information systems, based on their architecture or criticality or scope. We have introduced some architectural differences but it is not our study in this thesis to go into details of these.

1.2 Scope

We have seen a highly dense relationship between the information systems and database systems, which lead us to study and present the various database systems, their architectures and their history of evolvement in this thesis. The final outcome of this study is a database modeling for man-power planning in a typical organization in manufacturing business. Since this is not a full implementation of an information system, we have excluded the system analysis phases.

Different types (i.e. manufacturing, financial) of organizations, where full system developments have been experienced - from a programmers level to project management level - provided the real life knowledge for the database development work in this thesis. Real life experience on different platforms of computer systems and development environments over the past years was the main motivation, along with the theoretical study in the information systems and database area.

This work is mainly concerned with the utilization of formal database development methods and a sophisticated database design and development tool to implement a management information system database; a model for man-power planning in an organization. We will usage E/R Diagram method to define the model and use the conceptual and physical design methods for developing the actual database.

Before the actual development work, we have introduced various definitions and concepts of information systems and database systems. This is done as an outline of a generic picture of what happened and is happening in the information systems area and database management systems in the organizations. We have worked on a relational database model with the believe that the relational model is the most standardized and widely utilized model in many organizations to date.

Many big database providers such as Oracle, Sybase have come up with powerful and sophisticated database management systems over the last years for relational database systems. The current trend in database management systems is towards the object-relational and object-oriented databases. We did not include any object-oriented design and development in our model. Object-relational model is an extension to relational database management systems with added abstract data types, nested tables, varying arrays and large objects. Even though some of these features are used in our model, we still call it a relational-database since main model behind it is the relational database model.

1.3 Organization

The rest of this thesis is organized as follows

- Chapter 2 describes various concepts of information systems, their evolution, different information system types and their architectures.
- Chapter 3 describes various database system concepts, architectures, organizational relations and different database types. The elaboration is on the relational database systems as this is the model for our database development. Relational model concepts, functional dependency, normalization and schema refinement is given in this chapter.
- Chapter 4 describes the various concept on designing a database, E/R modeling, conceptual design and physical design.
- Chapter 5 contains the actual work of the database development using the modeling and design tool Power Designer.
- Chapter 6 is the concussion, wrapping up various discussions and future research and development opportunities in the area we have worked in this thesis.

Chapter 2

INFORMATION SYSTEMS

2.1 Basic Concepts: Data and Information

The new Oxford Dictionary of English defines data as "facts and statistics collected together for reference and analysis"[2] with the sub sense "the quantities, characters or symbols on which operations are performed by a computer, which may be stored and transmitted in the form of electrical signals and recorded on magnetic, optical, or mechanical recording media."

Webster's Dictionary defines data as "things known or assumed; facts or figures from which conclusion can be inferred."[3]

For a complete definition of "data", both definitions above could be combined. As for computer people and the way their minds sometimes work, the sub sense given above probably makes more sense.

A more dual and standard definition for data is given by the American National Standards Institute (ANSI):

1. A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by humans or by automatic means.
2. Any representation such as characters or analog quantities to which meaning is or might be assigned. Generally, we perform operations on data or data items to supply some information about an entity.[4]

Everyone, in computer terms or not, has data. Organizations, individuals, or systems generate and use data in their activities to form different kind of knowledge or information. The next step is to compare and distinguish data from information so that we can clearly understand both of these concepts and their relationship.

According to the International Standards Organization (ISO) and ANSI, information is the meaning that human assigns to data by means of the known conventions used in their representation.[5]

There is now a rather widely accepted distinction between data and information. The distinction in definitions for data and information also has a potential circularity when considering data as the encoded representation of information and information as derived from data and useful in solving problems.[6]

---

[2] The New Oxford Dictionary of English, Oxford University Press, Oxford, New York, 1998, page 468
[3] Webster's New World Dictionary, College Edition, Toronto, Canada, page 374, 1962
[4] American National Dictionary for Information Processing, Washington, DC : Computer and Business Equipment Manufacturers Association (CBEMA), Report No. X3/Tr-1-77, 1977 September.
[5] Ibid
[6] Everest Gordon C., *Database Management Objectives, System Functions & Administration,* McGraw-Hill, p. 7, 1986

McDonough offered an early distinction between data and information. (See Figure 2.1). He argued, that data becomes information when evaluated in a specific situation or applied to solving a particular problem. That is, data becomes information when used to make a decision. Since value derives solely from solving problems, it is meaningful to speak only of the value of information and not of the value of data. Information is formed in the human mind when data and a problem come together; the supplier and the user of information must both contribute to making the product. The supplier or supplying system cannot produce a complete product - information – without the user who is faced with a problem.

Some consider McDonough' s position extreme, yet his opinion led to a key point: data requires interpretation to derive information, and the interpretation must stem from a specific problem situation. Consequently, it is meaningless to speak of an information processing system. No matter how much data is processed, it cannot be turned into information until a person uses it to solve a problem.

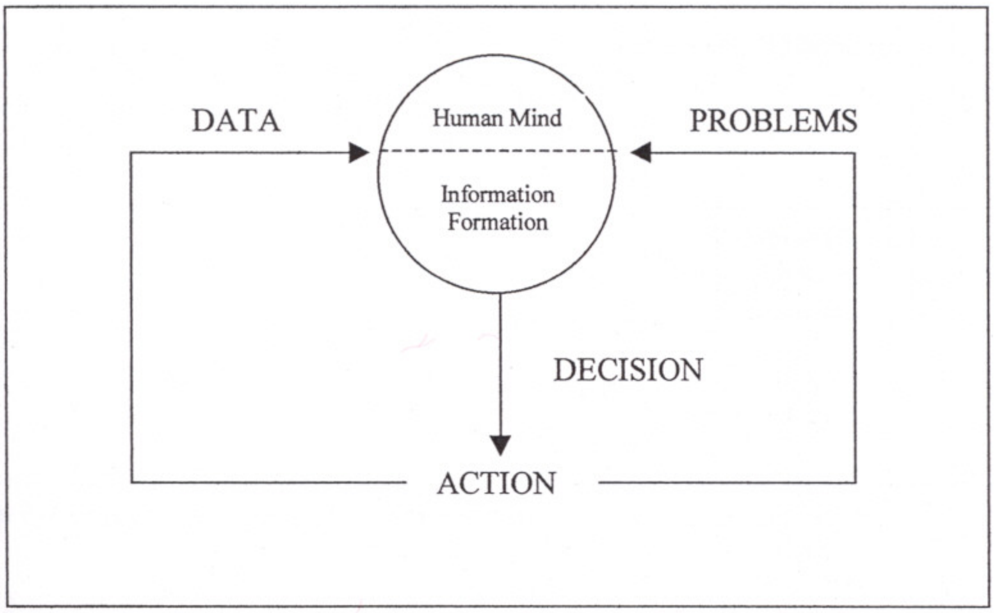


Figure 2.1 McDonough' s Distinction between Data and Information.

The term data is used here to represent messages that can be available to the individual but which have not as yet been evaluated for their worth to him in a specific situation. All communications in a firm may be considered as some form of data processing. Information is used here as the label for evaluated data in a specific situation. When the individual singles out one of his problems and finds among his data material that help him solve the problem, he is converting or isolating information from data. Note that a given message may remain constant in content and yet under this approach, change from data to information when it is put to use in making decision.[7]

Terence Hanold draws an interesting and clarifying distinction between data and information: Information has to do with communication of knowledge inspired by observation – with the interchange of thoughts and ideas proceeding from experience. Information is different in kind from data. Information has the attribute of

[7] Ibid, p.10.

communication which data does not have. In context of business, data is merely the digital shadow of haphazardly events indifferently recorded. Yet information begins with data. Data is transformed into information through the infusion of purposeful intelligence. Thus, information is data refined by intelligence so that it communicates meaning or knowledge.[8]

We have tried to clarify the definitions of data and information in this section, along with their interrelationships and logical dependencies. Different views for defining data and information leads us to a common understanding: Data has to be processed, filtered, reviewed, communicated, validated to produce information, which is useful to decision makers. Figure 2.2 shows the high-level process from data to information.

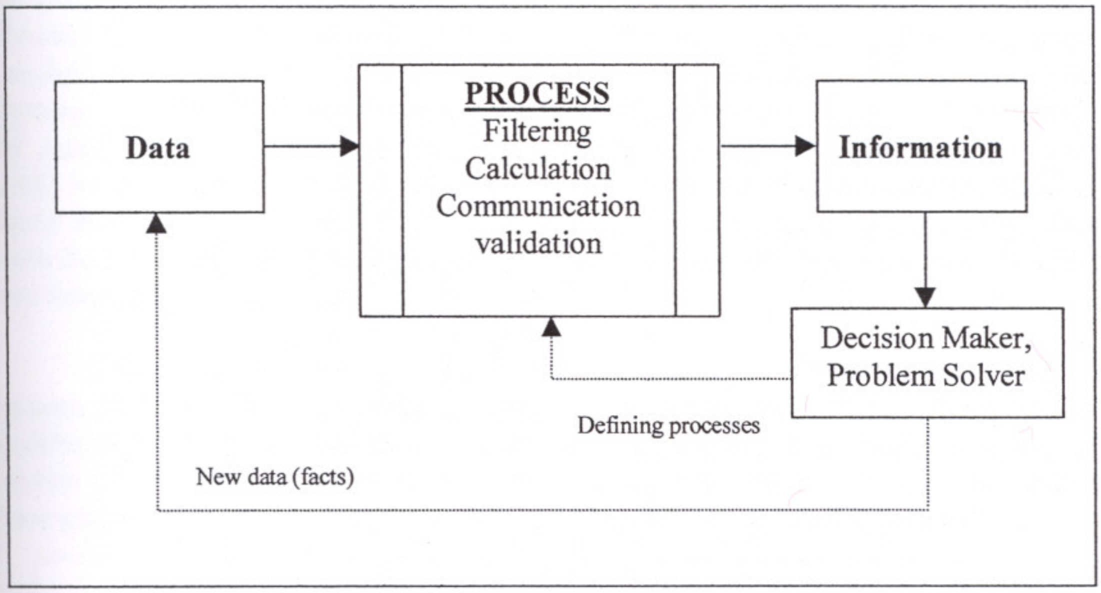


Figure 2.2 Process from Data to Information.

Information in general must have the following characteristics to be useful and valuable:

1. **Accuracy**: The known rule: "garbage in garbage out". No garbage data should be in the loop of generating information. Data must be accurate.
2. **Completeness**: Even though some might rightly argue that information is never complete and frozen, because of the continuing learning process, we still state the completeness of information - at a given time within given boundaries - as a requirement for its value.
3. **Simplicity**: Out of scope, too much information can distract the process of decision making by hiding the actually required information.
4. **Timeliness**: Out of date information or invalidated information would lead to wrong results and decisions.
5. **Verifiability**: Any information should be able to be traced back to the data behind it, and the processes that generated it for proof and verification purposes.
6. **Economical**: No process and system has unlimited sources for generating information. Therefore information must be economical to obtain.

[8] Terrance Hanold, An Executive View of MIS , Datamation (18:11), page 66, November-1972

2.2 Systems

As suggested by the name, information systems are specific types of systems. We will start from the basic system concepts to be able to describe and understand the nature of information systems.

We live in a world, full of systems, some of them physical and some others more abstract. A system can be defined as a collection of interrelated parts which taken together forms a whole such that, the collection has some purpose and a change in any of the parts leads to or results from a change in some other parts.[9]

This is a very broad definition of a system. The importance here is that, the elements or components of a system are interrelated. A single object or structure can not be defined as a system unless the elements of it are fulfilling the requirement of dynamic interrelation. Systems have inputs, processes, outputs, feedback and control mechanism. Furthermore we can say that components in a system could be as well sub-systems with their own inputs, processes and outputs. Most systems can be illustrated by general model given in the Figure 2.3, which also shows the intermediate storage entity of a system[10]. In this figure, internal details on various elements are shown as black boxes. These will be detailed and described in the following paragraphs, along with the other definitions such as system boundaries, connections between sub-systems and functional hierarchies within a system.

Another attribute of systems is their objectives, by which we understand the reasons of their existence. Objectives of a system can be clear; "easy" to observe and understand or they may be difficult and very complicated. The engine of a car as a system has a simple, clear objective whereas, political systems or economic systems have complex, multi objectives, difficult to define, understand and formulate.

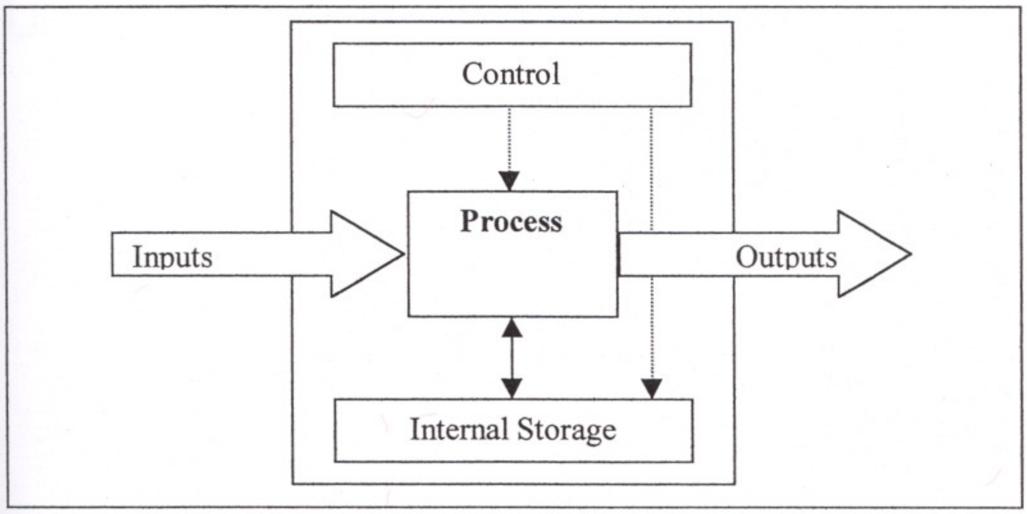


Figure 2.3 General System Model.

Inputs and outputs of a system depend on the nature of the system. These can be physical or abstract structures. The petrol for the car engine is a physical input whereas

[9] Curtis Graham, *Business Information Systems Analysis Design and Practice*, Third Edition, Addison Wesley Longman Ltd., p.13, 1998
[10] Ibid, p.14.

the view of a politician is an abstract input to an economic system. Followings are examples for system input and outputs:

- Physical materials
- Data / information
- Money
- Energy
- Labor
- Decision

Control mechanism in a system is the set of rules, conditions to ensure that the system objectives are fulfilled. Changing system objectives, processes, inputs and outputs would lead to changes in the control mechanisms. Control mechanism can be viewed as internal input to the system process.

Internal storage is the temporarily or permanent structures within the system, that the process needs to store, retrieve and update. A system would need to keep every intermediate output until the final one is achieved. Storage is also an environment for storing set of control rules and structures.

Figure 2.4 shows a more detailed model of a system where sub systems together compose the main system.

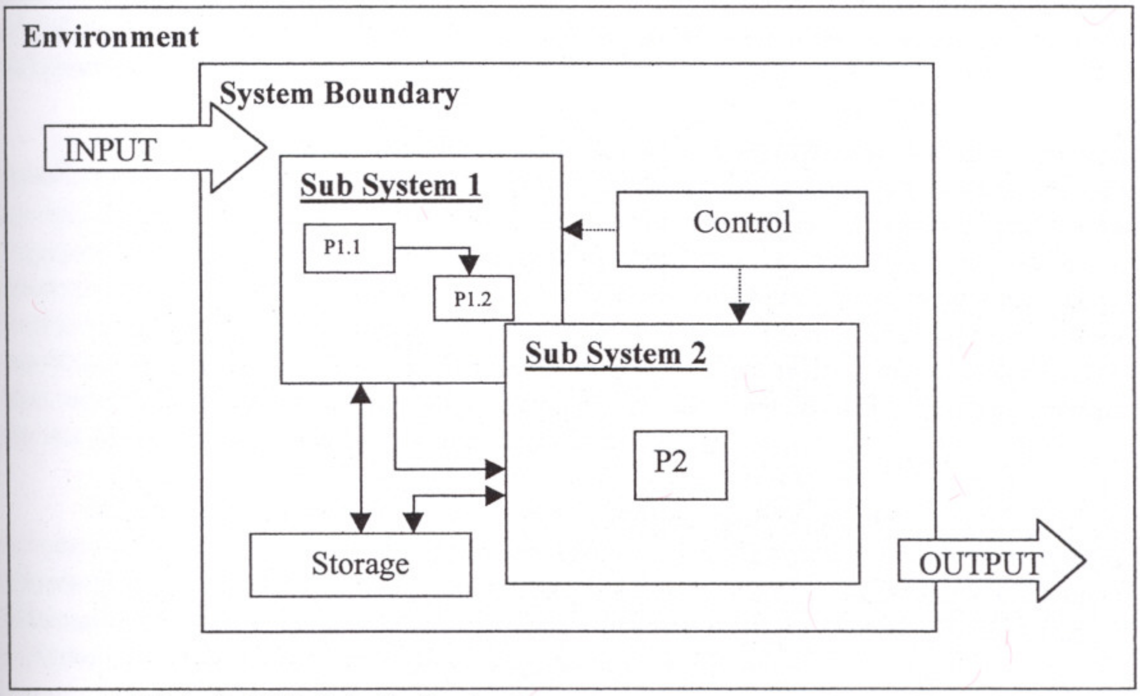


Figure 2.4 Detailed Model of a System.

Inputs come from, and outputs are transferred to, the environment of a system. Any entity outside the boundaries of a system can be described as the environment. The environment of a system is in interaction with it by giving inputs and receiving outputs. System boundary is a separation between the environment and system scope.

2.3 Information Systems

Information systems are the subject of the rest of this chapter. We will be describing various components, functionality and structures of information systems in the following sections. We will also present various definitions and discuss these definitions from different angels.

Information systems here will refer to computer based information systems, other than manual systems. The system part of the information system represents a way of seeing the set of interacting components, such as: people, objects, procedures and functions.[11] People follow procedures and execute functions to develop or generate information from the information systems as an output of data processing.

The term information systems has been defined as the effective design, delivery, use and impact of information technology in organizations and society. [12]

Buckingham *et al.* (1987b) defines information system as a system which assembles, stores, processes and delivers information relevant to an organization (or to society), in such a way that the information is accessible and useful to those who wish to use it, including staff, managers, clients, citizens. An information system is a human activity (social) system which may or may not involve the use of computer systems.

The definition of Buckingham emphasizes the "social" and "human" elements of information systems. According to this definition, information systems are not technology driven. What is more, the driving power behind the information systems are social and organizational requirements. Technology is seen more as a tool utilized in information systems.

We will review the evolution of the information systems from a technology and functional point of view – which also represents the social and human elements. Even though information systems are originating from the needs of various parties in organizations and society, technology behind them has become a more and more powerful resource, effecting the nature and utilization of the information systems. This, in return, has given wider perspectives for information consumers. Information system requirements and information scopes in organizations and societies are now far bigger than what they would have been fifty years ago. On the other hand information systems are not just technology and do not just mean automation.

Information systems study is at the center of many concepts mainly, people, software, data, communication, hardware, organization and procedures which makes information system development more than just an information technology (IT) task. Information systems are a part of an organizational solution, based on information technology, to a challenge posed by environment.

2.4 Architecture and Components of Information Systems

[11] Avison D.E, and Fitzgerald, G., *Information Systems Development : Methodologies, Techniques and Tools*, Second Edition, McGraw-Hill Companies, p.1,1995
[12] Ibid, p.2.

Computer based information systems are composed of various components interacting with each other to perform the desired goal. We will try to picture the general architecture and components of information systems. We will be considering the "modern" up to date architectures. This means that we will be taking samples from the up to date technological platforms and architectures. We will be also using the industrial experiences we have over the past years, where we have observed many types of information systems infrastructures in different types of organisations. The experience of real life information systems will be driving force for the discussions in this section.

People; managers, information workers, customers, developers are the first and main component of an information system. Information is only valid and valuable if it is utilised by people. Information systems are for people. Another view is that, it is people who design, implement and use information systems. It is nothing different than the Neanderthal Man making hand-axes for survival. We would not like to reduce the scope of information systems to a hand-axe development by saying this, but we would like to emphasize the needs and tool building characteristics of human being. Today, we need information systems, and so we build them.

Users, of an information systems can be in different levels. Data collectors, business experts, office clerks, production engineers, junior and senior managers and customers are examples of these users.

Another human group is the information system developers in organizations. Operators, programmers, business experts, managers, information workers, system analysts and database administrators are members of this group. No surprise that some of these people exists in both groups as users and developers. This is because, some people developing a system are also the very same people using the system. For example business experts are in these category.

The second component of an information system is the computer technology and various computer systems. Nowadays, computer systems in organizations are more like a collection of various architectures. Distributed systems, centralized mainframes, client server systems, various operating systems, database management systems and file structures co-exist to build up the overall computer platform for information systems.[13] Using local and wide area networks, these architectures can communicate, and various application programs can share the organizational, and external data.

Database management systems are an essential part of information system architecture. Everyone has data.[14] Data is stored and controlled in databases. An automated database is a mechanized, shared, formally defined and centrally controlled collection of data used in an organization.[15] We will explore the database concept in much more detail in the following chapter.

Many organizations have information systems with in-house developed applications and third-party packages. Third party packages more often are for a

[13] Uludag Memet, 00 Yilina Hazirmisiniz?, TMMOB Elektrik Muhendisleri Odasi Izmir Subesi Bulteni, Yil: 11, Sayi:110, Haziran 1999, p.20

[14] Looney Kevin, Koch George, *Orace 8I: The Complete Reference,* Osborne/McGraw-Hill, p.6, 2000.

[15] Everest Gordon C., *Database Management Objectives, System Function & Administration*, international student edition, McGraw-Hill Book Company, p.11, 1986

specific process within the organization. These are developed and maintained by an outside "expert" company. Payroll system, financial systems are a few of the examples where such packages come into picture.

End-user computing systems (EUCS) are a relatively new approach in information systems. In parallel to using in-house systems and packages, various business departments take the approach of developing small mostly stand-alone systems to assist on their daily activities. These systems are called the end-user computing systems. End-user computing also refers to the ability of non-computer employees to create their own systems.[16] There are various views on EUCS systems and on their advantages and disadvantages which we will be not discussing in this study.

Local area (or organization) computer network is the communicator for the systems. It serves for all well known benefits described in network studies. As for our purpose we know that developments in the communication technology made much faster and reliable communication networks possible.

A general organizational information system computer architecture is shown in Figure 2.5. It is clear that this representation is not always the case. By giving such a view we are trying to add every possible component into the picture to have a possible wider understanding and to see a bigger scope of information systems.

A higher level (functional) view of information systems is pictured by including external (customers, suppliers, co-operating organizations) and internal functional entities (departments, data flow, information flow, users). This is shown in Figure 2.6, where the computer component detailed in Figure 2.5 is shown as a black box.

Various functions within an information system scope are processed by different groups. Development, execution and maintenance of an information system are carried out together by business experts, workers, analysts, and managers. Various business areas will continuously feed in data and gather information from the information system. In fact, data and information will be shared by these different groups. Each of them will probably be more interested in a different format and content of information generated from the same pool of data. Data will be administrated by the database management (administration) entity; a collection of systems and people. External components (environment) will have a bi-directional data and in fact information flow to and from an organizations information system. Customers, co-operating organizations, vendors, institutes are types of external components. In Figure 2.6, data and information flow is shown with arrows. This is a general view of an organizational information system. Again we need to highlight that not always each of these functional relationships exist between various components. In reality, different kind of business areas (manufacturing, finance, health etc.) will have different structures, dictated by the nature of the organization.

[16] Department of Accounting and Business Law of the James J. Nance College of Business Administration at Cleveland State University, URL: http://www.csuohio.edu/accounts/

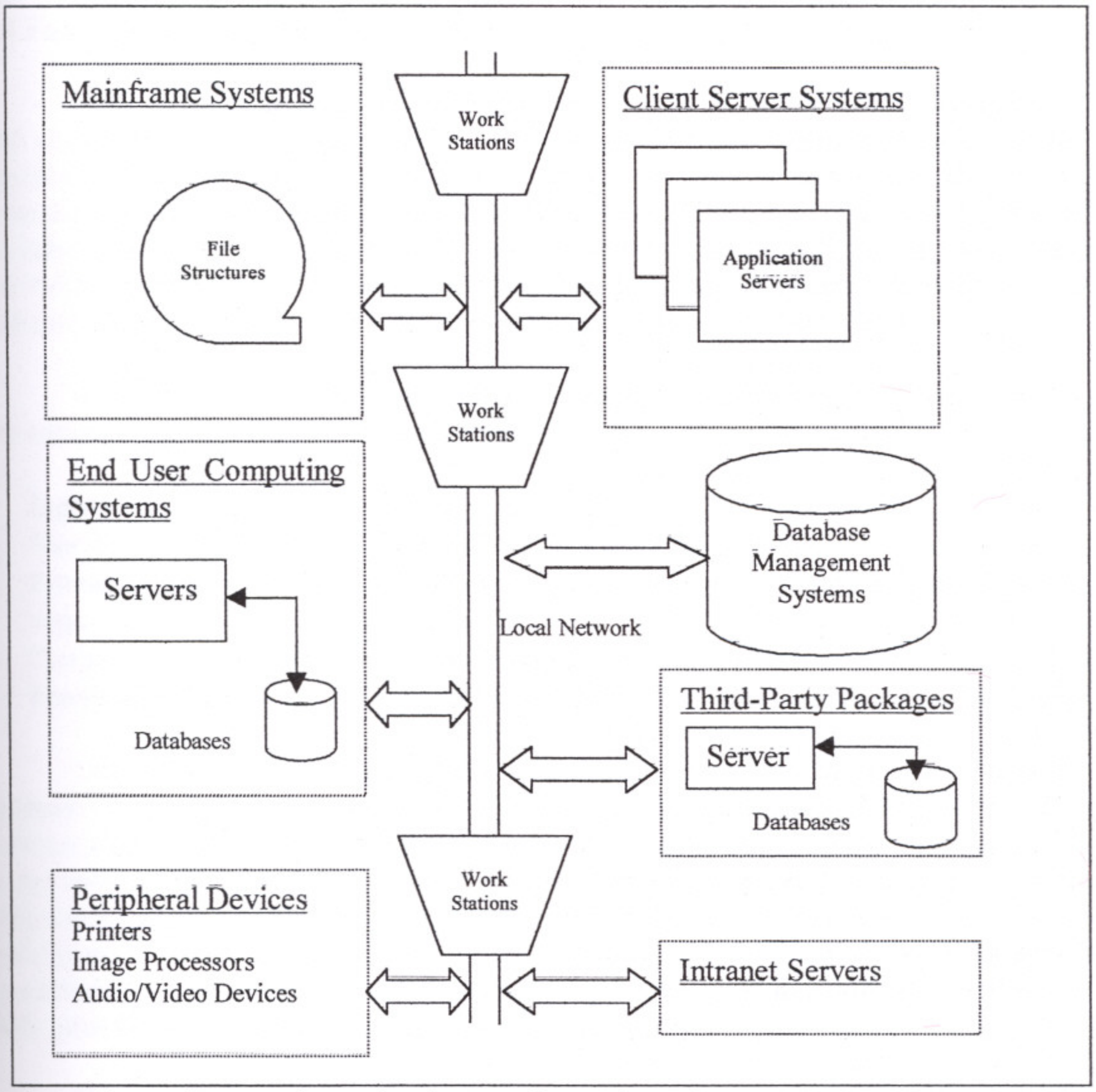


Figure 2.5 A General Organisational Information (Computer) Architecture.

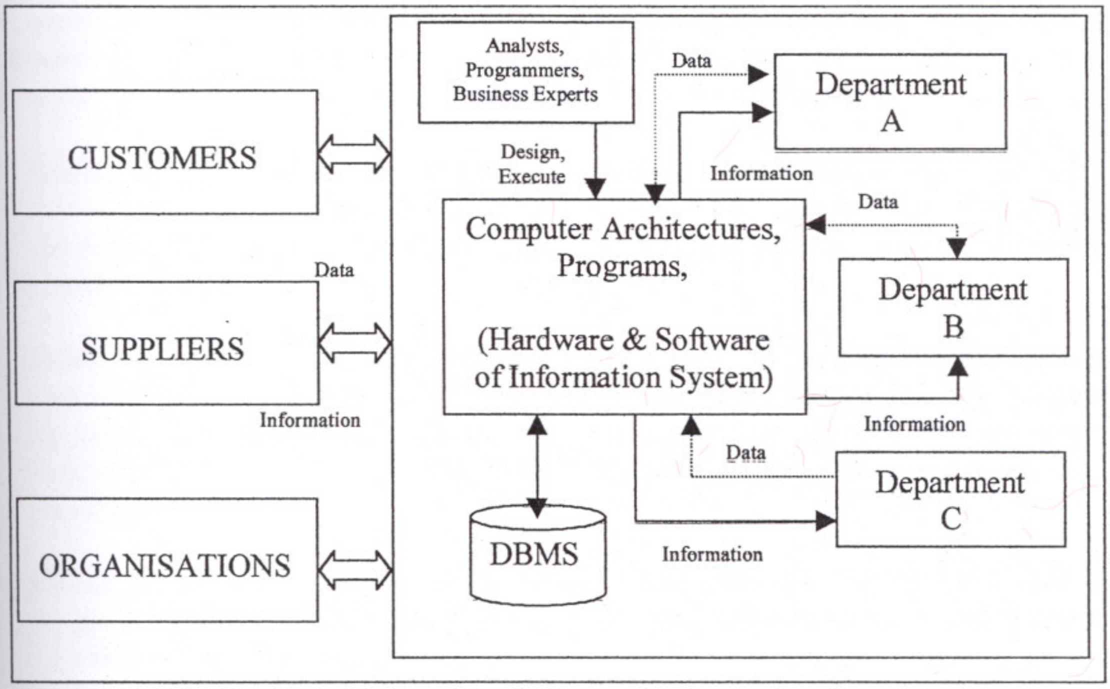


Figure 2.6 A Higher Level (Functional) View of Information Systems.

2.5 Functions of Information Systems

Until now we have described information systems, defined their architectures and components. During the previous sections we have also given indications on the functions of information systems. In this section we like to summarise the overall information systems functions. Regardless what the architecture, complexity or type of an information systems is, the general function descriptions we will give in this section are valid. Since we are not working on a specific information system, we will not give any specific functionality of any specific organisational information system.

A high level list of functions of an information system are given below. These are common, general functions valid for almost every information system.

- **Input**: Data collection, transaction processing.
- **Storing**: Storing business data. Involves database management.
- **Processing**: Converting data to information. Manipulation of data in many different ways.
- **Output**: Producing different types of outputs (as information).
- **Feedback**: New data generation from produced information. Serving as a feedback.

The list of functions we have given above are defined in isolation from the business functions of an organization. In fact, to be able to describe the functions of an information systems, first we need to describe the business functions of an organization it has born to. We need to understand the structure of the organization along with the business processes and objectives. Another very important aspect is that we need to understand the decision making methods. At the end of the day, we define information systems as very sophisticated tools for supporting the business functions and workers on daily activities, planning, reporting and decision making.

Every organization is unique. As complex systems, no two organizations can be exactly the same. But we can still give a general description of common function in organizations. The exact internal details of these functions (how they are executed, managed etc.) can be different from one organization to the other. Below, we have given the brief descriptions of these common functions in organizations:

- Purchasing: Regardless of their type, organizations need material or service from environment. An organization cannot be fully isolated to operate without receiving any service or material from outside. Therefore purchasing is a common function for organizations.

- Sales: Organizations produce “something” to sell or provide. Whatever an organization sells can be in fact the purchase of another. Service and products are the sellable outputs of organizations. We should note that products are not always finished goods, since they can be sometime raw material for another organization to process it.

- Human Resources & Man-Power Planning: “Things” are done by people. Labor is the greatest value. To do things, people need to be recruited, organized and planned. This is a common function and requirement for every organization.

- Manufacturing: To sell or deliver "something", it has to be produced. Sometimes this can be a service development where not a physical product is produced. Manufacturing is the function of the production in organization.

- Logistics: "How are products going to be delivered?" or "how will the raw material be available for production?", or "how will people be transferred from and to work place ?" These are some of the questions and problems the logistics function is dealing with.

- Accounting / Finance / Operations: This is a function sitting on top of the others. It can be seen as a function for assessing how things are going in the organization. In a sense, accounting and finance is the function that probes, measures and report on the status of other functions.

We like to include information technology, training, engineering and planning to the list of common functions in organizations at present.

All given functions above have their own objectives and tasks. What is more, they have different priorities and problems. They may be independent within them selves but in reality all of them actually depend and rely to each other to process. None of these function can be isolated from the others to meet the objectives of the organization. Different level of data and information flow exist between these functions. They have interfaces between each other.

We have to now place information systems to the organizational picture drawn above, and overlap the general information system functions with the business functions so that they make sense. The purpose of information systems in organizations is to integrate the activities of different departments (functions) into a single business system that produces coordinated, integrated responses to its environment.[17]

In Figure 2.7 we show the organizational functions and the relationship between them. As we have mentioned before, every organizational function has its own local objectives and structures. Therefore, up to a certain level they will utilize information systems to support their own operational information needs. Furthermore, different functions can have dedicated software packages (information sub-systems) for themselves. But from a higher level view, the data these departments provide to the information systems will in fact be shared and utilized by others which will build up the organizational information. Considering the human resources department for example, we could say that it would use a specific payroll system for its own operations. The payroll system would not be a tool for other departments as such. But the data provided by human resources department would be utilized by other departments and other sub systems, which then will make this data an organizational level data.

Considering the structure given in Figure 2.7, it is not a real life exercise to have just one information system component to equally support all functions in an organization, Therefore we have given the information systems entity within each function to show how in reality these are utilized.. This does not imply that these

[17] Kroenke David, *Management Information Systems*, McGraw-Hill Book Company, p.454,1989

information systems are in isolation or physically divided. In fact it shows the different functional systems with local information structures and their communication. The total of these information systems entities compose the organizational information system.

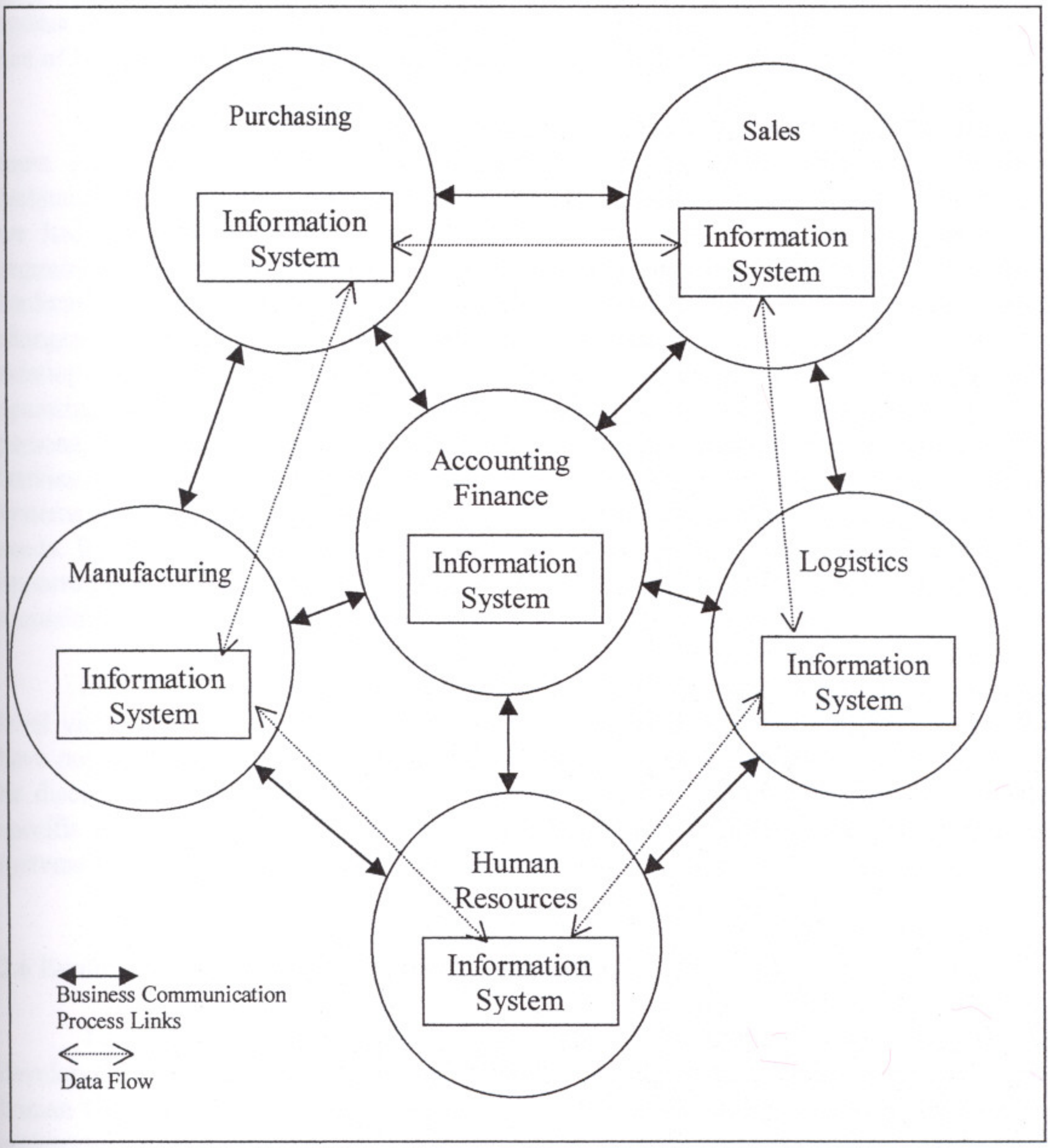


Figure 2.7 Organisational Functions and Their Relationships.

Because organizations are complex enough, information system for them need to have certain characteristics. First of all, there should be agreed and documented standards for usage, maintenance and scope. Workers, managers, everybody involved in the process should know and use these standards. Standards must be useful and they must be more people-oriented than technology oriented. Because it is people who will apply them or not, it is important to have business people, users, involved in agreeing on standards. Sometimes having standards and not applying them is worse than having no standards at all.

Information systems change by time. Parallel to the organizational changes. Some of them die, some of them get re-designed or some of them grow in scope and developed further. Changes required for information systems should be controlled. As we have mentioned above, in general, no information system in any function of the business is just isolated and stand-alone. Therefore, the impacts of changes to one system need to be analyzed and verified before any change is applied. Changes are a fact of life. It is to the control mechanism how smooth these will occur or how painful.

Modern technology in information systems has introduced many benefits for users and developers. It has also given new directions to information system architectures. But there is another side to this fast advancing technology. In Figure 2.5, we had given some of the technological architectures which can co-exist in an organization. The base for co-existence is the technological compatibility of various hardware and software systems. Especially in software industry, we see very rapid changes and upgrades. It can be sometimes chaotic for organizations to keep the existing compatible systems and at the same time upgrade versions, platforms and operating systems. Because information systems are not just a few programs on a personal computer, uncontrolled and unplanned changes, causing to interruption of the business can not be acceptable and affordable. Changes on technology in information systems should be coming mainly from the business requirements and organizational needs. By this we do not ignore the fact that there are also technological reason and opportunities why changes to the system should be applied. But this is should be a secondary reason after the business needs.

In this section we have analyzed the functions of information systems at a high level view. These have been mapped to the business functions of organizations. We have not yet brake down the general information systems into various types. This will be discussed in Section 2.7. For each type given, we will also briefly describe the specific functions. This will enable us to understand how different types of information systems fit together into organizational functions.

2.6 Evolution of Information Systems Technology

To understand the evolution of the information systems, we need to study the developments in the computing area over the last decades. Despite emphasising the human (information workers, managers, etc.) role in the overall information systems architecture, developments in the hardware and software technology have made important impacts on forming the modern system architectures. We would like to repeat that information systems development is not just equal to automation but automation plays an important role on the architecture, development and utilisation of information systems.

The field of information systems has grown dramatically over the past three decades. Recent trends have transformed the information system landscape. These trends include the evolution of implementation technology from centralised mainframe environments towards distributed client server architectures, embracing the internet and intranets; changes in the user interface technology from character -based to graphical user interfaces, multimedia, and the World Wide Web; changes in applications from transaction processing systems towards system supporting collaborative work; and the use of information technology as an enabler of business process reengineering and redesign.[18]

We have studied the evolution of computers and information systems by the known method of "generations approach" because, computer hardware, and software, can be considered to have evolved through a series of "generations".[19]

<u>First Generation, mid 1940's – mid 1950's:</u>

The science of electronic allowed the fist electronic computers to be build in 1940s. This was centered around the electronic valve, a device of the size of domestic light bulb. It consists of electrodes enclosed in a glass bulb, which is then evacuated. The valve is responsible for regulating and amplifying flows of electricity. It is usually agreed that the first general-purpose electronic computer was built in USA and was called ENIAC.

In 1946, a group of scientists and engineers at the University of Pennsylvania's Moore School of Electrical Engineering quietly inaugurated a revolutionary way of managing information. They called it the ENIAC (Electronic Numerical Integrator and Computer). It gave rise to the modern computer industry and would eventually transform people's lives to a degree that even its inventors could not have imagined.[20]

First Generation computer were

- Vacuum tube based (short-lived, generated a lot of heat, bulky, slow)
- Huge machines in size
- Large amount of manual work involved in running programs
- Used for specialized numerically based applications (scientific, census, engineering and military.)

<u>Second Generation, mid 1950's – mid 1960's:</u>

- The second generation computers were based on the new technology of transistors.
- Magnetic core memory increased the size to multiples of Kbytes.
- These were still, larger machines, requiring dedicated human (operator) activities to run.
- They used less power, produced less heat and therefore were more reliable.
- The timesharing mainframe machines were developed which would be for lease more than for sell.
- Programs were written in specialized languages such as COBOL which enabled the development of more complex business systems. Batch processing was applied. Payroll systems, scientific computing systems were developed.
- File systems were used for data storing and processing.

---

[18] Hirschheim Rudy, A Comparison of Five Alternative Approaches to Information Systems Development, Australian Journal of Information Systems Volume 5, Number 1, 1997

[19] Hart Dennis & Toomey Warren, History of Computer and Information Systems, URL: http://www.cs.adfa.edu.au/teaching/studinfo/csis/Lectures/topic3.html

[20] University of Pennsylvania School of Engineering & Applied Science, ttp://www.seas.upenn.edu:8080/ ~museum/ overview.html

Third Generation, mid 1960's – late 1970's:

- Developments in LSI technology enabled the third generation computers.
- Third generation computers were smaller, cheaper and more powerful computers than second generation.
- Availability of more sophisticated software (operating systems) made them easier to use. Computers began to be interactive. Office computers emerged running business applications.
- Networking technologies were used which increasing the shared utilization of computing powers.

Fourth Generation, early 1980's –now:

- VLSI technology is developed. Much more power, memory and disk capacity is available in this generation of computers.
- Fast, powerful computers lead to development of more complex, business oriented user-friendly information systems. More people have now access to personal computers and workstations.
- Database systems (relational) have progressed dramatically supporting development of information systems. Development tools and programming environments are now suitable for complex business system implementations.
- Multi-media systems are introduced. Distributed systems, client server architectures are developed.
- Communication technology has made dramatic progress. Internet is being used by millions of people.
- Computers became the essential daily tools in offices and personal activities.

As we see from the brief history, computer information systems evolved along with the developments in the technology. Huge computers within the walls of laboratories or military bases became powerful workstations available to offices and information workers. Thus leading to the development of very complex but user friendly information systems of any type.

The future of computing is being shaped in the present days. The impact and opportunities Internet brought to our life is already there. Geographical boundaries on communication and information sharing are disappearing whereas the privilege of owning the technology and information becomes a more and more powerful force, may be introducing new digital boundaries between nations and continents.

2.7 Types of Information Systems

Information systems can be categorized in various ways depending on what would be used as a criteria on categorizing them. In the following sections we will be giving a more functional and organizational category of information systems. Some other perspective would be the "criticality", "scope", "infrastructure" (hardware / software), "architecture" (centralized information systems, distributed systems architecture, Internet technology based architectures, etc.).

We will be identifying various information system categories according to their functionality within the organizations. The types of information systems described below are Transaction Processing Systems, Office Automation Systems, Knowledge Work System, Decision Support Systems, Management Information Systems and Executive Information Systems.

2.7.1 Transaction Processing Systems (TPS)

Transaction processing systems can be viewed as the most data oriented and least "intelligent" category of information systems. These are more for storing or retrieving data rather than generating information. Transaction processing system support day-to-day operations.[21]

A reservation system, order placement system are examples of transaction processing system. These system are often at or near the boundary of the organizations and close to their environment such as customers, suppliers. Transaction processing systems can be seen as the data source of an organizations, required to operate. These systems are often the providers of data for other types of information systems.

Transactions processing systems are the oldest of all information systems types, developed in 1950s in accounting department of major corporations.[22] This does not change the fact that we are still heavily using them in organizations and daily life. Starting from early terminal based centralized mainframe architectures to internet technologies and web pages, transaction processing systems are still well in service of organizations and people. Many credit card transactions made on today's Internet sites are actually typical examples of transaction processing systems with running on highly advanced technological platforms. Another specific area of transaction processing systems is the so-called "hand-held devices" utilized mostly by mobile sales and order processing workers.

Another characteristics of transaction processing systems is that these are exposed to a more public usage unlike some other type of information systems which are more management or business experts oriented, within the organization. Recalling the example of web sites for credit card transactions, will give us an idea how many different types of customers, with different approaches, understandings and needs will actually utilize such a system. Despite of their limited functionality in comparison to other information system types, transaction processing systems will surely have a wider range of users. Therefore functions and routine processes within such a system should be well defined and supported.

Figure 2.8 shows a typical architecture of a classical transaction processing system. Figure 2.9 in contrast shows a more distributed and internetworking architecture of transactions processing systems.

In Figure 2.8, in a classical centralized environment, transactions are generated by a terminal given as the transaction medium and directed to the application(s) for

---

[21] Kroenke David, *Management Information Systems*, McGraw-Hill Book Company, p.29,1989
[22] Ibid, p.29.

processing, storing data and generating straight forward reports. Data storage can be in any form from file structures to databases.

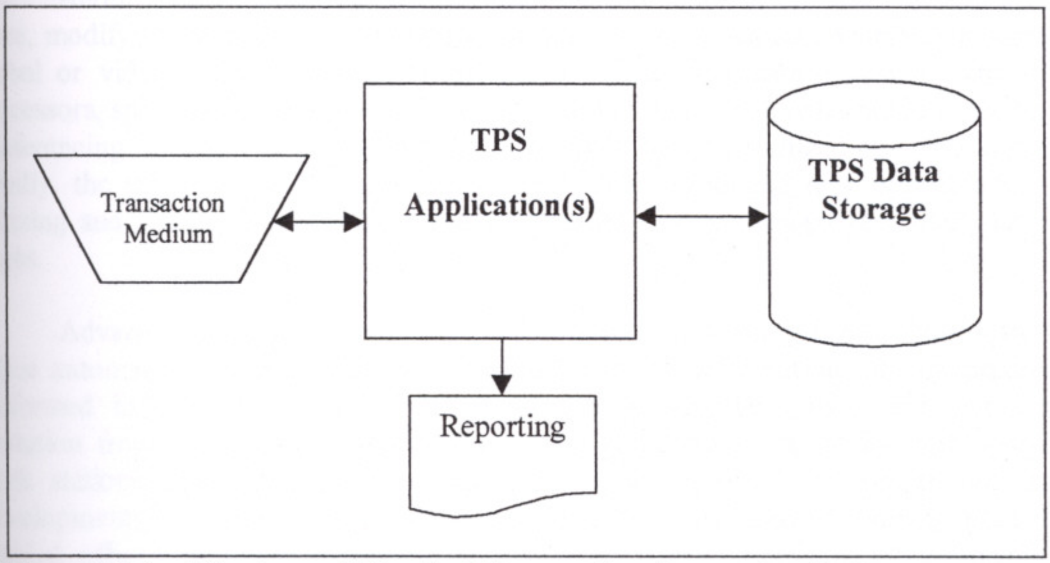


Figure 2.8 Architecture of Classical Transaction Processing Systems.

In Figure 2.9, the concept of geographically distant transaction mediums is introduced where, the TPS application is receiving transactions through the Internet from various decentralized medium. Internal components are like in the classical architecture, the TPS application, reporting and data storage. Transaction medium are external components of the overall system.

Transaction processing systems can be both, batch or on-line. Figure 2.9 is an example for on-line systems. On line systems provide immediate results for a single transaction at a time. In batch systems, transactions are collected, grouped together and processed as a set. This is called batch processing.

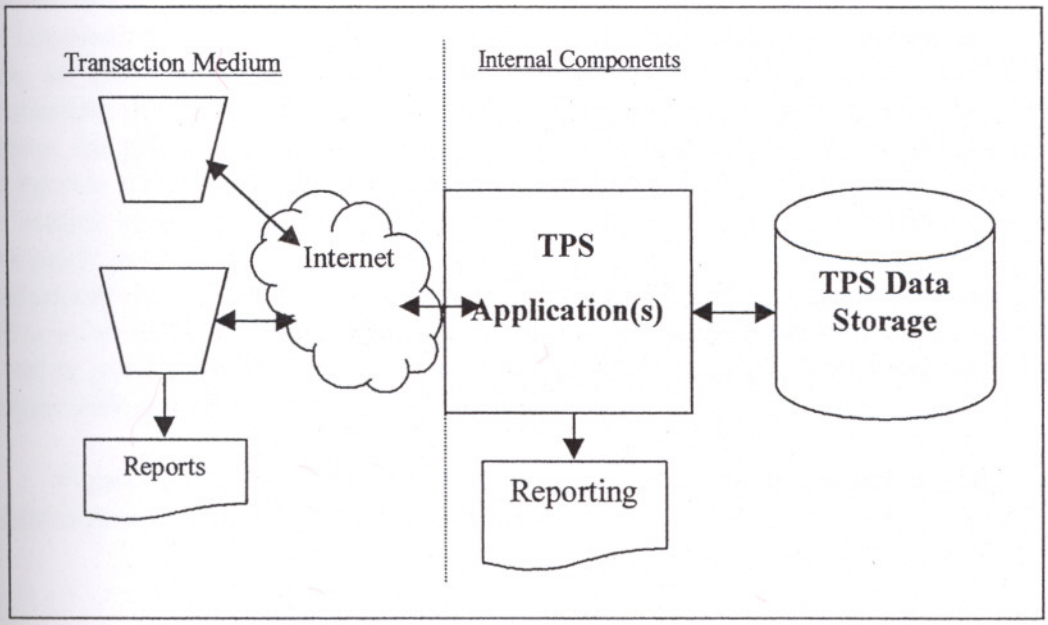


Figure 2.9 Internetworking Architecture of Transaction Processing Systems

2.7.2 Office Automation Systems (OAS)

Another type of information systems is office automation systems, that create, store, modify display and communicate business correspondence, whether in written, verbal or video form.[23] Typical examples for office automation systems are word processors, spreadsheet tools, e-mail, voice mail, planning and presentation tools, video conferencing facilities, and stand alone, relatively simple customized database systems. Finally, the utilization of Intranet technologies has introduced new opportunities on utilizing and sharing business data and information between employees over the web pages.

Advances in computer hardware and software technology have a big impact on office automation systems, changing the way how communication, documentation is performed in today's office environments. The computing power and technology evolution from centralized mainframe and user terminals to powerful multi purpose work stations have changed in many ways how the tools in offices are used. Developments in communication technology and fast local area network architectures enabled offices to move from initial standalone word processing environments to networked, shared processing powers.

Utilization of office automation systems has lead to many different business oriented systems like document archiving and imaging systems; on the job training utilities, internal (organization) libraries, standards and procedures catalogues. Furthermore, physically separate offices of an organizations can now share, access data and information without the barriers of distances. The more workers became familiar with the computing tools, the more customized information they could gather or produce.

Developments described above have also lead to new concepts in the offices. Sharable and easy to access data has lead to the emphasis of security and privacy issues. Because of the wider range of access to the office automation system and therefore to the information, the vulnerability of the data and information has become more and more an issue. Protection for digitally accessible files, e-mails with confidential information, databases with sensitive data has become more important than ever. What is more, the privacy of the employees, customers and other parties in an organization has become critical. We believe in future these issues will be more under discussion. We assume common sense ethical parameters are in place in every organization, but how much more has to be done to ensure security, privacy of personal and business information? This is not clear yet. Is it, for example, ethical, managers to monitor their offices by utilizing video conferencing or close circuit camera systems which normally should be great tools for overseas communications? Or should it be normal, monitoring workers desktops with some "office spy" tools?

Figure 2.10 shows typical architecture of automation systems with most common hardware and software components.

[23] Kroenke David, *Management Information Systems*, McGraw-Hill Book Company , p.55, 1989

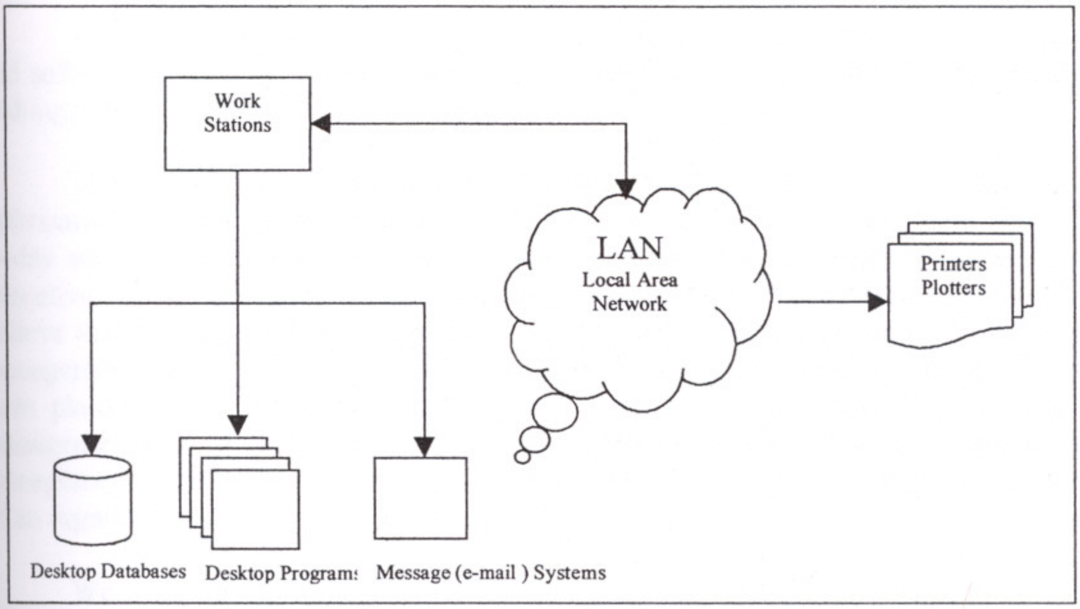


Figure 2.10 Architecture of Office Automation Systems

2.7.3 Knowledge (Engineering & Scientific) Work Systems (KWS)

A very specific information system type is the knowledge work systems. These are sophisticated special purpose engineering and scientific systems utilized by experts in an area. Most typical example of such a system would be the computer aided design (CAD) tools. Other examples are, special testing and analysis systems, simulation tools. Unlike office automation systems, these are very much customized systems. Knowledge work systems, are not general-purpose tools for many different types of usage. They generally need very powerful computer hardware to run. Surrounding equipment for such systems are generally special hardware unlike a shared printer in an office. One commonality between office automation and knowledge work systems is that both are mostly third party products developed by expert companies and are not in-house built.

2.7.4 Management Information Systems (MIS)

Probably the most popular name in information systems area is the management information systems or otherwise known the MIS. In this study, we are categorizing information systems into various types according to their architectures and functionality. But some resources use the name management information systems to cover all different types. In this approach, the name management information systems replaces the general term information systems. Some other resources call it business information systems. We will use the name management information systems and we will distinguish MIS from other types. The database development model and case study given in this study will based on a management information systems exercise.

Gordon Davis describes management systems as an integrated, user-machine system form providing information to support operations, management analysis and decision making functions in an organization. The system utilizes computer hardware and software; manual procedures; models for analysis, planning, control, and decision making; and a database.[24]

The description of Davis puts emphasis on three aspects of management information systems: decision support and planning, operations management and day-to-day activities, database component. Considering the decision support systems and executive information system more for decision making and long term planning we believe that the management information systems fall more into the area of operations management. Their main objective is providing information, enabling short - middle term planning for managers. Databases are essential components of computerized management information systems. An effective MIS cannot be built without viable data management tools. An important key to a successful MIS is the effective management of an organization's data resources.[25]

We will describe in the section 2.8 the interrelationship between various information system types more in detail but as for now, generally, transaction processing systems are the main data providers for management information system. Transactional data provided by TPS are transformed to valuable planning and monitoring information in MIS which then could as well serve as an input to more higher level decision making support systems like decision support systems and executive information systems.

A typical architecture for management information systems is given in Figure 2.11. The data flow from TPS to MIS is shown in the figure where TPS is an external but important entity for MIS. Report represent any type of information obtained from MIS. These can be text reports, graphical outputs, etc. Experts and users represent the human component of the MIS architecture for whom the information systems are developed for and who master the system. Database component of the MIS will be discussed in detail in the following chapter. What is more database themselves are complex enough systems. Database management systems composed from both, machine and human components are developed to handle the data background aspects of management information systems, and in fact every, information system.

Day-to-day reporting and information needs are handled by management information systems. Short-term planning, historical analysis are possible with management information systems because of the intensive data collection by transaction processing systems at the background.

2.7.5 Decision Support Systems (DSS)

Organisations do not only need routine, day–to-day data collection and reporting tools (TPS and MIS). They also require systems for helping decision making. Decision support systems are interactive, computer based facilities for assisting people making business decision. Decision making is not always a routine task. In fact it is more an ad hoc rather than a standard process. The difference of DSS from TPS and MIS is that,

[24] Gordan B. Davis and Margrethe H. Olson, *Management Information Systems: Conceptual Foundations, Structures, and Development*, second edition, McGraw-Hill Book Company, 1985
[25] Everest Gordon C., *Database Management Objectives, System Function & Administration*, international student edition, McGraw-Hill Book Company, p.18, 1986

DSS do not always support an ongoing process. They support the even driven, opportunity and problem related decision making.

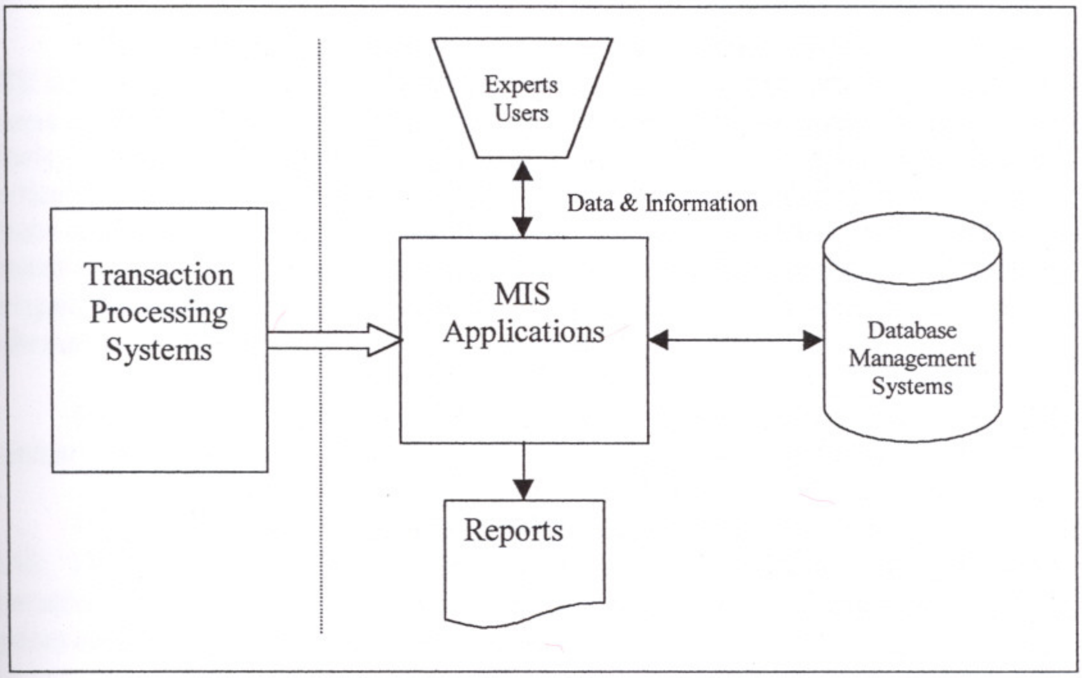


Figure 2.11 Architecture of Management Information Systems.

Transactions and regular reports are mostly standard, so are the TPS and MIS. But decision making is not. It is more flexible, variant and less structural. So are the decision support systems.[26]

Decision support systems are more sophisticated. They have more modeling and analysis power than management information systems. TPS and MIS are the internal data sources for DSS but external data sources are utilized as well. Decision support systems have relatively smaller number of users. Strategic decision makers, long term planners are the two groups of users in DSS area.

A general architecture for DSS is given in Figure 2.12. Here we have shown the data flow from TPS and MIS to DSS. Office automation tools are mostly used in decision support systems because they are flexible for ad hoc reporting, graphical presentation, documentation. Here links to external data sources and DSS model data is shown. DSS can store and use its own model data.

2.7.6 Executive Information Systems (EIS)

After seeing so many different types of information systems one may ask the question: "Why was there a reason to developed executive information systems?" An answer to this question comes from Watson & Rainer in Floyd Kelly's paper: "Information systems have long been used to gather and store data, to produce reports

[26] Kroenke David, *Management Information Systems*, McGraw-Hill Book Company, p.55, 1989

for workers, managers. However, senior managers rarely use these systems directly, and often find the information to be of little use without the ability to explore underlying details."[27]

This may explain the need for executive information systems. Differing from MIS and DSS, an executive information system is a tool that provides direct on-line access to relevant information in a useful and navigable format. Relevant information is timely, accurate and actionable information about aspects of a business that are of particular interest to the senior manager. The useful and navigable format of the system means that it is specifically, designed to be used by individuals with limited time, limited keyboarding skills and little direct experience with computers. An EIS is easy to navigate so that managers can identify broad strategic issues and then explore the information to find the root causes of those issues.[28]

For executives to see issues, executive information systems should consolidate, summarize and present information at the very high level within the organizations.

Figure 2.13 shows a general architecture of executive information systems. OAS, TPS, MIS and DSS are information systems providing data and model to executive information systems. Like in decision support systems, executive information system can build and store their own model data.

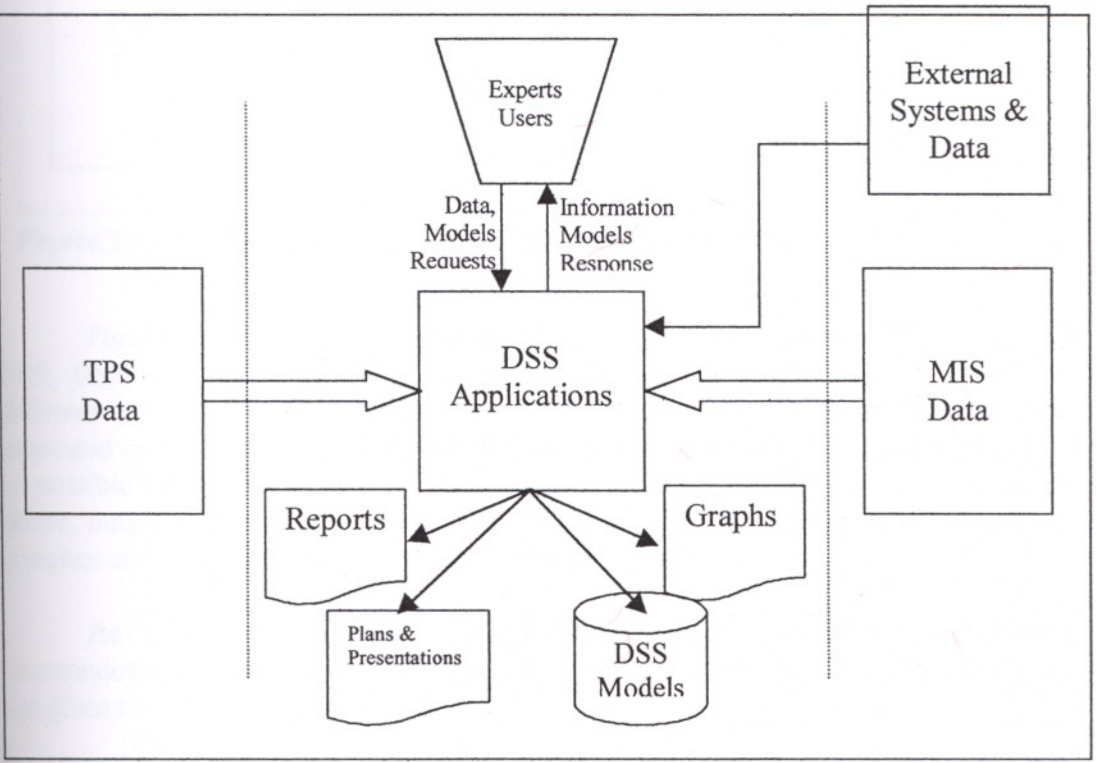


Figure 2.12 Architecture of Decision Support Systems.

[27] Kelly Floyd, Implementing an EIS (Executive Information System), EIS References, (Watson & Rainer, 1991), URL: http://www.ceoreview.com/papers/eis.htm
[28] Ibid.

2.8 Interrelationship between Information Systems Types

The final section of this chapter is about the interrelationship between various information system types we have seen so far. In DSS and EIS we have shown the links to other types of information systems where data from other types were used by DSS and EIS. To have a complete picture of information systems we have put them together and shown their data flow relationship.

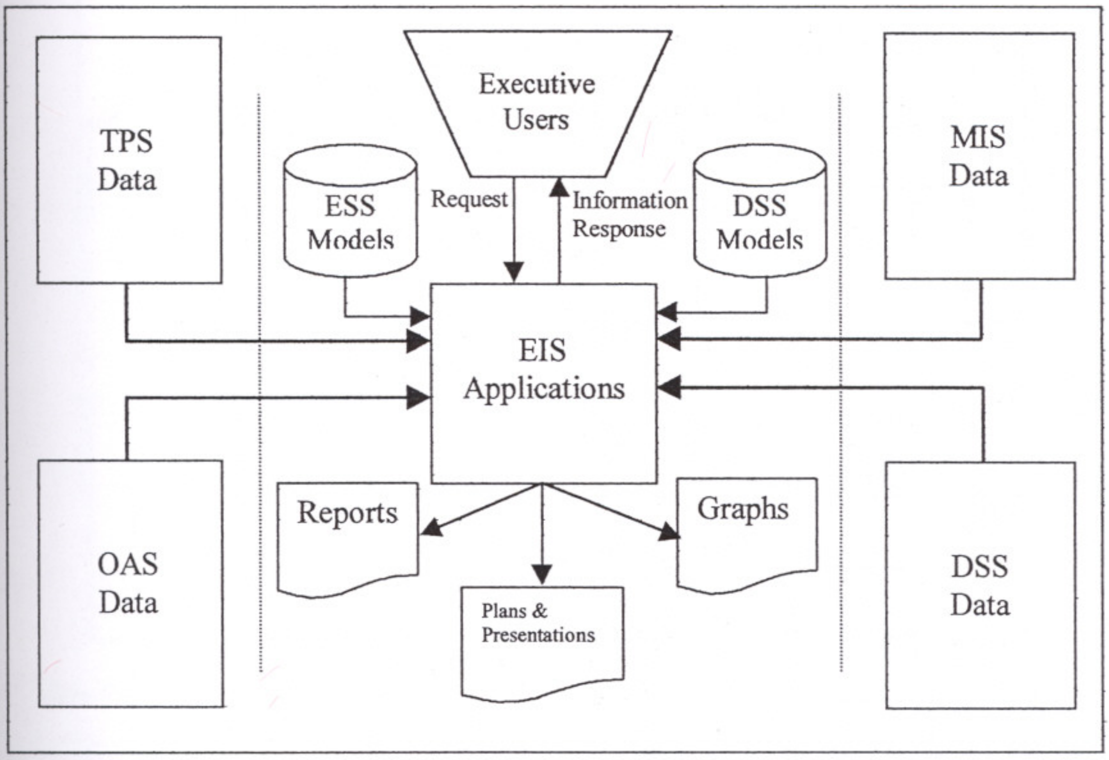


Figure 2.13 Architecture of Executive Information Systems.

Figure 2.14 shows the data dependency and flow between TPS, OAS, KWS, MIS, DSS and EIS. This relationship does not always exist. What is more, not all different types of information systems co-exist in every environment. But since we are interested in the rather wider and full picture than to some specific cases, we have given all possible links and co-existence. As we move from TPS to EIS, in the given sequence above, outputs are more and more consolidated. The data background becomes more complex and data comes from various sources.

As we have mentioned in the previous sections, information systems are also communicating to third party packages and external systems. For the simplicity we have not given these link in Figure 2.14.

We also think that KWS are different in nature from the other information system types. Because they are more for engineering, design and scientific areas, the usual data link between them and other types would not necessarily exist. We assume there is actually a human entity between the KWS and other types. Furthermore we assume that the flow of data from/to KWS and MIS or DSS is through the human entity what we call as a more manual link

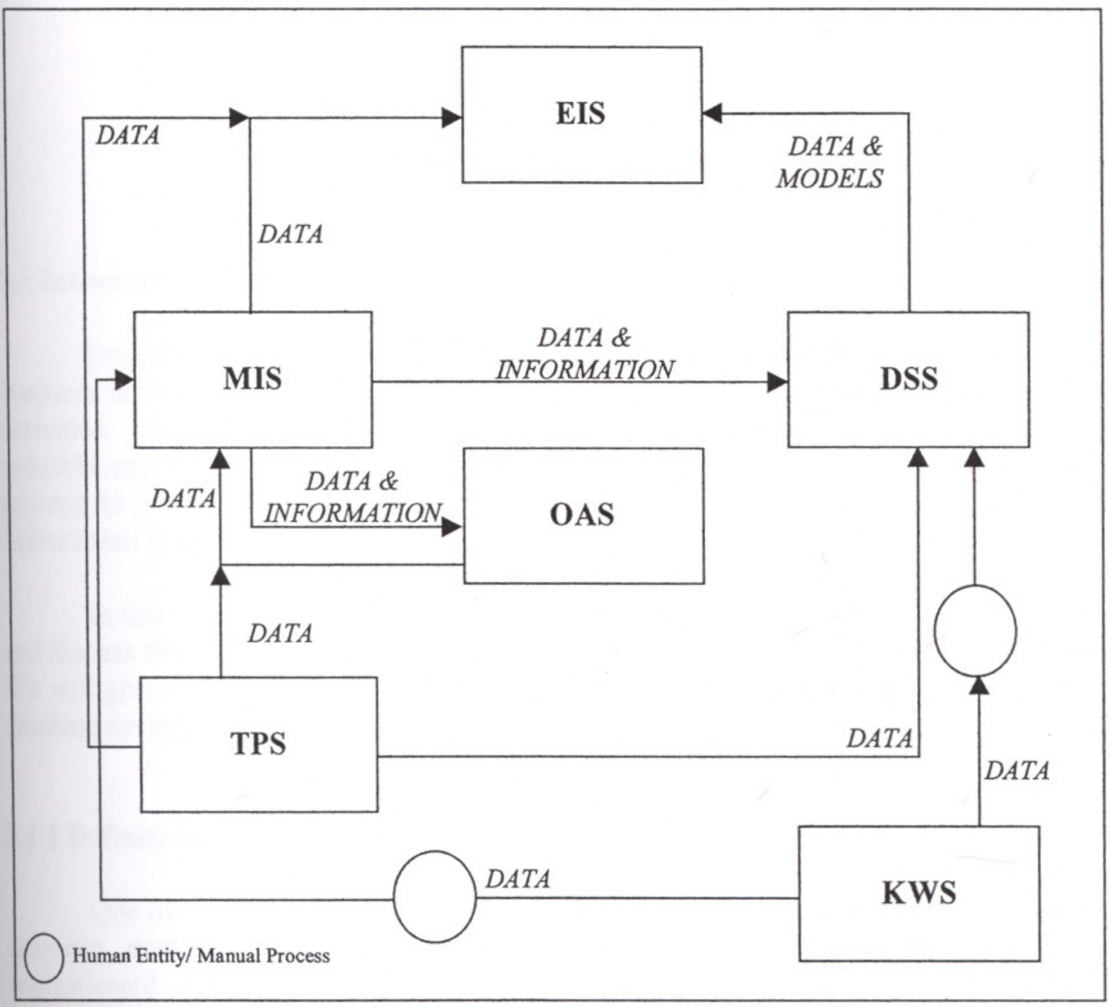


Figure 2.14 Interrelationship between Different Information System Types

IZMIR YUKSEK TEKNOLOJİ ENSTİTÜSÜ
REKTÖRLÜĞÜ
Kütüphane ve Dokümantasyon Daire Bşk.

Chapter 3

DATABASES AND INFORMATION SYSTEMS IN ORGANIZATIONS

3.1 Introduction To Database Systems

Organizations and people use data as a valuable resource for many kind of business activities every day. Data is organized, stored, processed as part of these activities. Information systems, are developed for processing data and generating valuable results for different levels within the organization. Database systems are utilized to handle the data component in this overall picture of organizations and information systems.

In this chapter we will study various database and database management concept and discuss the organizational relationships between information systems and databases. We will give definitions from various sources to understand the various components of database systems.

3.1.1 Definitions

One of the major entities in the overall information systems architectures are the database systems. Modern information systems are developed along with the sophisticated databases architectures and database management systems (DBMS). DBMS have evolved along with the developments in the computer technology, thus supporting the data requirements of organizations and information systems within the organizations more efficiently. *Today, more than at any previous time, the success of an organization depends on its ability to acquire accurate and timely data about its operations, to manage this data effectively, and use it to analyze and guide its activities.*[28]

The major part of our work in this thesis well be a relational database development process for a management information system model. And in this chapter we will first discuss the databases and database management systems theory as a base to our practical work in the following chapters. Some different definitions of databases and database management systems are given below.

A **database** is a collection of data, typically describing the activities of one or more related organizations. For example, a university database might contain information about entities such as students, faculty, courses and classroom; relationships between entities, such as students enrolments in courses, faculty teaching courses and the use of classrooms for courses. A **database management systems**, or DBMS, is a software designed to assist in maintaining and utilizing large collections of

[28] Ramakrishnan Raghu, Gehrke Johannes, *Database Management Systems*, Second Edition, McGraw-Hill Higher Education, p.3, 2000.

Database processing requires the database management system [DBMS] to provide an interface between application programs and these on-line database tables containing user-data. These DBMS are sold as packages developed by large experienced software firms such as Oracle, IBM, Sybase, and Microsoft. They may be classified as a Horizontal Application Software Packages, but some experts would instead classify a DBMS as a systems utility program. A modern DBMS is a bit of both; it provides general utility services and also assists in providing application support for a wide variety of commercial purposes."[33]

A very good representation of DBMS internal structure is given by Ramakrishnan and Gehrke. This representation is based on the relational model which we will be using in this thesis for the database development. Figure 3.1 Shows the structure of a relational DBMS. The given structure in the Figure 3.1 includes more recent technologies such as WEB as well as the classical front-end and structured query language (SQL) interfaces. These are in fact not a part of the DBMS internal structure but we have included them for the completeness of the picture.

External entities shown as application front end, world-wide-web (WWW) front end are for business experts and users, who we can call as the non-technical users. SQL tools and interfaces are for technical users such as system developers, database administrators, system programmers.

SQL tools and front-ends pass SQL commands to the DBMS. DBMS produces query execution plans, executes these plans against the database and returns the results. When triggered, SQL queries are parsed and presented to the query optimiser. Query optimiser uses the information how the data is stored and produces efficient execution plans. Execution plan is the key for evaluating a query and is generally represented as a tree of relational operators.

Disk space manager is responsible for keeping track of available disk space. The code that implements the operators sits on top of the file and access methods layer. File and access methods layer includes software for supporting the files, which are collection of pages or collection of records. File manager issues requests to the disk space manager to obtain and relinquish space on disk. The file and access methods layer requests and frees disk space in units of a page. The size of these pages is a DBMS parameter. Buffer manager brings pages from the disk into the main memory.

DBMS supports concurrency control by using a transaction manager and lock manager. Transaction manager ensures that transactions request and release locks according to a protocol. It also schedules the execution of transactions. Lock manager organises and keeps tracks of lock requests and implements the locks in the database on the objects. Recovery manager is responsible for maintaining a log and restoring the system to a consistent state after a crash situation. For creating their operational logs, disk space manager, buffer manager and files and access methods layer communicate with the recovery manager and concurrency control.

We will not present any further detail on the internal architecture of DBMS. The

[33]Topics Reading on Database Management Systems by A.A. Verstraete, Revised: May 22,1998, URL: http://misweb.smeal.psu.edu/database/

study in this thesis has the scope of information systems and databases.

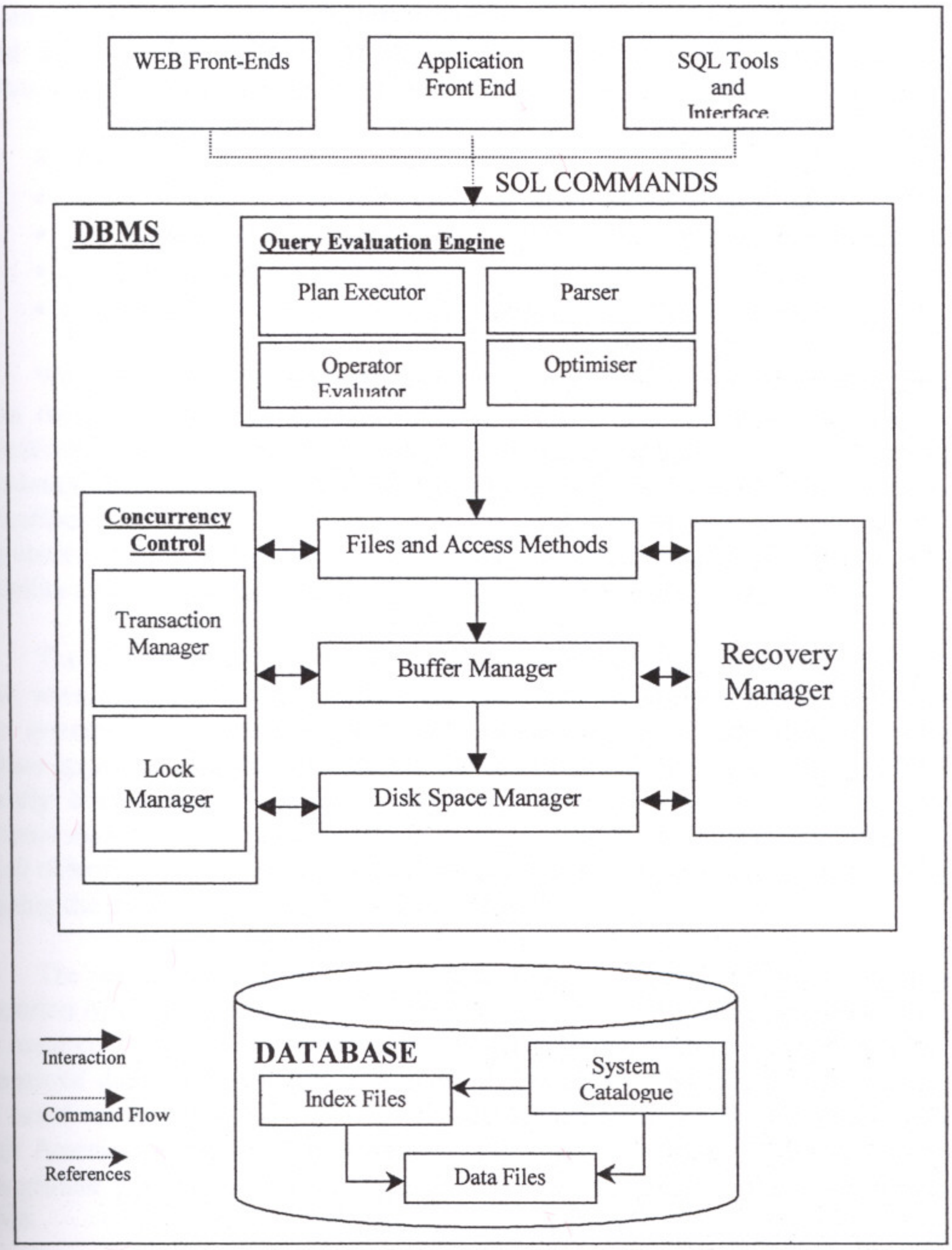


Figure 3.1 Structure of a DBMS.

3.2 Evolution of Database Systems : From File Organisations to Databases

Databases were developed to solve the problems and limitations of file systems which were painful platforms to do development, to maintain and use. Programmers and users were limited with the file structures on developing business application and information systems. Today, thanks to the modern database systems, organisations can develop and make use of highly sophisticated information systems supported by strong DBMS.

Traditional file structures and application development environments forced the development task to focus on applications and the processes, thus defining data within individual programs which furthermore introduced many kinds of other problems. Each time an application is be developed, the problem of data needed to be addressed and solved all over again. Some of the common problems with using files can be summarised in a few points given below :

- Data integrity problems and redundancy.
- Data inconsistency introduced by multiple copies of same data.
- Inflexibility and limited sharing of data because of application dependency.
- Poor development efficiency.
- Maintenance difficulties and data dependency of programs.

We can in fact add more items to this above. To solve some of these problems within the file structures, developers have tried to share data among the various applications by applying some techniques. This was an approach to try to minimise the redundancy problem, but in fact this required excessive physical data transfers between applications and data storage. Considering the limited networking, storage capacity and computation power in the days of file usage, the data transfer was not be always a possibility and an efficient solution.

The earliest database systems research were based on the hierarchical method. These were an extension of COBOL file systems. To provide more flexible access, these systems were extended to network databases. Following this, the relational database approach emerged and became the dominant and most common type available. Recently object-oriented approach has been developed which is an extension of relational model in some cases and a very much different architecture in others.[34] The overall object-oriented approach is still evolving and is still far away from the point of replacing the database systems developed before it.

The early programming languages COBOL and FORTRAN became the foundation of creating enterprise information systems. To operate, these systems, data was required to be stored somewhere. In 1964 General Electric developed the first commercial database management system called IDS – Integrated Data source. This was based on the early network data model developed by C. W. Bachman. IBM and North American Aviation (Rockwell International) developed MIS – Information Management System and its Language DL/1 as the first commercial hierarchical DBMS.

In 1970 Edgar F. Codd published an article which offered a fundamentally different approach. Codd suggested that all data in a database could be represented as a tabular structure (tables with columns and rows, which he called relations) and that these relations could be accessed using a high-level non-procedural (or declarative) language. Instead of writing algorithms to access data, this approach only needed a predicate that identified the desired records or combination of records. This would lead

[34] Post V. Gerald, *Database Management Systems, Designing and Building Business Applications*, Irwin McGraw-Hill, p.15, 1999

to higher programmer productivity. And in the beginning of the 1980s several Relational DBMS (RDBMS) products emerged (Oracle, Informix, Ingres and DB2).[35]

In the early-mid 1980s research started on another type of database. This research was among other things, motivated by the need of a database system capable of handling complex objects and structures like those used in CAD systems and CASE systems[36]. To accomplish these objectives the database had to be able to store classes, objects, objects associations and methods. Thus, the object-oriented DBMS (OODBMS) emerged. In the late 1980s and 1990s several vendors have developed OODBMSs.

During the same period, the relational databases already had its standard - SQL-92, defined by its ANSI committee and ISO. And so did the network database vendors as well; CODASYL (Conference on Data Systems Languages defined in 1986 by the ANSI X3H2 committee).

In Table 3.1 the historical developments and milestones in the database systems are given. Nowadays, relational database systems are the most commonly used and very much standardised systems in technological terms. Strong products are available in the market. Parallel to this another developing area are the object-oriented databases. But despite the research and new developments in the object-oriented databases, they lack of widely established standards and foundations.

The database development work in this thesis is based on the relational model.

Today, the features and performance of the database systems are bigger then ever. Increasing computing and communication power providing a platform for more sophisticated database management systems and database development tools, thus enabling business people and developers to concentrate more on the real life requirements and solutions. Some of the main features and solutions database systems provide can be listed as below, Database systems:

- reduce ***data redundancy***: data is more consistent since only one master version exists for any given data entity.
- improve ***data integrity***: data is more likely to be accurate and up-to-date, and available when it is needed. Data quality standards can be enforced and security and privacy of the data is more easily guarded.
- enable ***data independence***: data and programs are not dependent to each other. Data can therefore be reorganized without revising the programs (Provided that "good" programming techniques are applied). Programs can be written or revised without reorganizing the data.

All these features enable the development of more business-oriented systems. Changing roles in the development process create new interfaces between the business experts and developers. Database administration in the organisations is becoming a

[35] Codd E F., 1970, A Relational Model for Large Shared Databanks, Communications of the ACM, Volume 13, Number 6, p.377-390, June 1970.
[36] Zdonik S., What Makes Object-Oriented Database Management Systems Different, Advances in Object-Oriented Database Systems, NATO ASI Series, Series F: Computer and System Science, Vol. 130, 3-26, Springer Verlag, Berlin Heidelberg New York, 1994

more and more required role, which not only demands just technical knowledge but also a concentration on business functions of the organisations. We will discuss various component of a database system in the organisation later in this chapter.

Table 3.1 Evolution of Database Systems

Database System / DBMS	**Description**
1961 : Integrated Data Source (IDS)	• The first general purpose DBMS system. Developed by Charles Bachman at General Electric in early 1960s. • Formed the basis for Network data mode (standardised by the Conference on Data System Languages – CODASYL)
1968 : Information Management System (IMS)	• Developed by IBM in late 1960s. • This is still being used in some organisations. • Based on the Hierarchical data model.
SABRE System	• Developed for making airline reservation by American Airline and IBM in late 1960s.
1970 : Edgar Codd – The Relational Database Model and SQL.	• In 1970, Codd in IBM's San Jose Research Centre proposed the relational data model. • This enabled rapid development of many DBMS systems based on the relational model. • In 1980's the relational model became the dominant and most commonly used. • SQL query language became the standard as part of IBM's System R Project. (1980)
1980–1990: DB2, ORACLE, INFORMIX, SYBASE)	• Research on powerful query language and richer data models. • Specialised systems are developed. • Relational data model used in management resource planning (MRP), enterprise resource planning (ERP) systems
1990 – 2000 : Object Oriented Databases, WWW architectures.	• A new and still evolving method. • Less common, foundations still being established. • Not standardised as relational model. • All database vendors are adding internet features to their DBMS to make it suitable for systems developed for internet.

3.3 Components of Database Systems in Organisations.

Database systems in organisations consist of various components including both human components and machine/software components. A database system, (in fact, considering the in modern days we could say multiple database systems) are at the centre of the data processing in organisations. Database systems surrounded by various type of information systems, applications, tools, processes, people and the information needs of real life environment, The scene described above is a very much alive and ever-changing, evolving scene. Processes, business rules change and so do the Database systems and data processing tool.

In Figure 3.2, a high level relationship diagram between different entities of data processing in organisations is given.

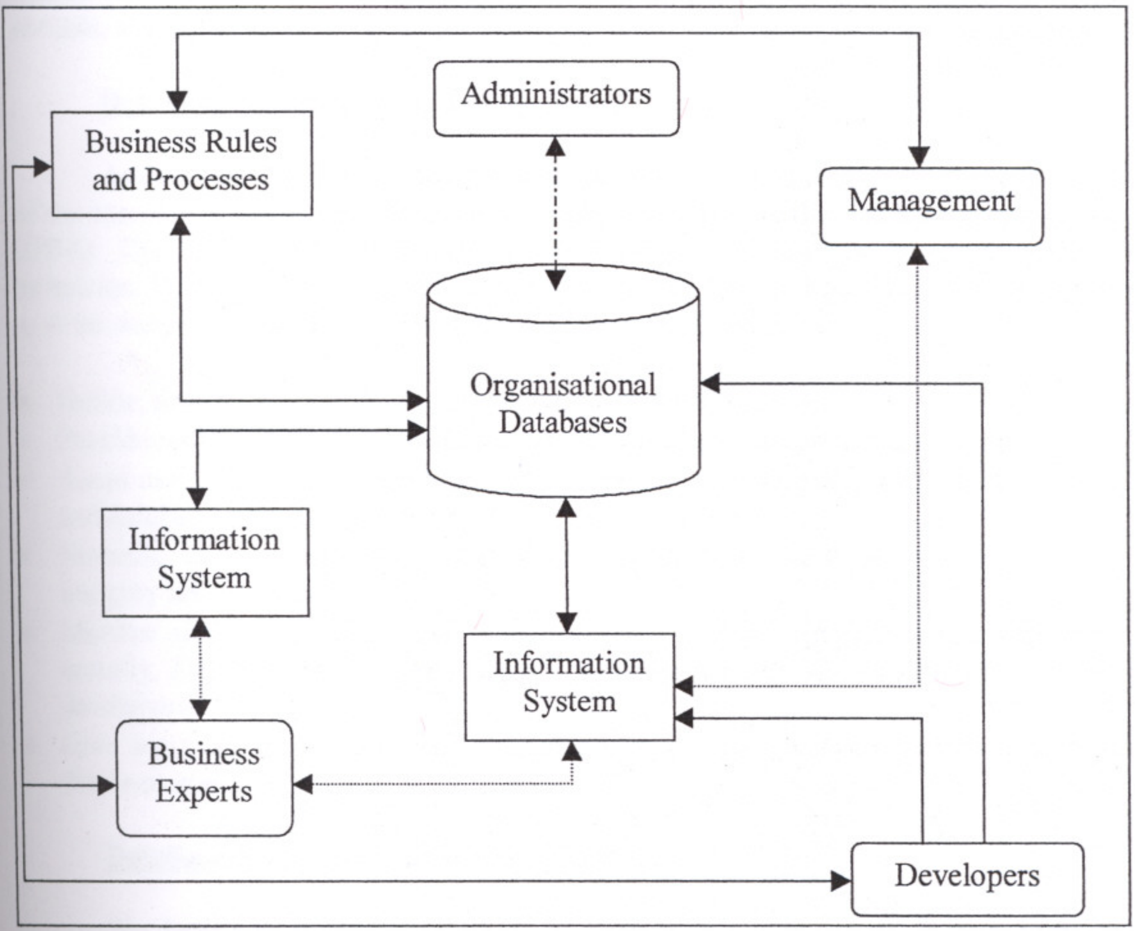


Figure 3.2 Relationships Between Entities in Data Processing in Organisations

Business rules and procedures are the formal definitions of how the business is conducted, how processes are executed. These rules and procedures are defined by business experts and managed by the and managers. Business experts have a role in information systems development as well, by providing business problems and proposals for solutions during the analysis and development of information systems within the organisation. The rules and processes also define the way databases are organised and data is maintained. Data is administrated by database administrators. Information systems, process the data and generate results for business problems. This whole set of relationships within an organisation defines the overall data processing and information management. Dotted lines represent the human processes interaction.

Figure 3.2 is just one way of putting various components and relationships together. Some may argue that, the centre piece of all these are actually business rules and processes which drive the rest. This would be an equally valid definition. But our aim here is to present the relationships in an organisations from a database point of view and we like to emphasise the importance of data and databases such that everybody else needs and uses them.

Regardless of an organisations structure, whether matrix, or pyramidal, where operational activities are at the bottom, management planning and control activities in the middle, and strategic planning in top, corporate databases - including all databases in an organisation - contain data, relating to the organisation; its operations, its plans, and its environment. Various types of information systems are put in place to access, process, and report data and generate information for the organisation and its functions.

Database Administrator (DBA):

As described earlier, database system have human and machine components. On of the essential human components of a database systems is the database administrator (DBA). The DBA must be an analyst and designer and not just an implementation technician. DBA must have the ability of problem solving and guidance to information systems designer. The typical roles of a DBA can be listed as :

- Define, acquire and retire data according to the business needs.
- Provide tools for developers and users to access the database, generate results.
- Assist users and information system developers in planning and using data resources and database management tools
- Maintain and manage day-to-day database functions such as backup, security, integrity and standards.
- Monitor daily operations and take necessary actions for increasing efficiency, security. Fine tune the database system to make it more efficient for the users and developers.
- Give input at the analysis and design phases of information systems. Carry-out database implementations.

Database Management System (DBMS):

We have described the DBMS in Section 3.1.1 in detail. Along with the functions already described, DBMS is essentially, a set of programs utilities, procedures to manage the database and data. It is also a tool for the DBA to administer the database system. DBMS provides tools and methods to access data by applications and users.

DBMS selection is a critical task for technical decision makers in organisations. In 1986, Everest loosely estimated the number of DBMSs developed as over 600 in the past 25 years. (Everest 1986). Considering the acceleration in the computer hardware and software technology and the increase of computer usage in organisations since 1986, this number can be easily be seen as doubled or tripled. In selecting a DBMS, an organisation should understand its own data processing requirements and environment. Required functional capabilities should be listed and verified to make sure the DBMS selected is a suitable one for the business functions. Business data load, data volume and geographical facts are the other criteria for selecting a DBMS.

Business Data:

A commercial database management system can exists in many organisations. For example, an ORACLE database can be purchased by many organisations and used. Many of the tools, functions of the DBMS would be very much the same in every organisation purchased it. But, on the other hand, business data is the one component which is almost unique to each organisation. Business data consist of in-house developed user databases, tables and related programs. The reason to use a database system is actually store and process the business data accumulated along with the daily operations of business. The business data is the most critical and valuable component of a database system. It is the business data feeding into the information systems enabling people to conduct their functions. Business data cannot be purchased. It can only be produced. Business data has to be maintained safely, structurally by the database management system.. Overhead data or system data (metadata : data about data) is invisible to the users and is mostly irrelevant to the actual business. The business data, databases, schemas, tables, procedures developed by the people of an organisation are intellectual properties of common effort between various parties.

Figure 3.3 shows the relationship between DBA, DBMS and the Business data (database). We have also shown the information system and tools in the figure.

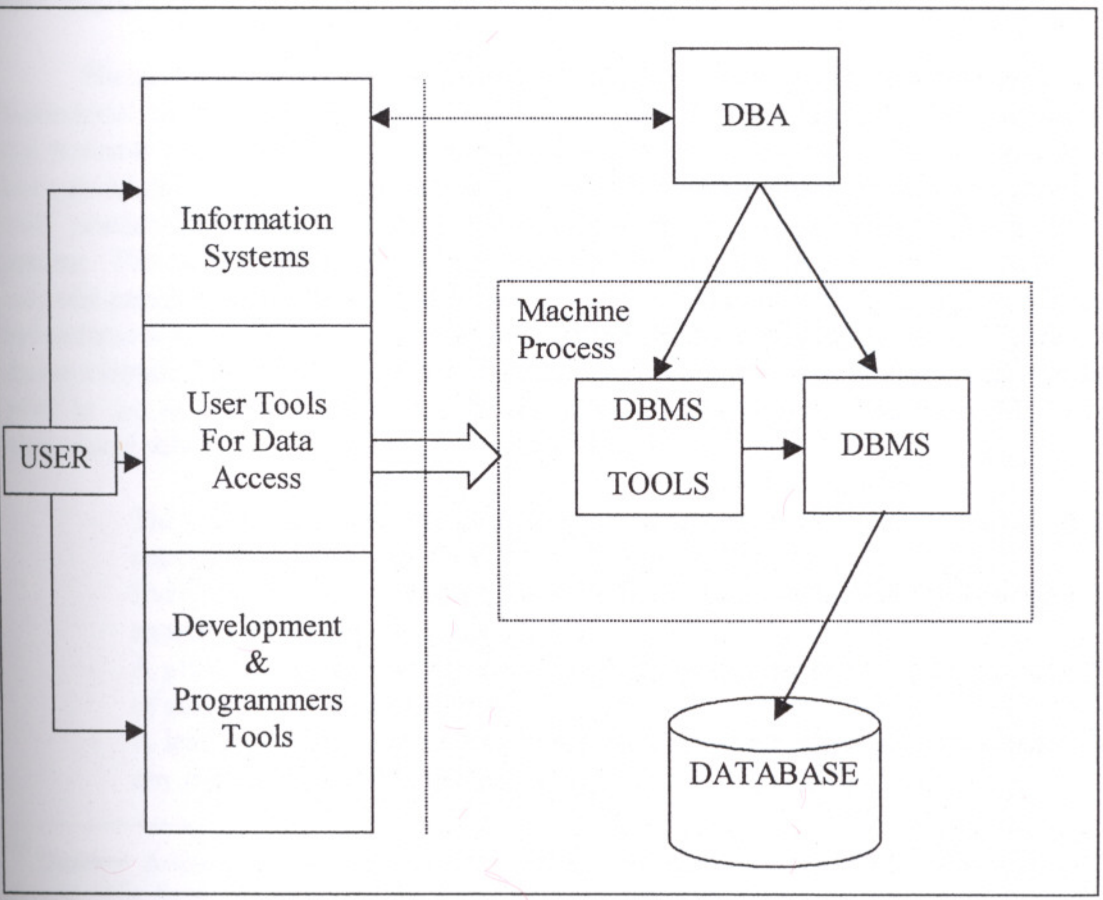


Figure 3.3 Relationships in Database Systems.

Users interact with the database via information systems, user tools for data access and development and programmers tools and tools provided by the DBMS. DBA is in charge of setting necessary platforms, granting access rights and monitoring the process. DBA also uses the DBMS tools for administrative purposes. DBMS provides a layer between the raw data in the database and users. DBMS is a machine process that enables and controls the data traffic between the users and the database. The DBMS has multiple components described earlier, to do the automatic tasks.

3.4 Types of Databases and Database Management Systems

Through out the evolution of database systems various architectures have been developed and implemented. As history of evolution shows, different database management systems have been developed during different times, and these have all different capabilities regarding organization and modeling of data and access to data.[37] The earliest of these were the hierarchical databases, followed by network databases, relational databases and the relatively new concept object oriented databases.

In this section we will briefly discuss the hierarchical, network and object oriented databases but go into more detail of relational databases. We will leave the relational databases to a separate section at the end of this chapter.

3.4.1 Hierarchical Databases

Hierarchical databases are the earliest database systems. As the name suggests, hierarchical databases use the hierarchical data model for structuring the data, assuming that business data often shows a hierarchical nature. The best way of understanding a hierarchical data structure is to picture an upside down tree which constructs a parent – child relationship between records types. These record types define a hierarchical schema. The parent –child relationship enforces that a child record can have only and only one parent record whereas the parent record can have multiple child records in the hierarchical schema. The parent record links to the child records via pointers. Figure 3.4 shows a simple hierarchical database (schema tree) where the root record is the starting point to get to a low level record in the hierarchical schema. The properties of a hierarchical schema must obey the following rules:

- The root record-type cannot be a child of any record type in the parent-child relationship.
- Every record type - except the root participates as a child record type in exactly one parent-child relationship.
- A given record type can participate as the parent record type in any number of parent-child relationship.
- A leaf is defined as a record type which does not participate as a parent in any parent-child relationship.

[37] Danielsen Asbjørn, *The Evolution Of Data Models And Approaches To Persistence In Database Systems,* M Sc. Essay, University of Oslo, Department of Informatics, 1998.

An example hierarchical database schema is given in Figure 3.5. This schema is designed to represent a university database where different courses thought in the departments and students enrolled to different courses. Each department has also staff members. Departments have classrooms where the courses are assigned to. The record types and the fields in each of them are given in the figure.

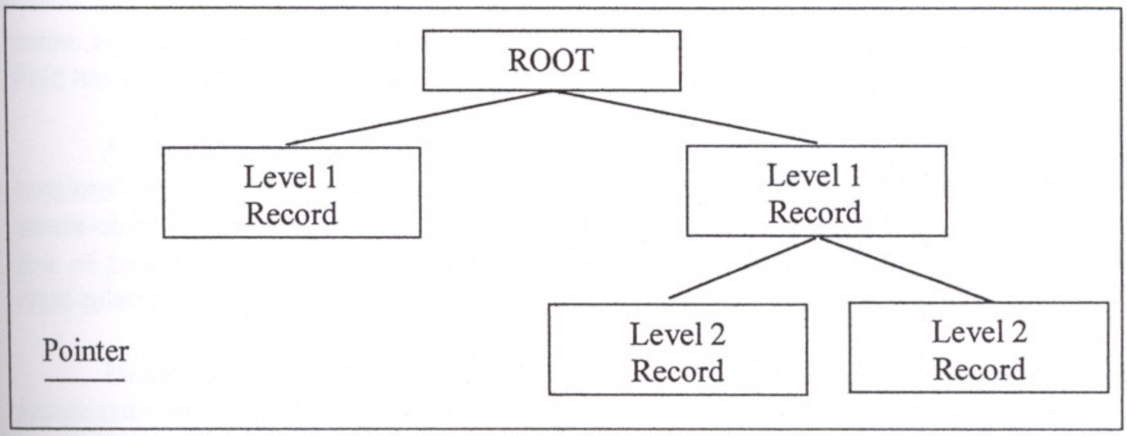


Figure 3.4 A Hierarchical Database (Schema Tree).

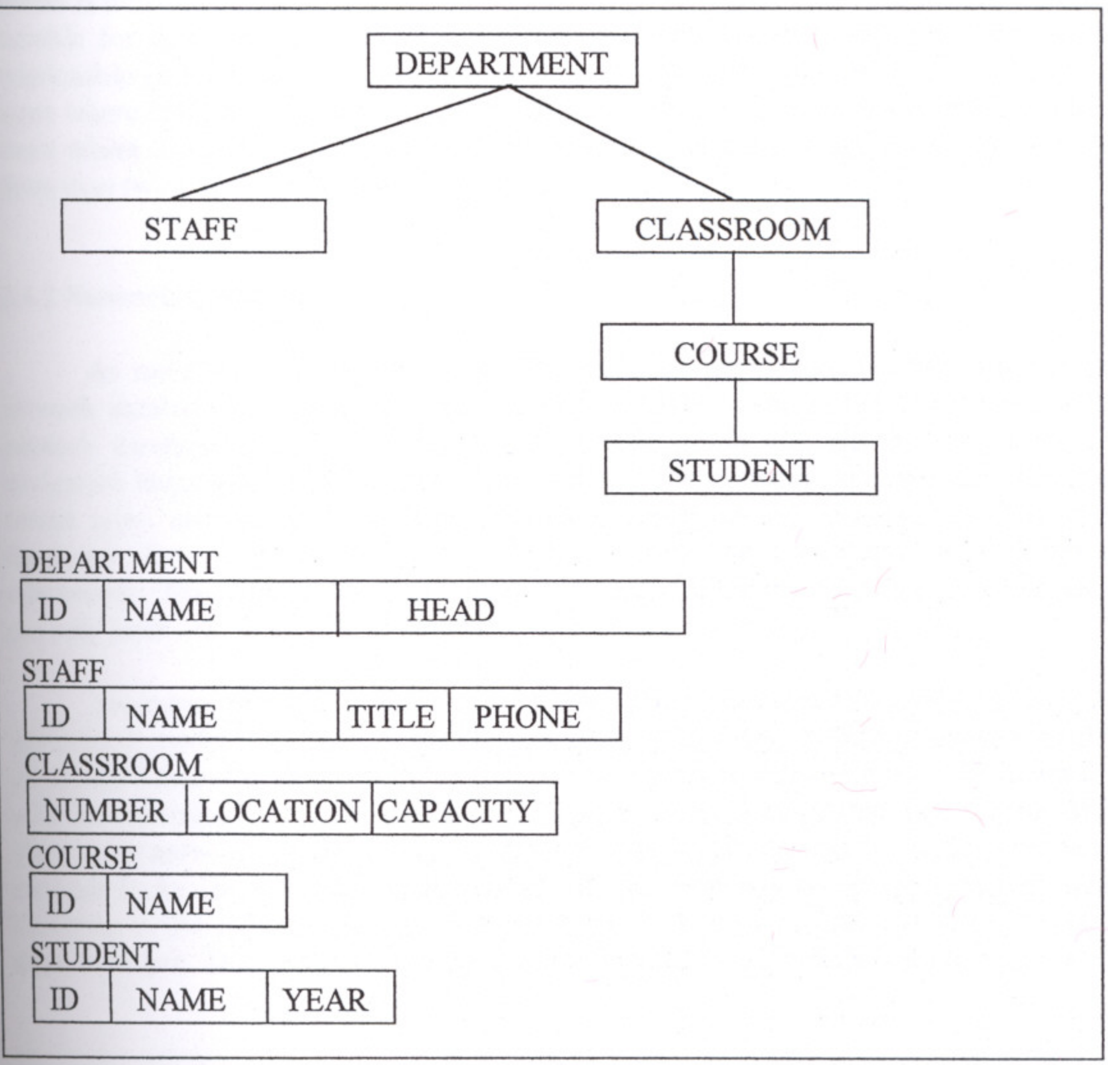


Figure 3.5 An Example Hierarchical Database.

Hierarchical model employs two main data structuring concepts : records and parent-child relationship. A record is a collection of field values (i.e. the DEPARTMENT field values in Figure 3.5 are values for ID, NAME, HEAD) that provide information on an entity or a relationship instance.

Records of the same type are grouped into record types. A record type is a given name, and its structure is defined by a collection of named fields or data items. Each field has a certain type, such as integer, real or string.[38]

A parent-child relationship in hierarchical databases is a one –to-many relationship where a parent record type instance has multiple child record types. The parent-child relationship in a hierarchical database is generally shown by listing the pair of parent and child record type) in parenthesis. In Figure 3.5 two of the parent-child relationships are (DEPARTMENT, STAFF); (DEPARTMENT-CLASSROOM).

Hierarchical databases can be very useful for some applications such as the functional mapping between parent and children and the hierarchical structure, whereas others pose restriction. We cannot have dangling children unconnected to a parent. If a parent is deleted, so must be all its children. While the hierarchical model can be very suitable for some applications such as organisational structures and parts explosion relationship (bill of materials), it can be too restrictive for some others.[39] The main areas where the hierarchical model has problems are, on many-to-many relationships, cases where a record type is the child of multiple parent and n-ary relationships with more than two participating relationships.

3.4.2 Network Databases

As mentioned in Section 3.2 C. W. Bachman developed the first commercial network database management system called the IDS – Integrated Data Store. The network database arranges its data as a directed graph and has its own standard navigation language, called DBTG. The nodes in the network database represent the record types and the arch between the nodes represent the relationships. Network databases allow a given node to have more than one arc, each for a different relationship. This feature does not exists in hierarchical databases where a parent node can only have one arc to link to its child.

The network model is very similar to the hierarchical model actually. In fact, the hierarchical model is a subset of the network model. However, instead of using a single-parent tree hierarchy, the network model uses set theory to provide a tree-like hierarchy with the exception that child tables were allowed to have more than one parent. This allowed the network model to support many-to-many relationships. Visually, a network database looks like a hierarchical database in that you can see it as a type of tree. However, in the case of a network database, the look is more like several trees which share branches. Thus, children can have multiple parents and parents can have multiple

[38] Elmasri R., Navathe S., *Fundamentals of Database Systems,* 2nd Edition, The Benjamin / Cummins Publishing Company, p.344, 1994

[39] Ozkarahan Esen, *Database Management. Concepts, Design and Practice*, Prentice-Hall International Editions, p.25, 1990

children.[40]

The network databases offers an efficient access-path to its data and is capable to represent almost any informational structure containing simple types (e.g. integers, floats, strings and characters). This is accomplished using different kinds of mapping mechanisms called sets. A set is a container of pointers identifying which sets of data can be reached from the current record. Three sets are defined by the CODASYL standard - singular/system sets, multimember sets and recursive sets. Using these sets, the database designer and programmer may represent and navigate on one-to-one, one-to-many, and many-to-many relationships. To be able to do this, the programmer has to know the physical representation of the database and access the database using a low-level navigational language (Bachman 1973). This approach to DBMS is more flexible than the hierarchical approach, but still the programmer has to know the physical representation of data to be able to access it, and accordingly applications using a network database has to be changed every time the structure of the database changes.[41]

Network databases were designed and implemented to solve some of the difficulties and problems of hierarchical databases. Hierarchical database would not allow the existence of many-to-many relationships without duplications of record types. Network databases can represent many-to-many relationships. Data redundancy is another problem of hierarchical databases, that network database solve.

A simple network database architecture and Bachman Diagram for a car-hire network database is given in Figure 3.6. In Figure 3.7, the many-to-many mapping between record types is shown. Figure 3.8 shows the schema data definition language (DDL) for record types and Figure 3.9 shows the DLL for relationships database. [42]

Data definition language enables to perform the following tasks on a schema :

- Create, alter, and drop schema objects.
- Grant and revoke privileges and roles.
- Analyze information on a table, index, or cluster.
- Establish auditing options.
- Add comments to the data dictionary.

In Figure 3.6, the many-to-many relationship between owner and member record types (i.e. MECHANIC and CAR, CAR and CUSTOMER) are constructed by inserting link record types between them. These link record types are SERVICES and HIRING. The many-to-many functional mapping between MECHANIC and CAR is constructed by setting a one-to-many mapping between MECHANIC and SERVICES and a one-to-many mapping between CAR and SERVICES. Similar approach is used for CAR and HIRING, CUSTOMER and HIRING functional mappings, which enables the many-to-many functional mapping between CAR and CUSTOMER record types.

[40] Sol Selena, Network Databases, Web Developers Virtual Library, 16 August 1998, URL: ttp://www.stars.com/Authoring/DB/

[41] Danielsen Asbjørn, *The Evolution Of Data Models And Approaches To Persistence In Database Systems,* M Sc. Essay, University of Oslo, Department of Informatics, 1998.

[42] University of Wolverhampton, The School of Computing and Information Technology (SCIT), *CP4011 Database Concepts and Techniques,* URL: http://scitsc.wlv.ac.uk/

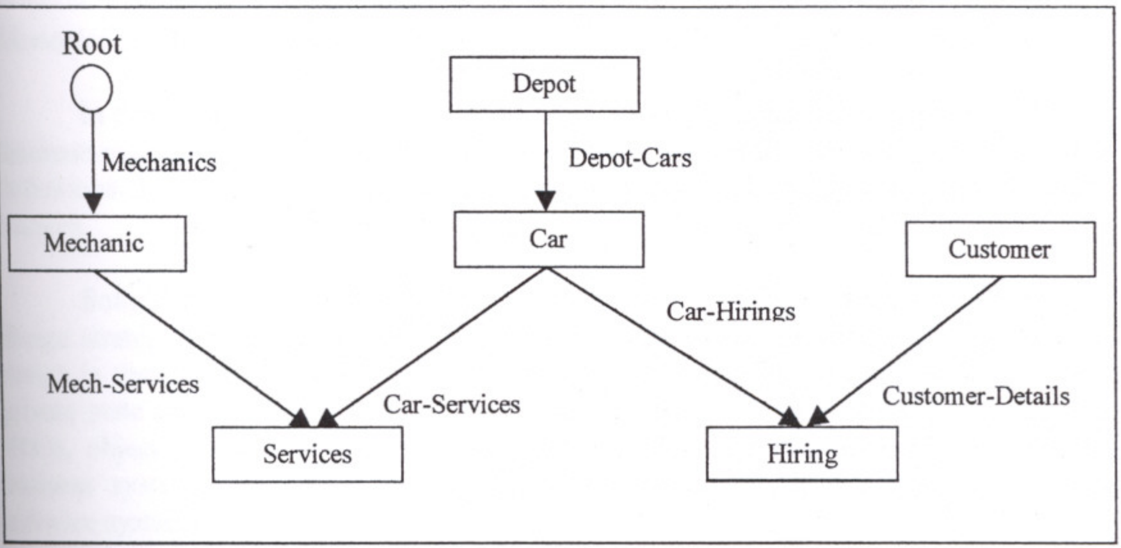


Figure 3.6. Bachman Diagram For Car-Hire Network Database.

To summarise the characteristics of the network databases following rules can be given:

- There must be a one-to-many relationship between pairs of record types (nodes) related with respect to an owner-with-member relationship.
- A given owner instance must posses a unique set of member instances
- A given record type cannot be both owner and member of the same set type.

A network database has the following constructs :

- Owner record types.
- Member record types.
- Set types relating owner and member record types.

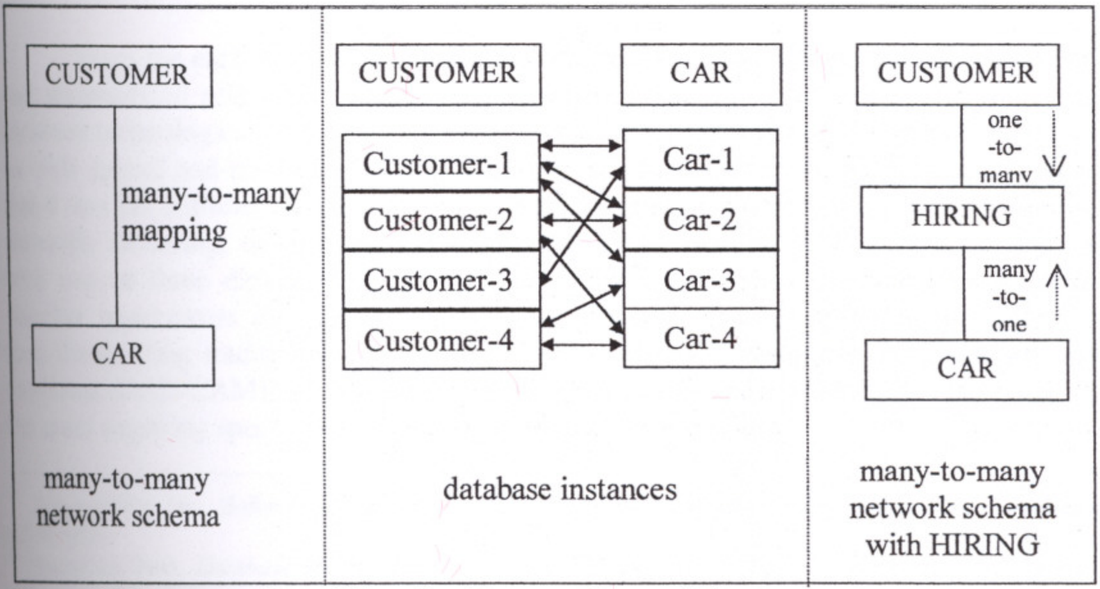


Figure 3.7 Many-to-Many Mapping Between Record Types.

3.4.3 Object-Oriented Databases

- You know my methods, Watson. Apply them. (Arthur Conan Doyle, *The Memories of Sherlock Holmes*).

In this section we will describe the object-oriented approach and object oriented database management systems without going into much detail. We will present various definitions for object-oriented design, programming and databases from different sources.

Sommerville defines object-oriented design as: "Object-oriented design is a design strategy based on information hiding. It differs from the functional approach to design in that it views a software system as a set of interacting objects, with their own private state rather than as a set of functions that share a global state. Since the late 1980s, object oriented design has been widely publicized and adopted. Outside the business systems domain, it is perhaps the pre-dominant design strategy for new software systems."[43]

Another definition for object-oriented design is given as: The foundation of the object-oriented approach lies in encapsulation which enables restricting the effects of change on software, and inheritance, which allows classes (i.e., types) to pass properties to each other. A class is a set of object, i.e., a type in databases. An object, therefore is an instant of a class similar to an entity being instance of an entity set or type in traditional databases. A method is a part of an object's properties (i.e., behavior) that describes how the object carries out its operations.[44]

The characteristics of an object-oriented design can be summarized as:

- Objects are abstraction of real world having their private states and offering service to other objects to achieve the goals of the overall system.
- Objects are independent entities and can be modified independently since the state and representation information is kept within them.
- No shared data exists in objects oriented design. Objects communicate through calling each other services.

From the early hierarchical and network databases to more recent well-establish and standardized relational databases systems, organizations used more and more the database technologies to conduct business. Even though the relational database model is so wide spread and standardized, special types of operations in daily life made it clear that it was not sufficient to model the needs of businesses and organizations. Specially, scientific modeling, design and manufacturing areas, dealing with geometric objects with two or three dimensions, robotics and automation made this very much clear. Another requirement emerged for new data types and structures, which are different then the existing traditional data structures. For example, computer aided design and modeling (CAD/CAM), document management or multi-media repositories are some of the areas requiring specialized data structures and data handling.

[43] Sommerville Ian, *Software Engineering*, Fifth Edition, Addison-Wesley Publishing Company, p.248,1996
[44] Ozkarahan Esen, *Database Management. Concepts, Design and Practice*, Prentice-Hall International Editions, p.264, 1990

```
SCHEMA NAME IS CAR-HIRE-DATABASE;
        PRIVACY LOCK FOR LOCKS IS LOCK-LOCK;
        PRIVACY LOCK FOR DISPLAY IS DISP-SCHEMA-LOCK;
        PRIVACY LOCK FOR INCLUSION IS INCL-SCHEMA-LOCK;
        PRIVACY LOCK FOR ALTER IS ALTER-SCHEMA-LOCK.

AREA NAME IS CAR-AREA;
        PRIVACY LOCK FOR RETRIEVAL IS AREA-READ-LOCK;
        PRIVACY LOCK FOR UPDATE IS AREA-WRITE-LOCK.

RECORD NAME IS DEPOT;
        LOCATION MODE IS CALC USING DEPOT-NAME
        DUPLICATES ARE NOT ALLOWED;
        PRIVACY LOCK FOR STORE IS DEPOT-ADD-LOCK;
        PRIVACY LOCK FOR DELETE IS DEPOT-DEL-LOCK.
        01 DEPOT-REC.
                02 DEPOT-NAME;          PICTURE 'X(20)'.
                02 DEPOT-ADDRESS;       PICTURE 'X(32)'.

RECORD NAME IS CAR;
        LOCATION MODE IS CALC USING CAR-REG
        DUPLICATES ARE NOT ALLOWED.
        01 CAR-REC.
                02 CAR-REG;             PICTURE 'X(6)'.
                02 CAR-MAKER;           PICTURE 'A(20)'.
                02 TYPE-OF-CAR;         PICTURE 'A(15)'.
                02 CAPACITY;            PICTURE '9999'.
                02 YEARS-OLD;           PICTURE '99'.

RECORD NAME IS HIRERS;
        LOCATION MODE IS CALC USING HIRER-NO
        DUPLICATES ARE NOT ALLOWED.
        01 HIRER-REC.
                02 HIRER-NO;            PICTURE '9(6)'.
                02 HIRER-NAME;          PICTURE 'A(20)';
                PRIVACY LOCK FOR GET IS CUST-NAME-GET-LOCK.
                02 HIRER-ADDRESS;       PICTURE 'A(32)';
                PRIVACY LOCK FOR GET IS CUST-ADDRESS-GET-LOCK.

RECORD NAME IS HIRINGS;
        LOCATION MODE IS VIA CAR-HIRINGS SET.
        01 HIRING-REC.
                02 HIRE-DATE;           PICTURE 'X(8)'.
                02 HIRE-TIME;           PICTURE 'X(5)'.
                02 HIRE-LENGTH;         PICTURE '999'.

RECORD NAME IS SERVICES;
        LOCATION MODE IS VIA CAR-SERVICES SET.
        01 SERVICE-REC.
                02 SERVICE-CODE;        PICTURE '9999'.
                02 SERVICE-DATE;        PICTURE 'X(8)'.
                02 MILES-CLOCKED;       PICTURE '9(6)'.

RECORD NAME IS MECHANIC;
        LOCATION MODE IS CALC USING MECH-NO
        DUPLICATES ARE NOT ALLOWED.
        01 MECH-REC.
                02 MECHNO;              PICTURE '9999'.
                02 MECH-NAME;           PICTURE 'X(20)'.
                02 MECH-ADDRESS;        PICTURE 'X(32
```

Figure 3.8 Record Type Definitions For Car-Hire Database.

```
SET NAME IS DEPOT-CARS;
   ORDER IS SORTED;
   PRIVACY LOCK FOR INSERT IS CAR-INSERT-LOCK;
   PRIVACY LOCK FOR REMOVE IS CAR-DELETE-LOCK;
   OWNER IS DEPOT.
   MEMBER IS CAR MANDATORY AUTOMATIC;
   ASCENDING KEY IS CAR-REG DUPLICATES ARE NOT ALLOWED;
   SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.

SET NAME IS CAR-HIRINGS;
   ORDER IS SORTED;
   OWNER IS CAR.
   MEMBER IS HIRINGS MANDATORY AUTOMATIC;
   ASCENDING KEY IS HIRE-DATE,HIRE-TIME DUPLICATES ARE NOT ALLOWED;
   SEARCH KEY IS HIRE-DATE DUPLICATES ARE ALLOWED;
   SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.

SET NAME IS HIRER-DETAILS;
   ORDER IS SORTED;
   OWNER IS HIRERS.
   MEMBER IS HIRINGS MANDATORY AUTOMATIC LINKED TO OWNER;
   ASCENDING KEY IS HIRE-DATE,HIRE-TIME DUPLICATES ARE NOT ALLOWED;
   SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.

SET NAME IS CAR-SERVICES;
   ORDER IS SORTED;
   OWNER IS CAR.
   MEMBER IS SERVICES MANDATORY AUTOMATIC LINKED TO OWNER;
   ASCENDING KEY IS SERVICE-DATE DUPLICATES ARE LAST ALLOWED;
   SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.

SET NAME IS MECH-SERVICES;
   ORDER IS SORTED;
   OWNER IS MECHANIC.
   MEMBER IS SERVICES MANDATORY AUTOMATIC LINKED TO OWNER;
   ASCENDING KEY IS SERVICE-DATE DUPLICATES ARE LAST ALLOWED;
   SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.

SET NAME IS MECHANICS;
   ORDER IS SORTED;
   PRIVACY LOCK FOR FIND IS MECH-LIST-LOCK;
   PRIVACY LOCK FOR REST IS MECH-UPDATE-LOCK;
   OWNER IS SYSTEM.
   MEMBER IS MECHANIC MANDATORY AUTOMATIC;
   ASCENDING KEY IS MECH-NO DUPLICATES ARE NOT ALLOWED.

END SCHEMA.
```

Figure 3.9 Relationship Definitions For Car-Hire Database.

Object-oriented database is a new and evolving method of organizing data[45] to fulfill the requirements of organizations and people. To understand the differences between a traditional (relational) and object-oriented databases we need to consider various characteristics of both and compare them to observe how they effect the whole database development and management processes.

Ozkarahan identifies the main difference between traditional and object-oriented databases as the passive and active behavior of the underlying system and the way these are implemented. According to this definition, traditional databases are passive and object-oriented databases are active. Data in traditional systems are accessed by the

[45] Post Gerald V., Database Management Systems, Designing and Building Business Applications, Irwin McGraw-Hill, p.17, 1999

application program via DBMS, stored in the programs data structures, processes to produce required outputs and results are finally written back to the passive database. Here the active component is the application program and not the database. In the object-oriented database, the database contains objects that are made up from passive data and active procedures to reflect the behavior of the object. A mechanic arm of robot for example, contains both data and programs that together compose the robot arm object and its behavior (i.e. movements). Figure 3.10 shows how the traditional and object-oriented databases differ from each other in the way described above. Application logic in object-oriented databases is passed to the database instead of being developed in the application program. The analogy here is, just like DBMS took over the details that the had to worry about in file systems, the object-oriented databases take over the details of modeling and implementing real-life objects from traditional DBMS.

Object-database systems have been developed into two different categories: Object-relational databases and object-oriented databases. Object - relational databases are extension to the relational databases which combines some of the object-oriented features with the existing relational. These extended features will be briefly presented in the following section .

3.4.3.1 Object-Relational Databases

There is an increasing need to store and manipulate complex data in relational databases. Complex data is imminent not only in multimedia applications for the Web, but also in specialized application domains such as medical care (including X-rays, MRI imaging, and EKG traces); geographical, space, and exploration systems (such as maps, seismic data, and satellite images); and even financial systems (such as time series data). [46]

The Object-relational databases (ORDBMS) are an extension to the existing RDBMS, which includes some of the features of object-oriented architecture, in other words ORDBMS encapsulates some features into an RDBMS, creating an ORDBMS.
ORDBMS add new object storage capabilities to the relational systems at the core of modern information systems. These new facilities integrate management of traditional fielded data, complex objects such as time-series and geospatial data and diverse binary media such as audio, video, images, and applets. By encapsulating methods with data structures, an ORDBMS server can execute complex analytical and data manipulation operations to search and transform multimedia and other complex objects.

As an evolutionary technology, the object-relational (OR) approach has inherited the robust transaction- and performance-management features of its relational ancestor and the flexibility of its object-oriented cousin. Database designers can work with familiar tabular structures and data definition languages (DDL) while assimilating new object-management possibilities. Query and procedural languages and call interfaces in ORDBMSs are familiar: SQL3, vendor procedural languages, and ODBC, and proprietary call interfaces are all extensions of RDBMS languages and interfaces. The leading vendors in the market for ORDBMS are IBM, Informix, and Oracle.[47]

---

[46] Rennhackkamp Martin, Extending Relational DBMSs, DBMS, Volume 10, Number 13, December 1997, p. 45
[47] Grimes Set, Modeling Object/Relational Databases, DBMS, Volume 11, Number 4, April 1998, p.51

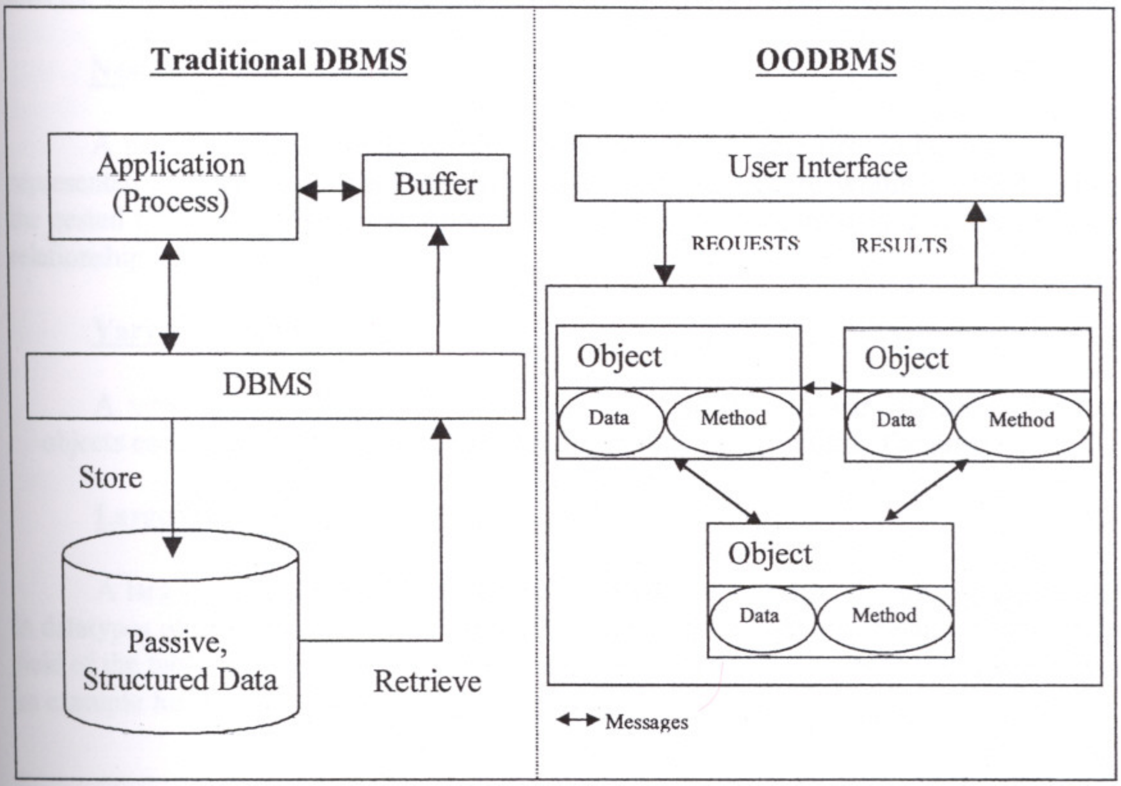


Figure 3.10 Difference Between Traditional and Object-Oriented Databases.

Object-relational databases organize information in the familiar relational tabular structures. In fact, object-relational implementations subsume the relational database model. ORDBMSs are an incremental upgrade to their RDBMS predecessors, and, unlike the move to object database systems, object-relational migration will not necessarily entail wholesale recording.

As we defined before; the traditional RDBMS are extended to include object-oriented concepts and structures such as abstract data types, nested tables and varying arrays. In ORACLE terms these are defined as following :

Abstract Datatypes

Abstract datatypes are datatypes, that consists of one or more subtypes. Rather than being constrained to the standard DBMS datatypes like NUMBER, DATE/TIME, CHAR, abstract datatypes can more accurately describe the data. For example an abstract datatype for project may contain the following fields (columns):

NAME	VARCHAR (50)
STARTDATE	DATE
ENDDATE	DATE
MANAGER	VARCHAR(25)
BUDGET	NUMBER(11,2)

When creating a table that uses project information , a column should be created that uses the abstract datatype for projects.[48]

Nested Tables

A nested table is a table within a table. A nested table is a collection of rows, represented as a column within the main table. For each record within the main table, the nested table may contain multiple rows. This can be seen as storing a one-to-many relationship in a table.[49]

Varying Arrays

A varying array is like a nested table, a collection. A varying array is a set objects each with the same datatype, that can be set while creating the array

Large Objects

A large object is capable of storing large volumes of data. Large objects can be in datatypes of binary, character. These objects can store G-bytes of data in one single field of the table. Storing pictures of students in student table as a binary large object is an example for the usage of large objects.

3.4.3.2 Object-Oriented Databases

Object-oriented database systems are proposed as an alternative to relational systems and are aimed at application domains where complex objects play a central role. The approach is heavily influenced by object-oriented programming languages and can be understood as an attempt to add DBMS functionality to a programming language environment.[50]

In an OODBMS anything represented as an object, or part of an object, may be stored in the database, regardless of type. All DBMSs, except the ORDBMSs, are only able to handle simple types and in some cases simple objects (e.g. Binary Large Objects - BLOB) or collections. The ORDBMS data model defines limitations to its ability to model data due to its organization of tables. The OODBMSs lack a common data-model like the ORDBMS/RDBMS and some consider this a major weakness of the OODBMSs upon it from a different perspective - OODBMSs are new and only research into the area will give the answer.

One thing is for sure: OODBMS has still a long way to go for fully replacing the existing RDBMS and ORDBMS, which are used very widespread in organizations as the data source for information systems. Computing is becoming object-oriented. But traditional information systems and data processing methods, database systems, legacy system continue their existence and serve the organizations as they did in the past. OODBMS does not look like to take over the world in the very near future. As King

[48] Looney K., Koch George, *Oracle 8i, The Complete Reference*, Oracle Press, 2000, p.73.
[49] Ibid, p.74.
[50] Ramakrishnan R, Gehrke J, *Database Management Systems,* McGraw-Hill Higher Education, 2000, p.736.

describes; "*Like wine makers, database management people don't favor pouring old wine into new bottles. Call them conservative, but given the mass of data stored in existing relational and mainframe databases, the tendency is to look askance even at new data going into new types of containers. That, in a metaphoric nutshell, is the situation faced by object-oriented database management systems (OODBMSs) as they seek to become mainstream products.*" [51]

3.5 Relational Databases

3.5.1 History in Brief

Before the relational databases emerged, IBM had already developed its hierarchical database product IMS in 1968, Charles Bachman developed CODASYL-based network model product IDS in General Electric (1961).

While all these were happening, at least one researcher at IBM was dissatisfied with both the CODASYL products and IBM's database package. Edgar F. (Ted) Codd, an Oxford-trained mathematician, joined IBM in 1949 and later moved to IBM - San Jose. Codd found existing and new database technologies during this time as "*taking the old-line view that the burden of finding information should be placed on users" [In this view, the database management system] should only recognize simple commands and it would be up to the users to put together appropriate commands for finding what was needed*" [52]

With the dissatisfaction of the existing products, Codd produced some technical reports in IBM, followed by his milestone paper "A Relational Model of Data for Large Shared Data Banks", where he outlined the relational approach to organize data in databases. The relational model consisted of:

- Independence of data from machine (hardware) and storage implementations
- High-level non-procedural language for accessing data.

Codd summarized the relational model in his abstract as "*Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as' a result of changes in query, update, and report traffic and natural growth in the types of stored information.*"[53]

---

[51] King Nelson H., Object DBMSs : Now or Never, DBMS Online, Volume 10, Number 8, July 1997, p.62.

[52] The Rise of Relational Databases, *Funding a Revolution: Government Support for Computing Research*, National Research Council Report, National Academy Press, Washington, D.C. 1999, Part-II Case Studies in Computing Research.

[53] Codd, E. F., *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM (CACM), Volume 13, Number 6, June 1970, p.377-387.

In the same paper, Codd, then describes the advantages of relational model over the hierarchical and network modes : "*The relational view (or model) of data described [in Section 1] appears to be superior in several respects to the graph or network model presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only -- that is, without superimposing any additional structure for machine representation poses. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.*

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations.

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed." [54]

While all these were happening, IBM had already went further with IMS. The company has already made financial investments on IMS. Codd had to prove in many platforms his idea of relational model and convince people that it would be a success, both technically and commercially. After Codd published his paper openly. IBM had declared IMS as sole product and criticized Codd as acting against the company objectives. Codd, arranged a meeting with CODASYL supporter Charles Bachman. This was followed by two projects in 1970, to implement relational method. The first project was within IBM and was called the System R. The second project, Ingres, started at UC-Berkeley and was founded by National Science Foundation (NSF) and military.

In 1990, in his book, Codd defined the importance of the database management systems as : "Today, if you have a well-designed database management system, you have the keys to the kingdom of data processing and decision support."[55] The relational model of Codd is based on the mathematical theory of relations. The relational model is simple and elegant, where a database is defined as the set of relations and a relation is a table with rows and columns. The relational model enables the usage of SQL for data querying.

Relational model has various properties and concepts we will be defining in the following section and go into more detail of relational design concepts and topics.

3.5.2 Definitions

<u>Relation:</u>

[54] Ibid.

[55] Codd E. F, *The Relational Model for Database Management: Version 2*, Addison-Wesley Publishing Company, 1990.

Relational model is based on the mathematical theory of relations. In mathematical terms, a relation is defined as :

Given sets S_1, S_2,..., S_n (not necessarily distinct), R is a relation on these *n* sets if it is a set of *n*-tuples, the first component of which is drawn from the S_1, the second component from S_2, and so on. R is a subset of Cartesian product $S_1 \times S_2 \times \ldots \times S_n$. Relation R here is the degree of *n*. Each of the sets S_1, S_2,..., S_n on which a relation is defined is called a domain. We will study domains later in this chapter in more detail.

A relation R of the degree *n* that has the following attributes :

Attribute A_1 getting its values from D_1
Attribute A_2 getting its values from D_2,
...
Attribute A_n getting its values from D_n,

where D_1, D_2, ..., D_n are domains; the relation R is shown as

$R(A_1{:}D_1\ A_2{:}D_2 \ldots A_n{:}D_n)$, or more often shown as :

$R(A_1\ A_2 \ldots A_n)$ given that domains are listed seperately.

The ordering of A_1, A_2,... A_n is not of any significance in the theory. The mathematical relation is a set with special properties. All of its elements are tuples. These are all of the same type. The tuples are not of any particular order; the relation is an unordered set. This is juts what is needed for commercial databases, since many of the relations in such databases are each likely to have thousands of tuples, sometimes millions[56]. Considering large databases in organizations today, even millions of tuples in a relation is not a imaginary large number. Ordering, sorting, numbering addressing tuples in the storage is not the aim of the relational model, which also avoids to give the task of ordering of tuples to the users. The relational databases model follows the mathematical version very closely. Physically as a table, a relation R has the following distinctive properties.

- A tuple is represented by each row.
- The order of the rows in R are not relevant.
- Content of each rows are distinct form each other, they are not the same.

Table 3.2 lists the Relations in Mathematics versus Relations in the Relational Model.[57]. The atomic value means that, each value is indivisible in the relational model.

Another definition of relation can be given as :

A relation *r* of the relation schema $R(A_1, A_2,\ldots, A_n)$, also denoted by $r(R)$, is a set of n-tuples, $r = (t_1, t_2, \ldots t_n)$. Each n-tuple t is an ordered list of n values. $t = <v_1, v_2, \ldots, v_n)$, where each value v_i $1 <= i <= n$, is an element of domain A_i, or is specially null value.

[56] Ibid, p.2
[57] Ibid, p.4

A relation schema R, denoted by R(A_1, A_2,..., A_n) is made up of relation name R and a list of attributes A_1, A_2,..., A_n. Each attribute A_i, is the name of a role some played by some domain D in the relation schema R. D is the domain of A_i. Domain D is represented as dom(A_i).

The main construct to represent data in relational model is a relation. A relational database is a collection of relations. Relations are like tables, having columns (attributes, fields), and rows (tuples, records). Relations have unique names. A sample relation is shown in Figure 3.11, where columns and rows and multi-dimensional set structure of the relation is described.

In the Figure 3.11 we have a multi-set collection which makes up the relation. There is the set of tuples(row), set of columns (attribute) values within each column and finally set of tuple values within each tuple. These sets are shown as A, B, C respectively, in the figure.

A relational database is a collection of relations with unique and distinct names. A relational database schema is the collections of schemas for the relations in the database.

Table 3.2 Relations in Mathematics Versus Relations in the Relational Model

Mathematics	**Relational Model**
Unconstrained values	Atomic Values
Columns not named	Columns named
Columns distinguished from each other by position	Columns distinguished from each other and from domains by name
Normally constant	Normally varies with time

Domain:

A relational schema R, denoted by R(A_1, A_2,..., A_n), is made up of a relation name R and a list of attributes A_1, A_2,..., A_n. Each attribute A_i is the name of a role played by some domain D, in the relational schema R. D is called the domain of Ai and is represented by dom(A_i) .

A relation schema defines the domain of each field or column in the relation. A domain is referred to in a relation schema by the domain name. Domain D is the set of atomic values, which are indivisible within each domain. A domain defines the rules and possible values for each column in the relation. The values that appear in the column must be drawn from the domain associated with that column. This is essentially defining the data type of the field. Domains can be sometimes very useful to implement. Domains have to be defined with a name and a type of data.

A real life example of domain usage and benefits are given below:

Twelve countries in European Union have initiated the conversion to single currency (euro). The conversion will happen during the last week of December-2001 and the first week of January-2002. Organizations, governments have to convert their databases from their local currency amounts to euro amounts according to the given *conversion ratio. For example the conversion ratio for Irish Punt is 0.787564, i.e. 1 Euro = 0.787564 Irish Punts.*

In addition to many technical difficulties, one problem is to identify the monetary fields within the organizations databases. In our case, Organization has a relational databases system and monetary fields are defined as NUMBER(11,2), i.e. field type is number with precision (integer) 9 and scale (decimal) 2 positions. But this is not enough to quickly say that every NUMBER(11,2) field is monetary. Some of them can be rates, units or percentages (i.e. 14.50%) or any other numeric but non-monetary fields.

The Domain usage here becomes very much important to categorize numeric fields into business domains.

Relation : **STUDENT**

SID	NAME	FACULTYNO	DOB
3456	Joe Murray	123	12/01/1972
8976	Helen Sherlock	123	17/05/1970
1655	Mick Reidy	123	12/12/1972
1020	Sandeep Ahuja	134	01/07/1968
4040	Niamh O'Malley	135	10/06/1969

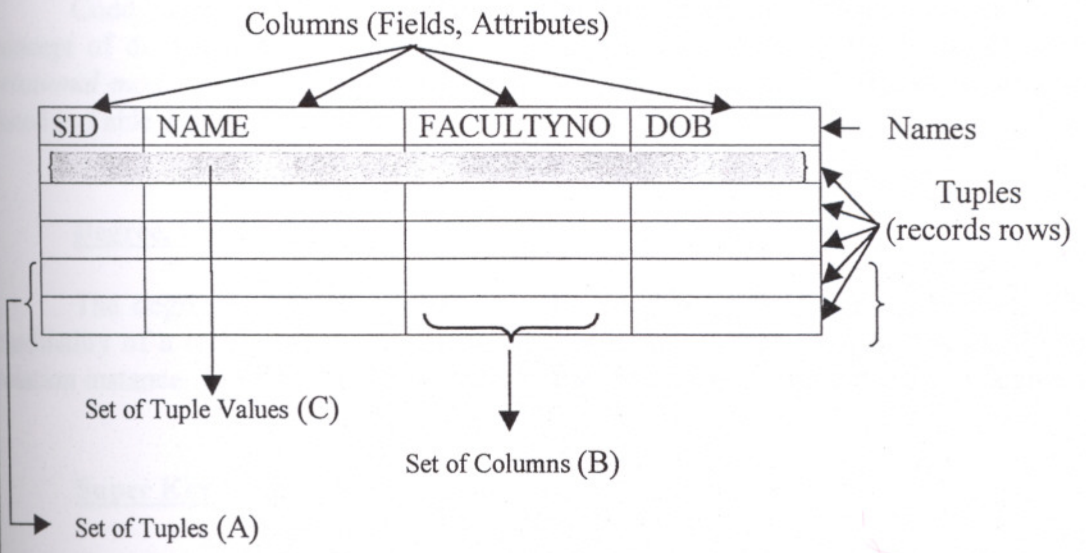


Figure 3.11 Relation.

Take the relation MEMBER having the listed fields and domain declarations. A member, the salary of the member and monthly pension contributions in 3 categories are given in this relation.

- MEMBERID	: D_STRING
- SALARY	: D_MONETARY
- CONTRIBUTION1	: D_ MONETARY
- CONTRIBUTION2	: D_PERCENTAGE
- CONTRIBUTION3	: D_ MONETARY

When the given domains are defined as

- D_STRING	: STRING 5
- D_MONETARY	: NUMBER 11,2
- D_PERCENTAGE	: NUMBER 11,2

it will imply that MEMBERID *is* STRING *type*, SALARY, CONTRIBUTION1, CONTRIBUTION3 *are* NUMBER *type but within the monetary domain and* CONTRIBUTION2 *is* NUMBER *type as well but within the percentage domain which clearly distinguishes monetary and non-monetary fields.*

Given the definition of domain, we can restate the definition of a relation within the context of domain:

A relation r(R) is a subset of the Cartesian product of the domains that define R $r(R) \subseteq (dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n))$. The Cartesian product defines all possible combinations of values from the domains. No value can exists in the relation which is not drawn for the underlying domain.

Codd, defined various practical reasons for relational DBMS to support the concept of domains, as he states that *"full support for many of the featured of the relational model depends on full support of the domain concept."* These reasons are listed in Table 3.3 as a summary.

Degree, Cardinality:

The degree (arity) of a relation is the number of fields in the relation. The cardinality of a relation instance, is the number of tuples (set of tuple values) in the relation instance. The STUDENT relation in Figure 3.11 has cardinality 5 and degree 4.

Super Key

Superkey of relation $R(A_1, A_2, \ldots, A_n)$ is the set of attributes $A \subseteq R$ with the rule that no two tuples t_1 and t_2 in any legal relation r on R have the property $t_1[A] = t_2[A]$.

Key

A key K is a superkey with the rule that removal of any attribute from K will cause it not to be a superkey anymore. Key is minimal as a difference from superkey.

Given the EMPLOYEE relation with attributes (EID, NAME, BIRTH-DATE, SALARY) , {EID} is a key. {EID, NAME} and {EID, NAME, BIRTH-DATE } are superkey.

Table 3.3 Practical Reasons for Supporting Domains.

Reason	**Description**
Integrity	Single most important concept in determining whether a given relational database is integrated. No relation is permitted from a non-existing domain.
Data Type Declaration	By defining the data types of domains only permitted data types can be used in the columns. Standardisation of a specific data type can be achieved.
Domain Integrity	Required id domain integrity is to be supported. Domain integrity consists of those integrity constraints that are shared by all the columns that draw their values from that domain. Frequently applied domain constraints are: Regular data types, Value ranges allowed, whether or not ordering functions (<,>) are applicable to the values
Domain-constrained Operators	Protect users from costly operations of comparison between semantically non-comparable values.
Transaction Support	It is necessary to support domains in order to support transactions that single out all occurrences of some value.
User-defined Integrity Checks	Enable users to define customised integrity checks for the values. If values in column C1 have to be sub-set of column C2 then it is required to have the domains defined for C1 and C2 which enables the user-defined constraint to work.
Support for Definitions	Primary key, foreign key, union compatibility, referential integrity and constraints
Naming Correspondence for UNION operation	When forming R UNION T is executed, it is required that the degree of R is equal to degree of T and there exists one-to-one mapping of the columns of R onto columns of T such that the two columns of each pair belonging to the mapping have a common domain.
Performance	Domains are important for performance of the database by supporting domain based indexes. Domain based index is a single index on the combination of all columns that draw their values for the domain. Once such an index is defined, a new column introduced to the database will automatically expand the domain based index.

3.5.3 Constrains in Relational Model

When considering constraints, we have to think of the importance of preserving the accuracy of data in organizational databases. Every day, critical or non-critical decisions are made in organizations, based on the results of various queries on the databases or outputs of information systems that generate reports for users. Garbage data will produce garbage output which may lead to garbage decision. The database therefore is only good and reliable if the stored data is good and reliable. In the previous section we have discussed the business data as one of the most important component of a DBMS. So it is up to the DBMS and database model to make sure the business data is valid and correct at any given moment. To achieve this relational database model has some integrity constraints.

An integrity constraint is a condition that is specified on a database schema, and restricts the data that can be stored in an instance of the database. If a database instance satisfies all the integrity constraints specified on the database schema, it is a legal instance. A DBMS enforces integrity constraints, in that it permits only legal instances to be stored in the database.[58]

The relational model has the preventive approach in maintaining the integrity and accuracy, which means that illegal instances are not permitted at first place. The opposite to this preventive approach would be in theory the corrective, which would be letting the illegal instances exists but enabling the users or DBA to correct them. Integrity constraints cannot be placed and executed in the user tools and applications connecting to the database. It must be a centralized task where there cannot be any by-pass.

Constraints that apply in every relational database are the entity integrity, such that no primary key or foreign key is allowed to have a missing value and referential integrity, such that for each distinct foreign-key value in relational database, there must exist in the database an equal value of a primary key from the same domain. If the foreign key is a composite key, the components that are foreign keys themselves, must exists in the database As components of at least one primary key value from the same domain. In the following section different types of constraints are discussed. These are - additional to entity integrity and referential integrity constraints- domain constraints, key constraints, foreign key constraints and user constraints

Codd, has a different way of categorizing the various constraints. In his categorization he has the following types of constraints D-TYPE, Domain Constraints ; C-TYPE, Column Integrity; E-TYPE, Entity Integrity; R-TYPE, Referential Integrity; U-TYPE, User defined Integrity. These are the so-called CURED constraints (CURED composed of the first letters of the five types)

3.5.3.1 Domain Constraints

A given value for an attribute A must be an atomic value from the domain of A (dom(A)). The type of the attribute A must also follow the defined domain type it draws its values from. Normally domains data-types are defined as the standards types such as Integer, String, Real etc. Any violation to the domain constraint is not permitted. An application program violating this constraint will be returned by an error code from DBMS which will be indicating that the specific command given by the application is not executed. This is an error situation where the front-end application has to normally a way of interpreting the error code from the DBMS and generate a understandable, user-friendly message or log for the users, specially for online front-end applications.

[58] Ramakrishnan R, Gehrke J, *Database Management Systems,* McGraw-Hill Higher Education, p.56, 2000

A good practice of system development is to be able to minimize the domain violations at the point of their occurrence, i.e. control them before they can occur. This may sound as a contradiction to what we have stated in Section 3.5.4 but it is in fact not such. If a given database column can only accept a set of numbers, for example the possible ages of employees cannot be less then 16, then it is always a good practice to limit the entry selection to numbers which are greater than 16.

3.5.3.2 Key Constraints

As we have defined before, in a relation composed of tuples, each tuple must be unique, i.e. no two tuples can be the same. There are other subsets of attributes of a relation schema R with the property that no two tuples in any relation instance *r* of R should have the same combination of values for these attributes. If a such subset of attributes is denoted as SK, then for any two distinct tuples t_1 and t_2 in a relation, instance *r* of R, we have the constraint that

$$t_1[SK] \neq t_2[SK]$$

Here, SK is called the superkey of the relation schema R. Superkey is the set of attributes that contains a key. A key uniquely identifies a tuple. A superkey can have redundant attributes but a key cannot. Given key K of a relation schema R is a superkey of R with the additional property that removing any attribute A from K leaves a set of attributes K^1 that is not a superkey of R. A key is a minimal superkey; a superkey from which we cannot remove any attributes and still have the uniqueness.

To further examine the key constraints, we have the sample EMPLOYEE relation in Figure 3.12 where we have same sample tuples and attributes. Set {EID} is a key for EMPLOYEE because no two employee tuples can have the same data value for EID. Take a set of attributes, for example {EID, FNAME, LNAME, DOB}which contains EID. This set of attributes is a superkey. This superkey is not a key for EMPLOYEE, since when removing FNAME, LNAME, or DOB or all of them from the set leaves still behind a superkey. A relation may have multiple keys, which are called the candidate keys. One of the candidate keys is designed to be the primary key which identifies the tuples in the relation.

EMPLOYEE

EID	FNAME	LNAME	DOB	PHONE
1226	Mick	Reidy	13/08/1945	7042020
1235	Joe	Murray	04/10/1950	7043239
1256	Helen	Sherlock	01/06/1965	7032929
1500	Joe	Soap	19/05/1072	7052101
1345	Grainne	Whelan	12/03/1967	7032001
1200	Nataraj	Balajee	04/02/1975	7042921

Figure 3.12 EMPLOYEE Relation.

Set {EID} listed below is a key EMPLOYEE.

EID
1226
1235
1256
1500
1345
1200

Set {EID}

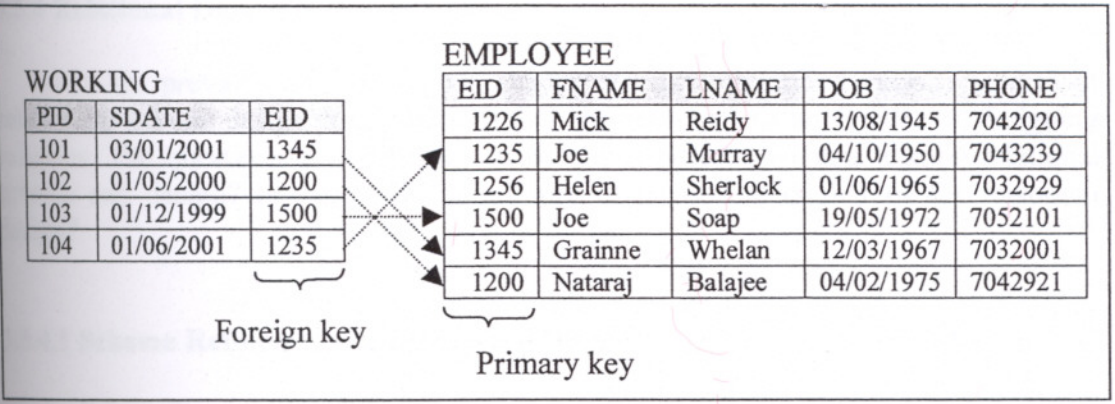
Set {EID, FNAME, LNAME, DOB} listed below is a superkey but not a key for EMPLOYEE. When removing the FNAME, LNAME, DOB or all of them, we still get a superkey

EID	FNAME	LNAME	DOB
1226	Mick	Reidy	13/08/1945
1235	Joe	Murray	04/10/1950
1256	Helen	Sherlock	01/06/1965
1500	Joe	Soap	19/05/1072
1345	Grainne	Whelan	12/03/1967
1200	Nataraj	Balajee	04/02/1975

Set{EID,FNAME, LNAME ,DOB}

3.5.3.3 Foreign Key Constraints

In the cases where the information stored in a relation is linked (via a foreign key) to another information stored in another relation we have to consider the foreign key constraint which enables the DBMS to check and ensure the consistency of the data across the multiple relations. In Figure 3.13, we have an additional relation, WORKING along with the EMPLOYEE relation where WORKING has the attributes, PID, SDATE, EID. To ensure the consistency of the data, we have to make sure that the EID attribute in WORKING tuples must also appear in the EID attribute of some tuples of EMPLOYEE. Here, the EID in WORKING relation is a foreign key and refers to the EMPLOYEE relation, where the EID is a primary key.


WORKING

PID	SDATE	EID
101	03/01/2001	1345
102	01/05/2000	1200
103	01/12/1999	1500
104	01/06/2001	1235

EMPLOYEE

EID	FNAME	LNAME	DOB	PHONE
1226	Mick	Reidy	13/08/1945	7042020
1235	Joe	Murray	04/10/1950	7043239
1256	Helen	Sherlock	01/06/1965	7032929
1500	Joe	Soap	19/05/1972	7052101
1345	Grainne	Whelan	12/03/1967	7032001
1200	Nataraj	Balajee	04/02/1975	7042921

Figure 3.13 WORKING and EMPLOYEE Relations.

Inserting a tuple <103,01/10/2001,9999> to WORKING foreign key constraint because there is no tuple in EMPLOYEE with the primary key 9999. If we delete the tuple <1500, Joe, Soap, 19/05/1972, 7052101> from the EMPLOYEE relation, the foreign key constraint will be violated because there are still at least one tuple referring to the one we are trying to delete.

3.5.3.4 User-defined Constraints

In addition to the previously given integrity constraints, user-defined integrity constraints are required for DBMS to enforce which include different organizational, legal rules and governmental regulations that can not be already built-in to the DBMS but can be defined by the DBA. User-defined integrity constraints are compiled and entered to the database catalog. An example of user-defined constraints can be given as follows.

Assume the employees will be given a salary increase which the DBMS cannot control on how to set the new salaries. When the new salaries are entered from a front-end user application or calculated and becomes a row to be committed to take effect, the front-end tool does not have anymore control on it. This is the moment where the DBMS takes action and check that the new salaries are compliant with he constraints defined. The constraint conditions for this can be given as

```
Condition 1:  SALARY_NEW < MAX_SALARY
              Where TITLE_CODE  = t

Condition 2:  (SALARY_NEW–SALARY-OLD)<(SALARY_OLD x
              INC_PERCENT)
              Where TITLE_CODE  = t
```

When the update of salary is executed, the testing of the constraints will be triggered and the following actions will be taken by the DBMS:

```
    UPDATE SALARY : If Condition 1 = TRUE Then ACCEPT
    UPDATE SALARY : If Condition 2 = TRUE Then ACCEPT
```

3.5.4 Relational Database Design Concepts

In the previous sections of this chapter we have described various aspects of relation model and design considerations that are necessary to implement on designing relations. The questions that still need to be answered are: What is a good relational schema and design? Is there any formal way of measuring quality of the relational design?

3.5.4.1 Schema Refinement and Decomposition

Informally, we may address some measures for high-quality relational design. These can be listed as:

- Semantics of the attributes: When we design a relation, we assume that certain meanings are assigned to the relation. Meaning are associated with the attributes it holds. The meaning (semantics) identifies what sort of attribute values would be in the relation and what is the relation of attribute values to one another. This is a very much subjective and common-sense measure.

- Reducing the redundant values in tuples: As described in this section, redundancy is the enemy of good design and efficient implementation

- Reducing null values in the tuples of relations: If many of the attributes do not apply to the tuples of a relation, then we will end-up with null values. Null values are difficult to handle and interpret. Sometimes even the meaning of the null value may be unclear. Is it null because the attribute does not apply to this tuple ? Is it null because the attribute applies to the tuple but the value is currently unknown ? or is it null because the attribute applies to the tuple, the value is know but not processed (i.e. updated) yet ?

Conceptual database design provides a set of relational schemas and integrity constraints. These are the definitely start-points for good database design but are not the only points to consider during the design phase. When ER designs are translated into relational schemas these inherit nonetheless redundancy problems which forces us to refine the schema and eliminate these problems. We can list the problems caused by redundancy as follows.

- Update Problems: Multiple copies of stored data has the problem of updating each copy correctly and has a big risk of inconsistency after the update operation.

- Insertion Problem: It may not be possible to insert data unless some other data has to be inserted as well.

- Deletion Problems: It may be impossible to delete a data without loosing some other required information.

- Storage redundancy : Waste of storage. Considering the cheap availability of storage media in the modern days still does not justify the storage redundancy, since this has also an effect on the overall performance of the database system.

Many real life redundant design cases today exist in some organizations because of lack of understanding of the importance of the subject or lack of quality measures and audit on designs. Production support, data verification become easily overwhelming tasks which could be in fact minimized when redundancy is taken care of. Many business users suffer from the inconsistency of multiple instances of the same data in the databases and require technical support from developers or database administrators when processing the data. The tools and the database at the background becomes easily a struggle for the users to achieve business objectives and perform required tasks.

For the purpose of examining the problems above and discuss the schema refinement and decomposition we will work on the following example:

The relation is defined as below with the given attributes where the key is the EID (employee id). The DAILY-SALARY is defined by RATING. For a given RATING value there is a defined DAILY-SALARY figure.

SALARY (SSN, NAME, RATING, DAILY-SALARY, D-WORKED)

Figure 3.14 shows the populated relation of SALARY relation on which we will examine various redundancy related issues. Same value of RATING attribute will lead to the same value of DAILY-SALARY attribute. Information in this case is stored multiple times, where RATING value 8 and DAILY-SALARY 100 are repeated 3 times in the relation. Similar case is valid for RATING value 6 and DAILY-SALARY 75

SALARY

EID	NAME	RATING	DAILY-SALARY	D-WORKED
1226	Mick Reidy	8	100	5
1235	Joe Murray	8	100	5
1256	Helen Sherlock	6	75	4
1500	Joe Soap	6	75	4
1345	Grainne Whelan	8	100	3
1200	Nataraj Balajee	5	50	4

Figure 3.14 SALARY Relation.

where the repetition is twice. If we plan update the DAILY-WAGES for the first tuple, where the RATING 8 will now define 125 instead of 100, we cannot do this without updating the second and firth tuples as well. This is an update problem in the relation. We cannot add a new tuple to the relation without giving the new RATING and corresponding DAILY-SALARY. This a typical insertion problem in the relation. If we delete the tuple for the employee Nataraj Balajee then we loose the RATING, DAILY-SALARY information for RATING 5. This is an example for delete problem.

To overcome the given situation above there is a process defines as the decomposition. Basically, decomposition means to replace a given relation with set of smaller relations where each of them has a subset of attributes of the original relation. This is done with the help of identifying functional dependencies in the original redundant relation.

In our case we can apply the following decomposition and end-up with the following relations :

DAYS-EMPLOYEE (EID, NAME, RATING, D-WORKED)
SALARY (RATING, DAILY-SALARY)

Figure 3.15 shows the populated relations after decomposition. We have eliminated addition problem, for both DAYS-EMPLOYEE and SALARY where we can add new tuples to both relation without the requirement of knowing any other information. Changing the DAILY-SALARY is an single tuple update in SALARY relation, unlike in the original version where multiple tuples had to be updated. We can delete an employee without loosing any SALARY information.

Even though everything sounds great with this decomposition approach, there are still issues and questions around it : Is a relation required to be decompose ? Did we introduce new problems by decomposing a relation as in example case ? These questions will be answered with the normalization methods and functional dependency analysis described in the following sections. Another derived problem of the decomposition is the potential requirement of the joins for the queries on the original relation. This over and over joining may cause to performance issues which may be more severe than the original redundancy problems. We will be discussing further relational database design aspect in the next sections.

DAYS-EMPLOYEE

EID	NAME	RATING	D-WORKED
1226	Mick Reidy	8	5
1235	Joe Murray	8	5
1256	Helen Sherlock	6	4
1500	Joe Soap	6	4
1345	Grainne Whelan	8	3
1200	Nataraj Balajee	5	4

SALARY

RATING	DAILY-SALARY
8	100
6	75
5	50

Figure 3.15 DAYS-EMPLOYEE and SALARY Relations.

3.5.4.2 Functional Dependency

We already have raised the questions regarding good relational design and how to distinguish a "good" version from a "bad" version with solid mathematical evidence and not with subjective design criteria. Relational model enables us to do so with the help of functional dependency and normalization theory.

The single most important concept in relational design is that of a functional dependency.[59] Normalization methods are based on the functional dependency theory. Functional dependency is a constraint between two sets of attributes from the relation. Before formally defining the functional dependency we shall define the functional mapping between attributes of a relation and work on a sample relation for clarifying the definition of functional dependency. Functional dependencies allow us to express constraints that cannot be expressed using superkey.

[59] Elmasri R, Navathe S. B., *Fundamentals of Database Systems,* Second Edition, The Benjamin / Cummins Publishing Company, p.401, 1994

Assume the two attributes EID (employee id) and TILE define the employees. EID is given as unique, i.e. representing one and only one employee. Each employee can have one title, but multiple employees can have the same title. This will lead us to the conclusion that there may exists a many-to-one mapping between EID and TITLE. This mapping is defined as the functional mapping and shown as EID → TITLE. Here the EID is named as domain and TITLE is named as range. In other words EID determines TITLE or TITLE depends on EID. Such a mapping is defined as functional dependency (FD).

In the given relation DPM (DEPT, PRODUCT, MANAGER) with the following populated data we will check the DEPT → MANAGER or MANAGER → DEPT as we claim they exist. According to the definition we can have a many-to-one or one-to-one DEPT → MANAGER if the FD holds. Same is valid for MANAGER → DEPT.

DPM

DEPT	PRODUCT	MANAGER
D1	P1	M1
D1	P2	M1
D1	P3	M2
D2	P1	M3
D2	P4	M3

For DEPT → MANAGER test, taken the values for DEPT we have D1 and D2. From the values above we se that DEPT → MANAGER cannot be validated because for D1 we have M1 and M2 which is not many-to-one or one-to-one.

For MANAGER → DEPT test, taken the values for MANAGER we have M1 and M2 and M3. From the values above we se that MANAGER → DEPT is valid since it satisfies the many-to-one and one-to-one condition.

Functional dependency analysis is a practical and structural method of analyzing problems and relational design anomalies introduced during the database design phases, specially entity-relationship (ER) modeling that we will study in the next chapter. Furthermore functional dependency is a way of refining the relational schema created from ER model. ER model by its nature cannot enable the designer make refinement decisions specially for:

- Constraints on relationship set. (Business rules or policies of the business which have to be incorporated to the database design)

- Problem of identifying attributes of entities in the ER model. (Does an Attribute A belong to Entity E_1 or E_2 ?)

- Problem of defining and identifying entities for representing business rules in the database design. (Do we need an entity E or not ?)

Definition

A functional dependency (FD), denoted by X → Y, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation instance *r* of R. The constraint states that, for any two tuples t_1 and t_2 in r such that $t_1[X] = t_2[X]$, we must also have $t_1[Y] = t_2[Y]$. This means that the values of the Y component of a tuple *r* depend on, or are determined by, the values of the X component; or alternatively, the values of the X component of a tuple uniquely (or functionally) determine the values of Y component.

Inference Rules for Functional Dependencies[60]

REFLEXIVITY: if $X \supseteq Y$, then $X \rightarrow Y$. Or, $X \rightarrow X$ and $Y \rightarrow Y$, where every X determines it self and every Y determines itself.

Proof:
If $X \supseteq Y$ and tuples t_1 and t_2 in some relation instance *r* of R such that $t_1[X] = t_2[X]$. Then $t_1[Y] = t_2[Y]$ because of $X \supseteq Y$ $\therefore$ $X \rightarrow Y$ and also $\therefore X \rightarrow Y$

AUGMENTATION: if $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z. Or, $XZ \rightarrow Y$.

Proof:
Assume that $X \rightarrow Y$ holds in a relation instance *r* of R but that $XZ \rightarrow YZ$ does not hold. Then there must exist two tuples t_1 and t_2 in *r* such that:

(a) $t_1[X] = t_2[X]$
(b) $t_1[Y] = t_2[Y]$
(c) $t_1[XZ] = t_2[XZ]$
(d) $t_1[YZ] \neq t_2[YZ]$ which is not possible since (e) is deduced from (a) and (b)
(e) $t_1[Z] = t_2[Z]$ and (f) is deduced from (b) and (e)
(f) $t_1[YZ] = t_2[YZ]$ which is contradicting (d).

TRANSITIVITY: If $X \rightarrow Y$, $Y \rightarrow Z$, then $X \rightarrow Z$.

Proof:
Assume that
(a) $X \rightarrow Y$ and
(b) $Y \rightarrow Z$ hold in a relation *r*. then for any two tuples t_1 and t_2 in *r* such that $t_1[X] = t_2[X]$, we must have
(c) $t_1[Y] = t_2[Y]$ (from (a)), and hence we must have
(d) $t_1[Z] = t_2[Z]$ (from (c) and (b)); $\therefore$ $X \rightarrow Z$ must hold in *r*.
DECOMPOSITION: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$.

Proof:
(a) $X \rightarrow YZ$ (given),
(b) $YZ \rightarrow Y$ (using reflexivity and $YZ \supseteq Y$)
(c) $X \rightarrow Z$ (using transitivity on (a) and (b))

[60] Ibid, p.404

ADDITIVE (UNION): If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$. Or, $X \rightarrow Y$ and $W \rightarrow Z$, then $XW \rightarrow YZ$ (if $X = W$ then $X \rightarrow YZ$)

Proof:
(a) $X \rightarrow Y$ (given),
(b) $X \rightarrow Z$ (given),
(c) $X \rightarrow XZ$ (using augmentation on (a) with X; where $XX = X$),
(d) $XY \rightarrow YZ$ (using augmentation on (b) with Y)
(e) $X \rightarrow YZ$ (using transitivity on (c) and (d))

PSEUDO-TRANSITIVITY: If $X \rightarrow Y$, $YW \rightarrow Z$, then $WX \rightarrow Z$

Proof:
(a) $X \rightarrow Y$ (given),
(b) $WY \rightarrow Z$ (given),
(c) $WX \rightarrow WY$ (using augmentation on (a) with W),
(d) $WX \rightarrow Z$ (using transitivity on (c) and (b))

The Membership Algorithm[61]

If an FD is derived from another existing FD, then this is called a redundant FD. During the various transformations and decompositions of a schema design, it has to be ensured that semantics are preserved. To do this the proof of equivalence of FDs between the two versions of the schemas is required. The staring point for the proof is the non-redundant set of FDs in both schemas which is obtained by minimality.

The inference rules reflexivity, augmentation and transitivity are known as Armstrong inference rules. With these three rules , given a functional dependency set F, on a schema R , any dependency that can be inferred from F by using reflexivity, augmentation and transitivity holds in every relation state *r* defined on R that satisfies the dependencies in F. Again by using reflexivity, augmentation and transitivity rules, we can infer dependencies until no further dependencies can be inferred, thus resulting to complete set of all possible dependencies that can be inferred from F. Furthermore, the set of dependencies F^+, which are called the closure of F, can be determined from F by using the first three rules. The set of all functional dependencies logically implied by F is the closure of F. In other words, the closure of F, F^+, is calculated from given set f of FDs by using Armstrong's rules.

Given the relation R, R(<u>**A**</u>, B, C, D, E, F, G), where A is the key, given the following FDs : $A \rightarrow ABCDEFG$, $CE \rightarrow A$, $BD \rightarrow E$, additional FDs can be computed as the members of the set F^+.

From $CE \rightarrow A$ and $A \rightarrow ABCDEFG$: $CE \rightarrow ABCDEFG$
From $BD \rightarrow E$ and augmentations : $BDC \rightarrow EC$
From $CE \rightarrow ABCDEFG$ and $BDC \rightarrow EC$: $BDC \rightarrow ABCDEFG$

[61] Ozkarahan Esen, *Database Management. Concepts, Design and Practice*, Prentice-Hall International Editions, p.108, 1990

From $A \rightarrow ABCDEFG$: $A \rightarrow A$, $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $A \rightarrow F$, $A \rightarrow G$

The of the membership algorithm is to show the redundancy of a FD by using the inference rules. Here we first need to apply the simplifications on the given set of FDs. Next step is the application of inference rules to the set F of FDs. The problem here is the be able to calculate the closure of F, F^+, to verify that a tested FD is implied by F^+. To avoid the difficult and costly task of calculating F^+ another alternative is proposed. X^+ is the closure of set of attributes X in R for FD set F. X^+ (the closure of set of attributes) is obtained such that the FD $X \rightarrow A$ can be deduced from F by the inference rules and enables the test of FD $X \rightarrow B$ in F without calculating the F^+. This algorithm is called the membership algorithm.

Given the set F of FDs below, we will test each FD $A \rightarrow B$ against the remainder $F' = F - (A \rightarrow B)$ to find out if $A \rightarrow B$ is member of F'^+ and if so to conclude that we can ignore $A \rightarrow B$ in F since F' will imply this.

R(A, B, C, G, H, I), F = { $A \rightarrow CB$, $CG \rightarrow HI$, $B \rightarrow H$, $A \rightarrow H$, $AG \rightarrow H$ }

Applying further simplification (decomposition) will lead to :

(1) $A \rightarrow B$
(2) $A \rightarrow C$
(3) $CG \rightarrow H$
(4) $CG \rightarrow I$
(5) $B \rightarrow H$
(6) $A \rightarrow H$
(7) $AG \rightarrow H$

Test of $A \rightarrow B$
1. X = A
2. X = AC from (2)
3. X = ACH from (6)
4. No more additions can be made to $X \Rightarrow A \rightarrow B$ is not redundant.

Test of $A \rightarrow C$
1. X = A
2. X = AB from (1)
3. X = ABH from (5)
4. No more additions can be made to $X \Rightarrow A \rightarrow C$ is not redundant.

Test of $CG \rightarrow H$
1. X = CG
2. X = CGI from (4)
3. No more additions can be made to $X \Rightarrow CG \rightarrow H$ is not redundant.

Test of $CG \rightarrow I$
1. X = CG
2. X = CGH from (3)

3. No more additions can be made to $X \Rightarrow CG \rightarrow$ I is not redundant.

Test of $B \rightarrow H$
1. $X = B$
2. No more additions can be made to $X \Rightarrow B \rightarrow$ H is not redundant.

Test of $A \rightarrow H$
1. $X = A$
2. $X = AB$ from (1)
3. $X = ABC$ from (2)
4. $X = ABCH$ from (5)
5. $H \subseteq ABCH$; $A \rightarrow H$ follows from F' and is redundant

Test of $AG \rightarrow H$
1. $X = AG$
2. $X = AGH$ from (6)
3. $H \subseteq AGH$; $AG \rightarrow H$ follows from F' and is redundant

We have end-up with this algorithm with a set of non-redundant FDs.

Minimal Set of Functional Dependencies

A set of FDs is minimal if
- Every dependent attribute is a single attribute.
- No determinant attribute is redundant.
- There is no redundant FD in the set of FDs

The database design process requiring the test of equivalence of two schemas requires. Given the set of FDs in both schemas are F_1 and F_2

- Equivalence of set of FDs in both schemas.
- Closure of F_1 (F_1^+)= Closure of F_2 (F_2^+).
- Deleting a FD from F_1 to obtain F_1' implies : $F2 \neq F_1'$

Minimality Example:

If $F = \{ ABCD \rightarrow E, E \rightarrow D, A \rightarrow B, AC \rightarrow D\}$

$ABCD \rightarrow E$:
Replace by $AC \rightarrow E$ (from $A \rightarrow B$, $AC \rightarrow D$); $AC \rightarrow E$

$AC \rightarrow E, E \rightarrow D \Rightarrow AC \rightarrow E$
By transitivity and can be removed

$AC \rightarrow E, E \rightarrow D, A \rightarrow B$;
No redundant FD : Minimal set of FDs

3.5.5 Normalization

Normalization process was first proposed by Codd. In general the normalization process takes a relation through a series of test to "certify" whether or not it belongs to a specific normal form. Normalization of data can be looked as a process during which unsatisfactory relation schemas are decomposed by breaking up their attributes into smaller relation schemas that posses desirable properties.[62] For a given schema, we should be able to tell whether it is a "good" design or a "bad" design. If it is a bad design we need to be able to understand the reasons and problems that makes it a "bad" design. Normal forms and normalization is a guidance for such measurement in the relational model.

The original normalization proposal was aiming to resolve the update anomalies on relations we have described before in this chapter. In a wider view, normalization provide the designers with

- A formal method of relation analysis based on the keys and functional dependencies.
- Test series, that can be applied on relations so that it can be normalized. Failing test will be an indication of decomposition requirement of the relation it is applied.

We do require normal forms and normalization of relations because, they are pure structures that do not carry multiple disjoint facts and they do not cause update anomalies. Additional to normalization followings are required for a "good" design, since normalization does not alone guarantee this.

- Lossless schemas and joins.
- Dependency preservation, which ensures that all FDs are represented in some of the individual relations.

Lossless Schemas (Lossless Join Decomposition)

Decomposition of a schema R is the process of replacing the schema R by more relations which contain subset of attributes of R and altogether they include the totality of attributes of schema R. More formally, the relational schema is the set of attributes, R={A_1, A_2,...,A_n} where A_i denotes an attribute. Decomposition D, creates a set of relational schemas from R, i.e. D ={ R_1, R_2,...,R_n }. D is called the decomposition of R.

Lossless join decomposition is defined as the decomposition which has the attribute preservation and relational preservation. Attribute preservation is achieved by satisfying the rule of no attributes being left out during decomposition or no spurious tuples are introduces. Given the schema R(X, Y, Z) , where decomposed relations are given as R_1(X, Y)and R_2 (Y, Z) . $R \neq R_1 \cup R_2$ concluding that the decomposition is not Lossless where tuples (x1, y1, z3) and (x3, y1, z1) are spurious an are not in the original relation. An algorithmic way of lossless join decomposition test is as follows:

[62] Elmasri R, Navathe S. B., *Fundamentals of Database Systems,* Second Edition, The Benjamin / Cummins Publishing Company, p.407, 1994

R

X	Y	Z
x1	y1	z1
x2	y2	z2
x3	y1	z3

$\rightarrow$

R_1

X	Y
x1	y1
x2	y2
x3	y1

R_2

Y	Z
y1	z1
y2	z2
y1	z3

$\longrightarrow$

$R_1 \cup R_2$

X	Y	Z
x1	y1	z1
x1	y1	z3
x2	y2	z2
x3	y1	z1
x3	y1	z3

Algorithm : Test for Lossless Join[63]

Step 1:
Create a matrix S with Row(i) for relation R_i of D and Column(j) for attribute A_j in R.

Step 2:
Assign $S(i, j) = b_{ij}\ \forall i,j$

Step 3:
For i = 1 to n representing R_i and j = 1 to m representing A_j (where n, m total number of rows and columns), if R_i includes Aj then $S(i, j) = a_j$

Step 4:
For each FD $A \rightarrow B$ in F,
For all rows in S which have the same symbol in the columns for attribute A; make the symbols in each column that corresponds attribute B be the same in all these rows such that if any of the rows has an "a" symbol for the column, set the other rows the that same "a" symbol in the column. If no "a" symbol exists for the attribute in any of the rows, choose one of the "b" symbols that appear in one of the rows for the attribute and set the other rows to that "b" symbol in the column.

Step 5: If a row is made of "a" symbol then the decomposition has the lossless join property, otherwise is not lossless.

Execution of the Algorithm on an Example

R = (A, B, C, D, E, F)
F = { $A \rightarrow B$, $C \rightarrow DE$, $AC \rightarrow F$ }
R_1 = (A, B), R_2 = (C, D, E) R_3 = (A, C, F)

Step 1 & 2:

S	A	B	C	D	E	F
R_1	b11	b12	b13	b14	b15	b16
R_2	b21	b22	b23	b24	b25	b26
R_3	b31	b32	b33	b34	b35	b36

İZMİR YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
REKTÖRLÜĞÜ
Kütüphane ve Dokümantasyon Daire Bşk.

[63] Ibid, p.428

Step 3:

S	A	B	C	D	E	F
R_1	a1	a2	b13	b14	b15	b16
R_2	b21	b22	a3	a4	a5	b26
R_3	a1	b32	a3	b34	b35	a6

Step 4 :

FD : A $\rightarrow$ B, Row 1 and Row 3 agree on values in column A (a1) : Change b32 to a2
FD : C $\rightarrow$ DE, Row 2 and Row 3 agree on values in column C (a3) : Change b34 to a4 b35 to a5

Step 4:

S	A	B	C	D	E	F
R_1	a1	a2	b13	b14	b15	b16
R_2	b21	b22	a3	a4	a5	b26
R_3	a1	a2	a3	a4	a5	a6

FD : AC $\rightarrow$ F: No two rows are the same in AC Columns, no change can be made.

Step 5: Row 3 is (a1, a2, a3, a4, a5, a6), so this decomposition is lossless.

Dependency Preservation

Dependency preservation is achieved by satisfying the rule, that FDs in set F on R do appear in the decomposed relations $R_1, R_2, \ldots, R_n$. If a schema S is given with the FD set F;

- The attributes of S are investigated and attributes which are not part of any FD are separated. These attributes are made to be a relation.
- If an FD A $\rightarrow$ B exists in F where AB contain all attributes of F then S is singe relation.
- Otherwise, a relation R_i is made from each $A_i \rightarrow B_i$
- If there exists a FD as A $\rightarrow B_i$ in F for all attributes B_i ($1 <= i <= n$) the a single relation of $AB_1B_2\ldots B_n$ is built.

As an additional rule; if the schema is a decomposition of R, then the key of R must be itself a relation or must be in one of the decomposed relations, to make the decomposition lossless and dependency preserving.

3.5.5.1 First Normal Form

- First normal form was defined to disallow multi-valued and composite attributes.
- In first normal form, every field must contain an atomic value.
- No nested or sub relations can exist in an relation.

The relation EMP_WORK, below is not in fist normal form. R is a relation with attributes employee id (EMPID), employee name (EMPNAME), projects worked

(PROJECTS_WORKED) which it self is a relation of project id (PROJID), total hours worked (H_WORK).

EMP_WORK (EMPID, EMPNAME, PROJECTS_WORKED)

EMPID	EMPNAME	PROJECTS_WORKED	
		PROJID	H_WORK
001	Joe Murray	{Y2K EURO}	{500 250}
002	Bob Hamilton	{Y2K EURO}	{1000, 325}

EMP_WORK as populated with the following attributes as first normal form and with redundancy. In this case the primary key is EMPID, PROJID

EMP_WORK (EMPID, EMPNAME, PROJID, H_WORK }

EMPID	EMPNAME	PROJID	H_WORK
001	Joe Murray	Y2K	500
001	Joe Murray	EURO	250
002	Bob Hamilton	Y2K	1000
002	Bob Hamilton	EURO	325

The relation EMP_WORK is decomposed into EMPLOYEE and EMP_PROJECTS to obtain non-redundant first normal form where the primary key is migrated to EMP_PROJECTS to combine with the partial key of PROJID in EMP_WORK relation. This process is taking out the nested relations into stand-alone relations.

EMPLOYEE (**<u>EMPID</u>**, EMPNAME)
EMP_PROJECTS (**<u>EMPID</u>**, **<u>PROJID</u>**, H_WORK)

EMPLOYEE

EMPID	NAME
001	Joe Murray
002	Bob Hamilton

EMP_PROJECTS

EMPID	PROJID	H_WORK
001	Y2K	500
001	EURO	250
002	Y2K	1000
002	EURO	325

3.5.5.2 Second Normal Form

- Full functional dependency is the base of second normal form, where a given FD $A \rightarrow B$ is a full functional dependency if any removal of attribute from A implies that the FD is not valid anymore. Formally, for a given $X \in A$ ($A - \{X\}\ x\rightarrow B$) where $x\rightarrow$ implies no functional dependency. A non-full functional dependency (partial functional dependency) A $\rightarrow$ B, is the case if the attribute X $\in$ A can be removed from set X and the FD still is valid. Formally, for a given X $\in$ A (A – {X} $\rightarrow$ B). In the relation EMP_WORK EMPID,PROJID $\rightarrow$ H_WORK is a full FD but EMPID,PROJID $\rightarrow$ EMPNAME is partial dependency.

- If two disjoint fact are in the same relation, the relation is not in second normal form.

- A given relation R is in second normal form if every nonprime attribute A id fully functional dependent on the primary key of the relation.

In the relation EMP_WORK below, the non-prime attribute EMPNAME violates the full functional dependency rule considering the FD

EMPID $\rightarrow$ EMPNAME.
EMPID,PROJID $\rightarrow$ EMPNAME and EMPID $\rightarrow$ EMPNAME

making EMPID partially dependent to the primary key. Also PROJNAME (project name) and PROJLOC (project location) are violating full functional dependency because of the FD

PROJID $\rightarrow$ PROJNAME,PROJLOC.
EMPID,PROJID $\rightarrow$ PROJNAME,PROJLOC and
PROJID $\rightarrow$ PROJNAME,PROJLOC

EMP_WORK (**EMPID**,**PROJID**, H_WORK, EMPNAME, PROJNAME, PROJLOC)

FD : EMPID, PROJID $\rightarrow$ H_WORK
FD : EMPID $\rightarrow$ EMPNAME
FD : EMPID $\rightarrow$ PROJNAME, PROJLOC

To obtain the second normal form, the EMP_WORK relation is decomposed into three relations EMPLOYEE, EMP_PROJ and PROJECT with the given attributes and keys and full functional dependencies.

EMPLOYEE (**EMPID**, EMPNAME) EMPID $\rightarrow$ EMPNAME
PROJECT (**PROJID**, PROJNAME, PROJLOC) PROJID $\rightarrow$ PROJNAME, PROJLOC
EMP_PROJ (**EMPID**, **PROJID**, H_WORK) EMPID, PROJID $\rightarrow$ H_WORK

3.5.5.3 Third Normal Form

- Transitive dependency is the base for third normal form which is defined such that ; given a FD in R A $\rightarrow$ B, if there is a set of attributes X which is not a subset of any key in R and A $\rightarrow$ X and X $\rightarrow$ B is valid.
- Codd defines a relation R in third normal form if "it is in second normal form and has no transitive dependency of any non-prime in primary key".

Relation EMP_DEPT below is in second normal form since no partial dependency exists but because of the functional dependencies EMPID → DEPTID (department id), DEPTID → DEPTNAME (department name). and DEPTID is not part of the key EMPID.

EMP_DEPT (**<u>EMPID</u>**, EMPNAME, GENDER, PHONE, DEPTID, DEPTNAME)
FD : EMPID → EMPNAME, GENDER, PHONE, DEPTID
FD : DEPTID → DEPTNAME

Decomposing the relation EMP_DEPT to EMPLOYEE and DEPARTMENT gives third normal form relations.

EMPLOYEE (**<u>EMPID</u>**, EMPNAME, GENDER, PHONE, DEPTID)
DEPARTMENT (**<u>DEPTID</u>**, DEPTNAME)

For a relation to be in third normal form, it has to be obeying the rules of being fully functional dependent on all keys of R, and no transitive dependent on every key of R.

3.5.5.4 Boyce-Codd Normal Form

Schemas with relations of third normal form are considered as "good" designs. Third normal form resolves anomalies of insertion, deletion and updates in the relations. Any design which is at least not in the third normal form will be introducing these anomalies. Boyce-Codd normal form is a normalization with additional restriction on third normal form. As seen in the previous section, a relation needs to be fully functional in other words there should be no partial dependency and should not allow transitional dependency which implies that no FD with non-prime dependent or key determinant should exist. In Boyce-Codd normal form the second condition, which allows in an FD A → B, B to be prime attribute if A is not a superkey, is not valid. A Boyce-Codd normal form is a normalized relation where all determinant attributes are candidate keys.

3.5.5.5 Forth Normal Form

Even when using FD and normalization approach to decompose relations to avoid anomalies and redundancies, in real life there can be cases where the FD theory does not apply and we may have to deal with such cases. Given the example In the relation R below with attributes PROJNAME, EMPNAME PROJLOC where a project has employees working and project has locations. Project employees and locations are not dependent to each other. Here we have a case of one-to-many PROJNAME, EMPLOYEE relation and one-to-many PROJNAME, PROJLOC relation. The existence of these two one-to-many unrelated relations are the base for another functional dependency, called the multivalued dependency. The rule for multivalued dependency in the given relations is to keep all possible combinations PROJNAME, EMPNAME and PROJNAME, PROJLOC. The relation is populated as a sample for this condition is shown below. Where the possible combinations of tuples of both one-to-many relations are in the relation.

R(PROJNAME, EMPNAME, PROJLOC)

PROJNAME	EMPNAME	PROJLOC
Y2K	Mick Reidy	L1
Y2K	Joe Murray	L2
Y2K	Mick Reidy	L2
Y2K	Joe Murray	L1

Multivalued Dependency

On a given relation R, the multivalued dependency (MVD), A $\rightarrow\rightarrow$ B, where A and B are subsets of R follows the rules below for an instance relation r of R : If t_1 and t2 exist in r such that $t_1[A] = t_2[A]$, then another two tuples must also exist in r as t_3 and t4 with the following rules:

- $t_3[A] = t_4[A] = t_1[A] = t_2[A]$
- $t_3[B] = t_1[B]$ and $t_4[B] = t_2[B]$
- $t_3[R-(AB)] = t_2[R-(AB)]$ and $t_4[R-(AB)] = t_1[R-(AB)]$

In the relation R above, PROJNAME $\rightarrow\rightarrow$ EMPNAME and PROJNAME $\rightarrow\rightarrow$ PROJLOC.

A given MVD on R, MVD A $\rightarrow\rightarrow$ B, is a trivial MVD if, B is a subset of A or $A \cup B = R$. Inference rules, reflexivity, augmentation, transitivity, projection, additivity, pseudo-transitivity, complementation and replication exist for MVD.

Reflexivity: A $\rightarrow\rightarrow$ B
Augmentation: If A $\rightarrow\rightarrow$ B, then AC $\rightarrow\rightarrow$ B
Transitivity: If A $\rightarrow\rightarrow$ B and B $\rightarrow\rightarrow$ C, then A $\rightarrow\rightarrow$ C – B
Projection: If A $\rightarrow\rightarrow$ B and A $\rightarrow\rightarrow$ C, then A $\rightarrow\rightarrow$ $B \cap C$, A $\rightarrow\rightarrow$ B – C, A $\rightarrow\rightarrow$ C – B
Additivity: If A $\rightarrow\rightarrow$ B and A $\rightarrow\rightarrow$ C, then A $\rightarrow\rightarrow$ BC
Pseudo-transitivity: If A $\rightarrow\rightarrow$ B and BC $\rightarrow\rightarrow$ D, then AC $\rightarrow\rightarrow$ D –BC
Complementation: If A $\rightarrow\rightarrow$ B and C = R – (AB), then A $\rightarrow\rightarrow$ C
Replication: If A $\rightarrow$ B, then A $\rightarrow\rightarrow$ B

The relation R above is not in fourth normal form. because it only consists of two unrelated one-to-many relationships. a given relation R is in fourth normal form with given F, set of dependencies, if for every nontrivial MVD A $\rightarrow\rightarrow$ B in F^+ A is a superkey in R. In the given relation R above, PROJNAME $\rightarrow\rightarrow$ EMPNAME and the PROJNAME $\rightarrow\rightarrow$ PROJLOC does not imply PROJNAME to be superkey of R. When the R above is decomposed into R1(PROJNAME, EMPNAME) and R2 (PROJNAME, PROJLOC) , then both R1 and R2 are in fourth normal form.

Given below the relation PROJECT(PROJNAME, EMPNAME. PROJLOC), identical to R above, where we have introduced another project name EURO along with three employee dependents (Helen Sherlock, Mick Reidy, Joe Murray, Bob Hamilton,) and four location dependents, L3, L4, L5, L6 we end-up with the following relation with 20 tuples.

PROJNAME	EMPNAME	PROJLOC
Y2K	Mick Reidy	L1
Y2K	Joe Murray	L2
Y2K	Mick Reidy	L2
Y2K	Joe Murray	L1
EURO	Helen Sherlock	L3
EURO	Mick Reidy	L3
EURO	Joe Murray	L3
EURO	Bob Hamilton	L3
EURO	Helen Sherlock	L4
EURO	Mick Reidy	L4
EURO	Joe Murray	L4
EURO	Bob Hamilton	L4
EURO	Helen Sherlock	L5
EURO	Mick Reidy	L5
EURO	Joe Murray	L5
EURO	Bob Hamilton	L5
EURO	Helen Sherlock	L6
EURO	Mick Reidy	L6
EURO	Joe Murray	L6
EURO	Bob Hamilton	L6

When decomposing PROJECT into PROJ_EMP and PROJ_LOC then we obtain the following tuples in both relations for PROJ_EMP(PROJNAME, EMPNAME) and PROJ_LOC (PROJNAME, PROJLOC) :

PROJNAME	EMPNAME
Y2K	Mick Reidy
Y2K	Joe Murray
EURO	Helen Sherlock
EURO	Mick Reidy
EURO	Joe Murray
EURO	Bob Hamilton

PROJNAME	PROJLOC
Y2K	L1
Y2K	L2
EURO	L3
EURO	L4
EURO	L5
EURO	L6

The total number of tuples is 12 with update, delete and insertion anomalies removed , which would exist in PROJECT relation.

In this chapter we have studied the various schema refinement and decomposition methods, along with the functional dependency theory and normal forms that are essential features to refine a schema. Even thought a database design mostly starts with ER modeling, it is not always possible to end-up with a non-redundant set of relations which don't have any insert, update or delete anomalies, translated directly from the ER model. ER modeling will be studied in the next chapter. We have not included the relational calculus in this chapter. We are also not presenting the details of the SQL but just give an overview of it. The theory we have studied in this chapter, specially normalization, will be a base for our database model and design we will be presenting in the following chapter.

3.5.6 Structured Query Language (SQL)

SQL was the first language used in the IBM's System R relational DBMS product. Over the past years since it was first used, SQL has become a popular relational database language. It is first standardized in 1986 by the American National Standards Institute (ANSI). Since then, it has been formally adopted as an International Standard by the International Organization For Standardization (ISO) and the International Electrotechnical Commission (IEC). It has also been adopted as a Federal Information Processing Standard (FIPS) for the U.S. federal government.[64]

The first SQL Standard developed by ANSI was in 1986. This was called as SQL-86. A minor revision came in 1989, and was called SQL-89. Next a major revision was introduced in 1992, and called SQL-92. This was a 580 page specification. ISO collaborated with ANSI to develop SQL-92. SQL-92 significantly increases the size of the original 1986 standard to include a schema manipulation language for modifying or altering schemas, schema information tables to make schema definitions accessible to users, new facilities for dynamic creation of SQL statements, and new data types and domains. Other new SQL-92 features include outer join, cascade update and delete referential actions, set algebra on tables, transaction consistency levels, scrolled cursors ,deferred constraint checking, and greatly expanded exception reporting. SQL-92 also removes a number of restrictions in order to make the language more flexible and orthogonal. Most commercial DBMSs currently support SQL-92. SQL-1999, As a major extension to SQL-92 has been approved in 1999. Table 3.4 summarizes the developments in SQL standards.

Table 3.4 Developments in SQL Standards

Version	**Description**
1986: SQL-86	First standard published by ANSI
1989: SQL-89	Minor revision on SQL-86
1992: SQL-92	Major revision. ISO collaborated with ANSI. Mainly Includes: - Schema manipulation language - Schema information tables - Facilities for dynamic creation of SQL statements - New Data types and domains - Outer join - Cascade update / delete
1999: SQL 1999	A Major extension to SQL-92

As mentioned before, our study does not include SQL implementations. We will not implement any programming, such as front-end or SQL. The programming phase would be an exercise for actual development of our model, for which we are designing the database, both conceptual and physical model.

[64] National Institute of Standards and Technology (NIST) SQL Project, URL: http://www.nist.gov/

Chapter 4

DATABASES DESIGN AND ENTITY-RELATIONSHIP MODELING

4.1 Database Development Process

Almost every design process of a database system contains the entity relationship (E/R) modeling. The E/R model is based on perception of the real world as consisting of a collection of basic objects (entities) and relationships among these objects. It is intended primarily for the database design process by allowing specifications of an enterprise scheme. This represents the overall logical structure of the database. The overall logical structure of a database can be expressed graphically by the E/R model using specific diagrams.[65]

Peter P. Chen proposed the E/R model: A data model, called the entity-relationship model, is proposed. This model incorporates some of the important semantic information about the real world. A special diagrammatic technique is introduced as a tool for database design. An example of database design and description using the model and the diagrammatic technique is given. Some implications for data integrity, information retrieval, and data manipulation are discussed. The entity-relationship model can be used as a basis for unification of different views of data: the network model, the relational model, and the entity set model.[66]

In this chapter we will give the various aspects of E/R model and start developing pieces of our database model for man-power planning with the E/R representation.

The general steps in a database development process can be listed as below. For the completeness of system development phases, we have given system analysis and implementation phases as headlines but we will not explore these areas in this chapter.

- Systems Analysis: Some sources also call this phase as requirements analysis. This phase includes initial processes and tasks of information system design. This phase collects facts, requirements, directions from the business environment. These collections involve both; application level and data level facts. Functional structures and existing data maps of the business environment are drawn in this phase. The outcome of this phase will lead to the application development and database design and implementation.

- System Design: Once the system analysis phase is completed, the next step is the identify the functional requirements and database requirements. The first one will be

[65] Han, J., *Database Systems and Structures, Lecture Notes, CPMT354, 1995*, School of Computer Science, Simon Fraser University, URL: http://www.csuohio.edu

[66] Chen P. P., The Entity-Relationship Model - Toward a Unified View of Data, ACM Transactions on Database Systems, Volume 1, Number 1, p.9, March 1976

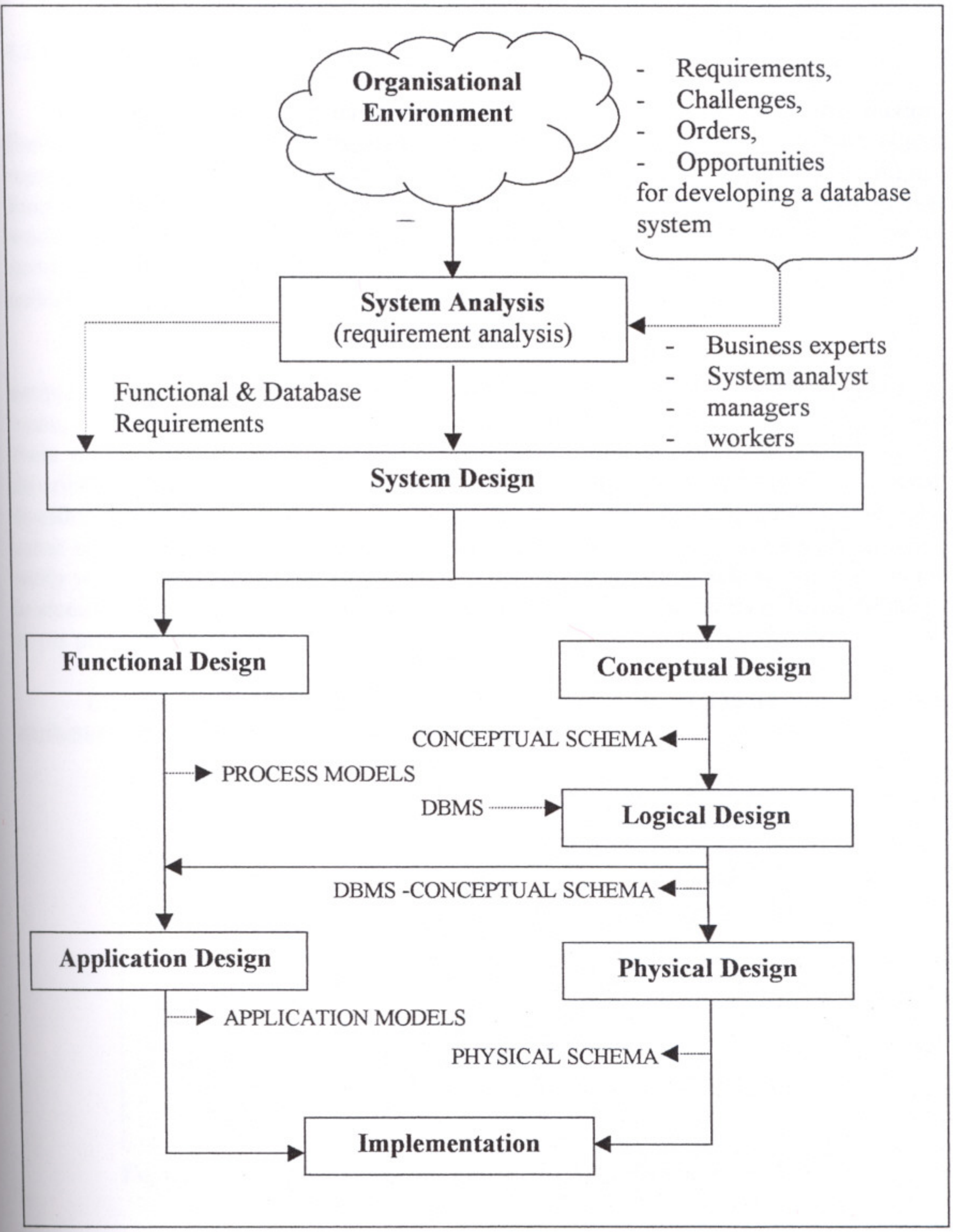


Figure 4.1 High-Level Phases of Database and Application Development.

The output of the conceptual design is the E/R Model (conceptual schema). for the rest of the chapter, we will be working in detail with the E/R model. We have seen the Logical design concepts in Chapter 2 where we have introduced the functional dependency and normalization in schema refinement. Various design tools can be used during the phases in the figure above. Major DBMS companies and vendors have produced analysis, design and development tools to assist the database designer. We will use the Sybase tool, Power Designer in our implementation of the relational model.

4.2 Entity-Relationship Model

4.2.1 Entity

E/R model has some basic components we will be studying in this section. Entity is the basic, main component in the E/R model. entity is an diagram object representing a real world object and it can be distinguished from other entities. Employee, title, department, project are some examples to possible entities in real world and in E/R model. Entities may be physical or conceptual objects. A set of entities is the collection of entities in a given database schema, or an entity type is the collection of entities that have the same attributes.

Entities have set of "**attributes**" as properties. The attributes describe the entity. For example, a department entity can be described by Department id, department name, manager. Attributes of an entity have "**values**", which are the stored data for them in the database. The attributes mostly have domains for their values. As we described a domain before, it is a set of possible values, that an attribute can have. domains are described with domain names, the data types and sometimes with valid value ranges. Entities have keys, as primary key, candidate key to identify a single entity in a given entity set. A key is the minimal set of attributes whose value is unique to identify the entity. There may be multiple candidate keys, one of them being defined as the primary key.

Entities in the E/R model are generally shown as in Figure 4.2 with their attributes as ovals attached to the rectangular entity.

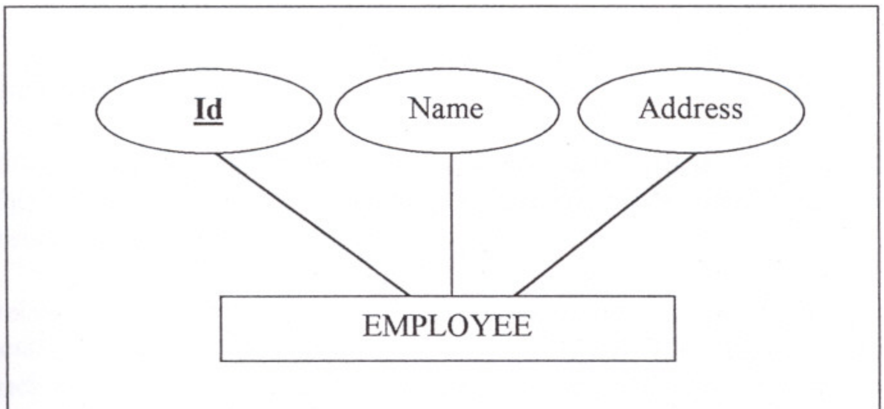


Figure 4.2 An Entity Representation in E/R Model.

In the Figure 4.2, the EMPLOYEE entity is represented with the rectangle with its name. the attributes are shown as ovals with attribute names in the middle. The key attribute is bold and underlined in E/R representation. Attributes of an entity can be in one of the following types:

- **Simple or composite**: A simple attribute cannot be divided into further sub-attributes and so it is atomic. On the other hand a composite attribute is composed of multiple atomic attributes. In Figure 4.2, the ADDRESS attribute can be given as an example to composite attributes because it may be composed of attributes such as street name, number, city. Here, any of the attributes composed to make-up the address attribute are called the simple attributes.

- **Single valued or multivalued** : Some entities can have single (unique) value. In the EMPLOYEE entity, the age would be an example of such an single-valued attribute. But the EMPLOYEE could have multiple phone numbers and therefore the phone attribute would be a multivalued attribute. Multivalued attributes mostly can have a group of values or range of values that the entity can have.

- **Stored or derived** : On the EMPLOYEE entity, if we have the birth date and age attributes together, then the birth date attribute could be a stored (recorded) entity and the age attribute would be the derived entity which is calculated from birth date and current date as : AGE = NOW-BIRTH DATE. The thing to notice here is that, even though the stored value is fixed, the derived value can change by time.

- **Null valued attributes**: One of the important concepts in attributes are the null valued attributes. for a given entity and its attribute, there may be a value not available or not applicable. For example, the apartment number of an employee may not exists since he or she may not be living in an apartment but may be living in a private house. such unknown values are called NULLS. It is important to understand that NULL ≠ "BLANK" or NULL ≠ ZERO. Blank and zero are known values whereas NULL is not known. Primary keys are not allowed to be NULL since this would violate all relational rules and integrity constraints.

4.2.2 Relationship

Entities which interact with each other are linked with relationships. Relationships tie entities. The second element of the E/R model are the relationships or the relationship types.

A relationship type is defined as, R, among n entity types E_1, E_2,...E_n, defines a set of associations among entities from these types. R is the set of relationship instances r_i, where each r_i associates n entities (e_1, e_2,...e_n), and each entity e_j in r_i is member of entity type E_j ($1 \leq j \leq n$).[67]

Relationships in E/R model are shown as diamond along with the attributes they hold from the entities they relate. A simple figure describing entities and relationship is given in Figure 4.3. Here, the entities are EMPLOYEE and TITLE and the relationship is EMPLOYEE_TITLE. The StartDate and EndDate attributes associated with the EMPLOYEE_TITLE relationship are called the descriptive attributes of the relationship. We show the properties (attributes) of the relationship above the diamond and the key properties are shown as boldface and underlined. If the key properties are

[67] Elmasri R., Navathe S., *Fundamentals of Database Systems,* 2nd Edition, The Benjamin / Cummins Publishing Company, p.49, 1994

associative, i.e. they are coming from the entity attribute set, then these are handled like foreign keys.

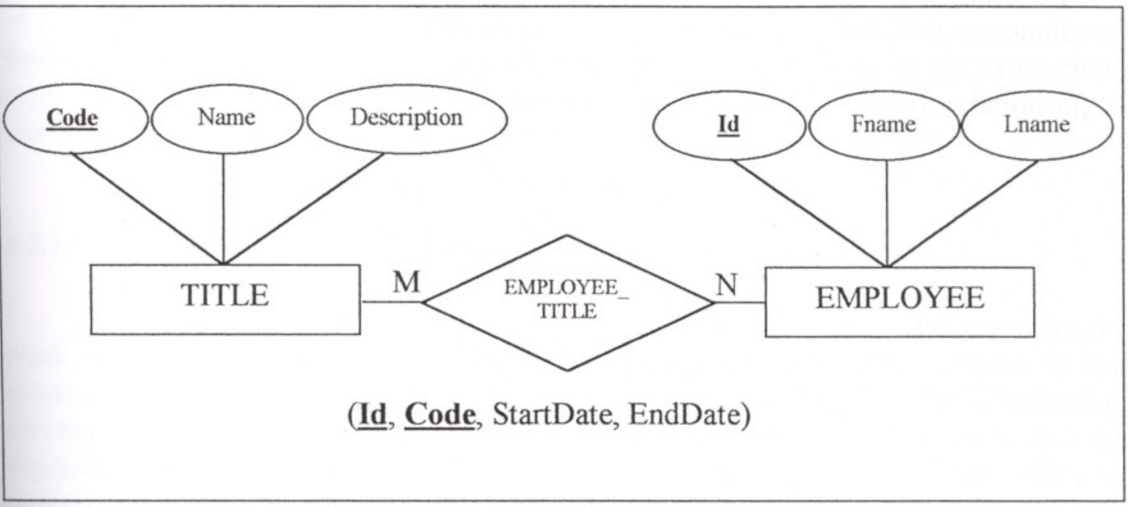


Figure 4.3 Entities and Relationships.

A relationships degree is the number of entity types in the relationship. The degree of EMPLOYEE_TITLE relationship is two which is called a binary relationship. A relationship of degree three is called a ternary relationship.

In the EMPLOYEE_TITLE, if a title can be held by many employees and an employee can have multiple titles than we have a many-to-many relationship. If a title can only be held by one employee and an employee can hold many titles then the relationship would be many-to-one from title to employee. If an employee can only have one title and a title can be held by many employees, the relationship would be one-to-many from title to employee. In the case of one employee can only have one title and a title can only be held by one employee, then we would have a one-to-one relationship. These constraints described are called the **mapping constraints**. We display the mapping constraints on the both side of the relationship diamond. In our case we have set a many-to-many relationship. Mapping constraints are also known as **cardinality constraints.**

If the existence of an entity depends on the existence of another entity through a relationship, then there exists an **participation constraint.** Participation constraint can be a total or partial. A given database requirement stating that every employee must be working in a department, would imply the total participation constraint on the relationship between employee and department. Every employee must be assigned to a department via the relationship. Database requirement stating that every department must have a manager (from the employee domain) implies that some employees will be managing departments but not all of them. Some employees will be related to the department via a relationship but not all of them. This constraint is the partial participation constraint. In E/R diagrams some sources show the total participation with double line (=) and partial participation with single line (-).

Entities play **roles** in relationships. This is identified by their **role name**. For example in the Figure 4.3, the TITLE entity plays the role of title. In some other cases, an entity in an relationship plays more than one roles and has different role names to distinguish each the participation and the role played by this participation. This kind of relationships, where an entity plays different roles are called the recursive relationships. in Figure 4.4, a recursive relationship is show. Here the employee entity plays the role of employee and also plays the role of supervisor, which is in fact also an employee. **Recursive relationships** occur where an entity type is related back to the same entity type, i.e. relations like "married", "reports-to", "manages". A recursive relationship is also called an unary relationship.

4.2.3 Weak Entity (or Weak Relation)

Entities, which don't have any key attribute for themselves are know as the **weak entity**. These entities are owned or identified by some other entities in the schema. Another way of defining the weak entities is to identify the relationship between an entity and its owner as a hierarchical relationship, which implies that the dependent cannot exists without the root it depends. In this way, the relationship is defined as weak and not the entity and the weakness of the relationship is shown with double lined diamond.[68] The weak relationship must obey the following rules :

- Leaf entity (weak entity) must have a total participation in the relationship. In other words, there cannot be a child entity without any parent entity.
- The mapping constraint must be one-to-or one-to-many from parent entity to child (dependent) entity.

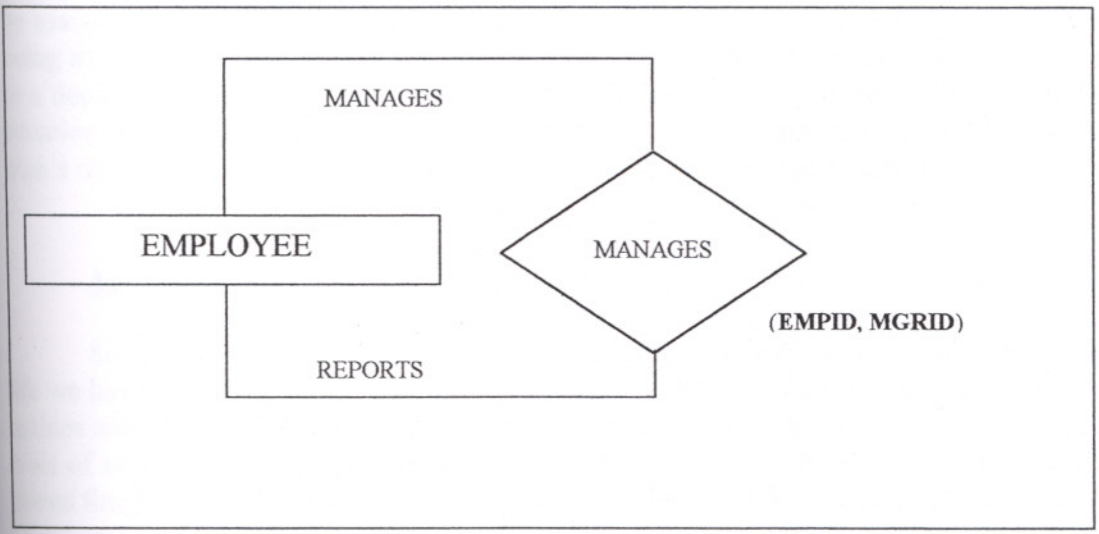


Figure 4.4 Recursive Relationships.

Child entities in weak relations can have a partial key which identifies uniquely the child entities dependent on a parent entity with one-to-many constraints. Any unique attribute of the child entity can be the partial key. In the physical implementation of the child entity, the key of the parent is migrated to it forming along with the partial key the identifier of the child. A weak relationship is shown in Figure 4.5, where the entity EMPLOYEE_HISTORY is dependent to the entity EMPLOYEE. The partial key of EMPLOYEE_HISTORY is shown as dotted-underlined.

[68] Ozkarahan Esen, *Database Management. Concepts, Design and Practice*, Prentice-Hall International Editions, p.212, 1990

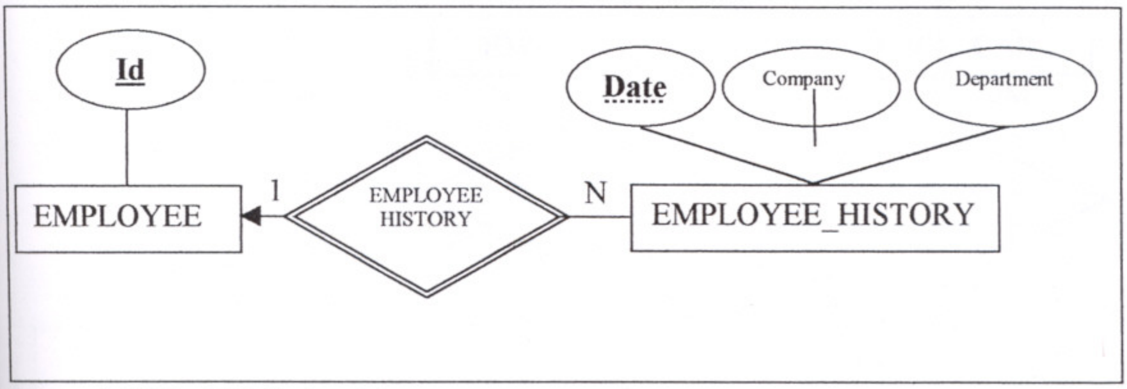


Figure 4.5 Weak Relationship.

4.2.4 Extended E/R Model [69]

In addition to the E/R model, certain extension are implemented, to conceptualize total mapping constraints, aggregation and generalizations in the E/R model. Extended E/R model is an additional abstraction to the existing E/R model.

Total Mapping Constraint

Consider the case, where departments have been allocated pool cars, which can be assigned to multiple departments. What is more, a pool car cannot exists without being assigned to a department. In other words, every POOL CAR instance is assigned to a department and not every department has to have a pool car assigned. This is the situation where we have total mapping constraint. Total mapping constraint is shown with a dot (.) on the side of the constrained entity, in this case the POOL CAR entity.

Aggregation

So far we have defined a relation to be an association between entities. In real life we have scenarios which required us to model relationships between a collection of entities and relationships. The feature enabling this is called the aggregation, indicating a set of relationships is participating to another relationship. Aggregation is shown in dotted line around the set of relationships and entities forming it. Consider the example of departments developing products and employees managing them given in Figure 4.6 where the DEVELOP relationship contains the DEPARTMENT and PRODUCT relationship. Aggregation enables us to use the DEVELOP relationship in the MANAGES relationship. This handles the problem of relationships only existing between entities. In the same figure we have also shown the total mapping constraint.

Generalization

[69] Ibid, p213

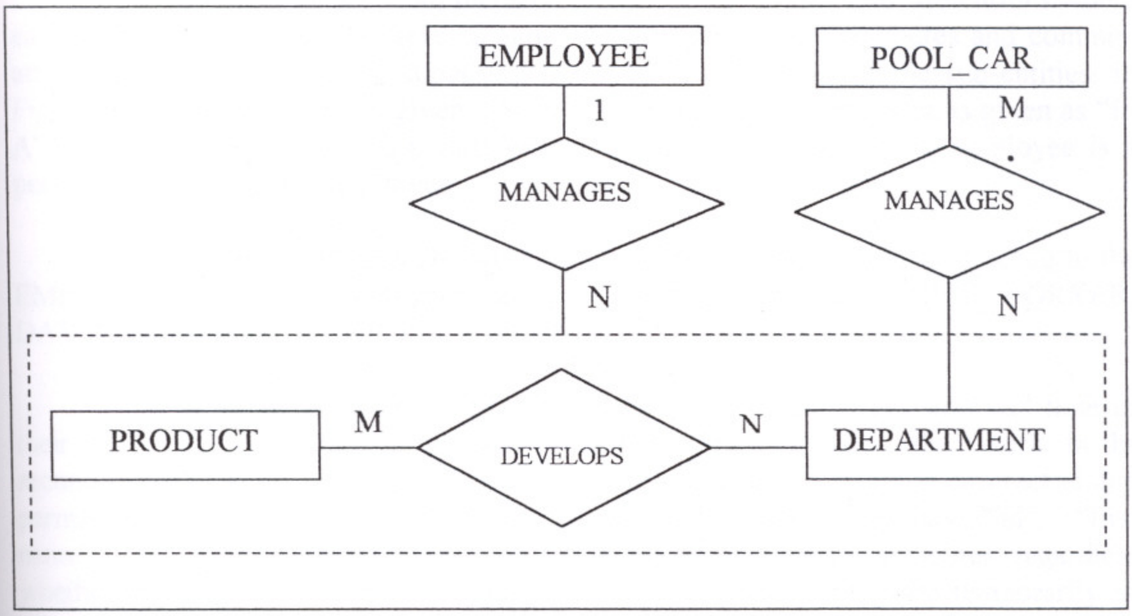


Figure 4.6 Aggregation.

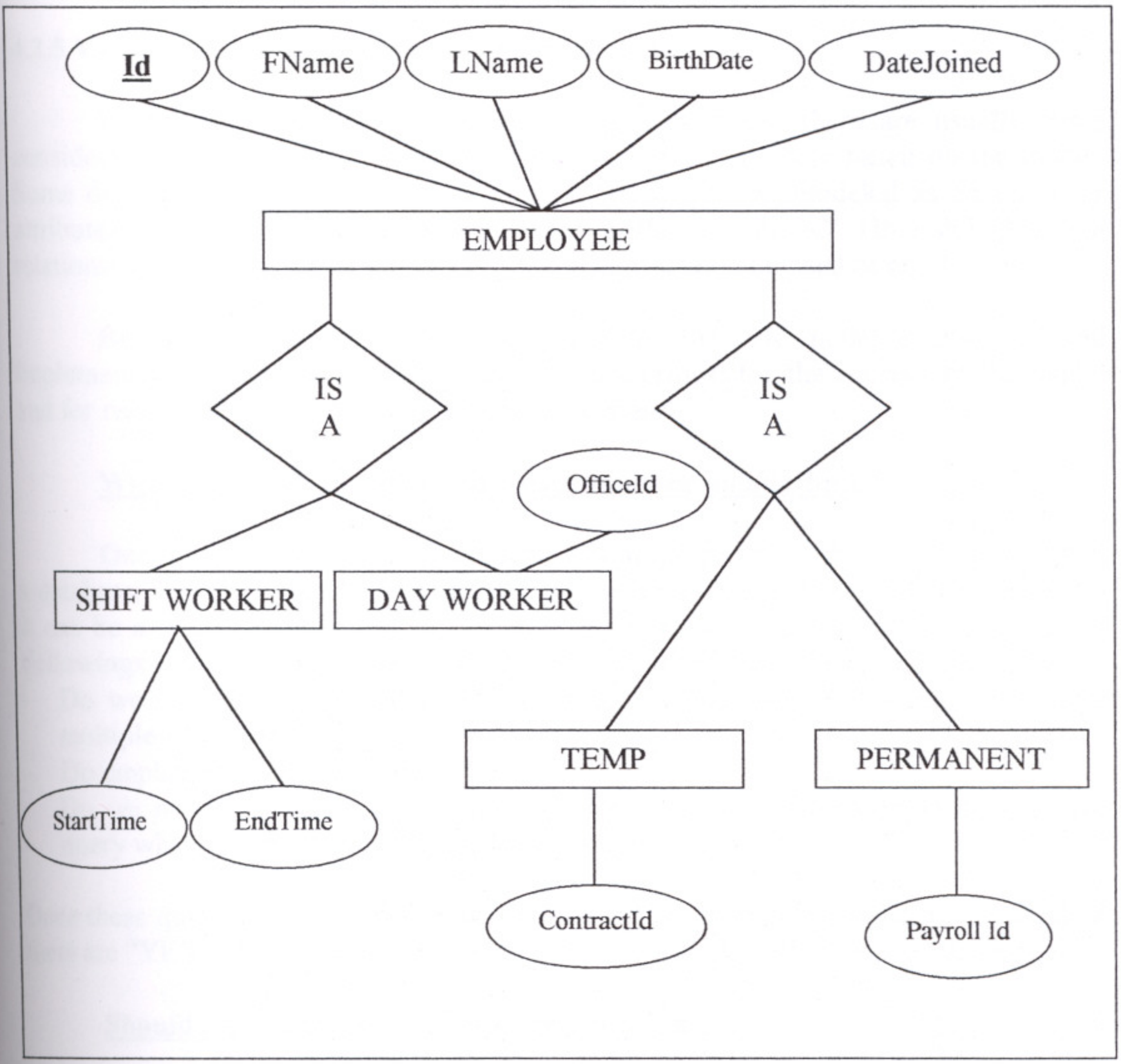


Figure 4.7 Generalization in E/R Model.

Generalization arises from the need to categorize entities into sub-entities where each of the sub-entities will have category specific descriptive attributes and common attributes are listed with the super-entity , which is above all of the sub-entities. in Figure 4.7 a generalization is given. The generalization is in the diagram is given as "IS A" where every employee is a shift worker or day worker and every employee is a permanent or temporary employee.

In Figure 4.7, common attributes, existing for all employees are attached to the EMPLOYEE entity. Employees are then categorized as SHIFT_WORKER, DAY_WORKER and TEMP, PERMANENT.

Shift working employees have been attributed with start time and end time of their shifts. Day workers have been assigned with office id, where they work. In the second "IS A" generalization, temporarily employees have the attribute contract id and permanent employees have the attribute payroll id. All employees have "id", "first name", "last name", "birth date" and "date joined company" attributes regardless whether they are shift working or day working and whether they are temporarily or permanent.

4.2.5 Various Design Consideration in E/R Model

When modeling database systems using E/R model, there are usually some considerations to be taken into account and decision to be made based on the model. Some decision points are whether a concept, (or object) be modeled as an entity or attribute?, Should it be a relation or an entity in the E/R model?, How should be the relationships designed, binary, ternary? Should aggregation be used or not ?

Based on the answers on these questions, the E/R model is designed and implemented. The answers to these questions are critical for the success of the model and for representing the real life cases in the database.

<u>What should be an entity and what should be an attribute ?</u>

One of the best examples to describe this problem is the address of employees in a database. There are two options, address can be an attribute of EMPLOYEE entity or it can be a separate entity and can be in a relationship with the EMPLOYEE entity. Followings are the questions we should be answering before a making a decision

- Do we need to capture many addresses for an employee ? Do employees have multiple addresses ?
- Do employees share the same addresses ?
- Do we need to report on sub-information on addresses. for example do we have query which reports all employees living in a given city ?

Once these questions are answered we will be able to design accordingly and if all of them are "YES" we will need to design address as an entity rather than an attribute.

<u>Should an object be an entity or relationship ?</u>

The best way of tackling this problem is to investigate the constraints given in the requirements and answering the following questions. Does the model fulfill the requirements and does it represent the given scenarios by using an object as an entity or relationship? Are the real life cases represented with entities and attributes or are they instances of relationships?

Given the example below of preliminary design of an E/R model for Employee and department entities, the employee entity has a reference to the department entity through the initial attribute DEPARTMENT. This clearly indicates that there is a need to a relationship among the two preliminary entities EMPLOYEE and DEPARTMENT. Most of the times, the initial design is refined to create more relationships for a database definition.

EMPLOYEE {ID, NAME, SEX, ADDRESS, **DEPARTMENT**}
DEPARTMENT {NUMBER, NAME, LOCATION}

4.2.6 E/R Diagram Symbols and Notations

Different source use different symbols and notations in modeling a database using E/R diagram. The symbols we will be using in this thesis are given in Figure 4.8.

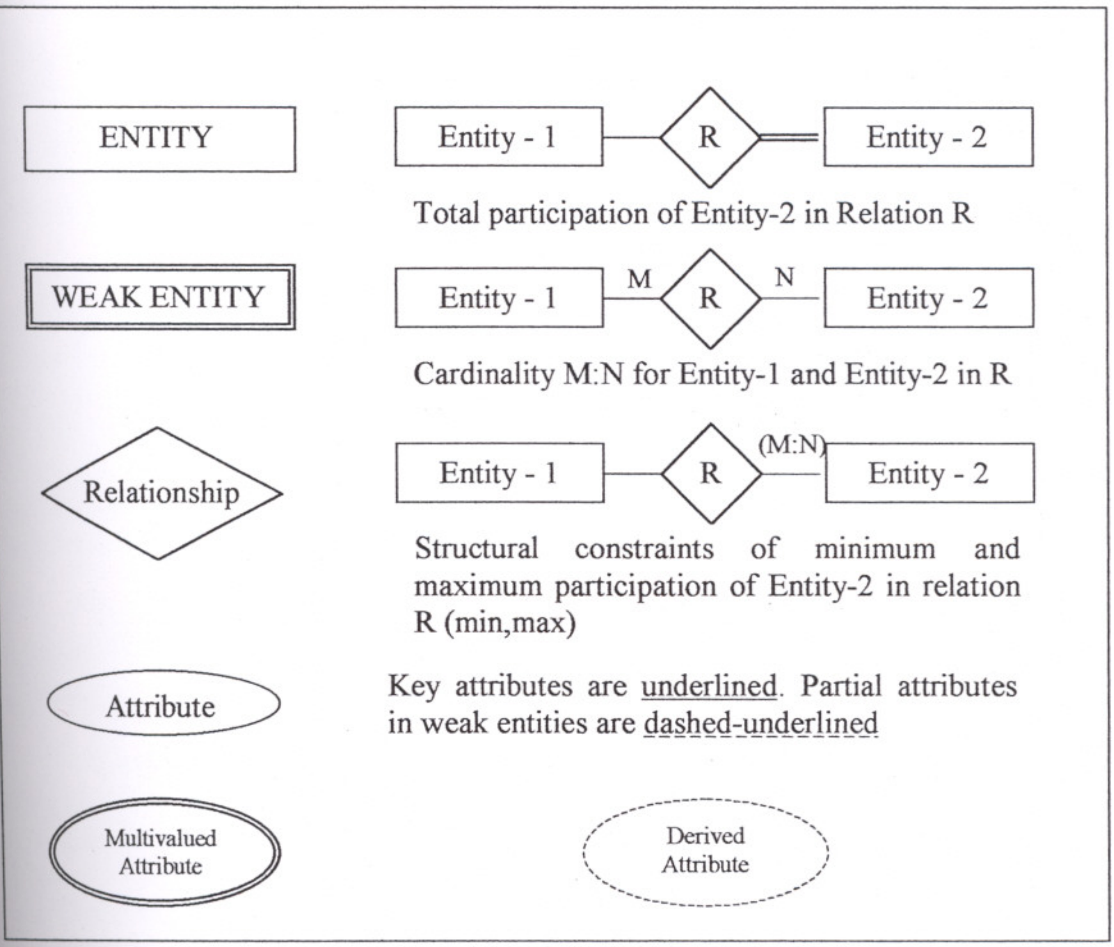

ENTITY
Entity - 1
R
Entity - 2
Total participation of Entity-2 in Relation R
WEAK ENTITY
M
N
Cardinality M:N for Entity-1 and Entity-2 in R
Relationship
(M:N)
Structural constraints of minimum and maximum participation of Entity-2 in relation R (min,max)
Attribute
Key attributes are underlined. Partial attributes in weak entities are dashed-underlined
Multivalued Attribute
Derived Attribute

Figure 4.8. Symbols used in E/R Diagram

In the next Chapter, where the E/R diagram is given, we will simplify the diagram by listing the attributes of the entities seperately and not showing in the E/R diagram.

Chapter 5

A MIS DATABASE MODEL FOR MAN-POWER PLANNING

5.1 Introduction

In this chapter we will give a MIS database model development for a man-power planning database. The development phase will start with the initial description of the system. the next step will be the E/R model of the database followed by the implementation of the conceptual model with Sybase tool Power Designer$^{©}$. The next step will be the creation of the relational database from the conceptual model.. The database model we have developed in this study is based on a theoretical organizational requirement of a database system.

We have made some assumptions during this design work. These are on the requirement analysis (system analysis) and actual implementation. We will not present the details of the requirement analysis and actual database implementation in this thesis. What is more we will not provide the business applications side of this work, i.e. implementation of front-end MIS applications working against our database.

Man-power planning and reporting is an important task in every organization. What is more, many organizational functions shown in Figure 2.7 actually require an efficient man-power planning tool or platform to carryout their tasks. Whether a company is in manufacturing sector or service provider, man-power is an essential part for delivering the products or the services. In our model we will be designing the database based on requirements of an organization in manufacturing area.

There are various problems and challenges in organizations to be tackled to carry out the organizational functions. These problems are in one way or the other also related to the man-power planning problem. As being the most valuable resource of operation, man-power has to be planned efficiently. There are multiple packages and solution providers in the market place today on enterprise resource planning systems, crew scheduling systems, human resources management packages, time attendance systems etc. Some of the major developers are SAP, PeopleSoft, Oracle, JDEdward, IBM and Microsoft.

Our study in this thesis is an attempt to understand information and database systems and try to develop a database model for an experimental environment for man-power planning. Following areas are considered in developing the database model.

Production and production planning: This includes to meet the production targets within the given constraints and quality metrics. Production requires man and machine as resources operate. Production and production planning are based on the short term requirements and plans. The output of production planning is the short term production schedules.

- Physical implementation issues such as security, views, DBMS selection, front-end user application design are not in the scope of this work. But we will be giving a sample physical database creation (scripts) for Sybase SQL Server Anywhere 5.5 ©

- Initial refinements are implemented on the E/R model. We are not giving every step of refinement during the E/R modeling.

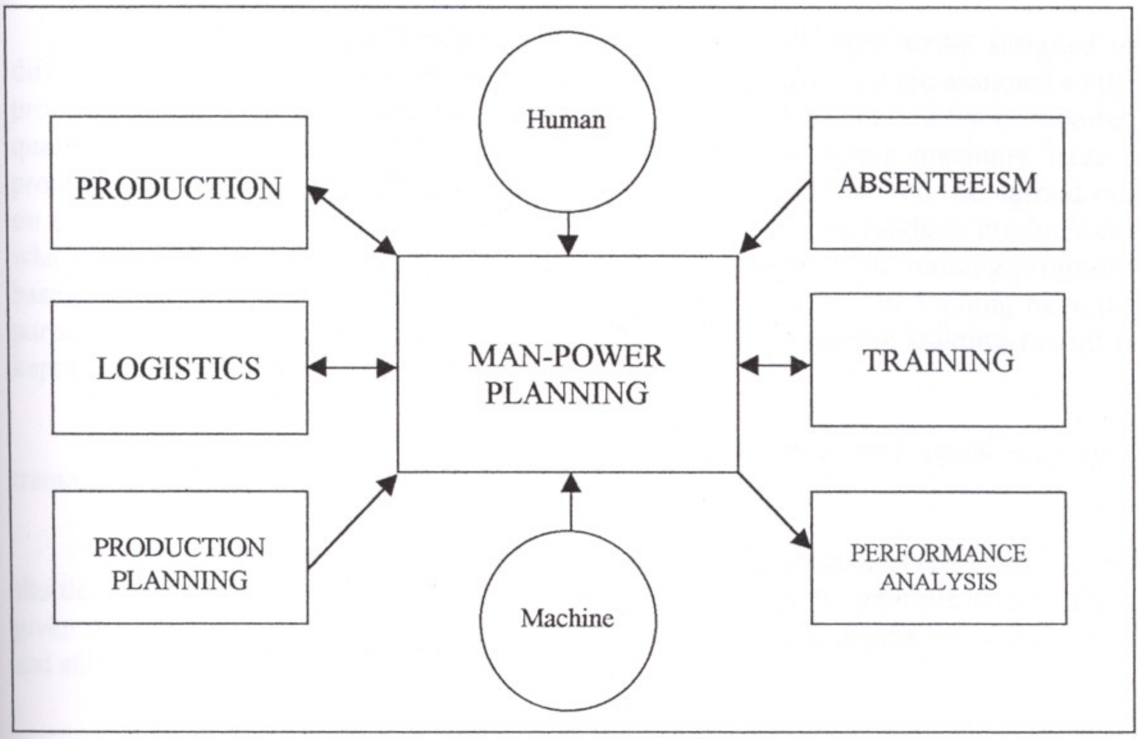


Figure 5.1 Relationships Between Functions and Man-Power Planning.

5.3 E/R Model Design

5.3.1 Database Requirements Summary

Database Name : MAN-POWER PLANNING DATABASE.

The Man-Power Planning Database is designed to accommodate data for an organization which manufactures a specific type of product.

The company is organized as various departments where a department can have multiple sub-departments. Within a department there are different work-teams. Departments own different production lines, composed of production machines where each production machine is in fact composed of different sub-machines making up the production line. Machines are located in different areas and locations. Each area has different locations assigned to it. The company operation is based on an overall organizational calendar with a planned version and actual version.

The company operates in daily shift basis. There are different shifts in a day starting and ending at certain times in a time sequential way. There is also a regular day shift operation in the company.

Crews are composed of employees. Crews are assigned to the different working shifts. A crew has a planned and actual calendar. Employees in a crew can come from different department. An employee can only be in one crew at a time.

Employees have qualifications and job titles enabling them to be assigned to different production machines. Workers with different qualifications are assigned to the production machines for each working shift. Production machines have a list of required qualifications. Production is carried out on a calendar basis where machines have a production calendar for operation. Different activities of operation can be carried out on a given machine on different dates and shifts. Machines also produce products for which data is stored in a daily basis for each shift. Employees attend training programs based on a training schedule and on given classrooms and dates. The training have the purpose to give more qualifications to the employees. An employee training record is kept for history of attended training.

Employee activities are planned by employee calendar and actual employee transactions are kept in actual employee calendar.

Employee transportation is carried out with shuttle system where stations and shuttles are assigned for each calendar date. A shuttle services multiple station for a given date and shift. A station can also be serviced by multiple shuttle on a given date and shift.

5.3.2 Preliminary Entity Design

{} Indicates relational information. ( ) Indicates composed attributes. Key attributes are underlined and bold face.

Below, the initial entity designs are given. These are not in any form or shape of refinement. These preliminary designs have to be converted into detailed entities and relationships in the E/R model.

```
ORGANIZATION CALENDAR
        {Working Calendar-Planned}
        {Working Calendar-Actual}

DEPARTMENT
        Code,                 /* Primary key for the DEPARTMENT entity type
        Name,                 /* Short name
        Description,          /* Long Description
        SuperDepartment,      /* Super-Department that this department reports to
        Manager,              /* Manager from EMPLOYEE entity
        {WorkTeam},           /* Work teams that belong to the department
        {ProductionMachines}/* Production machines owned by department
```

{Cost Center}   /* Cost Center areas in a department.

AREA
- **<u>Code</u>**
- Name
- {Location}

EMPLOYEE
- **<u>Code</u>**   /* Primary Key for EMPLOYEE entity type
- (Name(FirstName, LastName, UserName))
- SSN   /* Social Security Number
- (Personal Information(BirthDate, BirthPlace, Mothers Name, Fathers Name, Marital Status, Sex, Blood Type, Title, Nationality))
- Address(Address Line, City, Country, e-mail)
- {Job Title}
- {Qualifications}
- {Cost Center}
- {Education}
- {Company}
- {Department}
- {Working Calendar-Planned}
- {Working Calendar-Actual}
- {Crew}
- (Contact Information(FirstName, LastName, Address, Phone Number))
- {Shuttle Information}
- {Activity}
- {Vacation}

COMPANY
- **<u>Code</u>**
- Name
- Address
- Phone Number
- Fax Number

SHIFT
- **<u>Code</u>**
- Name
- (Time (Start Time, End Time))
- Order

CREW
- **<u>Code</u>**
- Name
- {Working Calendar-Planned}
- {Working Calendar-Actual}
- {Status}

PRODUCTION MACHINE
- {Working Calendar}

{Status}

TRAININGACTIVITY
{Training}
{Status}
{Type}
{Attendees}
{Trainer}
{Classroom}

5.3.3 Detailed Entity Design

Following the high-level requirement description and preliminary entity design, the next step is the detail the E/R model for the entities. In this section every entity of the model is described and attributes are given. The next step will be the E/R diagram given in Section 5.3.4. The key attributes are be underlined and bold-faced.

Table 5.1 summarizes the list of all entities followed by all entities with the attributes. Figure 5.2 (a), (b), (c), (d), (e), (f), (g), (h) in Section 5.3.4 show the E/R diagram of the model in Figure 5.2.

Table 5.1 Entity List

Name	**Description**
ACTIVITY	*Production activities of the Production Machines*
AREA	*Physical area in the company*
BLOODTYPE	*Different Blood Types*
CITY	*City*
CLASSROOM	*Class rooms for training*
COMPANY	*Company*
COSTCENTER	*Cost Center Codes in the company*
COUNTRY	*Country*
CREW	*Crew*
CREWCALENDAR_ACTUAL	*Actual crew operation calendar*
CREWCALENDAR_PLANNED	*Planned crew operation calendar*
CREWSTATUS	*Crew status codes*
DEPARTMENT	*Department*
EMPLOYEE	*Employee*
EMPLOYEEACTIVITY	*Employee activities*
EMPLOYEECALENDAR_ACTUAL	*Actual employee operation calendar*
EMPLOYEECALENDAR_PLANNED	*Planned employee operation calendar*
EMPTRAINING	*Employee Training*
INSTITUTE	*Educational Institutes*
JOBCATEGORY	*Job Categories*
JOBTITLE	*Job Titles*
LEVEL	*Education level for employees*
LOCATION	*Location in the company*

Table 5.1 Continued.

MACHINETYPE	*Production machine types*
ORGANIZATIONCALENDAR_ACTUAL	*Actual company operation calendar*
ORGANIZATIONCALENDAR_PLANNED	*Planned company operation calendar*
ORGANIZATIONSTATUS	*Company Operation status*
PMCALENDAR	*Production machine calendar*
PMSCHEDULE	*Production machine calendar*
PMSTATUS	*Production machine status codes*
PRODUCT	*product*
PRODUCTIONMACHINE	*Production machine*
QUALIFICATION	*Employee Qualification*
SHIFT	*Shift*
SHUTTLE	*Shuttle*
STATION	*Station*
TITLE	*Title (initials)*
TRAININGACTIVITY	*Training events*
TRAININGMASTER	*Training master setup*
TRAININGSTATUS	*Training status codes*
TRAININGTYPE	*Training types*
VACATIONENTITLEMENT	*Employee annual vacation entitlements*
WORKCALENDARTYPE	*Type of calendar (shift non-shift)*
WORKTEAM	*Work teams*

Entities :

ACTIVITY (Code *Name, Description, DefaultAmount*)

AREA (**Code**, *Name*)

BLOODTYPE (**Code**, *Name*)

CITY (**Code**, *Name*)

CLASSROOM (Code, *Name*)

COMPANY (Code, *Name, AddressHQ, PhoneNumber, FaxNumber*)

COSTCENTER (Code, *Name, Description*)

COUNTRY (Code, *Name, Abbreviation*)

CREW (Code, *Name, Description*)

CREWCALENDAR_ACTUAL (*CalendarDate, TotalAmount, Notes*)

CREWCALENDAR_PLANNED (*CalendarDate, TotalAmount, Notes*)

CREWSTATUS (<u>Code</u>, *Name, Description*)

DEPARTMENT (<u>Code</u>, *Name, Description, ManagerSince*)

DEPENDENTS (*FirstName, LastName, Relationship*)

EMPLOYEE (<u>Code</u>, *FirstName, LastName, UserName, Ssn, BankAccountNo, BirthDate, BirthPlace, MothersName, FathersName, MaritalStatus, Sex, AddressLine1, AddressLine2, AddressLine3, PostalCode, HomePhoneNumber, WorkPhoneNumber, WorkExtension, PostLocation, FaxNumber, Email, ContactFirstName, ContactLastname, ContactAddress, ContactPhoneNumber, DateJoined, LeftService, DateLeftService*)

EMPLOYEEACTIVITY (<u>Code</u>, *Abbreviation, Name, Description, ActivityType, WorkActivity, UnitofActivity, DisplayOrder, Display*)

EMPLOYEECALENDAR_ACTUAL (*CalendarDate, TotalAmount, Notes*)

EMPLOYEECALENDAR_PLANNED (*CalendarDate, TotalAmount, Notes*)

EMPLOYEEHISTORY (*RecordDate, Department, EduationLevel, Institute, JobTitle, Company, Notes*)

EMPTRAINING (*Result, Notes*)

INSTITUTION (<u>Code</u>, *Name*)

JOBCATEGORY (<u>Code</u>, *Name, Description*)

JOBTITLE (<u>Code</u>, *Name, Description*)

LEVEL (<u>Code</u>, *Name, Description*)

LOCATION (<u>Code</u>, *Name*)

MACHINETYPE (<u>Code</u>, *Name, Description*)

ORGANIZATIONCALENDAR_ACTUAL (*CalendarDate, TotalAmount, Notes*)

ORGANIZATIONCALENDAR_PLANNED (*CalendarDate, TotalAmount, Notes*)

ORGANIZATIONSTATUS (<u>Code</u>, *Name, Description*)

PMCALENDAR (<u>ProductionDate</u>)

PMDOWNTIMES (*StartTime, EndTime*)

PMSCHEDULE (*PlannedProductionAmount, ActualProductionAmount, Defects)*

PMSTATUS (<u>Code</u>, *Name, Description*)

PRODUCT *(*<u>**Code**</u>, *Name*, Description)

PRODUCTIONMACHINE (<u>**Code**</u>, *Name, Description, Active*)

QUALIFICATION (<u>**Code**</u>, *Name, Description*)

SHIFT (<u>**Code**</u>, *Name, StartTime, EndTime, Sequence*)

SHUTTLE (<u>**Code**</u>, *RegNumber, Driver, Capacity, Remarks*)

STATION (<u>**Code**</u>, *StationName, Address*)

TITLE (<u>**Code**</u>, *Name*)

TRAININGACTIVITY (*DateFrom, DateTo, Capacity, Notes*)

TRAININGMASTER (<u>**Code**</u>, *Name, Description, Notes*)

TRAININGSTATUS (<u>**Code**</u>, *Name*)

TRAININGTYPE (<u>**Code**</u>, *Name, Description*)

VACATIONENTITLEMENT (<u>**EntitlementYear**</u>, *VacationEntitlement, VacationRemaining, CompensationBalance*)

WORKCALENDARTYPE (<u>**Code**</u>, *Name*)

WORKTEAM (<u>**Code**</u>, *Name, Description*)

5.3.4 E/R Diagram

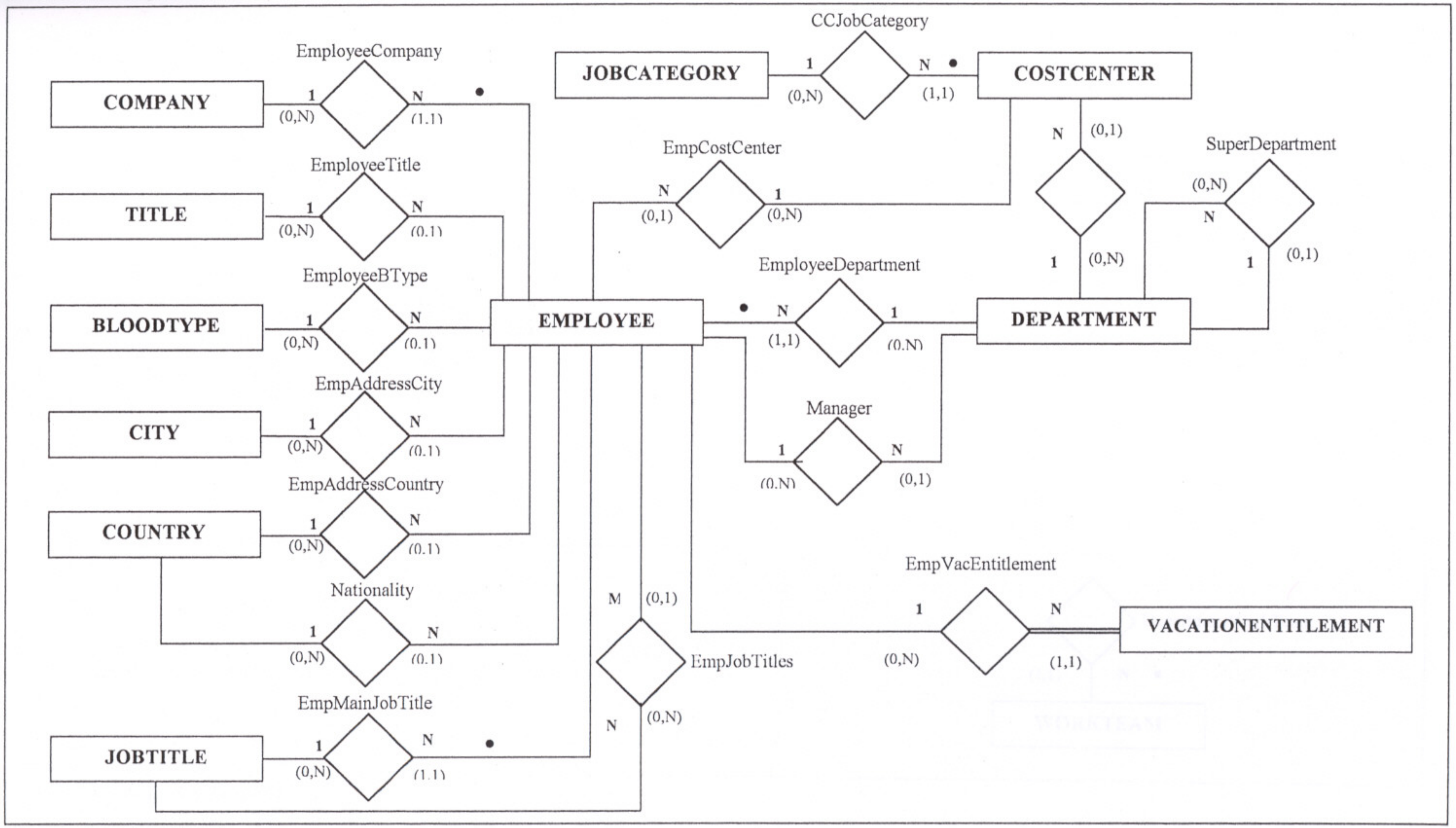


Figure 5.2 (a) E/R Diagram

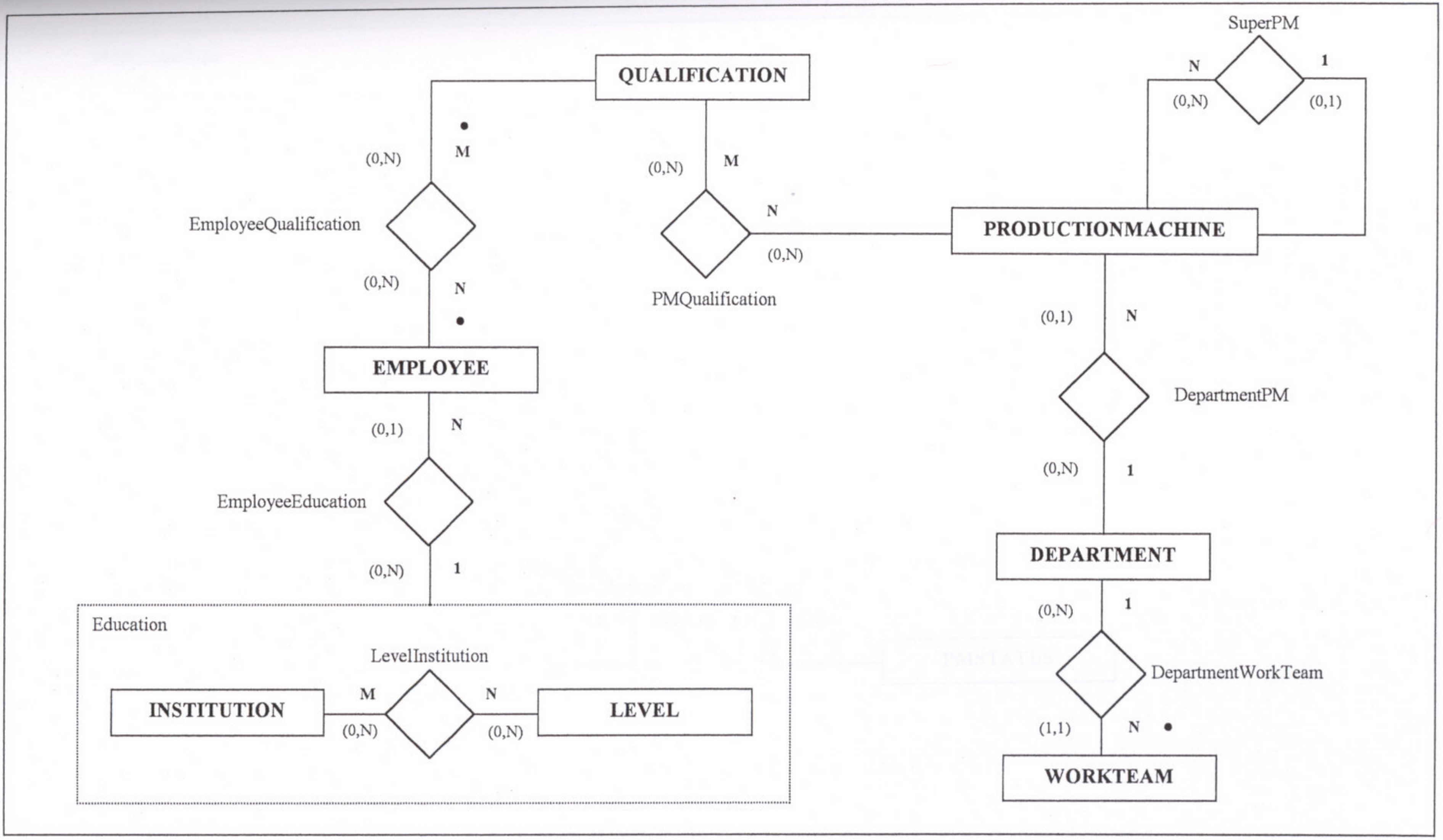


Figure 5.2 (b) E/R Diagram

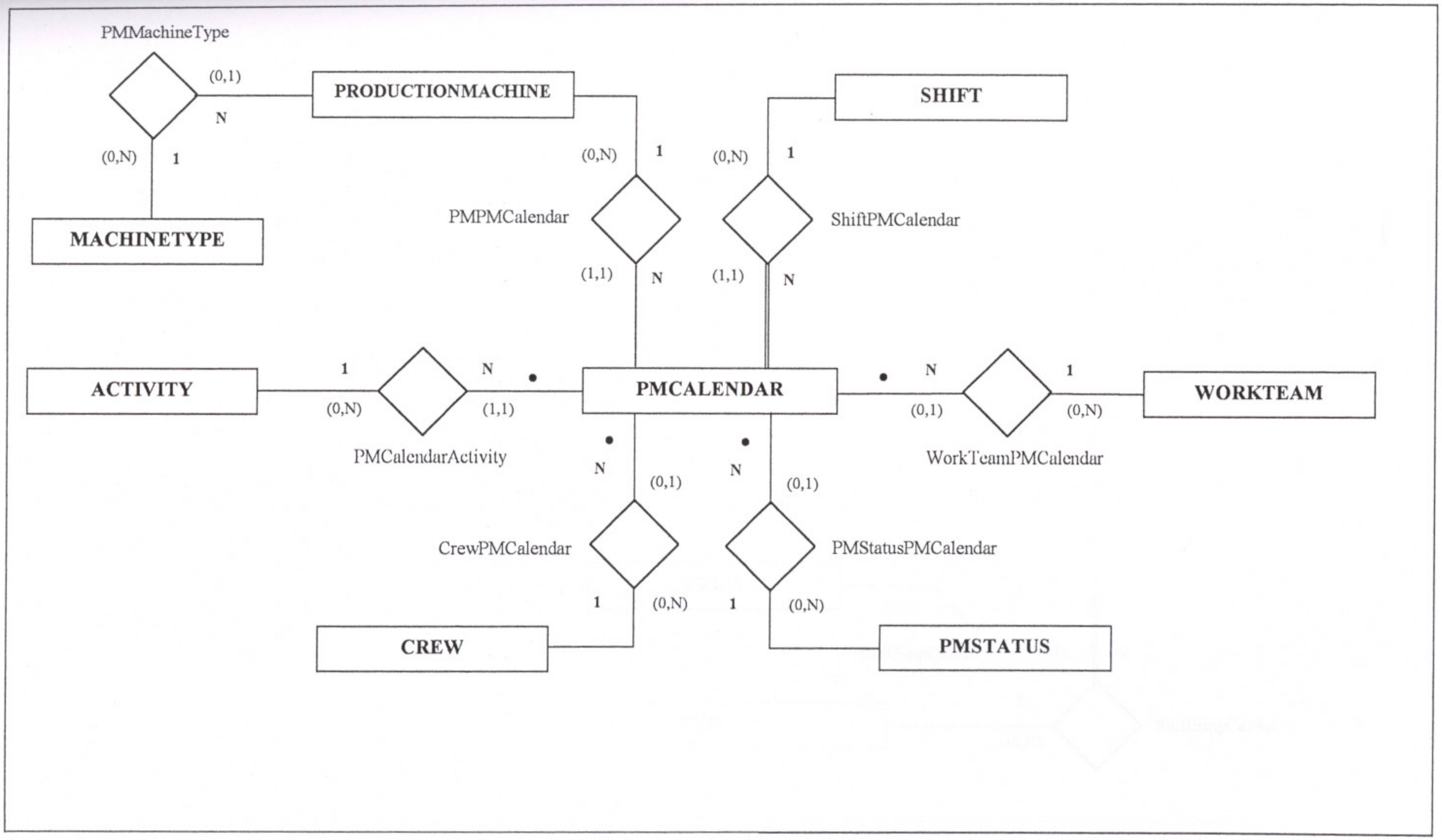


Figure 5.2 (c) E/R Diagram

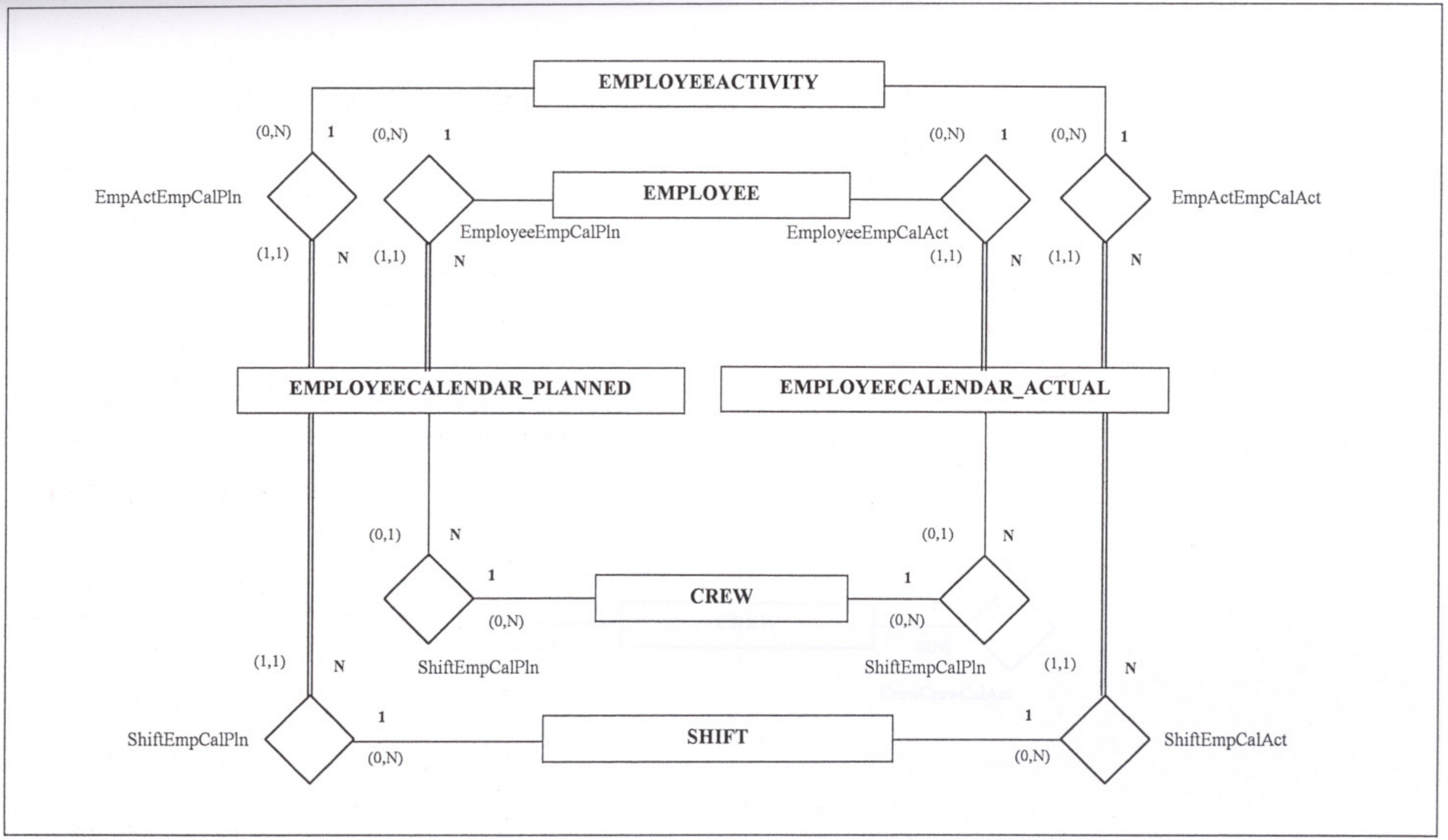


Figure 5.2 (d) E/R Diagram

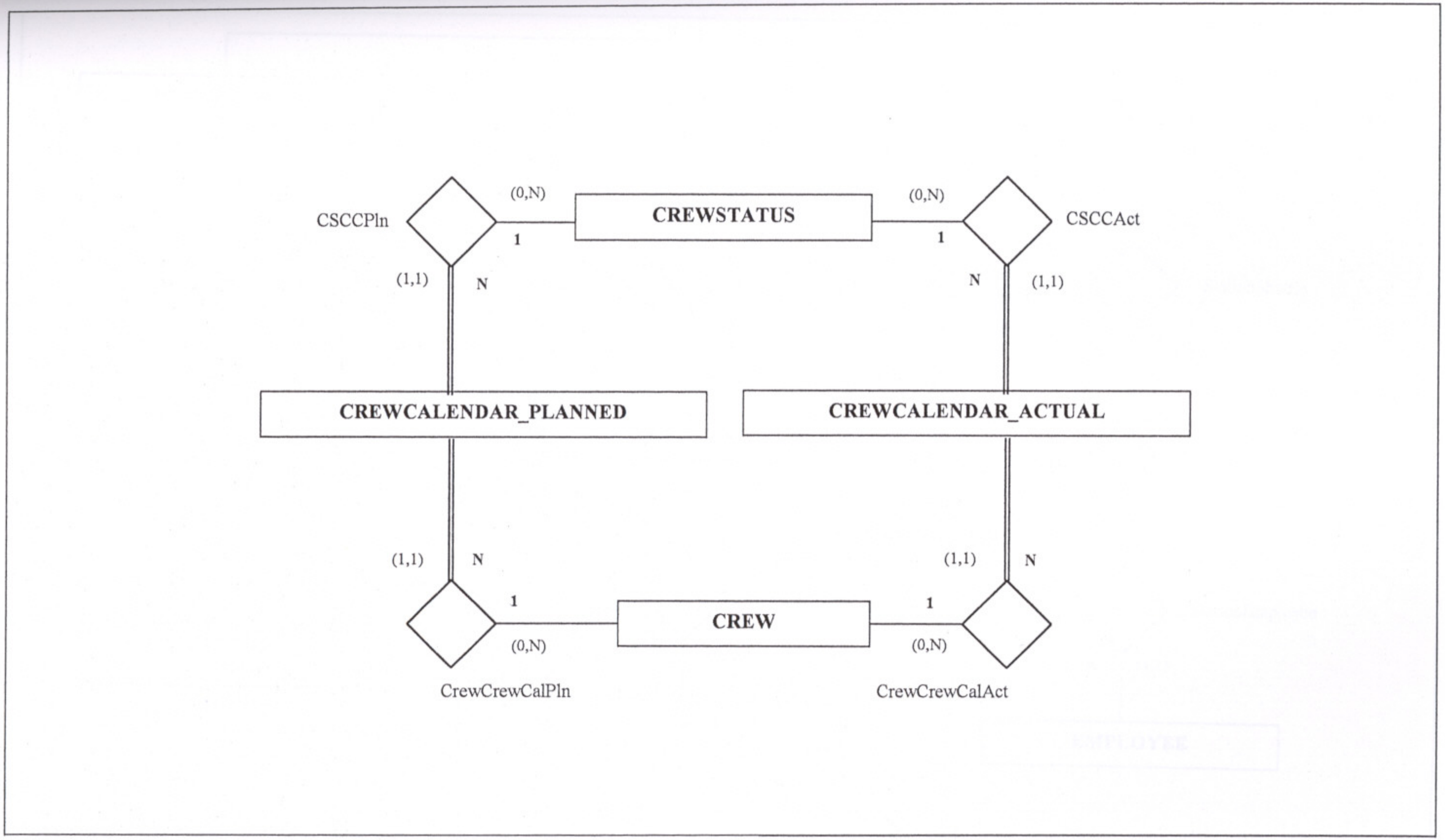


Figure 5.2 (e) E/R Diagram

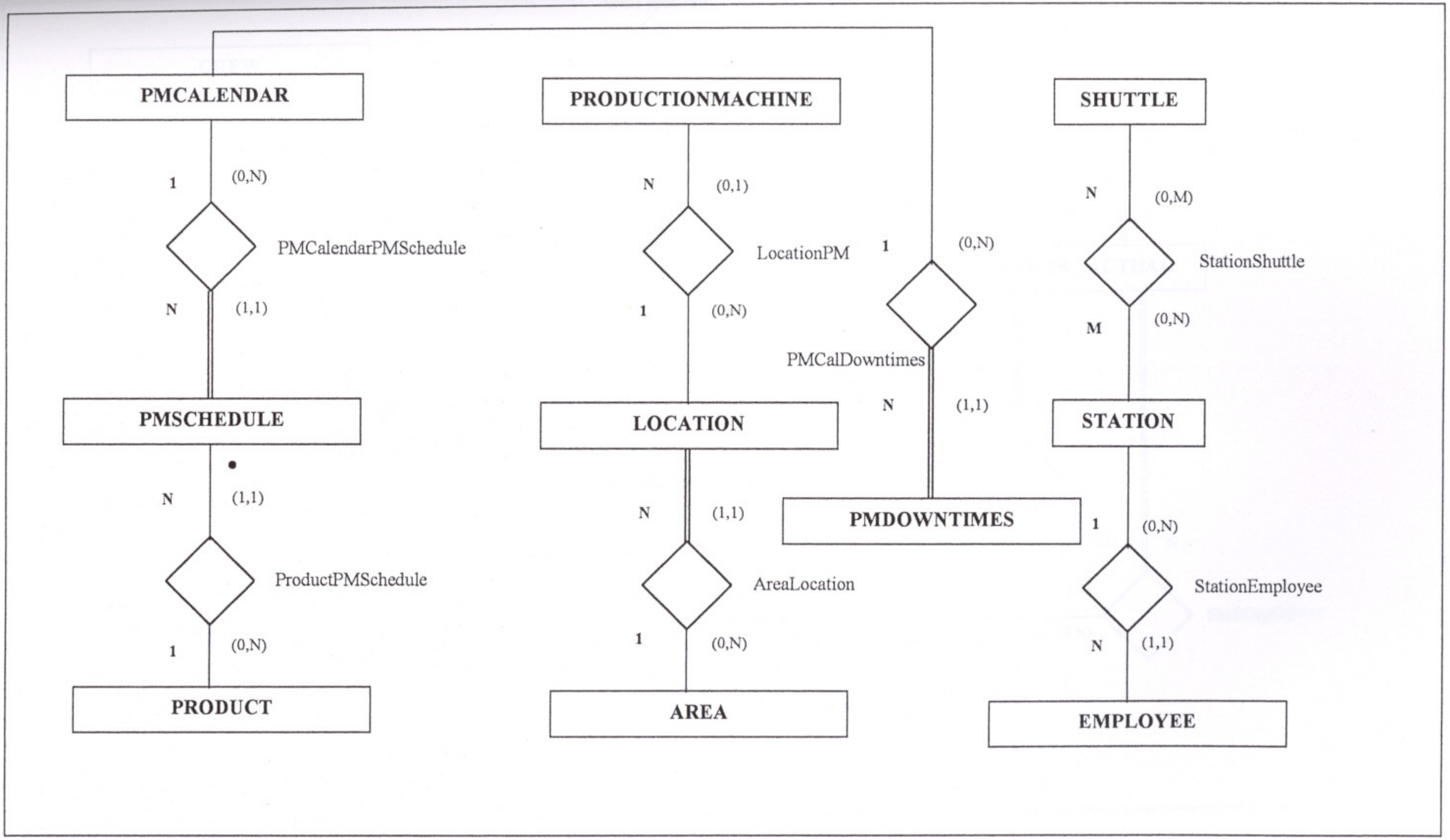


Figure 5.2 (f) E/R Diagram

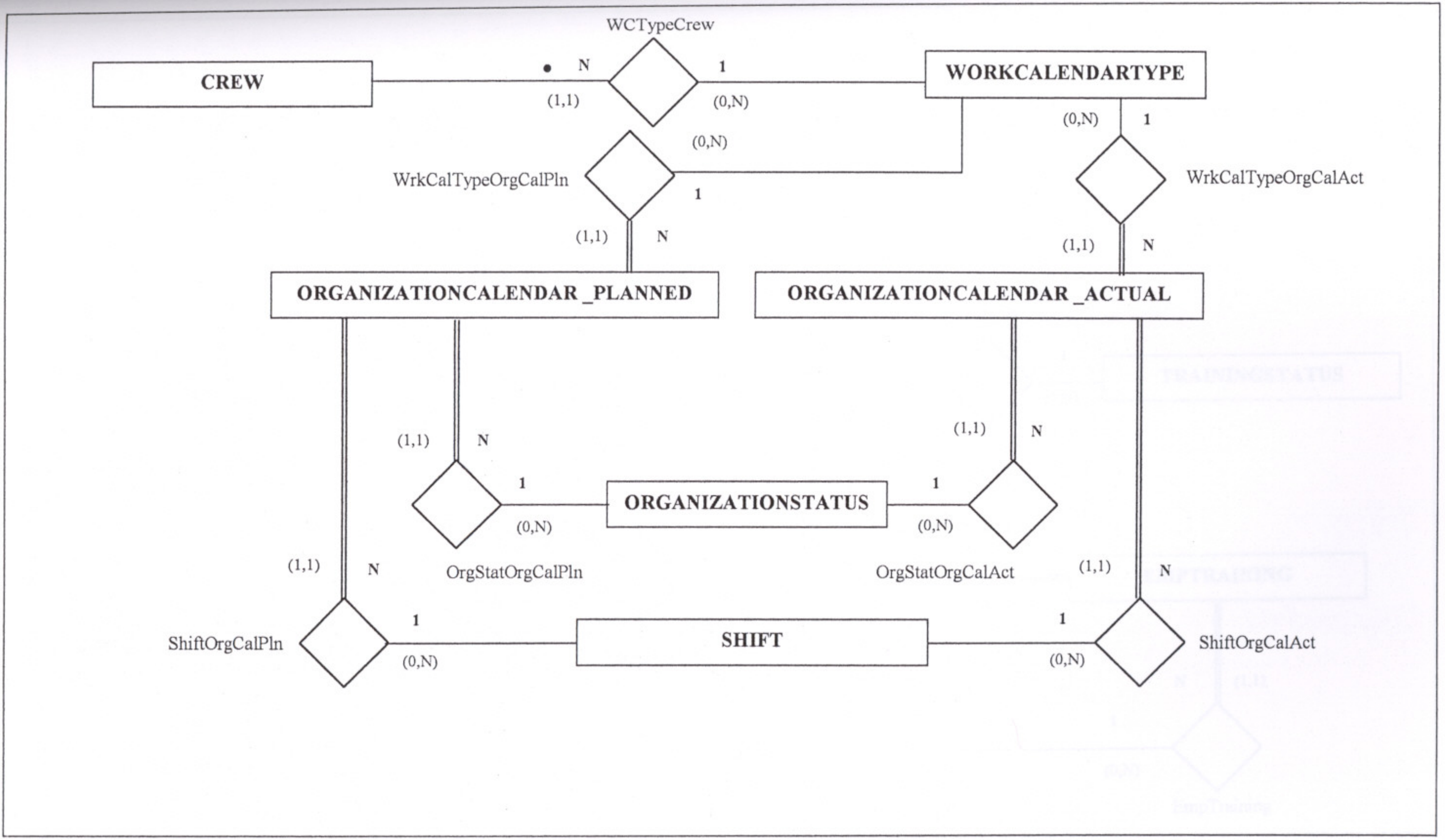


Figure 5.2 (g) E/R Diagram

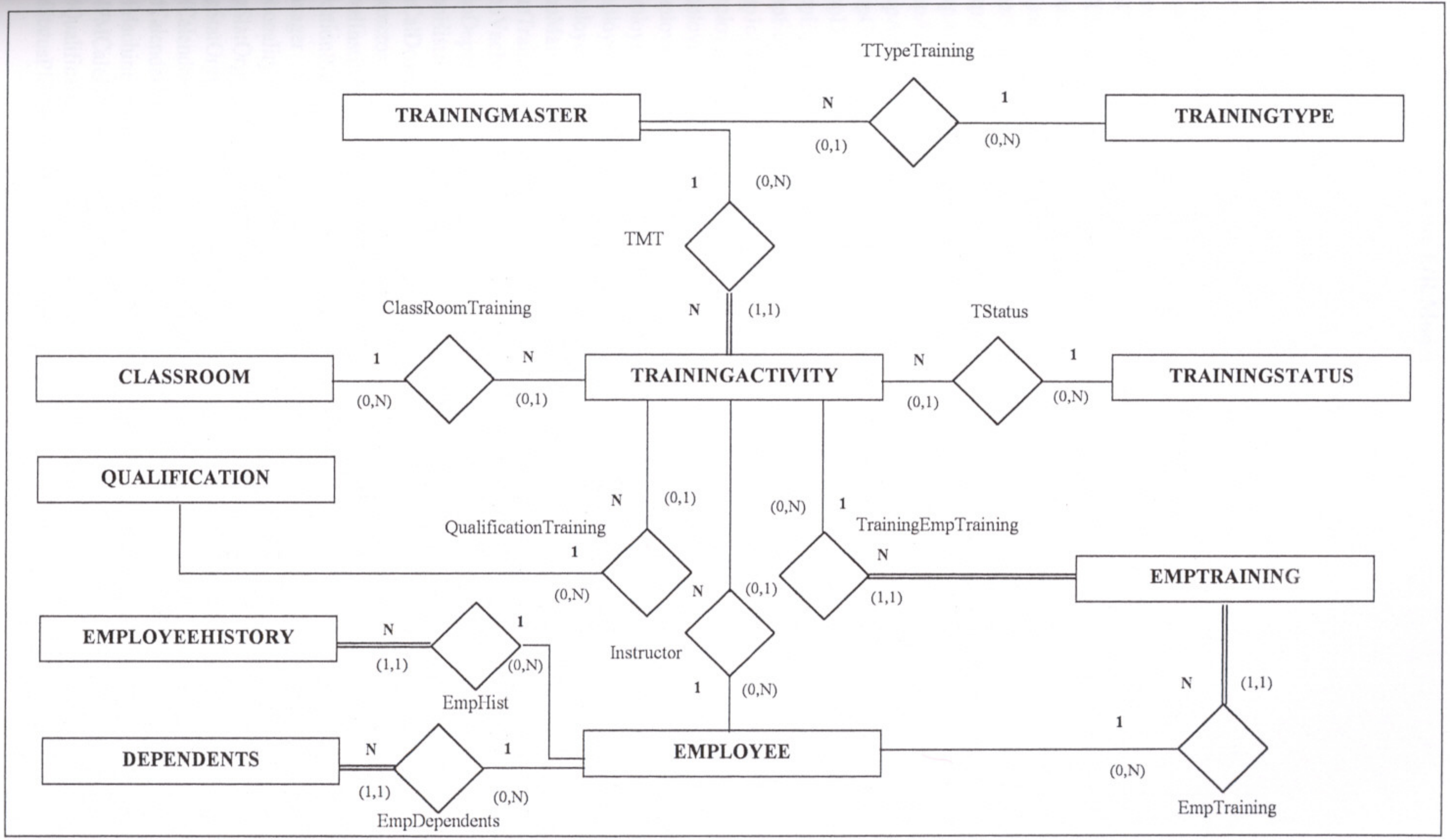


Figure 5.2 (h) E/R Diagram

5.3.5 Relationships in the E/R Model

AreaLocation
CCJobCategory
ClassRoomTraining
CrewCrewCalAct
CrewCrewCalPln
CrewEmpCalAct
CrewEmpCalPln
CrewEmployee
CrewPMCalendar
CSCCAct
CSCCPln
DepartmentPM
DepartmentWorkTeam
DeptCostCenter
EmpActEmpCalAct
EmpActEmpCalPln
EmpAddressCity
EmpAddressCountry
EmpCostCenter
EmpDependents
EmpEmpCalAct
EmpEmpHist
EmpJobTitles
EmployeCompany
EmployeeBType
EmployeeDepartment
EmployeeEmpCalPln
EmployeeQualification
EmployeeTitle
EmpMainJobTitle
EmpTraining
EmpVacEntitlement
EmpDependents
EmpHistory
PmCalDowntimes
Instructor
LevelInstitution
LocationPM
Manager
Nationality
OrgStatOrgCalAct
OrgStatOrgCalPln
PMCalendarActivity
PMCalendarPMSchedule
PMMachineType
PMPMCalendar
PMQualification
PMStatusPMCalendar

ProductPMSchedule
Qualification
ShiftEmpCalAct
ShiftEmpCalPln
ShiftOrgCalAct
ShiftOrgCalPln
ShiftPMCalendar
StationEmployee
StationShuttle
SuperDepartment
SuperPM
TMT
TrainingEmpTraining
TStatus
TTypeTraining
WCTypeCrew
WorkTeamPMCalendar
WrkCalTypeOrgCalAct
WrkCalTypeOrgCalPln

Detailed Description of Relationships

AreaLocation

Name:	AreaLocation
Entity 1:	Area
Entity 2:	Location
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes
Entity 1 → Entity 2: Role: Area Mandatory: No Dominant: No Min, Max: 0, n	
Entity 2 → Entity 1: Role: Location Mandatory: Yes Dominant: No Min, Max: 1, 1	

CCJobCategory

Name:	CCJobCategory
Entity 1:	JobCategory
Entity 2:	CostCenter
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No
Entity 1 → Entity 2: Role: JobCategory Mandatory: No Dominant: No Min, Max: 0, n	
Entity 2 → Entity 1: Role: CostCenter Mandatory: Yes Dominant: No Min, Max: 1, 1	

ClassRoomTraining

Name:	ClassRoomTraining
Entity 1:	ClassRoom
Entity 2:	TrainingActivity
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No
Entity 1 → Entity 2: Role: ClassRoom Mandatory: No Dominant: No Min, Max: 0, n	
Entity 2 → Entity 1: Role: Training Mandatory: No Dominant: No Min, Max: 0, 1	

CrewCrewCalAct

Name:	CrewCrewCalAct
Entity 1:	Crew
Entity 2:	CrewCalendar_Actual
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: Crew Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: CrewCalAct Mandatory: Yes Dominant: No Min, Max: 1, 1

CrewCrewCalPln

Name:	CrewCrewCalPln
Entity 1:	Crew
Entity 2:	CrewCalendar_Planned
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: Crew Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: CrewCalPln Mandatory: Yes Dominant: No Min, Max: 1, 1

CrewEmpCalAct

Name:	CrewEmpCalAct
Entity 1:	Crew
Entity 2:	EmployeeCalendar_Actual
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Crew Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: EmpCalAct Mandatory: No Dominant: No Min, Max: 0, 1

CrewEmpCalPln

Name:	CrewEmpCalPln
Entity 1:	Crew
Entity 2:	EmployeeCalendar_Planned
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Crew Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: EmpCalPln Mandatory: No Dominant: No Min, Max: 0, 1

CrewEmployee

Name:	CrewEmployee
Entity 1:	Crew
Entity 2:	Employee
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Crew Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Employee Mandatory: Yes Dominant: No Min, Max: 1, 1

CrewPMCalendar

Name:	CrewPMCalendar
Entity 1:	Crew
Entity 2:	PMCalendar
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Crew Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: PMCalendar Mandatory: Yes Dominant: No Min, Max 1, 1

CSCCAct

Name:	CSCCAct
Entity 1:	CrewStatus
Entity 2:	CrewCalendar_Actual
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: CS Mandatory: No Dominant: No Min, Max: 0, n

Entity 2 → Entity 1: Role: CCAct Mandatory: Yes Dominant: No Min, Max: 1, 1

CSCCPln

Name:	CSCCPln
Entity 1:	CrewStatus
Entity 2:	CrewCalendar_Planned
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: CS Mandatory: No Dominant: No Min, Max: 0, n

Entity 2 → Entity 1: Role: CCPln Mandatory: Yes Dominant: No Min, Max: 1, 1

DepartmentPM

Name:	DepartmentPM
Entity 1:	Department
Entity 2:	ProductionMachine
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Department Mandatory: No Dominant: No Min, Max: 0, n

Entity 2 → Entity 1: Role: PM Mandatory: No Dominant: No Min, Max: 0, 1

DepartmentWorkTeam

Name:	DepartmentWorkTeam
Entity 1:	Department
Entity 2:	WorkTeam
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Department Mandatory: No Dominant: No Min, Max: 0, n

Entity 2 → Entity 1: Role: Workteam Mandatory: Yes Dominant: No Min, Max: 1, 1

DeptCostCenter

Name:	DeptCostCenter
Entity 1:	Department
Entity 2:	CostCenter
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Department Mandatory: No Dominant: No Min, Max: 0, n

Entity 2 → Entity 1: Role: Costcenter Mandatory: No Dominant: No Min, Max: 0, 1

EmpActEmpCalAct

Name:	EmpActEmpCalAct
Entity 1:	EmployeeActivity
Entity 2:	EmployeeCalendar_Actual
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: EmpAct Mandatory: No Dominant: No Min, Max: 0, n

Entity 2 → Entity 1: Role: EmpCalAct Mandatory: Yes Dominant: No Min, Max: 1, 1

EmpActEmpCalPln

Name:	EmpActEmpCalPln
Entity 1:	EmployeeActivity
Entity 2:	EmployeeCalendar_Planned
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: EmpAct Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: EmpCalPln Mandatory: Yes Dominant: No Min, Max: 1, 1

EmpAddressCity

Name:	EmpAddressCity
Entity 1:	City
Entity 2:	Employee
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Address City Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Employee Mandatory: No Dominant: No Min, Max: 0, 1

EmpAddressCountry

Name:	EmpAddressCountry
Entity 1:	Country
Entity 2:	Employee
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Address Country Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Employee Mandatory: No Dominant: No Min, Max: 0, 1

EmpCostCenter

Name:	EmpCostCenter
Entity 1:	CostCenter
Entity 2:	Employee
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: CostCenter Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Employee Mandatory: No Dominant: No Min, Max: 0, 1

EmpEmpCalAct

Name:	EmpEmpCalAct
Entity 1:	Employee
Entity 2:	EmployeeCalendar_Actual
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: Emp Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: EmpCalAct Mandatory: Yes Dominant: No Min, Max: 1, 1

EmpEmpHist

Name:	EmpEmpHist
Entity 1:	Employee
Entity 2:	EmployeeHistory
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: Emp Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: EmpHist Mandatory: Yes Dominant: No Min, Max: 1, 1

EmpJobTitles

Name:	EmpJobTitles
Entity 1:	JobTitle
Entity 2:	Employee
Cardinality:	Many to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: JobTitle Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Employee Mandatory: No Dominant: No Min, Max: 0, n

EmployeeCompany

Name:	EmployeCompany
Entity 1:	Company
Entity 2:	Employee
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Company Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Employee Mandatory: Yes Dominant: No Min, Max: 1, 1

EmployeeBType

Name:	EmployeeBType
Entity 1:	BloodType
Entity 2:	Employee
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Blood Type Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Employee Mandatory: No Dominant: No Min, Max: 0, 1

EmployeeDepartment

Name:	EmployeeDepartment
Entity 1:	Department
Entity 2:	Employee
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Department Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Employee Mandatory: Yes Dominant: No Min, Max: 1, 1

EmpDependents

Name:	EmpDependents
Entity 1:	Employee
Entity 2:	Dependents
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: Emp Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Dependents Mandatory: Yes Dominant: No Min, Max: 1, 1

EmployeeEmpCalPln

Name:	EmployeeEmpCalPln
Entity 1:	Employee
Entity 2:	EmployeeCalendar_Planned
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: Employee Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: EmpCalPln Mandatory: Yes Dominant: No Min, Max: 1, 1

EmployeeQualification

Name:	EmployeeQualification
Entity 1:	Employee
Entity 2:	Qualification
Cardinality:	Many to Many
Entity 2 dependent of Entity 1:	No
Entity 1 → Entity 2:	Role: Employee Mandatory: Yes Dominant: No Min, Max: 1, n
Entity 2 → Entity 1:	Role: Qualification Mandatory: Yes Dominant: No Min, Max: 1, n

EmployeeTitle

Name:	EmployeeTitle
Entity 1:	Title
Entity 2:	Employee
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No
Entity 1 → Entity 2:	Role: Title Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1:	Role: Employee Mandatory: No Dominant: No Min, Max: 0, 1

EmpMainJobTitle

Name:	EmpMainJobTitle
Entity 1:	JobTitle
Entity 2:	Employee
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No
Entity 1 → Entity 2:	Role: JobTitle Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1:	Role: Employee Mandatory: Yes Dominant: No Min, Max: 1, 1

EmpTraining

Name:	EmpTraining
Entity 1:	Employee
Entity 2:	EmpTraining
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes
Entity 1 → Entity 2:	Role: Employee Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1:	Role: Training Mandatory: Yes Dominant: No Min, Max: 1, 1

EmpVacEntitlement

Name:	EmpVacEntitlement
Entity 1:	Employee
Entity 2:	VacationEntitlement
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes
Entity 1 → Entity 2:	Role: Emp Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1:	Role: VacEntitlement Mandatory: Yes Dominant: No Min, Max: 1, 1

Instructor

Name:	Instructor
Entity 1:	Employee
Entity 2:	TrainingActivity
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No
Entity 1 → Entity 2:	Role: Employee Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1:	Role: Training Mandatory: No Dominant: No Min, Max: 0, 1

LevelInstitution

Name:	LevelInstitution
Entity 1:	Level
Entity 2:	Institution
Cardinality:	Many to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Level Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Institution Mandatory: No Dominant: No Min, Max: 0, n

LocationPM

Name:	LocationPM
Entity 1:	Location
Entity 2:	ProductionMachine
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Location Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: PM Mandatory: No Dominant: No Min, Max: 0, 1

Manager

Name:	Manager
Entity 1:	Employee
Entity 2:	Department
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Manages Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Managed By Mandatory: No Dominant: No Min, Max: 0, 1

Nationality

Name:	Nationality
Entity 1:	Country
Entity 2:	Employee
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Country Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Employee Mandatory: No Dominant: No Min, Max: 0, 1

OrgStatOrgCalAct

Name:	OrgStatOrgCalAct
Entity 1:	OrganizationStatus
Entity 2:	OrganizationCalendar_Actual
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: OrgStat Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: OrgCalAct Mandatory: Yes Dominant: No Min, Max: 1, 1

OrgStatOrgCalPln

Name:	OrgStatOrgCalPln
Entity 1:	OrganizationStatus
Entity 2:	OrganizationCalendar_Planned
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: OrgStat Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: OrgCalPln Mandatory: Yes Dominant: No Min, Max: 1, 1

PMCalDowntimes

Name:	PMCalDowntimes
Entity 1:	PMCalendar
Entity 2:	PMDownTimes
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: PMCal Mandatory: No Dominant: No Min, Max: 0, n

Entity 2 → Entity 1: Role: Downtimes Mandatory: Yes Dominant: No Min, Max: 1, 1

PMCalendarActivity

Name:	PMCalendarActivity
Entity 1:	PMCalendar
Entity 2:	Activity
Cardinality:	Many to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: PMCalendar Mandatory: Yes Dominant: No Min, Max: 1, n

Entity 2 → Entity 1: Role: Activity Mandatory: Yes Dominant: No Min, Max: 1, n

PMCalendarPMSchedule

Name:	PMCalendarPMSchedule
Entity 1:	PMCalendar
Entity 2:	PMSchedule
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: PMCalendar Mandatory: No Dominant: No Min, Max: 0, n

Entity 2 → Entity 1: Role: PMSchedule Mandatory: Yes Dominant: No Min, Max: 1, 1

PMMachineType

Name:	PMMachineType
Entity 1:	MachineType
Entity 2:	ProductionMachine
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: MachineType Mandatory: No Dominant: No Min, Max: 0, n

Entity 2 → Entity 1: Role: PM Mandatory: No Dominant: No Min, Max: 0, 1

PMPMCalendar

Name:	PMPMCalendar
Entity 1:	ProductionMachine
Entity 2:	PMCalendar
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: PM Mandatory: No Dominant: No Min, Max: 0, n

Entity 2 → Entity 1: Role: PMCalendar Mandatory: Yes Dominant: No Min, Max: 1, 1

PMQualification

Name:	PMQualification
Entity 1:	Qualification
Entity 2:	ProductionMachine
Cardinality:	Many to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Qualification Mandatory: No Dominant: No Min, Max: 0, n

Entity 2 → Entity 1: Role: ProductionMachine Mandatory: No Dominant: No Min, Max: 0,n

PMStatusPMCalendar

Name:	PMStatusPMCalendar
Entity 1:	PMStatus
Entity 2:	PMCalendar
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 $\rightarrow$ Entity 2: Role: PMStatus Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 $\rightarrow$ Entity 1: Role: PMCalendar Mandatory: Yes Dominant: No Min, Max: 1, 1

ProductPMSchedule

Name:	ProductPMSchedule
Entity 1:	Product
Entity 2:	PMSchedule
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 $\rightarrow$ Entity 2: Role: Product Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 $\rightarrow$ Entity 1: Role: PMSchedule Mandatory: Yes Dominant: No Min, Max: 1, 1

Qualification

Name:	Qualification
Entity 1:	Qualification
Entity 2:	TrainingActivity
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 $\rightarrow$ Entity 2: Role: Qualification Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 $\rightarrow$ Entity 1: Role: Training Mandatory: No Dominant: No Min, Max: 0, 1

ShiftEmpCalAct

Name:	ShiftEmpCalAct
Entity 1:	Shift
Entity 2:	EmployeeCalendar_Actual
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 $\rightarrow$ Entity 2: Role: Shift Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 $\rightarrow$ Entity 1: Role: EmpCalAct Mandatory: Yes Dominant: No Min, Max: 1, 1

ShiftEmpCalPln

Name:	ShiftEmpCalPln
Entity 1:	Shift
Entity 2:	EmployeeCalendar_Planned
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 $\rightarrow$ Entity 2: Role: Shift Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 $\rightarrow$ Entity 1: Role: EmpCalPln Mandatory: Yes Dominant: No Min, Max: 1, 1

ShiftOrgCalAct

Name:	ShiftOrgCalAct
Entity 1:	Shift
Entity 2:	OrganizationCalendar_Actual
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 $\rightarrow$ Entity 2: Role: Shift Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 $\rightarrow$ Entity 1: Role: OrgCalAct Mandatory: Yes Dominant: No Min, Max: 1, 1

ShiftOrgCalPln

Name:	ShiftOrgCalPln
Entity 1:	Shift
Entity 2:	OrganizationCalendar_Planned
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: Shift Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: OrgCalPln Mandatory: Yes Dominant: No Min, Max: 1, 1

ShiftPMCalendar

Name:	ShiftPMCalendar
Entity 1:	Shift
Entity 2:	PMCalendar
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: Shift Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: PMCalendar Mandatory: Yes Dominant: No Min, Max: 1, 1

StationEmployee

Name:	StationEmployee
Entity 1:	Employee
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Station Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Employee Mandatory: No Dominant: No Min, Max: 0, 1

StationShuttle

Name:	StationShuttle
Entity 1:	Station
Entity 2:	Shuttle
Cardinality:	Many to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Station Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Shuttle Mandatory: No Dominant: No Min, Max: 0, n

SuperDepartment

Name:	SuperDepartment
Entity 1:	Department
Entity 2:	Department
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Super Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Sub Mandatory: No Dominant: No Min, Max: 0, 1

SuperPM

Name:	SuperPM
Entity 1:	ProductionMachine
Entity 2:	ProductionMachine
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Super Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Sub Mandatory: No Dominant: No Min, Max: 0, 1

TMT

Name:	TMT
Entity 1:	TrainingMaster
Entity 2:	TrainingActivity
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: TM Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Training Mandatory: Yes Dominant: No Min, Max: 1, 1

TrainingEmpTraining

Name:	TrainingEmpTraining
Entity 1:	TrainingActivity
Entity 2:	EmpTraining
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes

Entity 1 → Entity 2: Role: Training Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: EmpTraining Mandatory: Yes Dominant: No Min, Max: 1, 1

TStatus

Name:	TStatus
Entity 1:	TrainingStatus
Entity 2:	TrainingActivity
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: Status Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: TrainingStatus Mandatory: No Dominant: No Min, Max: 0, 1

TTypeTraining

Name:	TTypeTraining
Entity 1:	TrainingType
Entity 2:	TrainingMaster
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: TT Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Training Mandatory: No Dominant: No Min, Max: 0, 1

WCTypeCrew

Name:	WCTypeCrew
Entity 1:	WorkCalendarType
Entity 2:	Crew
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: WCType Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: Crew Mandatory: Yes Dominant: No Min, Max: 1, 1

WorkTeamPMCalendar

Name:	WorkTeamPMCalendar
Entity 1:	WorkTeam
Entity 2:	PMCalendar
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	No

Entity 1 → Entity 2: Role: WorkTeam Mandatory: No Dominant: No Min, Max: 0, n
Entity 2 → Entity 1: Role: PMCalendar Mandatory: Yes Dominant: No Min, Max: 1, 1

WrkCalTypeOrgCalAct

Name:	WrkCalTypeOrgCalAct
Entity 1:	WorkCalendarType
Entity 2:	OrganizationCalendar_Actual
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes
Entity 1 $\rightarrow$ Entity 2: Role: WrkCalType Mandatory: No Dominant: No Min, Max: 0, n	
Entity 2 $\rightarrow$ Entity 1: Role: OrgCalAct Mandatory: Yes Dominant: No Min, Max: 1, 1	

WrkCalTypeOrgCalPln

Name:	WrkCalTypeOrgCalPln
Entity 1:	WorkCalendarType
Entity 2:	OrganizationCalendar_Planned
Cardinality:	One to Many
Entity 2 dependent of Entity 1:	Yes
Entity 1 $\rightarrow$ Entity 2: Role: WrkCalType Mandatory: No Dominant: No Min, Max: 0, n	
Entity 2 $\rightarrow$ Entity 1: Role: OrgCalPln Mandatory: Yes Dominant: No Min, Max: 1, 1	

Conceptual Model of the Database

The initial E/R model is translated into the conceptual model in Sybase PowerDesigner© (Data Architect). In this representation, following symbols shown in Figure 5.3 are used for the entities and relationships in the model.

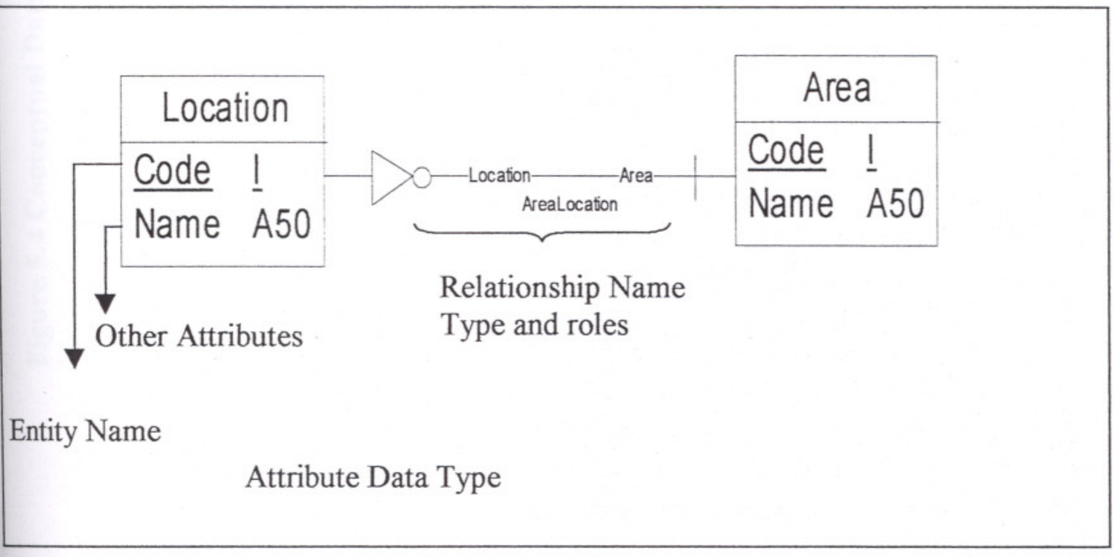
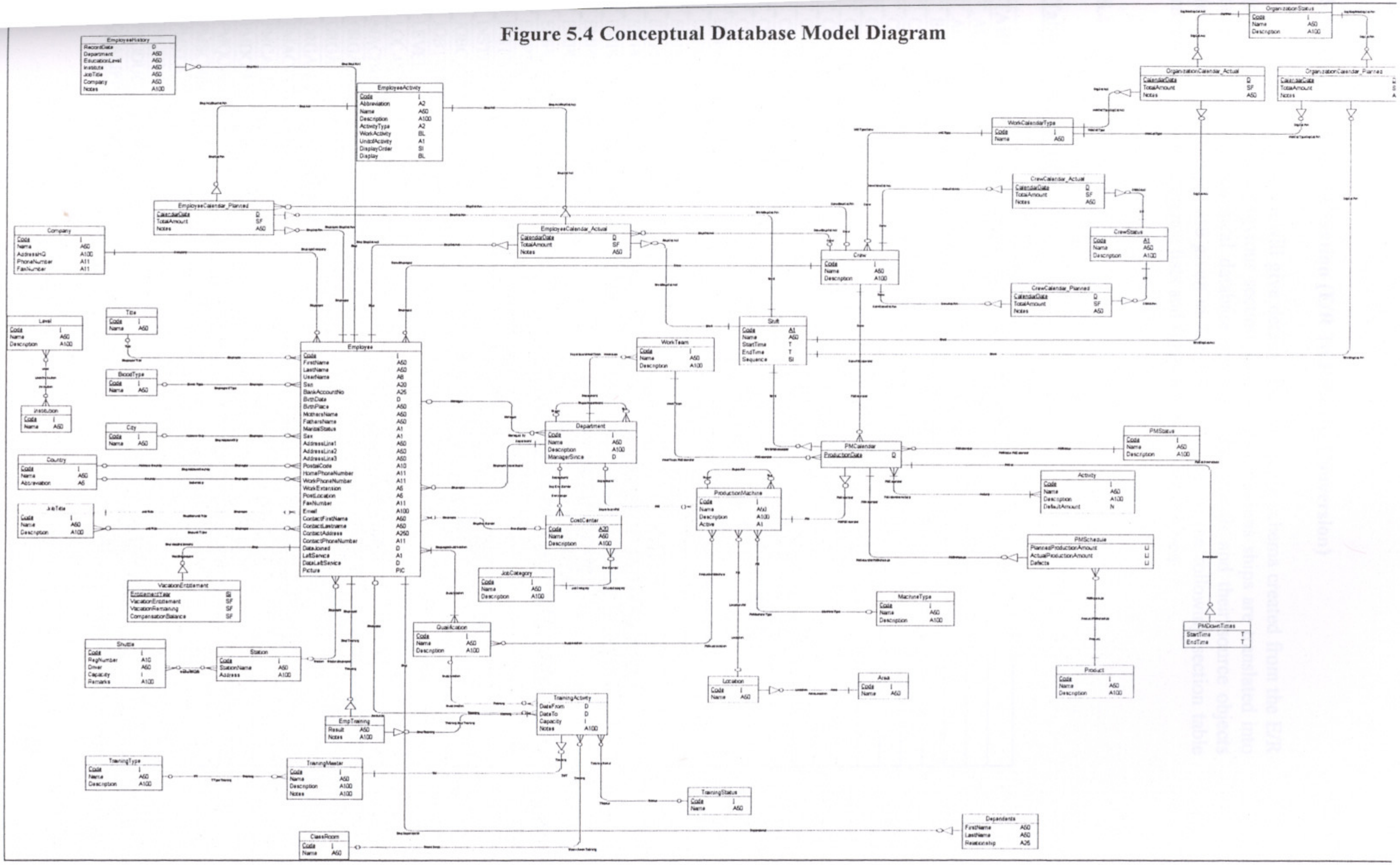


Figure 5.3. Entity Relationship Representation in Conceptual Model

In Figure 5.4 the conceptual model of the database created by Data Architect is given. This is a is a one-to-one translation of the original E/R Model into the Data Architects conceptual model format. We will be using this conceptual model and the features of the tool to compile and check to model and create the relational schema (physical model) from it. In the

Figure 5.4 Conceptual Database Model Diagram



5.4 Database Schema Creation (E/R To Relation Conversion)

In this Section we will give details of the database schema created from the E/R (conceptual model) in previous section. Entities and relations ships are translated into table objects in the physical database schema. Table details and their source objects (entity or relationship) are given in Table 5.2. Further in the following section table attributes, indexes, reference lists and schema domain list is given.

5.4.1. Tables

Table 5.2. Table List

TABLE NAME	SOURCE
T_ACTIVITY	*Entity ACTIVITY*
T_AREA	*Entity AREA*
T_BLOODTYPE	*Entity BLOODTYPE*
T_CITY	*Entity CITY*
T_CLASSROOM	*Entity CLASSROOM*
T_COMPANY	*Entity COMPANY*
T_COSTCENTER	*Entity COSTCENTER*
T_COUNTRY	*Entity COUNTRY*
T_CREW	*Entity CREW*
T_CREWCALENDAR_ACTUAL	*Entity CREWCALENDAR_ACTUAL*
T_CREWCALENDAR_PLANNED	*Entity CREWCALENDAR_PLANNED*
T_CREWSTATUS	*Entity CREWSTATUS*
T_DEPARTMENT	*Entity DEPARTMENT*
T_DEPENDENTS	*Entity DEPENDENTS*
T_EMPLOYEE	*Entity EMPLOYEE*
T_EMPLOYEEACTIVITY	*Entity EMPLOYEEACTIVITY*
T_EMPLOYEECALENDAR_ACTUAL	*Entity EMPLOYEECALENDAR_ACTUAL*
T_EMPLOYEECALENDAR_PLANNED	*Entity EMPLOYEECALENDAR_PLANNED*
T_EMPLOYEEHISTORY	*Entity EMPLOYEEHISTORY*
T_EMPLOYEEJOBTITLE	*Relationship EMPJOBTITLES*
T_EMPLOYEEQUALIFICATION	*Relationship EMPLOYEEQUALIFICATION*
T_EMPTRAINING	*Entity EMPTRAINING*
T_INSTITUTE	*Entity INSTITUTE*
T_INSTITUTELEVEL	*Relationship LEVELINSTITUTION*
T_JOBCATEGORY	*Entity JOBCATEGORY*
T_JOBTITLE	*Entity JOBTITLE*
T_LEVEL	*Entity LEVEL*
T_LOCATION	*Entity LOCATION*
T_MACHINETYPE	*Entity MACHINETYPE*
T_ORGANIZATIONCALENDAR_ACTUAL	*Entity ORGANIZATIONCALENDAR_ACTUAL*
T_ORGANIZATIONCALENDAR_PLANNED	*Entity ORGANIZATIONCALENDAR_PLANNED*
T_ORGANIZATIONSTATUS	*Entity ORGANIZATIONSTATUS*
T_PMACTIVITYHOURS	*Relationship PMCALENDARACTIVITY*
T_PMCALENDAR	*Entity PMCALENDAR*
T_PMDOWNTIMES	*Entity PMDOWNTIMES*
T_PMQUALIFICATION	*Relationship PMQUALIFICATION*
T_PMSCHEDULE	*Entity PMSCHEDULE*
T_PMSTATUS	*Entity PMSTATUS*
T_PRODUCT	*Entity PRODUCT*
T_PRODUCTIONMACHINE	*Entity PRODUCTIONMACHINE*
T_QUALIFICATION	*Entity QUALIFICATION*

Table 5.2 Continued

T_SHIFT	*Entity SHIFT*
T_SHUTTLE	*Entity SHUTTLE*
T_STATION	*Entity STATION*
T_STATIONSHUTTLE	*Relationship STATIONSHUTTLE*
T_TITLE	*Entity TITLE*
T_TRAININGACTIVITY	*Entity TRAININGACTIVITY*
T_TRAININGMASTER	*Entity TRAININGMASTER*
T_TRAININGSTATUS	*Entity TRAININGSTATUS*
T_TRAININGTYPE	*Entity TRAININGTYPE*
T_VACATIONENTITLEMENT	*Entity VACATIONENTITLEMENT*
T_WORKCALENDARTYPE	*Entity WORKCALENDARTYPE*
T_WORKTEAM	*Entity WORKTEAM*

Table 5.2 lists the tables created from the E/R (conceptual) model. Below, the details of each table is given.

TABLE: **ACTIVITY**

Column List

Name	Type	P*	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No
DEFAULTAMOUNT	numeric	No	No

Index List

Index Name	P**	F	U	C	Column Name	Sort
T_ACTIVITY_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_PMACTIVITYHOURS	CODE	ACTIVITYCODE

TABLE: **AREA**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_AREA_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_LOCATION	CODE	ARE_CODE

* P: Primary Key, M: Mandatory

** P: Primary Index, F: Foreign Index, U: User Defined, C: Clustered

TABLE: **BLOODTYPE**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_BLOODTYPE_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_EMPLOYEE	CODE	BLOODTYPECODE

TABLE: **CITY**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_CITY_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_EMPLOYEE	CODE	ADDRESSCITYCODE

TABLE: **CLASSROOM**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_CLASSROOM_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_TRAININGACTIVITY	CODE	CLA_CODE

TABLE: **COMPANY**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
ADDRESSHQ	char(100)	No	No
PHONENUMBER	char(11)	No	No
FAXNUMBER	char(11)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_COMPANY_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_EMPLOYEE	CODE	COMPANYCODE

TABLE: **COSTCENTER**

Column List

Name	Type	P	M
CODE	char(20)	Yes	Yes
DEPARTMENTCODE	integer	No	No
JOBCATEGORYCODE	integer	No	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_COSTCENTER_PK	Yes	No	Yes	No	CODE	ASC
DEPTCOSTCENTER_FK	No	Yes	No	No	DEPARTMENTCODE	ASC
CCJOBCATEGORY_FK	No	Yes	No	No	JOBCATEGORYCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_JOBCATEGORY	CODE	JOBCATEGORYCODE
T_DEPARTMENT	CODE	DEPARTMENTCODE

Reference by List

Referenced by	Primary Key	Foreign Key
T_EMPLOYEE	CODE	COSTCENTERCODE

TABLE: **COUNTRY**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
ABBREVIATION	char(5)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_COUNTRY_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_EMPLOYEE	CODE	ADDRESSCOUNTRYCODE
T_EMPLOYEE	CODE	NATIONALITYCODE

TABLE: **CREW**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
WORKINGCALENDARTYPECODE	integer	No	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_CREW_PK	Yes	No	Yes	No	CODE	ASC
WCTYPECREW_FK	No	Yes	No	No	WORKINGCALENDARTYPEC ODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_WORKCALENDARTYPE	CODE	WORKINGCALENDARTY PECODE

Reference by List

Referenced by	Primary Key	Foreign Key
T_CREWCALENDAR_ACTUAL	CODE	CREWCODE
T_CREWCALENDAR_PLANNED	CODE	CREWCODE
T_EMPLOYEECALENDAR_ACT UAL	CODE	CREWCODE
T_EMPLOYEECALENDAR_PLAN NED	CODE	CREWCODE
T_EMPLOYEE	CODE	CREWCODE
T_PMCALENDAR	CODE	CREWCODE

TABLE: **CREWCALENDAR_ACTUAL**

Column List

Name	Type	P	M
CREWCODE	integer	Yes	Yes
CALENDARDATE	date	Yes	Yes
CREWSTATUSCODE	integer	Yes	Yes
TOTALAMOUNT	real	No	No
NOTES	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_CREWCALENDAR_ACTUA L_PK	No	No	Yes	No	CREWSTATUSCODE CREWCODE CALENDARDATE	ASC ASC ASC
CREWCREWCALACT_FK	No	Yes	No	No	CREWCODE	ASC
CSCCACT_FK	No	Yes	No	No	CREWSTATUSCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_CREW	CODE	CREWCODE
T_CREWSTATUS	CODE	CREWSTATUSCODE

TABLE: **CREWCALENDAR_PLANNED**

Column List

Name	Type	P	M
CREWCODE	integer	Yes	Yes
CALENDARDATE	date	Yes	Yes
CREWSTATUSCODE	integer	Yes	Yes
TOTALAMOUNT	real	No	No
NOTES	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_CREWCALENDAR_PLANN	No	No	Yes	No	CREWSTATUSCODE	ASC
ED_PK					CREWCODE	ASC
					CALENDARDATE	ASC
CREWCREWCALPLN_FK	No	Yes	No	No	CREWCODE	ASC
CSCCPLN_FK	No	Yes	No	No	CREWSTATUSCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_CREW	CODE	CREWCODE
T_CREWSTATUS	CODE	CREWSTATUSCODE

TABLE: **CREWSTATUS**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_CREWSTATUS_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_CREWCALENDAR_ACTUAL	CODE	CREWSTATUSCODE
T_CREWCALENDAR_PLANNED	CODE	CREWSTATUSCODE

TABLE: **DEPARTMENT**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
SUPERDEPARTMENTCODE	integer	No	No
MANAGERCODE	integer	No	No
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No
MANAGERSINCE	date	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_DEPARTMENT_PK	Yes	No	Yes	No	CODE	ASC
R_SUPERDEPARTMENT_FK	No	Yes	No	No	SUPERDEPARTMENTCODE	ASC
R_EMPLOYEEDEPARTMENT	No	Yes	No	No	MANAGERCODE	ASC
MGR_FK						

Reference to List

Reference to	Primary Key	Foreign Key
T_EMPLOYEE	CODE	MANAGERCODE
T_DEPARTMENT	CODE	SUPERDEPARTMENTCODE

Reference by List

Referenced by	Primary Key	Foreign Key
T_PRODUCTIONMACHINE	CODE	DEPARTMENTCODE
T_WORKTEAM	CODE	DEPARTMENTCODE
T_COSTCENTER	CODE	DEPARTMENTCODE
T_EMPLOYEE	CODE	DEPARTMENTCODE
T_DEPARTMENT	CODE	SUPERDEPARTMENTCODE

TABLE: **DEPENDENTS**

Column List

Name	Type	P	M
EMPLOYEECODE	integer	Yes	Yes
FIRSTNAME	char(50)	No	No
LASTNAME	char(50)	No	No
RELATIONSHIP	char(25)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_DEPENDENTS_PK	Yes	Yes	Yes	No	EMPLOYEECODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_EMPLOYEE	CODE	EMPLOYEECODE

TABLE: **EMPLOYEE**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
COMPANYCODE	integer	No	Yes
DEPARTMENTCODE	integer	No	Yes
CREWCODE	integer	No	Yes
COSTCENTERCODE	char(20)	No	No
TITLECODE	integer	No	No
FIRSTNAME	char(50)	No	No
LASTNAME	char(50)	No	No
USERNAME	char(8)	No	No
SSN	char(20)	No	No
BANKACCOUNTNO	char(25)	No	No
BIRTHDATE	date	No	No
BIRTHPLACE	char(50)	No	No
MOTHERSNAME	char(50)	No	No
FATHERSNAME	char(50)	No	No
MARITALSTATUS	char(1)	No	No
SEX	char(1)	No	No
ADDRESSLINE1	char(50)	No	No
ADDRESSLINE2	char(50)	No	No
ADDRESSLINE3	char(50)	No	No
POSTALCODE	char(10)	No	No

Name	Type	P	M
ADDRESSCITYCODE	integer	No	No
ADDRESSCOUNTRYCODE	integer	No	No
HOMEPHONENUMBER	char(11)	No	No
WORKPHONENUMBER	char(11)	No	No
WORKEXTENSION	char(5)	No	No
FAXNUMBER	char(11)	No	No
EMAIL	char(100)	No	No
CONTACTFIRSTNAME	char(50)	No	No
CONTACTLASTNAME	char(50)	No	No
CONTACTADDRESS	char(250)	No	No
CONTACTPHONENUMBER	char(11)	No	No
DATEJOINED	date	No	No
LEFTSERVICE	char(1)	No	No
DATELEFTSERVICE	date	No	No
BLOODTYPECODE	integer	No	No
EDUCATIONLEVELCODE	integer	No	No
INSTITUTIONCODE	integer	No	No
JOBTITLECODE	integer	No	No
STATIONCODE	integer	No	No
NATIONALITYCODE	integer	No	No
PICTURE	long binary	No	No
POSTLOCATION	char(5)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_EMPLOYEE_PK	Yes	No	Yes	No	CODE	ASC
R_EMPLOYEEDEPARTMENT_FK	No	Yes	No	No	DEPARTMENTCODE	ASC
R_EMPLOYEETITLE_FK	No	Yes	No	No	TITLECODE	ASC
EMPLOYEEBLOODTYPE_FK	No	Yes	No	No	BLOODTYPECODE	ASC
EMPADDRESSCITY_FK	No	Yes	No	No	ADDRESSCITYCODE	ASC
EMPADDRESSCOUNTRY_FK	No	Yes	No	No	ADDRESSCOUNTRYCODE	ASC
EMPLOYECOMPANY_FK	No	Yes	No	No	COMPANYCODE	ASC
EMPCOSTCENTER_FK	No	Yes	No	No	COSTCENTERCODE	ASC
EMPJOBTITLE_FK	No	Yes	No	No	JOBTITLECODE	ASC
CREWEMPLOYEE_FK	No	Yes	No	No	CREWCODE	ASC
STATIONEMPLOYEE_FK	No	Yes	No	No	STATIONCODE	ASC
NATIONALITY_FK	No	Yes	No	No	NATIONALITYCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_CREW	CODE	CREWCODE
T_CITY	CODE	ADDRESSCITYCODE
T_COUNTRY	CODE	ADDRESSCOUNTRYCODE
T_COSTCENTER	CODE	COSTCENTERCODE
T_COMPANY	CODE	COMPANYCODE
T_BLOODTYPE	CODE	BLOODTYPECODE
T_DEPARTMENT	CODE	DEPARTMENTCODE
T_INSTITUTELEVEL	LEVELCODE	EDUCATIONLEVELCODE
	INSTITUTIONCODE	INSTITUTIONCODE
T_TITLE	CODE	TITLECODE
T_JOBTITLE	CODE	JOBTITLECODE
T_COUNTRY	CODE	NATIONALITYCODE
T_STATION	CODE	STATIONCODE

Reference by List

Referenced by	Primary Key	Foreign Key
T_DEPENDENTS	CODE	EMPLOYEECODE
T_EMPLOYEECALENDAR_ACTUAL	CODE	EMPLOYEECODE
T_EMPLOYEEHISTORY	CODE	CODE
T_EMPLOYEEQUALIFICATION	CODE	EMPLOYEECODE
T_EMPLOYEEJOBTITLE	CODE	EMPLOYEECODE
T_EMPLOYEECALENDAR_PLANNED	CODE	EMPLOYEECODE
T_EMPTRAINING	CODE	EMPLOYEECODE
T_VACATIONENTITLEMENT	CODE	EMPLOYEECODE
T_TRAININGACTIVITY	CODE	INSTRUCTORCODE
T_DEPARTMENT	CODE	MANAGERCODE

TABLE: **EMPLOYEEACTIVITY**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
ABBREVIATION	char(2)	No	No
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No
ACTIVITYTYPE	char(2)	No	No
WORKACTIVITY	numeric(1)	No	Yes
UNITOFACTIVITY	char(1)	No	No
DISPLAYORDER	smallint	No	Yes
DISPLAY	numeric(1)	No	Yes

Index List

Index Name	P	F	U	C	Column Name	Sort
T_EMPLOYEEACTIVITY_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_EMPLOYEECALENDAR_ACTUAL	CODE	EMPLOYEEACTIVITYCODE
T_EMPLOYEECALENDAR_PLANNED	CODE	EMPLOYEEACTIVITYCODE

TABLE: **EMPLOYEECALENDAR_ACTUAL**

Column List

Name	Type	P	M
CALENDARDATE	date	Yes	Yes
SHIFTCODE	char(1)	Yes	Yes
EMPLOYEECODE	integer	Yes	Yes
EMPLOYEEACTIVITYCODE	integer	Yes	Yes
TOTALAMOUNT	real	No	No
NOTES	char(50)	No	No
CREWCODE	integer	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_EMPLOYEECALENDAR_ACTUAL_PK	No	No	Yes	No	EMPLOYEECODE	ASC
					EMPLOYEEACTIVITYCODE	ASC
					SHIFTCODE	ASC

Index Name	P	F	U	C	Column Name	Sort
					CALENDARDATE	ASC
T_EMPLOYEECALENDAR_ACTUAL_PK	No	No	Yes	No	EMPLOYEECODE	ASC
					EMPLOYEEACTIVITYCODE	ASC
					SHIFTCODE	ASC
					CALENDARDATE	ASC
EMPACTEMPCALACT_FK	No	Yes	No	No	EMPLOYEEACTIVITYCODE	ASC
EMPEMPCALACT_FK	No	Yes	No	No	EMPLOYEECODE	ASC
CREWEMPCALACT_FK	No	Yes	No	No	CREWCODE	ASC
SHIFTEMPCALACT_FK	No	Yes	No	No	SHIFTCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_CREW	CODE	CREWCODE
T_EMPLOYEEACTIVITY	CODE	EMPLOYEEACTIVITYCODE
T_EMPLOYEE	CODE	EMPLOYEECODE
T_SHIFT	CODE	SHIFTCODE

TABLE: **EMPLOYEECALENDAR_PLANNED**

Column List

Name	Type	P	M
CALENDARDATE	date	Yes	Yes
SHIFTCODE	char(1)	Yes	Yes
EMPLOYEECODE	integer	Yes	Yes
EMPLOYEEACTIVITYCODE	integer	Yes	Yes
CREWCODE	integer	No	No
TOTALAMOUNT	real	No	No
NOTES	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_EMPLOYEECALENDAR_PLANNED_PK	No	No	Yes	No	EMPLOYEECODE	ASC
					EMPLOYEEACTIVITYCODE	ASC
					SHIFTCODE	ASC
					CALENDARDATE	ASC
EMPACTEMPCALPLN_FK	No	Yes	No	No	EMPLOYEEACTIVITYCODE	ASC
SHIFTEMPCALPLN_FK	No	Yes	No	No	SHIFTCODE	ASC
EMPLOYEEEMPCALPLN_FK	No	Yes	No	No	EMPLOYEECODE	ASC
CREWEMPCALPLN_FK	No	Yes	No	No	CREWCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_CREW	CODE	CREWCODE
T_EMPLOYEEACTIVITY	CODE	EMPLOYEEACTIVITYCODE
T_EMPLOYEE	CODE	EMPLOYEECODE
T_SHIFT	CODE	SHIFTCODE

TABLE: **EMPLOYEEHISTORY**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
RECORDDATE	date	No	No
EDUCATIONLEVEL	char(50)	No	No
INSTITUTE	char(50)	No	No
JOBTITLE	char(50)	No	No
COMPANY	char(50)	No	No
NOTES	char(100)	No	No
DEPARTMENT	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_EMPLOYEEHISTORY_PK	Yes	Yes	Yes	No	CODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_EMPLOYEE	CODE	CODE

TABLE: **EMPLOYEEJOBTITLE**

Column List

Name	Type	P	M
JOBTITLECODE	integer	Yes	Yes
EMPLOYEECODE	integer	Yes	Yes
DATEFROM	date	No	No
DATETO	date	No	No
ACTIVE	numeric(1)	No	Yes

Index List

Index Name	P	F	U	C	Column Name	Sort
T_EMPLOYEEJOBTITLE_PK	Yes	No	Yes	No	JOBTITLECODE	ASC
					EMPLOYEECODE	ASC
JOBTITLE_FK	No	Yes	No	No	JOBTITLECODE	ASC
EMPLOYEE_FK2	No	Yes	No	No	EMPLOYEECODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_EMPLOYEE	CODE	EMPLOYEECODE
T_JOBTITLE	CODE	JOBTITLECODE

TABLE: **EMPLOYEEQUALIFICATION**

Column List

Name	Type	P	M
EMPLOYEECODE	integer	Yes	Yes
QUALIFICATIONCODE	integer	Yes	Yes
DATEFROM	date	No	No
DATETO	date	No	No
ACTIVE	numeric(1)	No	Yes

Index List

Index Name	P	F	U	C	Column Name	Sort
T_EMPLOYEEQUALIFICATION_PK	Yes	No	Yes	No	EMPLOYEECODE	ASC
					QUALIFICATIONCODE	ASC
EMPLOYEE_FK	No	Yes	No	No	EMPLOYEECODE	ASC

Index Name	P	F	U	C	Column Name	Sort
QUALIFICATION_FK2	No	Yes	No	No	QUALIFICATIONCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_EMPLOYEE	CODE	EMPLOYEECODE
T_QUALIFICATION	CODE	QUALIFICATIONCODE

TABLE: **EMPTRAINING**

Column List

Name	Type	P	M
TRAININGCODE	integer	Yes	Yes
EMPLOYEECODE	integer	Yes	Yes
RESULT	char(50)	No	No
NOTES	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_EMPTRAINING_PK	Yes	No	Yes	No	TRAININGCODE	ASC
					EMPLOYEECODE	ASC
EMPTRAINING_FK	No	Yes	No	No	EMPLOYEECODE	ASC
TRAININGEMPTRAINING_FK	No	Yes	No	No	TRAININGCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_EMPLOYEE	CODE	EMPLOYEECODE
T_TRAININGACTIVITY	TRAININGCODE	TRAININGCODE

TABLE: **INSTITUTELEVEL**

Column List

Name	Type	P	M
LEVELCODE	integer	Yes	Yes
INSTITUTIONCODE	integer	Yes	Yes

Index List

Index Name	P	F	U	C	Column Name	Sort
T_RELATION_294_PK	Yes	No	Yes	No	LEVELCODE	ASC
					INSTITUTIONCODE	ASC
RELATION_294_FK2	No	Yes	No	No	LEVELCODE	ASC
RELATION_294_FK	No	Yes	No	No	INSTITUTIONCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_INSTITUTE	CODE	INSTITUTIONCODE
T_LEVEL	CODE	LEVELCODE

Reference by List

Referenced by	Primary Key	Foreign Key
T_EMPLOYEE	LEVELCODE	EDUCATIONLEVELCODE
	INSTITUTIONCODE	INSTITUTIONCODE

TABLE: **INSTITUTION**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_INSTITUTE_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_INSTITUTELEVEL	CODE	INSTITUTIONCODE

TABLE: **JOBCATEGORY**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_JOBCATEGORY_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_COSTCENTER	CODE	JOBCATEGORYCODE

TABLE: **JOBTITLE**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_JOBTITLE_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_EMPLOYEE	CODE	JOBTITLECODE
T_EMPLOYEEJOBTITLE	CODE	JOBTITLECODE

TABLE: **LEVEL**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_LEVEL_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_INSTITUTELEVEL	CODE	LEVELCODE

TABLE: **LOCATION**

Column List

Name	Type	P	M
ARE_CODE	integer	Yes	Yes
CODE	integer	Yes	Yes
NAME	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_LOCATION_PK	Yes	No	Yes	No	ARE_CODE	ASC
					CODE	ASC
AREALOCATION_FK	No	Yes	No	No	ARE_CODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_AREA	CODE	ARE_CODE

Reference by List

Referenced by	Primary Key	Foreign Key
T_PRODUCTIONMACHINE	ARE_CODE	AREACODE
	CODE	LOCATIONCODE

TABLE: **MACHINETYPE**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_MACHINETYPE_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_PRODUCTIONMACHINE	CODE	MACHINETYPECODE

TABLE: **ORGANIZATIONCALENDAR_ACTUAL**

Column List

Name	Type	P	M
CALENDARDATE	date	Yes	Yes
SHIFTCODE	char(1)	Yes	Yes
WORKCALENDARTYPECODE	integer	Yes	Yes
ORGANIZATIONSTATUSCODE	integer	Yes	Yes
TOTALAMOUNT	real	No	No
NOTES	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_ORGANIZATIONCALENDAR_ACTUAL_PK	No	No	Yes	No	WORKCALENDARTYPECODE ORGANIZATIONSTATUSCODE SHIFTCODE CALENDARDATE	ASC ASC ASC ASC
ORGSTATORGCALACT_FK	No	Yes	No	No	ORGANIZATIONSTATUSCODE	ASC
SHIFTORGCALACT_FK	No	Yes	No	No	SHIFTCODE	ASC
WRKCALTYPEORGCALACT_FK	No	Yes	No	No	WORKCALENDARTYPECODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_ORGANIZATIONSTATUS	CODE	ORGANIZATIONSTATUSCODE
T_SHIFT	CODE	SHIFTCODE
T_WORKCALENDARTYPE	CODE	WORKCALENDARTYPECODE

TABLE: **ORGANIZATIONCALENDAR_PLANNED**

Column List

Name	Type	P	M
CALENDARDATE	date	Yes	Yes
SHIFTCODE	char(1)	Yes	Yes
WORKCALENDARTYPECODE	integer	Yes	Yes
ORGANIZATIONSTATUSCODE	integer	Yes	Yes
TOTALAMOUNT	real	No	No
NOTES	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_ORGANIZATIONCALENDAR_PLANNED_PK	No	No	Yes	No	WORKCALENDARTYPECODE ORGANIZATIONSTATUSCODE SHIFTCODE CALENDARDATE	ASC ASC ASC ASC
ORGSTATORGCALPLN_FK	No	Yes	No	No	ORGANIZATIONSTATUSCODE	ASC
SHIFTORGCALPLN_FK	No	Yes	No	No	SHIFTCODE	ASC
WRKCALTYPEORGCALPLN_FK	No	Yes	No	No	WORKCALENDARTYPECODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_ORGANIZATIONSTATUS	CODE	ORGANIZATIONSTATUSCODE
T_SHIFT	CODE	SHIFTCODE
T_WORKCALENDARTYPE	CODE	WORKCALENDARTYPECODE

TABLE: **ORGANIZATIONSTATUS**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_ORGANIZATIONSTATUS_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_ORGANIZATIONCALENDAR_ACTUAL	CODE	ORGANIZATIONSTATUSCODE
T_ORGANIZATIONCALENDAR_PLANNED	CODE	ORGANIZATIONSTATUSCODE

TABLE: **PMACTIVITYHOURS**

Column List

Name	Type	P	M
PRODUCTIONDATE	date	Yes	Yes
SHIFTCODE	char(1)	Yes	Yes
PRODUCTIONMACHINECODE	integer	Yes	Yes
ACTIVITYCODE	integer	Yes	Yes
DURATION	numeric	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_PMCALENDARACTIVITY_PK	No	No	Yes	No	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE ACTIVITYCODE	ASC ASC ASC ASC
RELATION_583_FK2	No	Yes	No	No	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE	ASC ASC ASC
RELATION_583_FK	No	Yes	No	No	ACTIVITYCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_ACTIVITY	CODE	ACTIVITYCODE
T_PMCALENDAR	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE

TABLE: **PMCALENDAR**

Column List

Name	Type	P	M
PRODUCTIONDATE	date	Yes	Yes
SHIFTCODE	char(1)	Yes	Yes
PRODUCTIONMACHINECODE	integer	Yes	Yes
CREWCODE	integer	No	Yes
PMSTATUSCODE	integer	No	Yes
WORKTEAMCODE	integer	No	Yes

Index List

Index Name	P	F	U	C	Column Name	Sort
T_PMCALENDAR_PK	No	No	Yes	No	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE	ASC ASC ASC
RELATION_532_FK	No	Yes	No	No	PRODUCTIONMACHINECODE	ASC
CREWPMCALENDAR_FK	No	Yes	No	No	CREWCODE	ASC
SHIFTPMCALENDAR_FK	No	Yes	No	No	SHIFTCODE	ASC
PMSTATUSPMCALENDAR_FK	No	Yes	No	No	PMSTATUSCODE	ASC
WORKTEAMPMCALENDAR_FK	No	Yes	No	No	WORKTEAMCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_CREW	CODE	CREWCODE
T_PRODUCTIONMACHINE	CODE	PRODUCTIONMACHINECODE
T_PMSTATUS	CODE	PMSTATUSCODE
T_SHIFT	CODE	SHIFTCODE
T_WORKTEAM	CODE	WORKTEAMCODE

Reference by List

Referenced by	Primary Key	Foreign Key
T_PMDOWNTIMES	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE
T_PMSCHEDULE	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE
T_PMACTIVITYHOURS	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE

TABLE: **PMDOWNTIMES**

Column List

Name	Type	P	M
PRODUCTIONDATE	date	Yes	Yes
PRODUCTIONMACHINECODE	integer	Yes	Yes
SHIFTCODE	char(1)	Yes	Yes
STARTTIME	time	No	No
ENDTIME	time	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_PMDOWNTIMES_PK	No	Yes	Yes	No	SHIFTCODE PRODUCTIONMACHINECOD E PRODUCTIONDATE	ASC ASC ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_PMCALENDAR	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE	SHIFTCODE PRODUCTIONMACHINEC ODE PRODUCTIONDATE

TABLE: **PMQUALIFICATION**

Column List

Name	Type	P	M
QUALIFICATIONCODE	integer	Yes	Yes
PRODUCTIONMACHINECODE	integer	Yes	Yes
REQUIRED	numeric(1)	No	Yes
REQUIREDAMOUNT	numeric	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_PMQUALIFICATION_PK	Yes	No	Yes	No	QUALIFICATIONCODE PRODUCTIONMACHINECOD E	ASC ASC
QUALIFICATION_FK	No	Yes	No	No	QUALIFICATIONCODE	ASC
PRODUCTIONMACHINE_FK	No	Yes	No	No	PRODUCTIONMACHINECOD E	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_PRODUCTIONMACHINE	CODE	PRODUCTIONMACHINEC ODE
T_QUALIFICATION	CODE	QUALIFICATIONCODE

İZMIR YUKSEK TEKNOLOJİ ENSTİTÜSÜ
REKTÖRLÜĞÜ
Kütüphane ve Dokümantasyon Daire Bşk.

TABLE: **PMSCHEDULE**

Column List

Name	Type	P	M
SHIFTCODE	char(1)	Yes	Yes
PRODUCTIONMACHINECODE	integer	Yes	Yes
PRODUCTIONDATE	date	Yes	Yes
PRODUCTCODE	integer	No	Yes
PLANNEDPRODUCTIONAMOUNT	numeric	No	No
ACTUALPRODUCTIONAMOUNT	integer	No	No
DEFECTS	integer	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_PMSCHEDULE_PK	Yes	Yes	Yes	No	SHIFTCODE PRODUCTIONMACHINECOD E PRODUCTIONDATE	ASC ASC ASC
RELATION_573_FK	No	Yes	No	No	PRODUCTCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_PMCALENDAR	SHIFTCODE PRODUCTIONMACHINECODE PRODUCTIONDATE	SHIFTCODE PRODUCTIONMACHINEC ODE PRODUCTIONDATE
T_PRODUCT	CODE	PRODUCTCODE

TABLE: **PMSTATUS**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_PMSTATUS_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_PMCALENDAR	CODE	PMSTATUSCODE

TABLE: **PRODUCT**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_PRODUCT_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_PMSCHEDULE	CODE	PRODUCTCODE

TABLE: **PRODUCTIONMACHINE**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
DEPARTMENTCODE	integer	No	No
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No
ACTIVE	char(1)	No	No
AREACODE	integer	No	No
LOCATIONCODE	integer	No	No
SUPERPRODUCTIONMACCODE	integer	No	No
MACHINETYPECODE	integer	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_PRODUCTIONMACHINE_PK	Yes	No	Yes	No	CODE	ASC
DEPARTMENTPM_FK	No	Yes	No	No	DEPARTMENTCODE	ASC
LOCATIONPM_FK	No	Yes	No	No	AREACODE	ASC
					LOCATIONCODE	ASC
RELATION_317_FK	No	Yes	No	No	SUPERPRODUCTIONMACCODE	ASC
PMMACHINETYPE_FK	No	Yes	No	No	MACHINETYPECODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_DEPARTMENT	CODE	DEPARTMENTCODE
T_LOCATION	ARE_CODE	AREACODE
	CODE	LOCATIONCODE
T_MACHINETYPE	CODE	MACHINETYPECODE
T_PRODUCTIONMACHINE	CODE	SUPERPRODUCTIONMACCODE

Reference by List

Referenced by	Primary Key	Foreign Key
T_PMCALENDAR	CODE	PRODUCTIONMACHINECODE
T_PMQUALIFICATION	CODE	PRODUCTIONMACHINECODE
T_PRODUCTIONMACHINE	CODE	SUPERPRODUCTIONMACCODE

TABLE: **QUALIFICATION**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_QUALIFICATION_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_EMPLOYEEQUALIFICATION	CODE	QUALIFICATIONCODE
T_PMQUALIFICATION	CODE	QUALIFICATIONCODE
T_TRAININGACTIVITY	CODE	QUALIFICATIONCODE

TABLE: **SHIFT**
Column List

Name	Type	P	M
CODE	char(1)	Yes	Yes
NAME	char(50)	No	No
STARTTIME	time	No	No
ENDTIME	time	No	No
SEQUENCE	smallint	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_SHIFT_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_EMPLOYEECALENDAR_ACTUAL	CODE	SHIFTCODE
T_EMPLOYEECALENDAR_PLANNED	CODE	SHIFTCODE
T_ORGANIZATIONCALENDAR_ACTUAL	CODE	SHIFTCODE
T_ORGANIZATIONCALENDAR_PLANNED	CODE	SHIFTCODE
T_PMCALENDAR	CODE	SHIFTCODE
T_STATIONSHUTTLE	CODE	SHIFTCODE

TABLE: **SHUTTLE**
Column List

Name	Type	P	M
CODE	integer	Yes	Yes
REGNUMBER	char(10)	No	No
DRIVER	char(50)	No	No
REMARKS	char(100)	No	No
CAPACITY	integer	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_SHUTTLE_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_STATIONSHUTTLE	CODE	SHUTTLECODE

TABLE: **STATION**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
STATIONNAME	char(50)	No	No
ADDRESS	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_STATION_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_STATIONSHUTTLE	CODE	STATIONCODE
T_EMPLOYEE	CODE	STATIONCODE

TABLE: **STATIONSHUTTLE**

Column List

Name	Type	P	M
CALENDARDATE	d	Yes	Yes
SHIFTCODE	char(1)	Yes	Yes
STATIONCODE	integer	Yes	Yes
SHUTTLECODE	integer	Yes	Yes
PICKUPTIME	t	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_STATIONSHUTTLE_PK	No	No	Yes	No	STATIONCODE	ASC
					SHUTTLECODE	ASC
STATION_FK	No	Yes	No	No	STATIONCODE	ASC
SHUTTLE_FK	No	Yes	No	No	SHUTTLECODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_SHUTTLE	CODE	SHUTTLECODE
T_SHIFT	CODE	SHIFTCODE
T_STATION	CODE	STATIONCODE

TABLE: **TITLE**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_TITLE_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_EMPLOYEE	CODE	TITLECODE

TABLE: **TRAININGACTIVITY**

Column List

Name	Type	P	M
TRAININGCODE	integer	Yes	Yes
TRANINGSTATUSCODE	integer	No	No
QUALIFICATIONCODE	integer	No	No
INSTRUCTORCODE	integer	No	No
DATEFROM	date	No	No
DATETO	date	No	No
CAPACITY	integer	No	No
NOTES	char(100)	No	No
CLASSROOMCODE	integer	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_TRAININGACTIVITY_PK	Yes	Yes	Yes	No	TRAININGCODE	ASC
TSTATUS_FK	No	Yes	No	No	TRANINGSTATUSCODE	ASC
INSTRUCTOR_FK	No	Yes	No	No	INSTRUCTORCODE	ASC
QUALIFICATION_FK3	No	Yes	No	No	QUALIFICATIONCODE	ASC
CLASSROOMTRAINING_FK	No	Yes	No	No	CLASSROOMCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_CLASSROOM	CODE	CLASSROOMCODE
T_EMPLOYEE	CODE	INSTRUCTORCODE
T_QUALIFICATION	CODE	QUALIFICATIONCODE
T_TRAININGMASTER	CODE	TRAININGCODE
T_TRAININGSTATUS	CODE	TRANINGSTATUSCODE

Reference by List

Referenced by	Primary Key	Foreign Key
T_EMPTRAINING	TRAININGCODE	TRAININGCODE

TABLE: **TRAININGMASTER**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
TRAININGTYPECODE	integer	No	No
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No
NOTES	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_TRAININGMASTER_PK	Yes	No	Yes	No	CODE	ASC
TTYPETRAINING_FK	No	Yes	No	No	TRAININGTYPECODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_TRAININGTYPE	CODE	TRAININGTYPECODE

Reference by List

Referenced by	Primary Key	Foreign Key
T_TRAININGACTIVITY	CODE	TRAININGCODE

TABLE: **TRAININGSTATUS**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_TRAININGSTATUS_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_TRAININGACTIVITY	CODE	TRANINGSTATUSCODE

TABLE: **TRAININGTYPE**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_TRAININGTYPE_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_TRAININGMASTER	CODE	TRAININGTYPECODE

TABLE: **VACATIONENTITLEMENT**

Column List

Name	Type	P	M
ENTITLEMENTYEAR	smallint	Yes	Yes
EMPLOYEECODE	integer	Yes	Yes
VACATIONENTITLEMENT	real	No	No
VACATIONREMAINING	real	No	No
COMPENSATIONBALANCE	real	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_VACATIONENTITLEMENT_ PK	No	No	Yes	No	EMPLOYEECODE ENTITLEMENTYEAR	ASC ASC
EMPVACENTITLEMENT_FK	No	Yes	No	No	EMPLOYEECODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_EMPLOYEE	CODE	EMPLOYEECODE

TABLE: **WORKCALENDARTYPE**

Column List

Name	Type	P	M
CODE	integer	Yes	Yes

Name	Type	P	M
NAME	char(50)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_WORKCALENDARTYPE_PK	Yes	No	Yes	No	CODE	ASC

Reference by List

Referenced by	Primary Key	Foreign Key
T_CREW	CODE	WORKINGCALENDARTYPECODE
T_ORGANIZATIONCALENDAR_ACTUAL	CODE	WORKCALENDARTYPECODE
T_ORGANIZATIONCALENDAR_PLANNED	CODE	WORKCALENDARTYPECODE

TABLE: **WORKTEAM**

Column List

Name	Type	P	M
DEPARTMENTCODE	integer	No	Yes
CODE	integer	Yes	Yes
NAME	char(50)	No	No
DESCRIPTION	char(100)	No	No

Index List

Index Name	P	F	U	C	Column Name	Sort
T_WORKTEAM_PK	Yes	No	Yes	No	CODE	ASC
R_DEPARTMENTWORKTEAM_FK	No	Yes	No	No	DEPARTMENTCODE	ASC

Reference to List

Reference to	Primary Key	Foreign Key
T_DEPARTMENT	CODE	DEPARTMENTCODE

Reference by List

Referenced by	Primary Key	Foreign Key
T_PMCALENDAR	CODE	WORKTEAMCODE

5.4.2. Domains

<u>Table 5.3 Domain List</u>

Domain Name	Code	Data Type
d_bankaccountno	D_BANKACCOUNTNO	char(25)
d_code	D_CODE	Integer
d_description	D_DESCRIPTION	char(100)
d_name	D_NAME	char(50)
d_sex	D_SEX	char(1)
d_ssn	D_SSN	char(20)
d_username	D_USERNAME	char(8)
d_yesno	D_YESNO	char(1)

5.4.3. Physical Database Schema

Figure 5.5 shows the physical database schema diagram based on the definitions above. All tables and references between the tables are shown in the physical database model diagram. The physical model shows all actual attributes and attributes definitions (data types), primary keys and foreign keys of the tables. Indexes, key constraints, user constraints are also embedded in this model. Triggers, procedures for integrity constraints are defined within the physical model. In fact all these settings are the last step before the creation of the actual database scheme in the given DBMS environment.

We have not shown the implementation of the actual database for our schema since we are not going to include the database implementation phase in this thesis. The implementation phase would require the setup of views, database procedures, security grants, user groups and fine tuning the setting (i.e. table spaces, database log spaces) depending on the volume of the data, number of users and DBMS restrictions.

The next step of the implementation would be the development of user interfaces (applications for data processing, query and reporting). This would be, along with the implemented database schema, a management information systems for man-power planning. Another point to remember is the internal and external links of the database schema to other data sources. While some of the data would actually come from other information systems (i.e. transaction processing systems, automated data collection system), some of the data in our database would be utilized by other systems in the organizations (i.e. payroll, training department, logistics) Figure 5.6. shows the interfaces of our database systems to other possible systems. In other words the data flow from our system to other business systems in the organization and vice versa. These interfaces are shown at a high-level in the figure.

In this chapter, we have tried to implement a database model for a given problem, that we call the *man-power planning*, within a typical organization we have pictured for our model. We did not show the implementation our theoretical requirements analysis which lead us to the database requirements as a starting point for our model. We have detailed the process of entity design, E/R modeling, conceptual

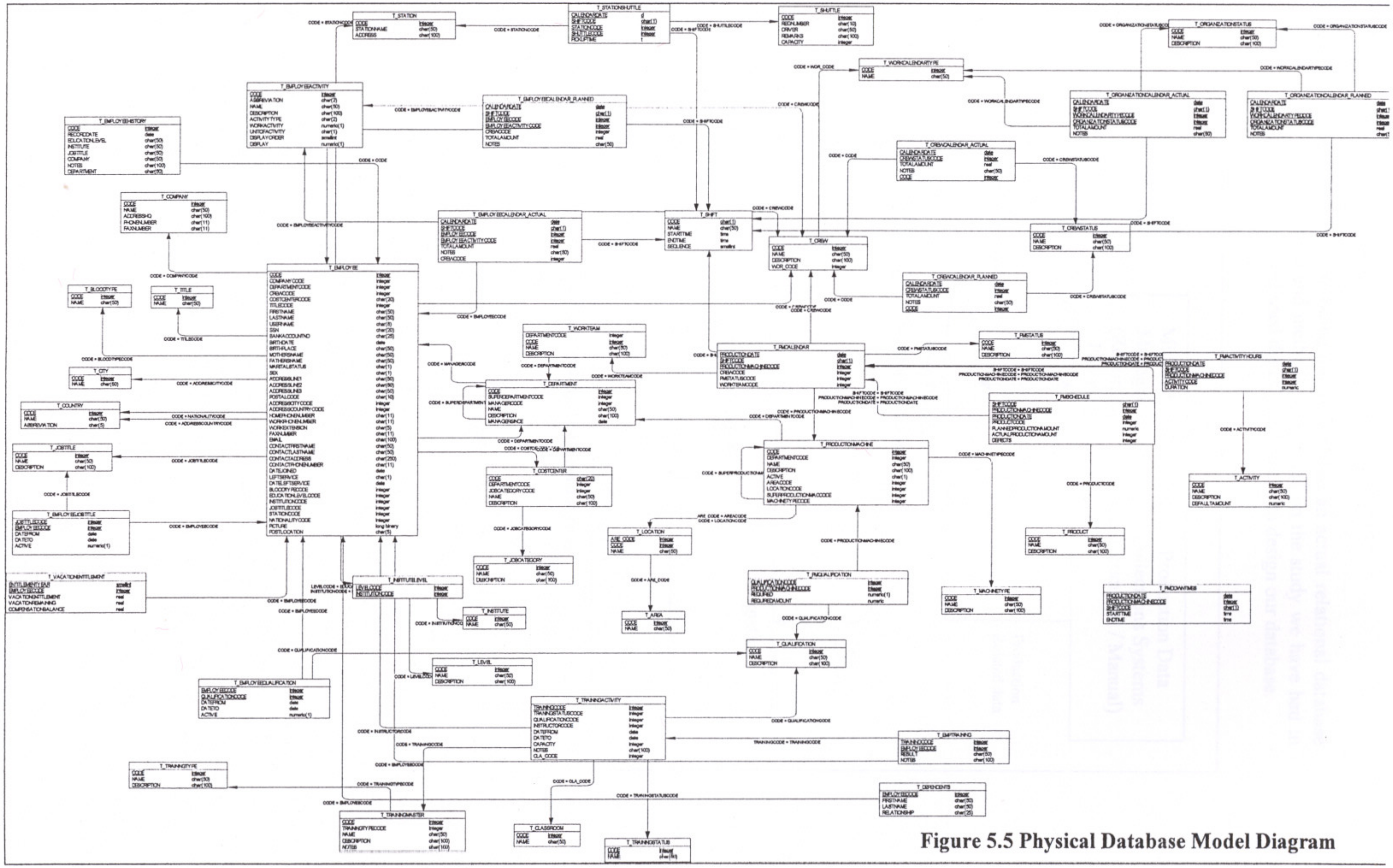


Figure 5.5 Physical Database Model Diagram

schema design and finally translating our system into an actual relational database schema. Our design criteria and methods were based on the study we have had in Chapter 3 and 4 where we have used the E/R techniques to design our database.

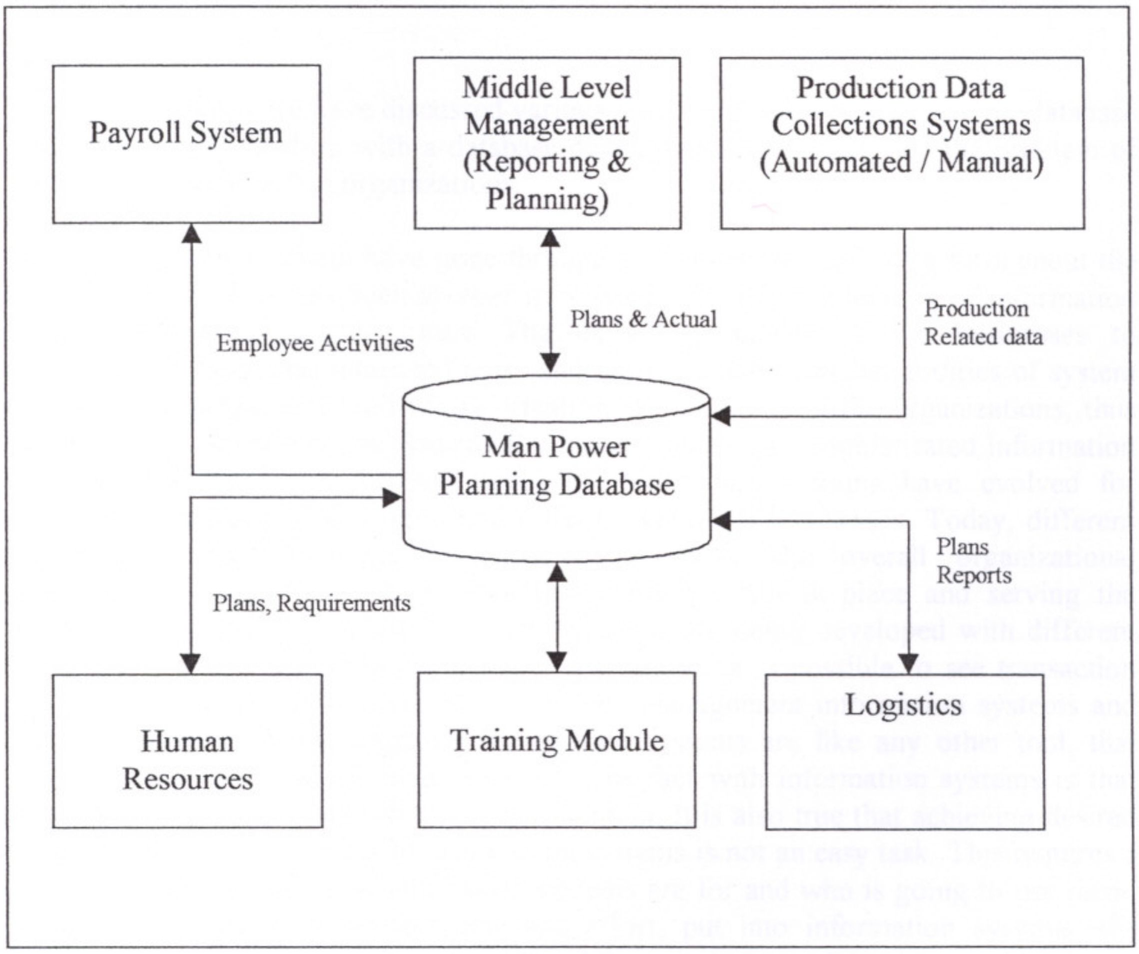


Figure 5.6 Interfaces To Other Business Systems and Areas

Chapter 6

CONCLUSION

In this study we have discussed various aspects of information systems, database systems and we ended-up with a database development case study for the problem of "*man-power planning*" in organizations.

Information system have gone through various phases and roles throughout the last fifty years. There has been an ever increasing volume of utilization of information systems within the organizations. The daily dependency of the businesses to information systems has increased more and more. As the technical abilities of system development increased so did the information requirements of the organizations, thus leading to functionally and architecturally more complex and sophisticated information systems. What is more, different types of information systems have evolved for different groups of people (users) and different nature of businesses. Today, different technological platforms work together to compose the overall organizational information system. While "old" legacy systems are still in place and serving the organizations in on way or the other, new systems are being developed with different architectures and features. In any given organizations it is possible to see transaction processing systems, office automation systems, management information systems and decision support systems together. Information systems are like any other tool, that people develop and use for their purposes. The fact with information systems is that they are difficult, and costly to build and maintain. It is also true that achieving desired targets and providing successful information systems is not an easy task. This requires a good understanding of what information systems are for and who is going to use them. Given the amount of expertise, time and effort, put into information systems, the expectancy of "*success*", "*effectiveness*" and "*user satisfaction*" is at a very high level. System developers are facing challenges from both business and technology angels.

A major component in the information systems are the databases and database management systems. A successful information systems should surely sit on a successfully implemented database system. Information systems are not just a few applications on users work stations. They process data. Data has to be well defined and organized. This has to reflect the nature of the business and the nature of the organization. Data has to be shared and available to multiple people. It has to be maintainable and accessible. Various database models have been in place since the early network model. Hierarchical model, relational model, object-relational model and finally the recent object oriented model are the various categories. Today, the most widely used database model is the relational model. Major DBMS solutions are based on relational and object-relational models. We have studied the relational model in this thesis. We have given various problems associated with relational model and also various solutions. E/R modeling, schema refinement, normalization are among the concepts we have discussed. These formed the base for our case study of database schema development.

As the case study we have worked on a database schema design and development task. This is a database model proposal for man-power planning problem. The database requirements are summarized at the start of the development process. The development process has been a demonstration of database development cycles from initial entity design to final physical schema generation. While doing this we have utilized the sophisticated design tool Sybase PowerDesigner.

It is important in any database development process to identify the entities and relationships among them. The design has to serve its purpose and should be as refined as possible. The whole schema in fact should be the answer of many questions in the problem area. In other words, a data solution for information needs of users. While the initial solution is still a manual task, using a design tool like in our case, enabled us to generate accurate, structural and well documented results. E/R model (conceptual model) was the initial solution for our problem. This is then followed by refinement and creation of the actual database schema. Every database design in one way or the other should go through an initial definition phase and modeling followed by a refinement process. Normalization, redundancy, integrity constraints have to be taken into consideration before the final product.

Man-power planning is chosen for our case study. Every organization somehow has the challenge of man-power planing. Activities around people at work are in many various nature, predictable, unpredictable, easy to plan etc. Man-power planning is associated with almost every business function in an organization. Production, training, logistics are some of the examples we can give here. In our model we have chosen a theoretical organization in manufacturing area. We have placed our employee entity at the center and stared building up the model as described in the database requirements summary. There are many enterprise resource planning tools, crew scheduling packages on the marketplace. Our development rather is based on a theoretical problem of man-power planning in a theoretical organization and is to provide a small proposal of how to solve it. and while doing so to show the well defined and documented database development process. We are not proposing any real life solution with this study. To be a real life solution, this study has to be brought further and deeper in analyzing the problem. What is more, it is not a complete solution. We have not developed any package or application. Our scope was defined as the database model development. Developing a complete package would require various analysis and decisions on technology and implementation. We have seen our solution as independent from any implementation strategy. The importance of such a customized design is the business orientation of it. Similar problems would exists in different organizations but how far the solution will go, would depend on the choices and decisions for each organizations.

Big business solution packages do have modules for human resource management, production planning, training scheduling etc. But this still does not solve the problem of achieving customized organization specific requirements quickly and easily. This is not a debate of package vs. in-house development. Along with different packages purchased organizations do have system analysts, designers and architects to implement in-house solutions. What we have done here is a very similar small approach. Only this time we have defined our organization, given it a problem and tried to solve it.

SUMMARY

Information systems study is at the center of many concepts mainly, people, software, data, communication, hardware, organization and procedures which makes information system development more than just an information technology (IT) task. Information systems are a part of an organizational solution, based on information technology, to a challenge posed by environment.

The term information systems has been defined as the effective design, delivery, use and impact of information technology in organizations and society. Another way of seeing them is that information system are systems which assemble, store, process and deliver information relevant to an organization (or to society), in such a way that the information is accessible and useful to those who wish to use it, including staff, managers, clients, citizens.

There are different architectures (types) of information systems. Defined by their functionality and processing. These are, transaction processing systems, office automation systems, knowledge work systems, management information systems, decision support systems and executive information systems. This categorization is made based on their target users and their funtionality. These different types can co-exists in organizations to built-up the overall organizational information system.

A major component of the information systems is the database system or the database management systems that provide a platform for storing, organizing and querying data by information systems. A database is a collection of data, typically describing the activities of one or more related organizations. For example, a university database might contain information about entities such as students, faculty, courses and classroom; relationships between entities, such as students enrolments in courses, faculty teaching courses and the use of classrooms for courses. A database management systems, or DBMS, is a software designed to assist in maintaining and utilizing large collections of data. Different database systems have been developed over the past years. These are network model, hierarchical model, relational / object-relation model and the recently object oriented model. The relation (and object-relational) model is the most widely used model today. Relation model is based on the relational theory in mathematics. Relations are the main entities in the relational model. Relational model has its own problems of schema refinement and normalization.

The design of a relational database schema mostly starts with the E/R modeling of the required systems. This is then followed by E/R to relational schema conversion and schema refinement leading to the final product. In real life, the full implementation of a database schema also includes various DBMS and platform dependent issues, such as security, access grants, DBMS tuning etc.

Our database development model and the process in this study stops at the point of the actual implementation. We have developed a database model for man-power planning problem, where we have used E/R method, a database design tool Sybase PowerDesigner. We have generated the physical database schema as a combination of the tool output and schema refinement .

ÖZET

Bilgi sistemleri araştırmaları farklı kavramların - insan, yazılım, veri, iletişim, donanım, organizasyon ve prosedürler- kesiştiği bir noktada bulunmaktadır. Bu nedenle de bilgi sistemleri tasarımı sadece bir teknolojik çalışma değildir. Bilgi sistemleri, organizasyonlarda, çevreden ve işin doğasından gelen problemlere ve gereksinimlere bir çözüm bulma çabasıdır.

Bilgi sistemleri, bilgi teknolojisinin etkin tasarımı, sürümü, ve kullanımı olarak görülebilir. Bir başka yaklaşım, bilgi sistemlerini veri toplayan, işleyen ve kullanıcılara faydalı ve gerekli bilgiyi üreten sistemler olarak görmektir.

Bilgi sistemlerin farklı tipleri bulunmaktadır. Bu farklılık fonksiyonel açıdan tanımlanmıştır. Farklı tipler, "transaction processing systems", "office automation systems", "management information systems","decision support systems" ve "executive information systems" şeklinde sıralanabilir. Bu farklı tipler organizasyonlarda bir arada bulunarak, organizasyonel bilgi sistemini oluşturur.

Bilgi sistemlerinin büyük bir parçası, veri tabanları ve veri tabanı yönetim sistemleridir. Veri tabanı, organizasyonel verilerin yapısal bir toplamıdır. Örneğin bir üniversite veri tabanı sistemi, ögrenciler, dersler, bölümler vs. ile ilgili verileri ve bunların arasındaki ilişkileri bulundurur. Veritabanı yönetim sistemleri, veri tabanlarının işletilmesini ve kullanımını sağlayan geniş çaplı yazılımlardır. Geçmişten bugüne, değişik veri tabanı sistemleri geliştirilmiştir. "Network model", "hierarchical model", "relational (object-relational) model" ve son dönemlerde geliştirilen "object oriented model" bu farklı sistemlerdir. "Relational model" en yagın kullanılan veri tabanı modelidir. "Relational model", "normalization" ve "schema refinement" gibi kendine özgü problemler barındırmaktadır.

"Relational" veri tabanı tasarımı E/R modellemesi ile başlar. Bunu izleyen işlem, E/R modelin "relational" bir yapıya dönüştürülmesidir. Gerçek yaşamda veri tababanı tasarımı ve uygulaması, veritabanı yönetim sisteminin seçimi ve veri güvenliği, kullanıcı tanımlanması gibi işletim ortamına bağlı seçeneklerin tanımlanmasını gerektirir.

Bu çalışmada verilen veri tabanı modeli tasarımı, gerçek veri tabanının uygulanması noktasında bırakılmıştır. Tasarladığımız veri tabanı modeli, insan gücü planlamasına yönelik bir çalışmadır. Bu tasarım sırasında, E/R modelleme yöntemi ve veri tabanı tasarım paketi Sybase PowerDesigner kullanılmıştır. Çalışmanın sonunda fiziksel veri tabanı modeli üretimiştir.

BIBLIOGRAPHY

American National Dictionary for Information Processing, Washington, DC : Computer and Business Equipment Manufacturers Association (CBEMA), Report No. X3/Tr-1-77, 1977 September.

Avison D.E, and Fitzgerald, G., *Information Systems Development : Methodologies, Techniques and Tools*, Second Edition, McGraw-Hill Companies, 1995

Bartholomew, D.J., Andrew F. F., *Statistical Techniques for Man Power Planning,* Wiley Series, John Wiley & Sons, 1979

Chanliau M., Power Designer 7.0. The Next Generation., White Paper, URL: www.sybase.com/, 1999.

Chen P. P., The Entity-Relationship Model - Toward a Unified View of Data, ACM Transactions on Database Systems, Volume 1, Number 1, March 1976, p.9.

Codd E F., 1970, A Relational Model for Large Shared Databanks, Communications of the ACM, Volume 13, Number 6, p.377-390, June 1970.

Codd E. F., *The Relational Model for Database Management: Version 2*, Addison-Wesley Publishing Company, 1990.

Curtis Graham, *Business Information Systems Analysis Design and Practice*, Third Edition, Addison Wesley Longman Ltd., 1998

Danielsen Asbjørn, *The Evolution Of Data Models And Approaches To Persistence In Database Systems,* M Sc. Essay, University of Oslo, Department of Informatics, 1998.

Department of Accounting and Business Law of the James J. Nance College of Business Administration at Cleveland State University, URL: http://www.csuohio.edu/accounts/

Elmasri R., Navathe S., *Fundamentals of Database Systems,* 2nd Edition, The Benjamin / Cummins Publishing Company, 1994

Everest C. Gordon, *Database Management Objectives, System Function & Administration*, international student edition, McGraw-Hill Book Company, 1986

Farris M,. Sybase Development Tools' "Best Kept" Secret, White Paper, URL : www.sybase.com, 1999

Fertuck L., *System Analysis and Design With Modern Methods*, Dubuque : Business & educational Technologies/Wm.C.Brown, 1995

Gordan B. Davis and Margrethe H. Olson, *Management Information Systems: Conceptual Foundations, structures, and Development*, second edition, McGraw-Hill Book Company, 1985

Grimes Set, Modeling Object/Relational Databases, DBMS, Volume 11, Number 4, April 1998, p.51

Han, J., Database Systems and Structures, Lecture Notes, *CPMT354*, *1995*, School of Computer Science, Simon Fraser University, URL: http://www.csuohio.edu

Hart Dennis & Toomey Warren, History of Computer and Information Systems, URL: http://www.cs.adfa.edu.au/teaching/studinfo/csis/Lectures/topic3.html

Hirschheim Rudy, A Comparison of Five Alternative Approaches to Information Systems Development, Australian Journal of Information Systems Volume 5, Number 1, 1997

Kanellakis P.C., Elements of Relational Database Theory, Technical Report CS-89-39, Brown University, Department of Computer Science, 1989

Kelly Floyd, Implementing an EIS (Executive Information System), EIS References, (Watson & Rainer, 1991), URL: http://www.ceoreview.com/papers/eis.htm

King Nelson H., Object DBMSs : Now or Never, DBMS Online, Volume 10, Number 8, July 1997, p.62.

Kroenke David, *Management Information Systems*, McGraw-Hill Book Company, 1989

Looney Kevin, Koch George, *Orace 8I: The Complete Reference,* Osborne/McGraw-Hill, 2000

Munshi J., A Framework for MIS Effectiveness, Working Paper, For presentation to the Academy of Business Administration, Athens, International Conference, July 1996, p.1

National Institute of Standards and Technology (NIST) SQL Project, URL: http://www.nist.gov/

Ozkarahan Esen, *Database Management. Concepts, Design and Practice*, Prentice-Hall International Editions, 1990

Post V. Gerald, *Database Management Systems, Designing and Building Business Applications*, Irwin McGraw-Hill, p.3, 1999

Ramakrishnan Raghu, Gehrke Johannes, *Database Management Systems*, Second Edition, McGraw-Hill Higher Education, 2000.

Rennhackkamp Martin, Extending Relational DBMSs, DBMS, Volume 10, Number 13, December 1997, p. 45

Sommerville Ian, *Software Engineering*, Fifth Edition, Addison-Wesley Publishing Company, 1996

Sol Selena, Network Databases, Web Developers Virtual Library, 16 August 1998, URL: ttp://www.stars.com/Authoring/DB/

Terrance Hanold, An Executive View of MIS, Datamation (18:11), 1972 November, p. 66.

The New Oxford Dictionary of English, Oxford University Press, Oxford, New York, 1998, page 468

The Rise of Relational Databases, *Funding a Revolution: Government Support for Computing Research,* National Research Council Report, National Academy Press, Washington, D.C. 1999, Part-II Case Studies in Computing Research.

Topics Reading on Database Management Systems by A.A. Verstraete, Revised: May 22,1998, URL: http://misweb.smeal.psu.edu/database/

Uludag Memet, 00 Yilina Hazirmisiniz?, TMMOB Elektrik Muhendisleri Odasi Izmir Subesi Bulteni, Yil: 11, Sayi:110, Haziran, p.20, 1999

University of Pennsylvania School of Engineering & Applied Science, http://www.seas.upenn.edu:8080/~museum/overview.html

University of Wolverhampton, The School of Computing and Information Technology (SCIT), *CP4011 Database Concepts and Techniques,* URL: http://scitsc.wlv.ac.uk/

Watson R.T., *Data Management : databases and organizations,* Second Edition, New York Chichester, Wiley, 1999

Webster's New World Dictionary, College Edition, Toronto, Canada, 1962

Zdonik S., What Makes Object-Oriented Database Management Systems Different, Advances in Object-Oriented Database Systems, NATO ASI Series, Series F: Computer and System Science, Vol. 130, 3-26, Springer Verlag, Berlin Heidelberg New York, 1994