CATHOLIC UNIVERSITY OF RWANDA

P.O. Box 49 BUTARE / HUYE - RWANDA
TEL: (00250) 0252 530 893 FAX: (00250) 0252 530 627
E-mail: administration@cur.ac.rw
Web: www.cur.ac.rw

FACULTY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

MODULE OF DATABASE MANAGEMENT AND INFORMATION SYSTEMS

'SEMESTER I'

LEVEL ONE CS

DMIS2512,CREDITS: 15, HOURS: 75

ACADEMIC YEAR 2018-2019

LECTURER: NSENGIYUMVA Jean Marie Vianney

Master of Science in Information Systems

Email : nsengiyumva@gmail.com

LEARNING OUTCOMES

LEARNING AND TEACHING STRATEGIES

Lectures

Practical works

Group & Individual work

Assignments

Practical works

Contents

PART ONE: COMPONENT OF DATABASE MANAGEMENT SYSTEMS.

CHAPTER I: INTRODUCTION

1.1. Database

Databases and database technology have a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, electronic commerce, engineering, medicine, genetics, law, education, and library science. The word *database* is so commonly used that we must begin by defining what a database is.

A **database** is a collection of related data. By **data**, we mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know. You may have recorded this data in an indexed address book or you may have stored it on a hard drive, using a personal computer and software such as Microsoft Access or Excel. This collection of related data with an implicit meaning is a database.

The preceding definition of database is quite general; for example, we may consider the collection of words that make up this page of text to be related data and hence to constitute a database. However, the common use of the term *database* is usually more restricted. A database has the following implicit properties:

■ A database represents some aspect of the real world, sometimes called the **mini world** or the **universe of discourse (UoD)**. Changes to the mini world are reflected in the database.

■ A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database.

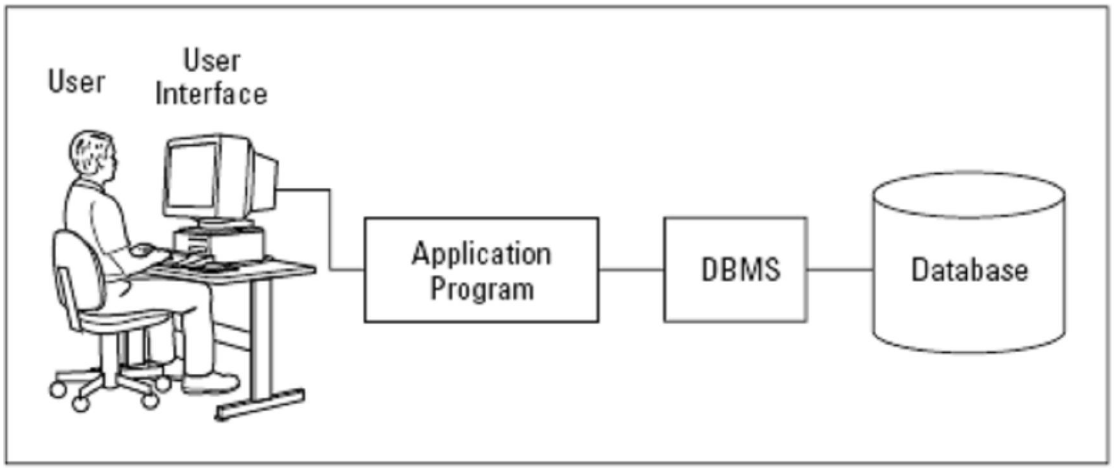
■ A database is designed, built, and populated with data for a specific purpose.

It has an intended group of users and some preconceived applications in which these users are interested. In other words, a database has some source from which data is derived, some degree of interaction with events in the real world, and an audience that is actively interested in its contents. The end users of a database may perform business transactions (for example, a customer buys a camera) or events may happen (for example, an employee has a baby) that cause the information in the database to change. In order for a database to be accurate and reliable at all times, it must be a true reflection of the miniworld that it represents; therefore, changes must be reflected in the database as soon as possible.

1.2 A database management system

A database management system *(DBMS)* is a set of programs used to define, administer, and process databases. A DBMS is the tool you use to build that structure and operate on the data contained within the database. The DBMS is a *general-purpose software system* that facilitates the processes of *defining, constructing, manipulating,* and *sharing* Databases among various users and applications. **Defining** a database involves specifying the data types, structures, and constraints of the data to be stored in the database. The database definition or descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called **meta-data**. **Constructing** the database is the process of storing the data on some storage medium that is controlled by the DBMS. **Manipulating** a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data. **Sharing** a database allows multiple users and programs to access the database simultaneously. An **application program** accesses the database by sending queries or requests for data to the DBMS. Many DBMS programs are on the market today. Some run only on mainframe computers, some only on minicomputers, and some only on personal computers. Whatever the size of the computer that hosts the database and regardless of whether the machine is connected to a network the flow of information between database and user is the same.

Figure 1.1 Block diagram of a DBMS based information system



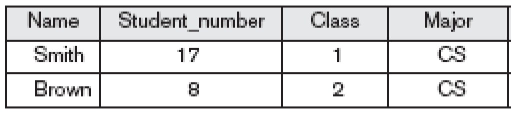
1.2.1 A query

A **query** typically causes some data to be retrieved; a **transaction** may cause some data to be read and some data to be written into the database. Other important functions provided by the DBMS include *protecting* the database and *maintaining* it over a long period of time. **Protection** includes *system protection* against hardware or software malfunction (or crashes) and *security protection* against unauthorized or malicious access. A typical large database may have a life cycle of many years, so the DBMS must be able to **maintain** the database system by allowing the system to evolve as requirements change over time. It is not absolutely necessary to use general-purpose DBMS software to implement a computerized database. We could write our own set of programs to create and maintain the database, in effect creating our own *special-purpose* DBMS software. In either case whether we use a general-purpose DBMS or not—we usually have to deploy a considerable amount of complex software. In fact, most DBMSs are very complex software systems. To complete our initial definitions, we will call the database and DBMS software together a **database system**.

An Example of a database system

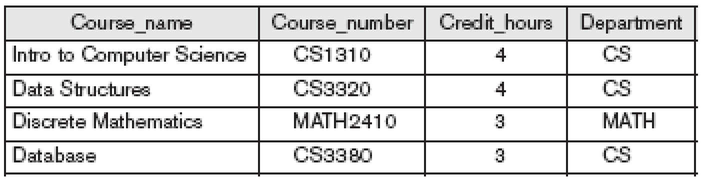
Let us consider a simple example that most readers may be familiar with: a UNIVERSITY database for maintaining information concerning students, courses,and grades in a university environment. Figure 1.2 shows the database structure and a few sample data for such a database. The database is organized as five files, each of which stores **data records** of the same type.3 The STUDENT file stores data on each student, the COURSE file stores data on each course, the SECTION file stores data on each section of a course, the GRADE_REPORT file stores the grades that students receive in the various sections they have completed, and the PREREQUISITE file stores the prerequisites of each course. To *define* this database, we must specify the structure of the records of each file by specifying the different types of **data elements** to be stored in each record. In tables below, each STUDENT record includes data to represent the student’s Name, Student_number, Class (such as freshman or ‘1’, sophomore or ‘2’, and so forth), and Major (such as mathematics or ‘MATH’ and computer science or ‘CS’); each COURSE record includes data to represent the Course_name, Course_number, Credit hours, and Department (the department that offers the course); and so on. We must also specify a **data type** for each data element within a record. For example, we can specify that Name of STUDENT is a string of alphabetic characters, Student number of STUDENT is an integer, and Grade of GRADE_REPORT is a single character from the set {‘A’, ‘B’, ‘C’, ‘D’, ‘F’, ‘I’}.We may also use a coding scheme to represent the values of a data item. For example, in Figure 1.2 we represent the Class of a STUDENT as 1 for freshman, 2 for sophomore, 3 for junior, 4 for senior, and 5 for graduate student. To *construct* the UNIVERSITY database, we store data to represent each student, course, section, grade report, and prerequisite as a record in the appropriate file. Notice that records in the various files may be related. For example, the record for Smith in the STUDENT file is related to two records in the GRADE_REPORT file that specify Smith’s grades in two sections. Similarly, each record in the PREREQUISITE file relates two course records: one representing the course and the other representing the prerequisite. Most medium-size and large databases include many types of records and have *many relationships* among the records. Let’s see the tables below

STUDENT



Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE



Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

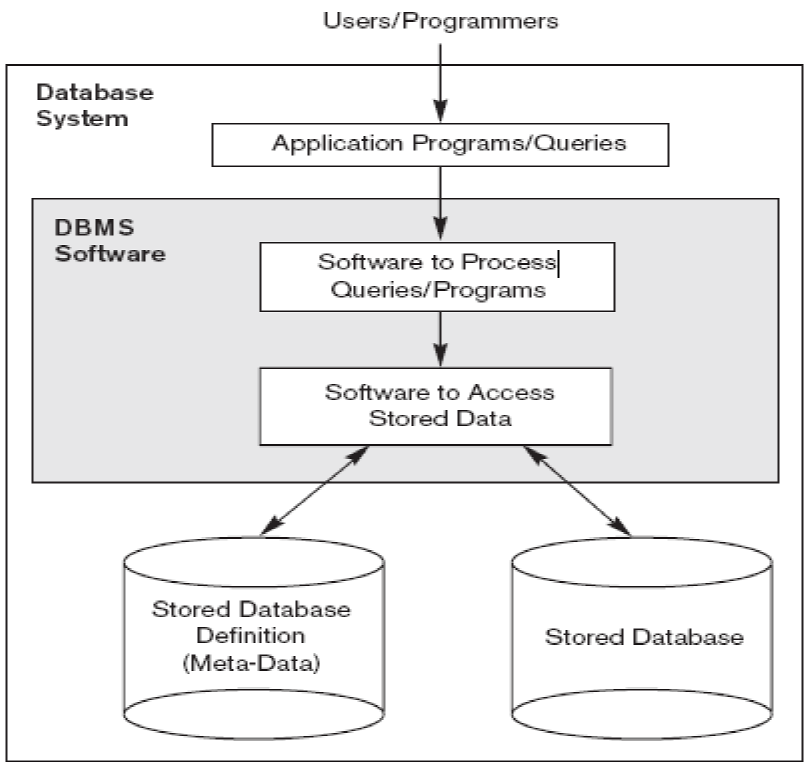
GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2. A simplified database system environment.


Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

The database itself can be regarded as a kind of repository or container for a collection of computerized data files. Users of the system can perform a variety of operations on such files for example:

- ✓ adding new files to the database
- ✓ inserting data into existing files
- ✓ retrieving data from existing files
- ✓ changing data in existing files
- ✓ deleting data from existing files
- ✓ removing existing files from the database

- ▪ The goal of a DBMS is to provide an environment that is both **convenient** and **efficient** to use in
  - o Retrieving information from the database.
  - o Storing information into the database.
- ▪ Databases are usually designed to manage **large** bodies of information. This involves
  - – Definition of structures for information storage (data modeling).
  - – Provision of mechanisms for the manipulation of information (file and systems structure, query processing).
  - – Providing for the safety of information in the database (crash recovery and security).
  - – Concurrency control if the system is shared by users.
  - – Help to solve the problem of data, redundancy and inconsistency.

1.3 Database Applications

- ✓ Banking: all transactions
- ✓ Airlines: reservations, schedules
- ✓ Universities: registration, grades
- ✓ Sales: customers, products, purchases
- ✓ Online retailers: order tracking, customized recommendations
- ✓ Manufacturing: production, inventory, orders, supply chain

- ✓ Human resources: employee records, salaries, tax deductions
- ✓ Telecommunication: for keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards and storing information about the communication network.

1.4 History of Database Systems

- **1950s and early 1960s:**
  - ➢ Data processing using magnetic tapes for storage (Tapes provide only sequential access)
- **Late 1960s and 1970s:**
  - ➢ Hard disks allow direct access to data.
  - ➢ Network and hierarchical data models in widespread use.

- **1980s:**

- ➢ Parallel and distributed database systems

- **1990s:**

- ➢ Large multi-terabyte data warehouses
- ➢ Emergence of Web commerce

3. Basic Definitions

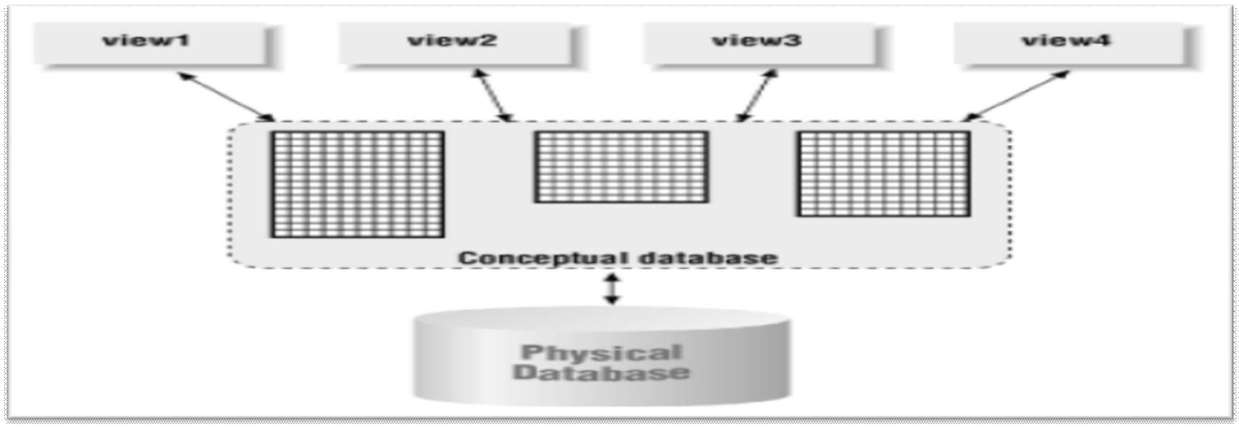
- ✓ Database: A collection of related data.
- ✓ Data: Known facts that can be recorded and have an implicit meaning.
- ✓ Database Management System (DBMS): A software package/ system to facilitate the creation and maintenance of a computerized database.
- ✓ **Physical level:** describes how a record (e.g., customer) is stored.
- ✓ **Logical level:** describes data stored in database, and the relationships among the data.
- ✓ **Data Model**: A set of concepts to describe the *structure* of a database

- ✓ **Data Model Operations**: Operations for specifying database retrievals and updates by referring to the concepts of the data model. Operations on the data model may include **basic operations and user-defined operations.**
- ✓ **View level:** A way to hide: (a) details of data types and (b) information (such as an employee's salary) for security purposes.
- ✓ **Tables:** alternatively known as a **relation**, is a two dimensional structure used to hold related information. A database consists of one or more related tables.
- ✓ **Rows:** in a table is a collection or instance of one thing, such as one employee or one line item on an invoice
- ✓ **Columns:** A **column** contains all the information of a single type, and the piece of data at the intersection of a row and a column,
- ✓ **Primary Keys:** The primary key is an attribute (or attributes) that uniquely identifies every record in the relation. This is often a unique identification number, such as an employee ID number or a serial number.
- ✓ **Foreign Keys:** A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables

Data types: A data type in a programming language is a set of data with values having predefined characteristics.

1.5 An architecture for a database system.

A *database system is often pictured as a three-level structure*



The three-level structure of a database system

(i) **Physical database**

- ✓ This is the lowest level of the structure
- ✓ At this level, the data have no logical meaning, as related to the database.
- ✓ The physical level of a database is often referred to as the *internal level.*

(ii) Conceptual database

- ✓ It gives the data a *logical structure.*
- ✓ The data are viewed as a collection of tables, with column headings describing the attributes of the corresponding entity class
- ✓ The conceptual model is intended to model the entire database. However, individual users may be interested in views of only specific portions of the data.

(iii) Views

- ✓ The highest level in the three-tier structure that may be held by users of the database and they are also referred to as *external level.*
- ✓ A user does not need to know anything about database programming to create a database in Microsoft Access, although he or she does need to have a familiarity with the *conceptual level of a relational database.*

1.6. Database Designers

Database designers are responsible for identifying the data to be stored in the database
and for choosing appropriate structures to represent and store this data. These tasks are mostly undertaken before the database is actually implemented and populated with data. It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements. In many cases, the designers are on the staff of the DBA and may be assigned other staff responsibilities after the database design is completed. Database designers typically interact with each potential group of users and develop **views** of the database that meet the data and processing requirements of these groups. Each view is then analyzed and

integrated with the views of other user groups. The final database design must be capable of supporting the requirements of all user groups.

1.7 End Users

End users are the people whose jobs require access to the database for querying, updating, and generating reports; the database primarily exists for their use. There are several categories of end users:

■ **Casual end users** occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests and are typically middle- or high-level managers or other occasional browsers.

■ **Naive** or **parametric end users** make up a sizable portion of database end users. Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates called **canned transactions** that have been carefully programmed and tested. The tasks that such users perform are varied:

_ Bank tellers check account balances and post withdrawals and deposits.

_ Reservation agents for airlines, hotels, and car rental companies check availability for a given request and make reservations.

■ Employees at receiving stations for shipping companies enter package identifications via bar codes and descriptive information through buttons to update a central database of received and in-transit packages.

■ **Sophisticated end users** include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirements.

■ **Standalone users** maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces. An example is the user of a tax package that stores a variety of personal financial data for tax purposes. A typical DBMS provides multiple facilities to access a database.

1.8 Advantages of Using the DBMS Approach

- **Controlling Redundancy**

In traditional software development utilizing file processing, every user group maintains its own files for handling its data-processing applications. For example, consider the UNIVERSITY database example of Section 1.2; here, two groups of users might be the course registration personnel and the accounting office. In the traditional approach, each group independently keeps files on students. The accounting office keeps data on registration and related billing information, whereas the registration office keeps track of student courses and grades. Other groups may further duplicate some or all of the same data in their own files.

- **Restricting Unauthorized Access**

When multiple users share a large database, it is likely that most users will not be authorized to access all information in the database. For example, financial data is often considered confidential and only authorized persons are allowed to access such data. In addition, some users may only be permitted to retrieve data, where as others are allowed to retrieve and update. Hence, the type of access operation retrieval or update must also be controlled. Typically, users or user groups are given account numbers protected by passwords, which they can use to gain access to the database. A DBMS should provide a **security and authorization subsystem**, which the DBA uses to create accounts and to specify account restrictions. Then, the DBMS should enforce these restrictions automatically. Notice that we can apply similar controls to the DBMS software. For example, only the dba's staff may be allowed to use certain **privileged software**, such as the software for creating new accounts. Similarly, parametric users may be allowed to access the database only through the predefined canned transactions developed for their use.

- **Providing Backup and Recovery**

A DBMS must provide facilities for recovering from hardware or software failures.The **backup and recovery subsystem** of the DBMS is responsible for recovery. For example, if the computer system fails in the middle of a complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing. Alternatively, the recovery subsystem could ensure that the transaction is resumed from the point at which it was interrupted so that its full effect is recorded in the database. Disk backup is also necessary in case of a catastrophic disk failure.

- **Providing Multiple User Interfaces**

Because many types of users with varying levels of technical knowledge use a database,a DBMS should provide a variety of user interfaces. These include query languages for casual users, programming language interfaces for application programmers, forms and command codes for parametric users, and menu-driven interfaces and natural language interfaces for standalone users. Both forms-style interfaces and menu-driven interfaces are commonly known as **graphical user interfaces (GUIs)**. Many specialized languages and environments exist for specifying GUIs. Capabilities for providing Web GUI interfaces to a database or Web enabling a database are also quite common.

- **Representing Complex Relationships among Data**

A database may include numerous varieties of data that are interrelated in many ways. Consider The record for 'Brown' in the STUDENT file is related to four records in the GRADE_REPORT file. Similarly, each section record is related to one course record and to a number of GRADE_REPORT records one for each student who completed that section. A DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve and update related data easily and efficiently.

- **Enforcing Integrity Constraints**

Most database applications have certain **integrity constraints** that must hold for the data. A DBMS should provide capabilities for defining and enforcing these constraints. The simplest type of integrity constraint involves specifying a data type for each data item.

1.9 DBMS Languages.

(i) **Data Definition Language** (**DDL**): Used to specify the *database schema*. In many DBMSs, the DDL is also used to define internal and external schemas (views).

Example: **create table** *account* (*account-number* **char**(10) *balance* **integer**).

(ii) Data Manipulation Language (**DML**):

- A DBMS must also provide a language designed to manipulate the data in a database. This language is called a *database manipulation language, or DML.*
- As you can see even from this small example, the DML is designed to perform a variety of actions, such as:

- ➢ Moving through the data in the database
- ➢ Adding data to the database
- ➢ Editing or updating data in the database
- ➢ Deleting data from the database.
- ➢ Querying the data and returning those portions of the data that satisfy the query

CH AP 2 RELATIONAL DATABASE DESIGN

- ✓ The key to understanding the database design process lies in understanding the way a relational database management system (tool), such as SQL, Microsoft Access etc store data.

- ✓ To efficiently and accurately provide you with information, tool needs to have the facts about different subjects stored in separate tables, for example, you might have one table that stores only facts about employees, and another that stores only facts about sales.
- ✓ A relational database is based on the relational data model.
- ✓ Data and relationships among the data is represented by a collection of tables.
- ✓ Includes both a DML and a DDL.
- ✓ Most commercial relational database systems employ the **SQL** query language.

2.1 Data Modeling

- In data modeling, the developer conceptualizes and documents all the tables for the database.
- One of the common methods for modeling a database is called ERA, which stands for entities, relationships, and attributes
- The database designer uses an application that can maintain entities, their attributes, and their relationships
- In general, an entity corresponds to a table in the database, and the attributes of the entity correspond to columns of the table
- The data-modeling process involves defining the entities, defining the relationships between those entities, and then defining the attributes for each of the entities.

- Once a cycle is complete, it is repeated as many times as necessary to ensure that the designer is capturing what is important enough to go into the database

2.2 Data-Modeling Process

- ✓ **Defining the Entities:** First, the designer identifies all of the entities within the scope of the database application. The entities are the persons, places, or things that are important to the organization and need to be tracked in the database.
- ✓ **Defining the Relationships between Entities:** Once the entities are defined, the designer can proceed with defining how each of the entities is related. Often, the designer will pair each entity with every other entity and ask, "Is there a relationship between these two entities?" Some relationships are obvious; some are not.

2.3 Three types of relationships in a relational database

2.3.1 One-to-one

One-to-one: In **one-to-one** relationship, a row in a table is related to only one or none of the rows in a second table. These relationships are not as common as one-to-many relationships, because if one entity has an occurrence for a corresponding row in another entity, in most cases, the attributes from both entities should be in a single entity.

2.3.2 One-to-many

One-to-many: The most common type of relationship is one-to-many. This means that for each occurrence in a given entity, the parent entity, there may be one or more occurrences in a second entity, the child entity, to which it is related.

2.3.3 Many-to-many

Many-to-many: In a many-to-many relationship, one row of a table may be related to many rows of another table, and vice versa. Usually, when this relationship is implemented in the database, a third entity is defined as an intersection table to contain the associations between the two entities in the relationship. For example, in a database used for school class enrollment, the STUDENT table has a many-to-many relationship with the course table—one student may take one or more courses, and a given course may be pursued by one or more students. The intersection table STUDENT_COURSE would contain the combinations of STUDENT and COURSE to track which students pursue which courses.

- **Assigning Attributes to Entities:** Once the designer has defined the entity relationships, the next step is to assign the attributes to each entity. This is physically implemented using columns

2.4 Data Modeling Using the Entity-Relationship (ER) Model

Outline

- Example of a database company
- ER Model Concepts

- ➢ Entities and Attributes
- ➢ Relationships and Relationship Types
- ➢ Roles and Attributes in Relationship Types

- ER Diagrams - Notation
- ER Diagram for COMPANY Schema

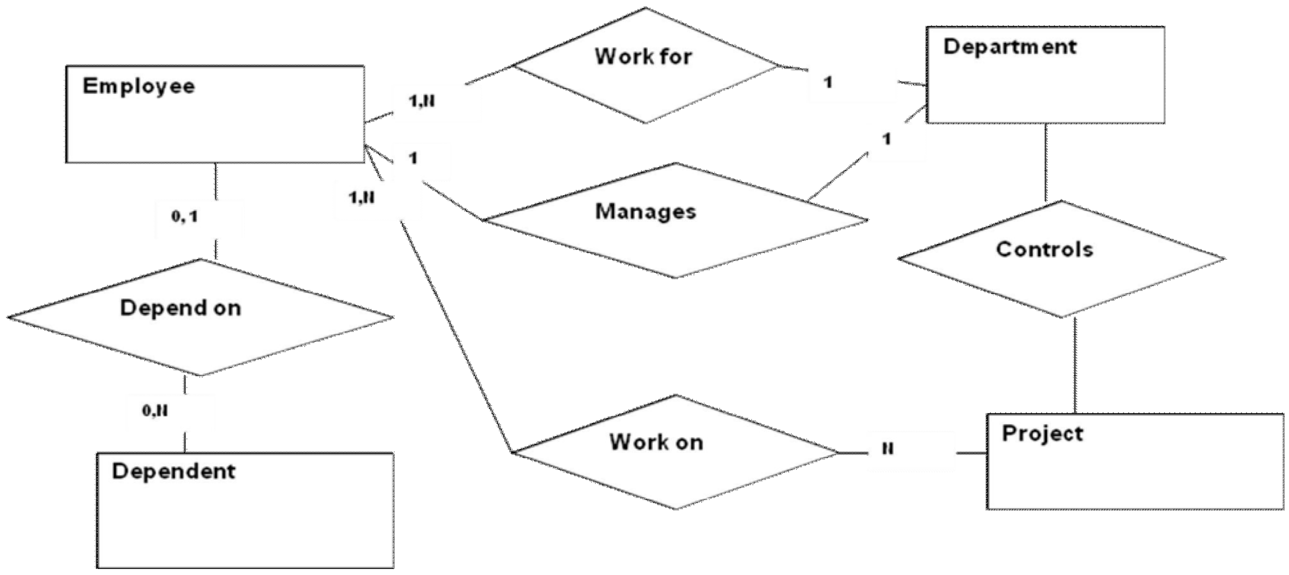
2.4.1 Example of a database company

- ➢ The company is organized into DEPARTMENTs. Each department has an employee who *manages* the department. Each department *controls* a number of PROJECTs.
- ➢ One or more employees *work for* one department and may also *work on* several projects.
- ➢ Each employee may *have* a number of DEPENDENTs. For each dependent, we keep track of their name, sex, birthdate, and relationship to employee.

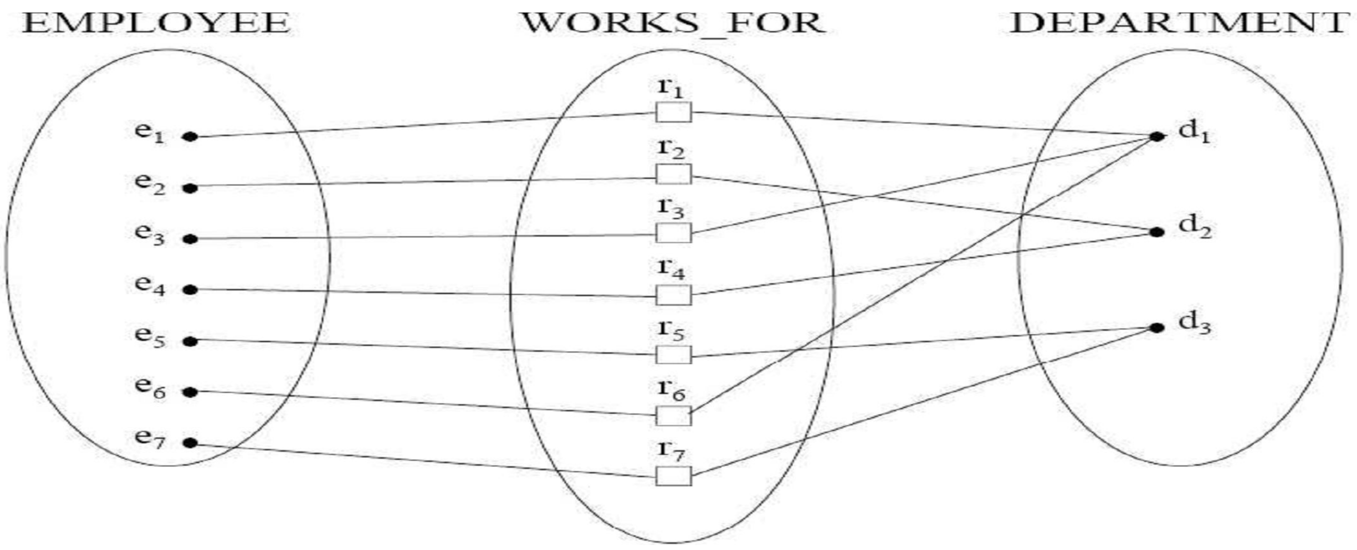
2.4.2 ER Model Concepts

(i) Entities and Attributes

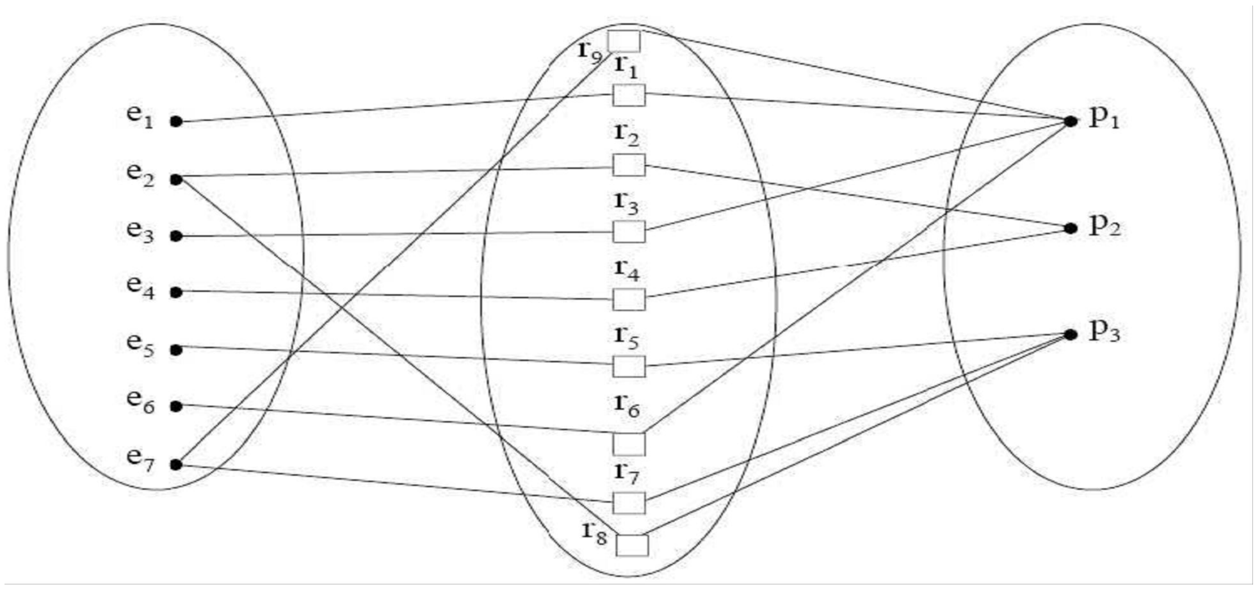
- Entities are specific objects or things that are represented in the database for example the EMPLOYEE, DEPARTMENT etc.
- Attributes are used to describe an entity. For example an EMPLOYEE entity may have a Name, SSN, Address, Sex, BirthDate. Each attribute has a *value set* (or data type) associated with it – e.g. integer, string,…
- Entity Relationship Diagram (ERD) of Employee, Department, Project and Dependent.



Example relationship instances of the WORKS_FOR relationship between EMPLOYEE and DEPARTMENT.



Example relationship instances of the WORK_FOR relationship between EMPLOYEE and PROJECT



<u>**Exercise**</u>

- The INES Ruhengeri has several departments and each department has a supervisor and at least one employee. Employees must be assigned to at least one department.

  Required:

(i) Show the important tables/entities

(ii) Show the type of relationship between tables/entities

(iii) Draw the Entity Relationship Diagram (ERD).

2.5. STEPS TO DESIGN A DATABASE

STEP 1: DETERMINE THE PURPOSE OF THE DATABASE

- The first step in designing an SQL database is to determine the purpose of the database and how it's to be used.
- This tells you what information you want from the database.
- From that, you can determine what subjects you need to store facts about (the tables) and what facts you need to store about each subject (the fields in the tables)
- Talk to the people who will use the database.
- Brain storm about the questions you'd like the database to answer.
- Sketch out the reports you'd like it to produce.
- Gather the forms you currently use to record your data.
- You'll use all this information in the remaining steps of the design process.

Step 2: Determine the tables you need.

Once you have a clear purpose for your database, you can divide your information into separate subjects, such as "Employees" or"Orders" Each subject will be a table in your database.

Step 3: Determine the fields you need.

- ✓ Decide what information you want to keep in each table.
- ✓ Each category of information in a table is called a *field* and is displayed as a column in the table.
- ✓ For example, one field in an Employees table could be “Last Name”; another could be “Date”

step 4: Determine the primary key fields

- ✓ The power in a relational database management system such as SQL comes from its ability to quickly find and bring together information stored in separate tables.
- ✓ The primary key is an attribute (or attributes!) that uniquely identifies every record in the relation.
- ✓ In order for SQL/MS-Access to work most efficiently, each table in your database should include a field or set of fields that uniquely identifies each individual record stored in the table.
- ✓ This is often a unique identification number, such as an employee ID number or a serial number.
- ✓ In database terminology, this information is called the *primary key* of the table.
- ✓ SQL/MS-Access uses primary key fields to quickly associate data from multiple tables and bring them together for you.
- ✓ SQL/MS Access doesn't allow duplicate values in a primary key field.
- ✓ For example, don't use people's names as a primary key, because names aren't unique.
- ✓ You could easily have two people with the same name in the same table.

If you don't already have a unique identifier in mind for a table, you can use a field that simply numbers the records consecutively.

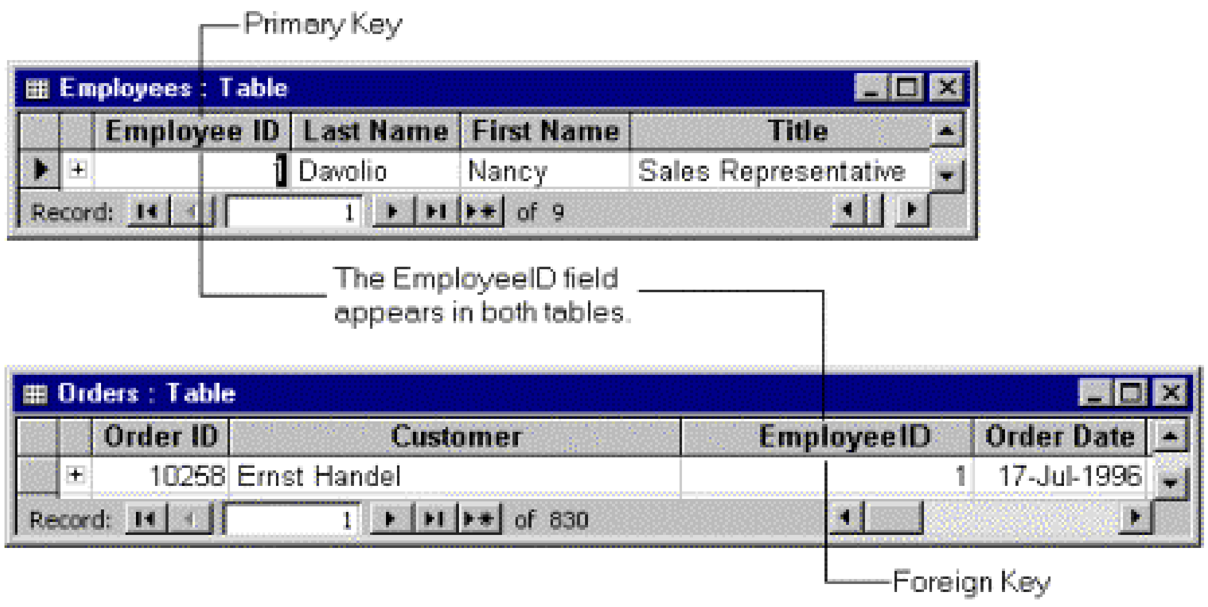
When choosing primary key fields, keep these points in mind:

- ✓ SQL/MS Access doesn't allow duplicate values in a primary key field.
- ✓ You may use the value in the primary key field to look up records, so it shouldn't be too long to remember or type. You may want it to have a certain number of letters or digits, or be in a certain range.
- ✓ The size of the primary key affects the speed of operations in your database. When you create primary key fields, you can set a property to limit the size of the field.

step 5: Determine the relationships

- ✓ Now that you've divided your information into tables, you need a way to tell SQL/ Microsoft Access how to bring it back together again in meaningful ways.
- ✓ SQL/Microsoft Access is a *relational* database management system.
- ✓ That means you store related data in separate tables.
- ✓ Then you define relationships between the tables, and Microsoft Access uses the relationships to find associated information stored in your database.

Relationship between two table (Employee & order).



Employees : Table

Employee ID	Last Name	First Name	Title
1	Davolio	Nancy	Sales Representative

Record: 1 of 9

Orders : Table

Order ID	Customer	EmployeeID	Order Date
10258	Ernst Handel	1	17-Jul-1996

Record: 1 of 830

✓ So, to set up a relationship between two tables -- Table A and Table B -- you add one table's **primary key** to the other table, so that it appears in both tables.

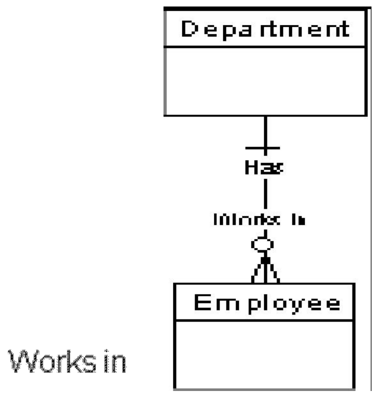
✓ But how do you decide which table's primary key to use? To set up the relationship correctly, you must first determine the nature of the relationship.

There are three types of relationships between tables:

- ✓ One-to-many relationships
- ✓ Many-to-many relationships
- ✓ One-to-one relationships

Creating a One-to-Many Relationship

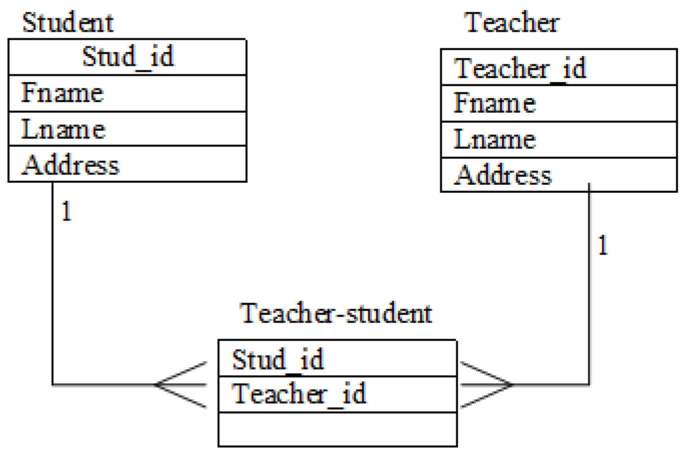
- ✓ A one-to-many relationship is the most common type of relationship in a relational database
- ✓ A relationship between tables is usually one-to-many affair.
- ✓ In a one-to-many relationship, a record in Table A can have more than one matching record in Table B, but a record in Table B has *at most* one matching record in Table A.
- ✓ For example a team has many players while players can play only for one team."
- ✓ For example, the Suppliers and Products tables in the database have a one-to-many relationship as supplier can supply many products.
- ✓ For example, one class has many students assigned to it. But what if you need to assign MANY students to MANY classes?
- ✓ To accomplish this, you can add a third table that acts as a go-between/junction table and handles all the possible combinations involved.
- ✓ To set up the relationship, you add the field or fields that make up the primary key on the "one" side of the relationship to the table on the "many" side of the relationship.
- ✓ In this case, you would add the Supplier ID field from the Suppliers table to the Products table, because *one* supplier supplies *many* products. SQL/Microsoft Access uses the supplier ID number to locate the correct supplier for each product.
- ✓ Alternative example, EMPLOYEES work in one DEPARTMENT; a DEPARTMENT has many EMPLOYEEs.
- ✓ A One-to-Many relationship is shown on the diagram by a line connecting the two entities with *crows feet* symbol denoting the "many" end of the relationship.
- ✓



Creating Many-to-Many Relationships

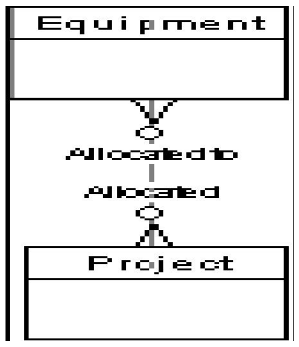
- ✓ This type of relationship takes place when many occurrences of an entity are related to many occurrences of the second entity and vice-versa
- ✓ In order to support a many-to-many dimension relationship, a primary key–foreign key relationship must be defined in the data source view between all the tables that are involved.
- ✓ Otherwise, you will not be able to select the correct intermediate measure group.
- ✓ Well imagine you are keeping track of teachers and the students in their classes at a college, where students have many teachers.
- ✓ For each student there are many teachers and for each teacher there are many students. This is a many to many relationship.

This type of relationship cannot be achieved just by linking the two tables. You need to use a joining/junction table.



- ✓ For example, EQUIPMENT is allocated to many PROJECTs; A PROJECT is assigned many items of EQUIPMENT.
- ✓ A Many-to-Many relationship is shown on the diagram by a line connecting the two entities with *crows feet* at each end of the line.

Many-to-Many Relationship



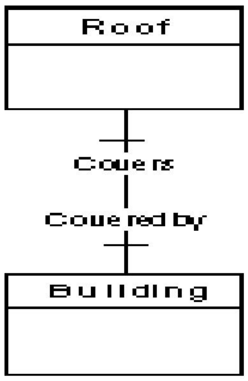
Steps taken to eliminate a many to many relationship to one to many relationship

- ✓ Create the two tables that will be used in the many-to-many relationship.
- ✓ Create another table, called a junction table.
- ✓ In the junction table, add fields with the same definitions as the Primary Key fields from each of the other two tables.
- ✓ Create a one-to-many relationship between each of the two primary tables and the junction table.

Creating a One-to-One Relationship

- ✓ In a one-to-one relationship, a record in Table A can have no more than one matching record in Table B, and a record in Table B can have no more than one matching record in Table A.
- ✓ For example, a ROOF covers one BUILDING; a BUILDING is covered by one ROOF.

One-to-One Relationship



Step 6: Refine/process your design.

- ✓ Analyze your design for errors.
- ✓ Create the tables and add a few records of sample data.
- ✓ See if you can get the results you want from your tables.
- ✓ Make adjustments to the design as needed.

2.6.RELATIONAL MODEL

- ✓ A branch of set theory that deals with logical relationships between sets.
- ✓ The relational model used the basic concept of a relation or table.
- ✓ The columns or fields in the table identify the attributes such as name, age, and so on.

✓ The relational model also includes concepts such as foreign keys, which are primary keys in one relation that are kept in another relation to allow for the joining of data.

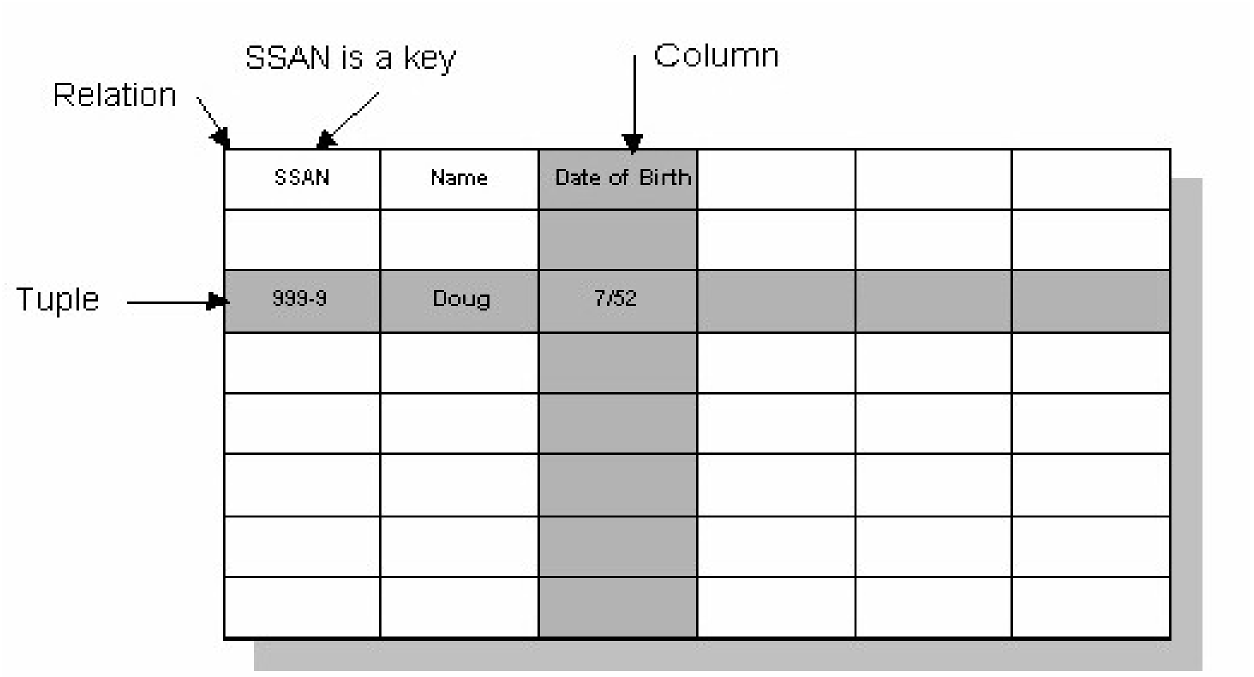
2.6.1 Types of Attributes

Simple: Each entity has a single atomic value for the attribute. For example, SSN or Sex

Composite: The attribute may be composed of several components. For example, Address (Apt#, House#, Street, City, State, ZipCode, Country) or Name (FirstName, MiddleName, LastName).

Multi-valued : An entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT. Denoted as {Color} or {PreviousDegrees}.

Relational model example.



<u>Relation Schema</u>

- $A_1, A_2, \ldots, A_n$ are *attributes*
- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

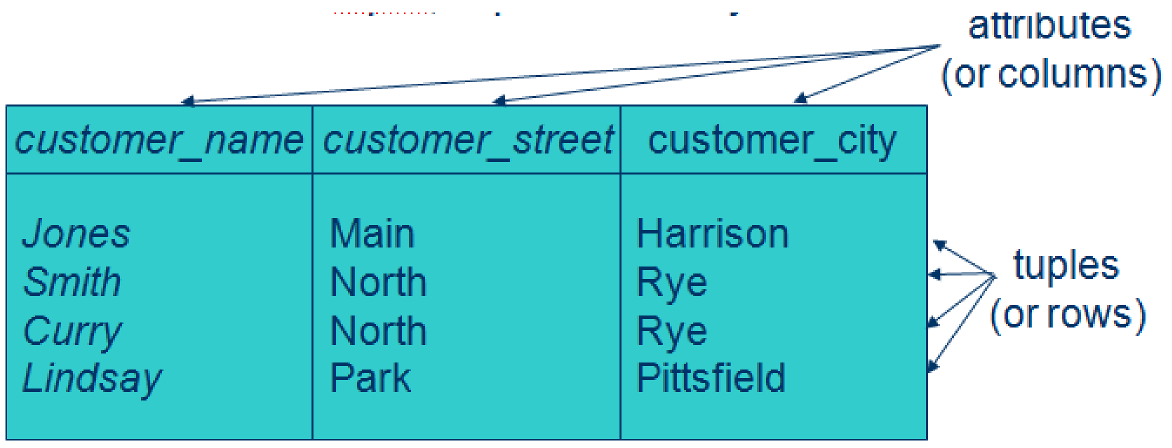
Example: *Customer_schema* = (*customer_name, customer_street, customer_city*)

- *r*(*R*) denotes a *relation r* on the *relation schema R*

Example:*customer (Customer_schema)*

Relation Instance

- The current values (*relation instance*) of a relation are specified by a table
- An element *t* of *r* is a *tuple*, represented by a *row* in a table


attributes
(or columns)
tuples
(or rows)

customer_name	*customer_street*	customer_city
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

customer

CHAPTER 3 LANGUAGE SQL

3.1 Definition

SQL often referred to as **Structured Query Language**, is a database computer language designed for managing data in relational database management systems (RDBMS), and originally based upon relational algebra.

3.2 SCOPE of SQL

A database language standard specifies the semantics of various components of a database management system (DBMS). In particular, it defines the structures and operations of a data model implemented by the DBMS, as well as other components that support data definition, data access, security, programming language interface, and data administration. The SQL standard specifies data definition, data manipulation, and other associated facilities of a DBMS that supports the relational data model.

3.3 Data Definition

The Data Definition Language (DDL) manages table and index structure. The most basic items of DDL are the CREATE, ALTER, RENAME, DROP and TRUNCATE statements:

CREATE creates an object (a table, database for example) in the database.

Syntax to create database:

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
```

CREATE DATABASE creates a database with the given name. To use this statement, you need the CREATE privilege for the database.

Syntax to create table:

```
CREATE TABLE table_name

(

   field1   datatype1,

   field2   datatype2,

   field3   datatype3

   );
```

e.g.:

```
CREATE TABLE My_table
```

```
(
   my_field1   INT,
   my_field2   VARCHAR(50),
   my_field3   DATE        NOT NULL
);
```

ALTER TABLE enables you to change the structure of an existing table. For example, you can add or delete columns, create or destroy indexes, change the type of existing columns, or rename columns or the table itself.

Syntax:

```
ALTER TABLE table_name ADD field4 datatype;
```

e.g.:

```
ALTER TABLE My_table ADD my_field4 NUMBER(3) NOT NULL;
```

You can rename a column using a CHANGE *old_col_name column_definition* clause. To do so, specify the old and new column names and the type that the column currently has. For example, to rename an INTEGER column from a to b, you can do this:

```
ALTER TABLE t1 CHANGE a b INTEGER;
```

If you want to change a column's type but not the name, CHANGE syntax still requires an old and new column name, even if they are the same. For example:

```
ALTER TABLE t1 CHANGE b b BIGINT NOT NULL;
```

You can also use MODIFY to change a column's type without renaming it:

```
ALTER TABLE t1 MODIFY b BIGINT NOT NULL;
```

TRUNCATE deletes all data from a table in a very fast way, deleting the data inside the table and not the table itself.

e.g.:

TRUNCATE TABLE My_table;

DROP deletes an object in the database, usually irretrievably, i.e., it cannot be rolled back.

e.g.:

DROP TABLE My_table;

3.4 Data types

Each column in an SQL table declares the type(s) that column may contain. ANSI SQL includes the following datatypes:

Character strings

- CHARACTER(*n*) or CHAR(*n*) — fixed-width *n*-character string, padded with spaces as needed
- CHARACTER VARYING(*n*) or VARCHAR(*n*) — variable-width string with a maximum size of *n* characters

Bit strings

- BIT(*n*) — an array of *n* bits
- BIT VARYING(*n*) — an array of up to *n* bits

Numbers

- INTEGER and SMALLINT
- FLOAT, REAL and DOUBLE PRECISION
- NUMERIC() or DECIMAL()

Date and time

- DATE — for date values (e.g., 2010-05-30)
- TIME — for time values (e.g., 14:55:37). The granularity of the time value is usually a *tick* (100 nanoseconds).
- TIMESTAMP — This is a DATE and a TIME put together in one variable (e.g., 2010-05-30 14:55:37).

3.5 Primary Key

A primary key is a unique identifier for a database record. When a table is created, one of the fields is typically assigned as the primary key. While the primary key is often a number, it may also be a text field or other data type. For example, if a database contains definitions of computer terms, it would make sense that each term is only listed once in the database. By defining the "Term" field as the primary key, it would ensure that no term is listed more than once in the database.

Example: To create a table customer with primary key, we proceed as follow:

```
CREATE TABLE Customer
(SID integer,
Last_Name varchar(30),
First_Name varchar(30),
PRIMARY KEY (SID));
```

3.6 Foreign Key

A foreign key is a field (or fields) that points to the primary key of another table. The purpose of the foreign key is to ensure referential integrity of the data. In other words, only values that are supposed to appear in database are allowed.

For example, suppose we have two tables, CUSTOMER table that includes all customer data, and an ORDERS table that includes all customers' orders. The constraint here is that all orders must b associated with a customer that is already in the CUSTOMER table. In this case, we will place a foreign key on the ORDERS table and have it relate to the primary key of the CUSTOMER table. Below there is an example of how to specify the foreign key when creating ORDERS table:

```
CREATE TABLE ORDERS
(Order_ID integer,
Order_Date date,
Customer_SID integer,
Amount double,
Primary Key (Order_ID),
Foreign Key (Customer_SID) references CUSTOMER (SID));
```

3.8 Reinforce Referential Integrity

Referential integrity is an important concept in database design. The term refers to a state when all the references in a database are valid and no invalid links exist between the various tables that make up the system. When referential integrity exists, any attempt to link to a record which does not already exist will fail; this helps prevent user errors, producing a more accurate (and useful) database.

Referential integrity is usually implemented through the use of foreign keys. For a long time, the popular open-source RDBMS MySQL did not support foreign keys, citing concerns that such support would erode RDBMS speed and performance. However, given the high volume of user interest in this feature, later versions of MySQL implemented support for foreign keys through the InnoDB table engine. Consequently, maintaining referential integrity within the tables that make up a database is significantly simpler.

In order to set up a foreign key relationship between two MySQL tables, three conditions must be met:

1. Both tables must be of the InnoDB table type.
2. The fields used in the foreign key relationship must be indexed.
3. The fields used in the foreign key relationship must be similar in data type.

Let’s use above example (CUSTOMER and ORDERS tables) to enforce referential integrity

Creation of customer table as Innodb type

```
CREATE TABLE Customer
(SID integer,
Last_Name varchar(30),
First_Name varchar(30),
PRIMARY KEY (SID))ENGINE =INNODB;
```

Creation of Orders table as Innodb type also and foreign key indexed

```
CREATE TABLE ORDERS
(Order_ID integer,
Order_Date date,
Customer_SID integer,
Amount double, Index (Customer_SID),
Primary Key (Order_ID),
Foreign Key (Customer_SID) references CUSTOMER (SID)) ENGINE=INNODB;
```

3.8 Data Manipulation

The Data Manipulation Language (DML) is the subset of SQL used to add, update and delete data:

- INSERT adds rows (formally tuples) to an existing table:

SYNTAX:

INSERT INTO table_name (field1, field2, field3, …) values (value1, value2, value3, …);

e.g.

INSERT INTO My_table

```
        (field1, field2, field3)

    VALUES

        ('test', 'N', NULL);
```

INSERT INTO Customer(CID, Last_Name, First_Name) VALUES (1, 'jojo', 'kagabo');

- UPDATE modifies a set of existing table rows:

SYNTAX:

UPDATE table_name SET field_to_update=”new value” where field_name=”value”

e.g.

```
UPDATE My_table

    SET field1 = 'updated value'

    WHERE field2 = 'N';
```

UPDATE customer SET Last_Name = 'joyeuse' WHERE Last_Name = 'jojo';

- DELETE removes existing rows from a table, e.g.,:

```
DELETE FROM My_table

    WHERE field2 = 'N';
```

3.9 Data Selection

The most common operation in SQL is the query, which is performed with the declarative SELECT statement. SELECT retrieves data from one or more tables, or expressions. Standard SELECT statements have no persistent effects on the database. Some non-standard implementations of SELECT can have persistent effects, such as the SELECT INTO syntax that exists in some databases.

Queries allow the user to describe desired data, leaving the database management system (DBMS) responsible for planning, optimizing, and performing the physical operations necessary to produce that result as it chooses.

A query includes a list of columns to be included in the final result immediately following the SELECT keyword. An asterisk ("*") can also be used to specify that the query should return all columns of the queried tables. SELECT is the most complex statement in SQL, with optional keywords and clauses that include:

- The FROM clause which indicates the table(s) from which data is to be retrieved. The FROM clause can include optional JOIN subclauses to specify the rules for joining tables.
- The WHERE clause includes a comparison predicate, which restricts the rows returned by the query. The WHERE clause eliminates all rows from the result set for which the comparison predicate does not evaluate to True.
- The GROUP BY clause is used to project rows having common values into a smaller set of rows. GROUP BY is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set. The WHERE clause is applied before the GROUP BY clause.
- The HAVING clause includes a predicate used to filter rows resulting from the GROUP BY clause. Because it acts on the results of the GROUP BY clause, aggregation functions can be used in the HAVING clause predicate.
- The ORDER BY clause identifies which columns are used to sort the resulting data, and in which direction they should be sorted (options are ascending or descending). Without an ORDER BY clause, the order of rows returned by an SQL query is undefined.

SELECT Statement

The SELECT statement is used to select data from a database.

The result is stored in a result table, called the result-set.

SQL SELECT Syntax

SELECT column_name (s) FROM table_name and SELECT *FROM table_name

Note: The asterisk (*) is used to select all columns of the table

Let’s consider the following example:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn10	Sandnes
2	Svendson	Tove	Borgvn23	Sandnes
3	Pettersen	Kari	Storgt20	Stavanger

Now we want to select the content of the columns named "LastName" and "FirstName" from the table above. We use the following SELECT statement:

SELECT LastName,FirstName FROM Persons

The result-set will look like this:

LastName	FirstName
Hansen	Ola
Svendson	Tove
Pettersen	Kari

SELECT DISTINCT Statement

In a table, some of the columns may contain duplicate values. This is not a problem; however, sometimes you will want to list only the different (distinct) values in a table.

The DISTINCT keyword can be used to return only distinct (different) values.

SQL SELECT DISTINCT Syntax

SELECT DISTINCT column_name(s) FROM table_name

Consider the same table “Persons”; now we want to select only the distinct values from the column named "City". We use the following SELECT statement:

SELECT DISTINCT City FROM Persons

The result-set will look as follow:

City
Sandnes
Stavanger

The WHERE Clause

The WHERE clause is used to extract only those records that fulfill a specified criterion.

SQL WHERE Syntax

SELECT column_name(s) FROM table_name WHERE column_name operator value

Here there is an example to select only the persons living in the city "Sandnes" from the table Persons. We use the following SELECT statement:

SELECT * FROM Persons WHERE City='Sandnes'

Operators Allowed in the WHERE Clause

With the WHERE clause, the following operators can be used:

Operator	**Description**
=	Equal
<>	Not equal
>	Greater than

<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	If you know the exact value you want to return for at least one of the columns

The LIKE Operator

The LIKE operator is used to search for a specified pattern in a column.

SQL LIKE Syntax

SELECT column_name(s) FROM table_name WHERE column_name LIKE pattern

To select the persons living in a city that starts with "s" from the table Persons. We use the following SELECT statement:

SELECT * FROM Persons WHERE City LIKE 's%'

Next, to select the persons living in a city that ends with an "s" from the "Persons" table. We use the following SELECT statement:

SELECT * FROM Persons WHERE City LIKE '%s'

Next, to select the persons living in a city that contains the pattern "tav" from the "Persons" table.

We use the following SELECT statement:

SELECT * FROM Persons WHERE City LIKE '%tav%'

It is also possible to select the persons living in a city that NOT contains the pattern "tav" from the "Persons" table, by using the NOT keyword. We use the following SELECT statement:

SELECT * FROM Persons WHERE City NOT LIKE '%tav%'

The IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

SQL IN Syntax

SELECT column_name(s) FROM table_name WHERE column_name IN (value1, value2...)

To select the persons with a last name equal to "Hansen" or "Pettersen" from the table above.We use the following SELECT statement:

SELECT * FROM Persons WHERE LastName IN ('Hansen','Pettersen')

The BETWEEN Operator

The BETWEEN operator selects a range of data between two values. The values can be numbers, dates.

SQL BETWEEN Syntax

SELECT column_name(s) FROM table_name WHERE column_name

BETWEEN value1 AND value2

SELECT P_Id FROM Persons WHERE P_Id BETWEEN 2 AND 3

The AND & OR Operators

The AND & OR operators are used to filter records based on more than one condition.

The AND operator displays a record if both the first condition and the second condition is true.

The OR operator displays a record if either the first condition or the second condition is true.

AND Operator Example

To select only the persons with the first name equal to "Tove" AND the lastname equal to "Svendson"

We use the following SELECT statement:

SELECT * FROM Persons WHERE FirstName='Tove' AND LastName='Svendson'

The result-set will look like this:

P_Id	**LastName**	**FirstName**	**Address**	**City**
2	Svendson	Tove	Borgvn 23	Sandnes

OR Operator Example

To select only the persons with the first name equal to "Tove" OR the first name equal to "Ola".

We use the following SELECT statement:

SELECT * FROM Persons WHERE FirstName='Tove' OR FirstName='Ola'

The result-set will look like this:

P_Id	**LastName**	**FirstName**	**Address**	**City**
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

Combining AND & OR

You can also combine AND and OR (use parenthesis to form complex expressions).

To select only the persons with the last name equal to "Svendson" AND the first name equal to "Tove" OR to "Ola". We use the following SELECT statement:

SELECT * FROM Persons WHERE LastName='Svendson'
AND (FirstName='Tove' OR FirstName='Ola')

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes

The ORDER BY Keyword

The ORDER BY keyword is used to sort the result-set by a specified column.

The ORDER BY keyword sorts the records in ascending order by default.

If you want to sort the records in a descending order, you can use the DESC keyword.

SQL ORDER BY Syntax

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC|DESC
```

```
SELECT P_Id FROM persons  ORDER BY P_Id DESC
```

3.10 Query on Multiple Tables

Assume that we have a table called "Persons" and another table called "Product_Orders". We will give the table aliases of "p" and "po" respectively.Now to list all the orders that "Ola Hansen" is responsible for; we use the following SELECT statement:

```
SELECT po.OrderID, p.LastName, p.FirstName
FROM Persons AS p,
Product_Orders AS po
WHERE p.LastName='Hansen' AND p.FirstName='Ola'
```

The same SELECT statement without aliases:

```
SELECT Product_Orders.OrderID, Persons.LastName, Persons.FirstName
FROM Persons,
Product_Orders
WHERE Persons.LastName='Hansen' AND Persons.FirstName='Ola'
```

```
SELECT customer.CID, Persons.LastName, Persons.FirstName
FROM Persons, customer
WHERE Persons.LastName = 'Hansen' AND Persons.FirstName = 'Ola'
```

3.11 SQL join

The JOIN keyword is used in an SQL statement to query data from two or more tables, based on a relationship between certain columns in these tables. Tables in a database are often related to each other with keys.

A primary key is a column (or a combination of columns) with a unique value for each row. Each primary key value must be unique within the table. The purpose is to bind data together, across tables, without repeating all of the data in every table.

Look at the "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Note that the "P_Id" column is the primary key in the "Persons" table. This means that **no** two rows can have the same P_Id. The P_Id distinguishes two persons even if they have the same name.

Next, we have the "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1

Note that the "O_Id" column is the primary key in the "Orders" table and that the "P_Id" column refers to the persons in the "Persons" table without using their names.

Note that the relationship between above two tables is the "P_Id" column.

SQL INNER JOIN Keyword

The INNER JOIN keyword return rows when there is at least one match in both tables.

SQL INNER JOIN Syntax

```
SELECT column_name(s) FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

SQL INNER JOIN Example

Let’s consider table “Persons” and table “Orders”, and then list all persons with their orders; we use the following SQL SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
INNER JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

The result-set will look like this:

LastName	**FirstName**	**OrderNo**
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678

The INNER JOIN keyword return rows when there is at least one match in both tables. If there are rows in "Persons" that do not have matches in "Orders", those rows will NOT be listed.

3.12 SQL Aggregate Functions

SQL aggregate functions return a single value, calculated from values in a column.

Useful aggregate functions:

- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

The AVG() Function

The AVG() function returns the average value of a numeric column.

SQL AVG() Syntax

```
SELECT AVG(column_name) FROM table_name
```

SQL AVG() Example

Let’s consider the following "Orders" table:

O_Id	**OrderDate**	**OrderPrice**	**Customer**
1	2008/11/12	1000	Hansen
2	2008/10/23	1600	Nilsen
3	2008/09/02	700	Hansen
4	2008/09/03	300	Hansen
5	2008/08/30	2000	Jensen
6	2008/10/04	100	Nilsen

To find the average value of the "OrderPrice" fields; we use the following SQL statement:

```
SELECT AVG(OrderPrice) AS OrderAverage FROM Orders
```

The result-set will look like this:

OrderAverage
950

To find the customers that have an OrderPrice value higher than the average OrderPrice value; we use the following SQL statement:

```
SELECT Customer FROM Orders
WHERE OrderPrice>(SELECT AVG(OrderPrice) FROM Orders)
```

SQL COUNT() Function

The COUNT() function returns the number of rows that matches a specified criteria.

SQL COUNT(column_name) Syntax

The COUNT(column_name) function returns the number of values (NULL values will not be counted) of the specified column:

```
SELECT COUNT(column_name) FROM table_name
```

SQL COUNT(*) Syntax

The COUNT(*) function returns the number of records in a table:

SELECT COUNT(*) FROM table_name

SQL COUNT (column_name) Example

To count the number of orders from "Customer Nilsen"; we use the following SQL statement:

SELECT COUNT (Customer) AS CustomerNilsen FROM Orders WHERE Customer='Nilsen'

The result of the SQL statement above will be 2, because the customer Nilsen has made 2 orders in total:

CustomerNilsen
2

SQL COUNT(*) Example

If we omit the WHERE clause, like this:

SELECT COUNT(*) AS NumberOfOrders FROM Orders

The result-set will look like this:

NumberOfOrders
6

This is the total number of rows in the table.

The MAX() Function

The MAX() function returns the largest value of the selected column.

SQL MAX() Syntax

SELECT MAX(column_name) FROM table_name

To find the largest value of the "OrderPrice" column; we use the following SQL statement:

SELECT MAX(OrderPrice) AS LargestOrderPrice FROM Orders

The MIN() Function

The MIN() function returns the smallest value of the selected column.

SQL MIN() Syntax

SELECT MIN(column_name) FROM table_name

To find the smallest value of the "OrderPrice" column; we use the following SQL statement:

SELECT MIN(OrderPrice) AS SmallestOrderPrice FROM Orders

The SUM() Function

The SUM() function returns the total sum of a numeric column.

SQL SUM() Syntax

To find the sum of all "OrderPrice" field; we use the following SQL statement:

SELECT SUM(OrderPrice) AS OrderTotal FROM Orders

Note

The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns; it has the following syntax:

SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name

Example:

To find the total sum (total order) of each customer; we have to use the GROUP BY statement to group the customers using the following SQL statement:

```
SELECT Customer,SUM(OrderPrice) FROM Orders GROUP BY Customer
```

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions; it has the following syntax:

```
SELECT column_name, aggregate_function(column_name) FROM table_name WHERE
column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value
```

Example:

To find if any of the customers have a total order of less than 2000; we use the following SQL statement:

```
SELECT Customer,SUM(OrderPrice) FROM Orders
GROUP BY Customer
HAVING SUM(OrderPrice)<2000
```

CHAPTER 4 SECURE A DATABASE

4.1 Database security

Database security concerns the use of a broad range of information security controls to protect databases (potentially including the data, the database applications or stored functions, the database systems, the database servers and the associated network links) against compromises of their confidentiality, integrity and availability. It involves various types or categories of controls, such as technical, procedural/administrative and physical.

Security risks to database systems include, for example:

- Unauthorized or unintended activity or misuse by authorized database users, database administrators, or network/systems managers, or by unauthorized users or hackers (e.g. inappropriate access to sensitive data, metadata or functions within databases, or inappropriate changes to the database programs, structures or security configurations);
- Malware infections causing incidents such as unauthorized access, leakage or disclosure of personal or proprietary data, deletion of or damage to the data or programs, interruption or denial of authorized access to the database, attacks on other systems and the unanticipated failure of database services;
- Design flaws and programming bugs in databases and the associated programs and systems, creating various security vulnerabilities (e.g. unauthorized privilege escalation), data loss/corruption, performance degradation etc.;
- Data corruption and/or loss caused by the entry of invalid data or commands, mistakes in database or system administration processes, sabotage/criminal damage etc.

4.2 Access control

Access control is a system that enables an authority to control access to areas and resources in a given physical facility or computer-based information system. An access control system, within the field of physical security, is generally seen as the second layer in the security of a physical structure.

Access control systems provide the essential services of identification and authentication (I&A), authorization, and accountability where:

- identification and authentication determine who can log on to a system, and the association of users with the software subjects that they are able to control as a result of logging in;
- authorization determines what a subject can do;
- Accountability identifies what a subject (or all subjects associated with a user) did.

In access control, let’s consider the following types of authentication.

There are three types (factors) of authenticating information:

- something the user knows, e.g. a password, pass-phrase or PIN
- something the user has, such as smart card
- something the user is, such as fingerprint, verified by biometric measurement

Passwords are a common means of verifying a user's identity before access is given to information systems. In addition, a fourth factor of authentication is now recognized: someone you know, where another person who knows you can provide a human element of authentication in situations where systems have been set up to allow for such scenarios.

4.3 Grant and Revoke commands

DCL commands are used to enforce database security in a multiple user database environment. Two types of DCL commands are GRANT and REVOKE. Only Database Administrators or owners of the database object can provide/remove privileges on a database object.

4.3.1 SQL GRANT Command

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

The Syntax for the GRANT command is:

```
GRANTprivilege_name
ONobject_name
TO{user_name}
[WITH GRANT OPTION];
```

- ***privilege_name*** is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
- ***object_name*** is the name of a database object like TABLE, VIEW, and STORED PROC
- ***user_name*** is the name of the user to whom an access right is being granted.
- ***WITH GRANT OPTION*** - allows a user to grant access rights to other users.

For Example: GRANT SELECT ON employee TO user1; this command grants a SELECT permission on employee table to user1.You should use the WITH GRANT option carefully because for example if you GRANT SELECT privilege on employee table to user1 using the WITH GRANT option, then user1 can GRANT SELECT privilege on employee table to another user, such as user2 etc. Later, if you REVOKE the SELECT privilege on employee from user1, still user2 will have SELECT privilege on employee table.

4.3.2 SQL REVOKE Command

The REVOKE command removes user access rights or privileges to the database objects.

The Syntax for the REVOKE command is:

```
REVOKE privilege_name
ON object_name
FROM {user_name }
```

For Example: REVOKE SELECT ON employee FROM user1;This commmand will REVOKE a SELECT privilege on employee table from user1.When you REVOKE SELECT privilege on a table from a user, the user will not be able to SELECT data from that table anymore. However, if the user has received SELECT privileges on that table from more than one user, he/she can SELECT from that table until everyone who granted the permission revokes it. You cannot REVOKE privileges if they were not initially granted by you.

3.2.3 Privileges

Privileges: Privileges defines the access rights provided to a user on a database object. There are two types of privileges.

1) System privileges - This allows the user to CREATE, ALTER, or DROP database objects.
2) Object privileges - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

4.4 Backup of data

In information technology, a backup or the process of backing up is making copies of data which may be used to *restore* the original after a data loss event. The verb form is back up in two words, whereas the noun is *backup*.

Backups have two distinct purposes. The primary purpose is to recover data after its loss, be it by data deletion or corruption. Data loss is a very common experience of computer users. The secondary purpose of backups is to recover data from an earlier time, according to a user-defined data retention policy, typically configured within a backup application for how long copies of data are required.

Though backups popularly represent a simple form of disaster recovery, and should be part of a disaster recovery plan, by themselves, backups should not alone be considered disaster recovery.

Since a backup system contains at least one copy of all data worth saving, the data storage requirements are considerable. In the modern era of computing there are many different types of data storage devices that are useful for making backups. There are also many different ways in which these devices can be arranged to provide geographic redundancy, data security, and portability.

CHAPTER 5- DATABASE ADMINISTRATION

5.1 Transaction

A database transaction is a unit of work performed against a database management system or similar system that is treated in a coherent and reliable way independent of other transactions. A database transaction, by definition, must be atomic, consistent, isolated and durable. These properties of database transactions are often referred to by the acronym ACID.

One or more operations collected as a single logical unit of database work is called a DATABASE TRANSACTION. Database transactions are usually activated by user requests in

SQL or other Data Manipulation Languages (DML). Most real-world database transactions are based on two or more requests. Transactions access the database via reading and/or writing operations: they may be INSERT, UPDATE, DELETE, or SELECT data; they may modify the values of attributes in tables; they may also change the physical structure of the database itself. The scope of a database transaction is defined by the user when he or she initiates it. Once started, a transaction must be ended either with COMMIT or ROLLBACK. A successfully completed transaction ends with COMMIT to permanently record the transaction results in the database. A failed and then aborted transaction ends with ROLLBACK to undo the transaction effects on the database. Committed transactions change the database from one consistent state to another.

5.2 Concurrency Control

Concurrency control is a database management systems (DBMS) concept that is used to address conflicts with the simultaneous accessing or altering of data that can occur with a multi-user system. Concurrency control, when applied to a DBMS, is meant to coordinate simultaneous transactions while preserving data integrity. The concurrency is about to control the multiuser access of Database

To illustrate the concept of concurrency control, consider two travelers who go to electronic kiosks at the same time to purchase a train ticket to the same destination on the same train. There's only one seat left in the coach, but without concurrency control, it's possible that both travelers will end up purchasing a ticket for that one seat. However, with concurrency control, the database wouldn't allow this to happen. Both travelers would still be able to access the train seating database, but concurrency control would preserve data accuracy and allow only one traveler to purchase the seat.

This example also illustrates the importance of addressing this issue in a multi-user database. Obviously, one could quickly run into problems with the inaccurate data that can result from several transactions occurring simultaneously and writing over each other.

5.2.1Concurrency control strategies

Pessimistic Locking: This concurrency control strategy involves keeping an entity in a database locked the entire time it exists in the database's memory. This limits or prevents users from altering the data entity that is locked. There are two types of locks that fall under the category of pessimistic locking: write lock and read lock.

With write lock, everyone but the holder of the lock is prevented from reading, updating, or deleting the entity. With read lock, other users can read the entity, but no one except for the lock holder can update or delete it.

Optimistic Locking: This strategy can be used when instances of simultaneous transactions, or collisions, are expected to be infrequent. In contrast with pessimistic locking, optimistic locking doesn't try to prevent the collisions from occurring. Instead, it aims to detect these collisions and resolve them on the chance occasions when they occur.

Pessimistic locking provides a guarantee that database changes are made safely. However, it becomes less viable as the number of simultaneous users or the number of entities involved in a transaction increase because the potential for having to wait for a lock to release will increase.

Optimistic locking can alleviate the problem of waiting for locks to release, but then users have the potential to experience collisions when attempting to update the database.

PART TWO: INFORMATION SYSTEMS

CHAP I: MANAGEMENT INFORMATION SYSTEMS

1.1. **What Is an Information System?**

Information system has been defined in terms of two perspectives: one relating to its function; the other relating to its structure. From a **functional perspective**; an information system is a technologically implemented medium for the purpose of recording, storing, and disseminating linguistic expressions as well as for the supporting of inference making. From a **structural perspective**; an information system consists of a collection of people, processes, data, models, technology and partly formalized language, forming a cohesive structure which serves some organizational purpose or function. The functional definition has its merits in focusing on what actual users -from a conceptual point of view- do with the information system while using it. They communicate with experts to solve a particular problem. The structural definition makes clear that IS are socio-technical systems, i.e., systems consisting of humans, behavior rules, and conceptual and technical artifacts.

An information system can be defined ***technically*** as a set of interrelated components that collect (or retrieve), process, store, and distribute information to support decision making and control in an organization. In addition to supporting decision making, coordination, and control, information systems may also help managers and workers analyze problems, visualize complex subjects, and create new products. Three activities in an information system produce the information that organizations need to make decisions, control operations, analyze problems, and create new products or services.

These activities are input, processing, and output. Input captures or collects raw data from within the organization or from its external environment. Processing converts this raw input into a more meaningful form. Output transfers the processed information to the people who will use it or to the activities for which it will be used. Information systems also require feedback, which is output that is returned to appropriate members of the organization to help them evaluate or correct the input stage.

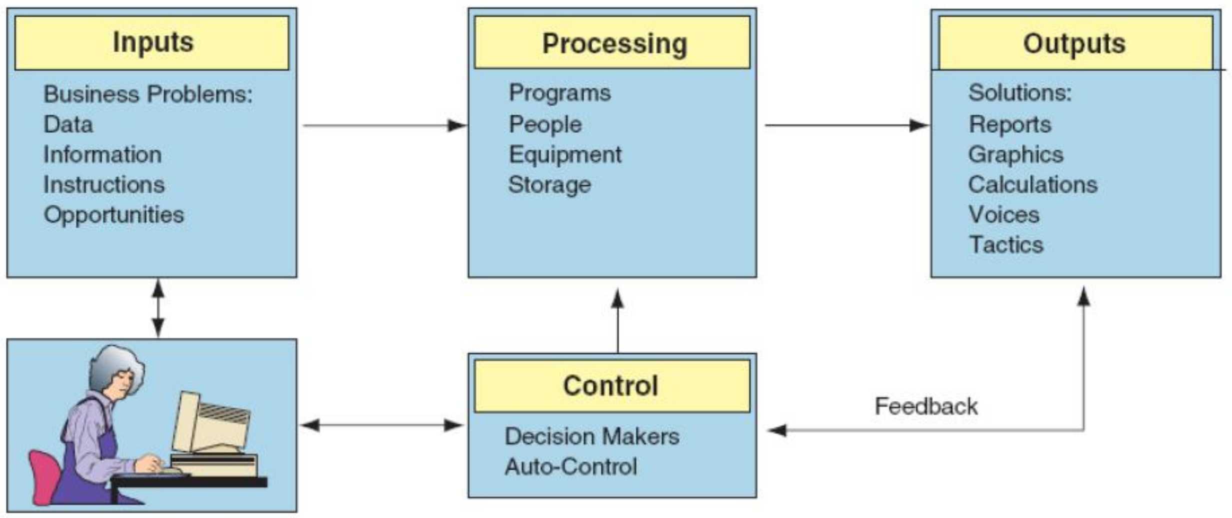


Figure 1: Functions of an information system

1.2. **A Computer-Based Information System**

A computer-based information system (CBIS) is an information system that uses computer technology to perform some or all of its intended tasks. Such a system can include as little as a personal computer and software. Or it may include several thousand computers of various sizes with hundreds of printers, plotters, and other devices, as well as communication networks (wire-line and wireless) and databases. In most cases an information system also includes people. The basic components of information systems are listed below. Note that not every system includes all these components.

1.3. Components of Information Systems

1. Resources of people: (end users and IS specialists, system analyst, programmers, data administrators etc.).
2. Hardware: (Physical computer equipments and associate device, machines and media).
3. Software: (programs and procedures).
4. Data: (data and knowledge bases), and
5. Networks: (communications media and network support).

People Resources

• End users: (also called users or clients) are people who use an information system or the information it produces. They can be accountants, salespersons, engineers, clerks, customers, or managers. Most of us are information system end users.

• IS Specialists: people who actually develop and operate information systems. They include systems analysts, programmers, testers, computer operators, and other managerial, technical, and clerical IS personnel. Briefly, systems analysts design information systems based on the information requirements of end uses, programmers prepare computer programs based on the specifications of systems analysts, and computer operators operate large computer systems.

Hardware Resources

• Machines: as computers and other equipment along with all data media, objects on which data is recorded and saved.

• Computer systems: consist of variety of interconnected peripheral devices. Examples are microcomputer systems, midrange computer systems, and large computer systems.

Software Resources

Software Resources includes all sets of information processing instructions. This generic concept of software includes not only the programs, which direct and control computers but also the sets of information processing (procedures). Software Resources includes:

System software, such as an operating system

- Application software, which are programs that direct processing for a particular use of computers by end users.
- Procedures, which are operating instructions for the people, who will use an information system. Examples are instructions for filling out a paper form or using a particular software package.

Data Resources

Data resources include data (which is raw material of information systems) and database. Data can take many forms, including traditional alphanumeric data, composed of numbers and alphabetical and other characters that describe business transactions and other events and entities. Text data, consisting of sentences and paragraphs used in written communications; image data, such as graphic shapes and figures; and audio data, the human voice and other sounds, are also important forms of data. Data resources must meet the following criteria:

- Comprehensiveness: means that all the data about the subject are actually present in the database.
- Non-redundancy: means that each individual piece of data exists only once in the database.
- Appropriate structure: means that the data are stored in such a way as to minimize the cost of expected processing and storage.

The data resources of IS are typically organized into:

- Processed and organized data-Databases.
- Knowledge in a variety of forms such as facts, rules, and case examples about successful business practices.

Network Resources

Telecommunications networks like the Internet, intranets, and extranets have become essential to the successful operations of all types of organizations and their computer-based information systems. Telecommunications networks consist of computers, communications processors, and other devices interconnected by communications media and controlled by communications software. The concept of Network Resources emphasizes that communications networks are a fundamental resource component of all information systems. Network resources include:

Communications media: such as twisted pair wire, coaxial cable, fiber-optic cable, microwave systems, and communication satellite systems.

• Network support: This generic category includes all of the people, hardware, software, and data resources that directly support the operation and use of a communications network. Examples include communications control software such as network operating systems and Internet packages.

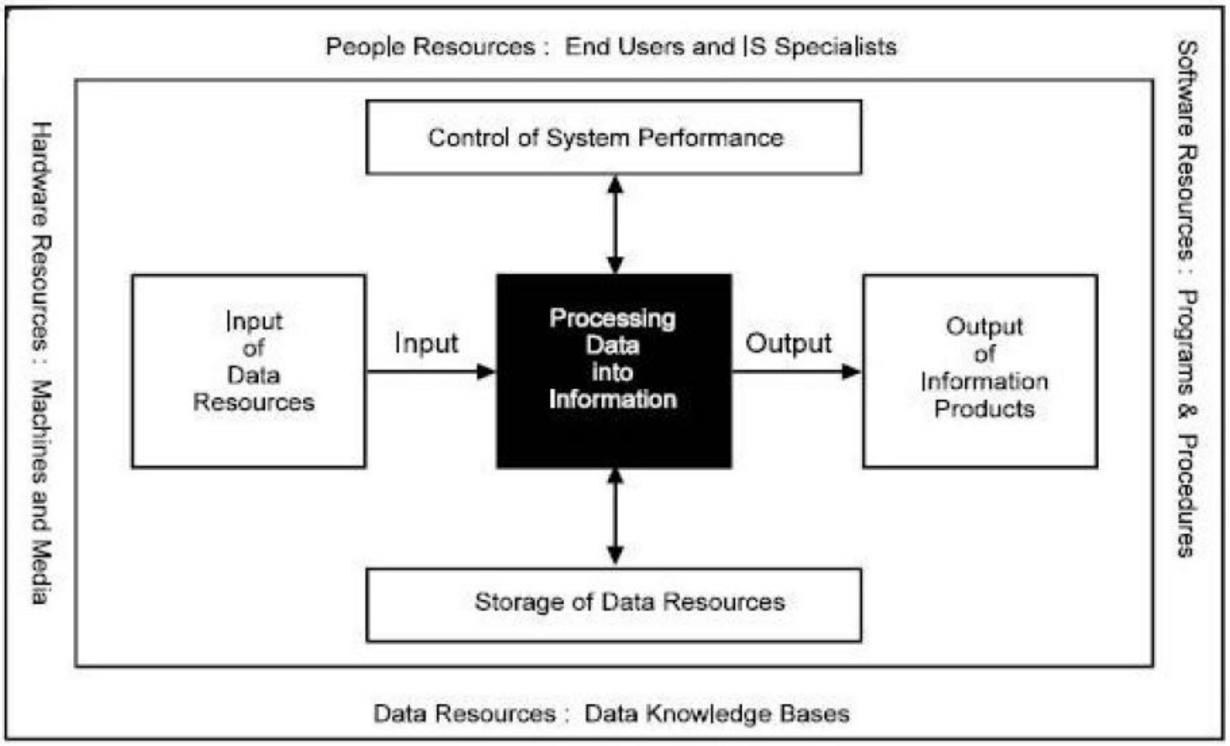

People Resources : End Users and IS Specialists
Hardware Resources : Machines and Media
Software Resources : Programs & Procedures
Control of System Performance
Input of Data Resources
Input
Processing Data into Information
Output
Output of Information Products
Storage of Data Resources
Data Resources : Data Knowledge Bases

Figure 2: Components of Information System

1.4 Difference between Computers and Information Systems

Computers provide effective and efficient ways of processing data, and they are a necessary part of an information system. An IS, however, involves much more than just computers. The successful application of an IS requires an understanding of the business and its environment that is supported by the IS. For example, to build an IS that supports transactions executed on the New York Stock Exchange, it is necessary to understand the procedures related to buying and selling stocks, bonds, options, and so on, including irregular demands made on the system, as well as all related government regulations. In learning about information systems, it is therefore not sufficient just to learn about computers. Computers are only one part of a complex system that must be designed, operated, and maintained. A public transportation system in a city provides an analogy. Buses are a necessary ingredient of the system, but more is needed. Designing the bus routes, bus stops, different schedules, and so on requires considerable understanding of customer demand, traffic patterns, city regulations, safety requirements, and the like. Computers, like buses, are only one component in a complex system.

1.4.1 Information Technology and Information Systems

Information technology broadly defined as the collection of computer systems used by an organization. Information technology, in its narrow definition, refers to the technological side of an information system. It includes the hardware, software, databases, networks, and other electronic devices. It can be viewed as a subsystem of an information system. Sometimes, though, the term information technology is also used interchangeably with information system.
The term IT in its broadest sense used to describe an organization's collection of information systems, their users, and the management that oversees them.

A major role of IT is being a *facilitator* of organizational activities and processes. That role will become more important as time passes. Therefore, it is necessary that every manager and professional staff member learn about IT not only in his or her specialized field, but also in the entire organization and in inter-organizational settings as well. Obviously, you will be more effective in your chosen career if you understand how successful information systems are built, used, and managed.

You also will be more effective if you know how to recognize and avoid unsuccessful systems and failures. Also, in many ways, having a comfort level with information technology will enable you, off the job and in your private life, to take advantage of new IT products and systems as they are developed. (Wouldn’t you rather be the one explaining to friends how some new product works, than the one asking about it?) Finally, you should learn about IT because being knowledgeable about information technology can also increase employment opportunities. Even though computerization eliminates some jobs, it also creates many more.

The demand for traditional information technology staff such as programmers, systems analysts, and designers is substantial. In addition, many excellent opportunities are appearing in emerging areas such as the Internet and e-commerce, m-commerce, network security, object-oriented programming, telecommunications, multimedia design, and document management. According to a study by the U.S. Bureau of Labor Statistics, each of the top seven fastest-growing occupations projected through 2010 fall within an IT- or computer related field. These top seven occupations are:

1.Computer software applications engineers
2. Computer support specialists
3. Computer software systems engineers
4. Network and computer systems administrators
5. Network systems and data communications analysts
6. Desktop publishers
7. Database administrators

To exploit the high-paying opportunities in IT, a college degree in any of the following fields, or combination of them, is advisable: computer science, computer information systems (CIS), management information systems (MIS), electronic commerce, and e-business. Within the last few years, many universities have started e-commerce or e-business degrees. Many schools offer graduate degrees with specialization in information technology.

1.5. Types of Management Information Systems

1.5.1Transaction-Processing Systems

Transaction-processing systems are designed to handle a large volume of routine, recurring transactions. They were first introduced in the 1960s with the advent of mainframe computers. Transaction-processing systems are used widely today. Banks use them to record deposits and payments into accounts. Supermarkets use them to record sales and track inventory. Managers often use these systems to deal with such tasks as payroll, customer billing and payments to suppliers.

1.5.2Operations Information Systems

Operations information systems were introduced after transaction-processing systems. An operations information system gathers comprehensive data, organizes it and summarizes it in a form that is useful for managers. These types of systems access data from a transaction-processing system and organize it into a usable form. Managers use operations information systems to obtain sales, inventory, accounting and other performance-related information.

1.5.3 Decision Support Systems (DSS)

A DSS is an interactive computer system that can be used by managers without help from computer specialists. A DSS provides managers with the necessary information to make informed decisions. A DSS has three fundamental components: database management system (DBMS), which stores large amounts of data relevant to problems the DSS has been designed to tackle; model-based management system (MBMS), which transforms data from the DBMS into information that is useful in decision-making; and dialog generation and management system (DGMS), which provides a user-friendly interface between the system and the managers who do not have extensive computer training.

1.5. 4.Expert Systems and Artificial Intelligence

Expert systems and artificial intelligence use human knowledge captured in a computer to solve problems that ordinarily need human expertise. Mimicking human expertise and intelligence requires the computer to do the following: recognize, formulate and solve a problem; explain solutions; and learn from experience. These systems explain the logic of their advice to the user; hence, in addition to solving problems they also can serve as a teacher. They use flexible thinking processes and can accommodate new knowledge.

CHAPTER 2: ETHICAL AND SOCIAL ISSUES IN INFORMATION SYSTEMS

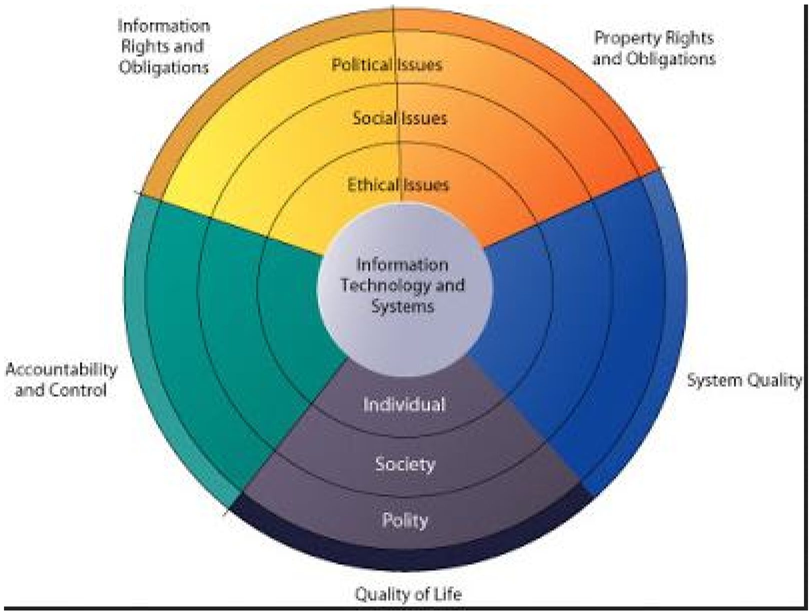
Technology can be a double-edged sword. It can be the source of many benefits but it can also create new opportunities for invading your privacy, and enabling the reckless use of that information in a variety of decisions about you.

2.1. Understanding Ethical and Social Issues Related to Systems

In the past 10 years, we have witnessed, arguably, one of the most ethically challenging periods for U.S. and global business. In today’s new legal environment, managers who violate the law and are convicted will most likely spend time in prison. ***Ethics*** refers to the principles of right and wrong that individuals, acting as free moral agents, use to make choices to guide their behaviors. When using information systems, it is essential to ask, “What is the ethical and socially responsible course of action?”

2.2 A Model for Thinking about Ethical, Social and Political Issues

Ethical, social, and political issues are closely linked. The ethical dilemma you may face as a manager of information systems typically is reflected in social and political debate.


Information Rights and Obligations
Property Rights and Obligations
Political Issues
Social Issues
Ethical Issues
Information Technology and Systems
Accountability and Control
System Quality
Individual
Society
Polity
Quality of Life

2.3. Five Moral Dimensions Of The Information Age

The major ethical, social, and political issues raised by information systems include the following moral dimensions:

Information rights and obligations. What information rights do individuals and organizations possess with respect to themselves? What can they protect?

Property rights and obligations. How will traditional intellectual property rights be protected in a digital society in which tracing and accounting for ownership are difficult and ignoring such property rights is so easy?

Accountability and control. Who can and will be held accountable and liable for the harm done to individual and collective information and property rights?

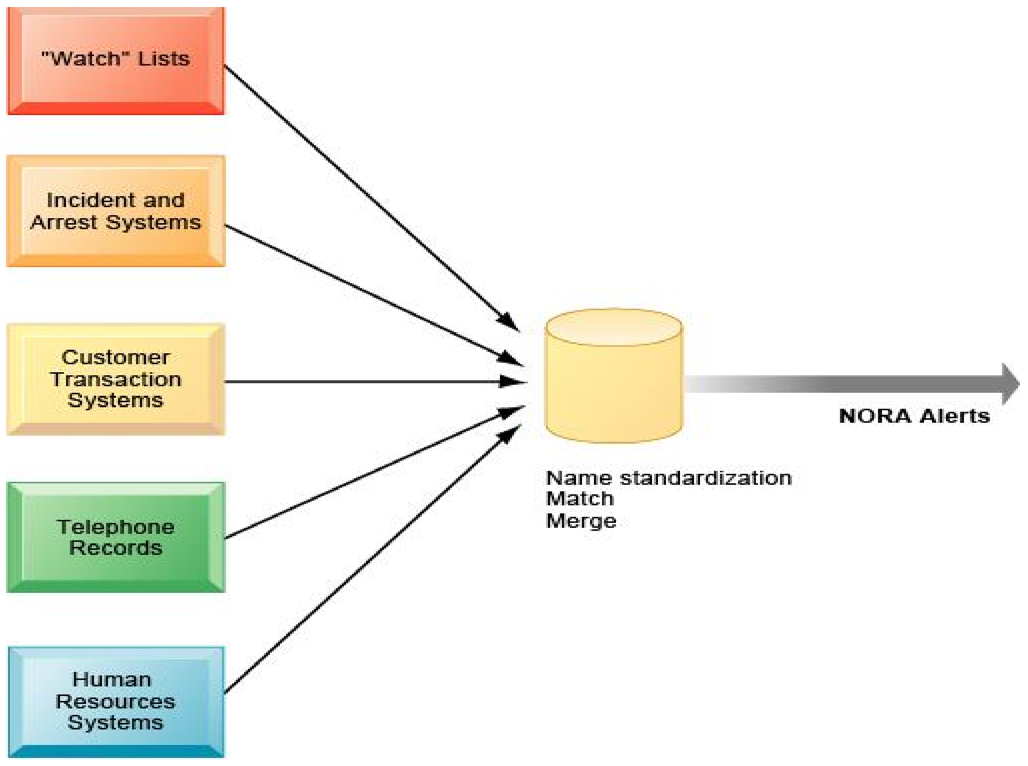
System quality. What standards of data and system quality should we demand to protect individual rights and the safety of society?

Quality of Life. What values should be preserved in an information- and knowledge-based society?

2.4. Key Technology Trends that Raise Ethical Issues

Profiling – the use of computers to combine data from multiple sources and create electronic dossiers of detailed information on individuals.

Non obvious relationship awareness (NORA) – a more powerful profiling capabilities technology, can take information about people from many disparate sources, such as employment applications, telephone records, customer listings, and “wanted” lists, and correlated relationships to find obscure hidden connections that might help identify criminals or terrorists.



Ethics In An Information Society

Basic Concepts: Responsibility, Accountability, and Liability

Ethical choices are decisions made by individuals who are responsible for the consequences of their actions.

Responsibility is a key element and means that you accept the potential costs, duties, and obligations for the decisions you make.

Accountability is a feature of systems and social institutions and means mechanisms are in place to determine who took responsible action, and who is responsible.

Liability is a feature of political systems in which a body of laws is in place that permits individuals to recover the damages done to them by other actors, systems, or organizations.

Due process is a related feature of law-governed societies and is a process in which laws are known and understood, and there is an ability to appeal to higher authorities to ensure that the laws are applied correctly.

The Moral Dimensions of Information Systems

Information Rights: Privacy and Freedom In The Internet Age

Privacy is the claim of individuals to be left alone, free from surveillance or interference from other individuals or organizations, including the state. Most American and European privacy law is based on a regime called Fair Information Practices (FIP) first set forth in a report written in 1973 by a federal government advisory committee (U.S. Department of Health, Education, and Welfare, 1973).

The European Directive on Data Protection

In Europe, privacy protection is much more stringent than in the United States. Unlike the United States, European countries do not allow businesses to use personally identifiable information without consumers' prior consent**.**

Informed consent can be defined as consent given with knowledge of all the facts needed to make a rational decision. Working with the European Commission, the U.S. Department of Commerce developed a safe harbor framework for U.S. firms.

A **safe harbor** is a private self-regulating policy and enforcement mechanism that meets the objectives of government regulators and legislation but does not involve government regulation or enforcement.

Internet Challenges to Privacy

Internet technology has posed new challenges for the protection of individual privacy. Information sent over this vast network of networks may pass through many different computer systems before it reaches its final destination. Each of these systems is capable of monitoring, capturing, and storing communications that pass through it.

Cookies are small text files deposited on a computer hard drive when a user visits to the web sites. Cookies identify the visitor's web browser software and track visits to the website. Web beacons, also called web bugs, are tiny objects invisibly embedded in e-mail messages and Web pages that are designed to monitor the behavior of the user visiting a web site or sending e-mail. Spyware can secretly install itself on an Internet user's computer by piggybacking on larger applications. Once installed, the spyware calls out to Web sites to send banner ads and other unsolicited material to the user, and it can also report the user's movements on the Internet to other computers.

Property Rights: Intellectual Property

Intellectual property is considered to be intangible property created by individuals or corporations. Information technology has made it difficult to protect intellectual property because computerized information can be so easily copied or distributed on networks. Intellectual property is subject to a variety of protections under three different legal traditions: trade secrets, copyright, and patent law.

Trade Secrets

Any intellectual work product – a formula, device, pattern, or compilation of data-used for a business purpose can be classified as a trade secret, provided it is not based on information in the public domain.

Copyright

Copyright is a statutory grant that protects creators of intellectual property from having their work copied by others for any purpose during the life of the author plus an additional 70 years after the author's death.

Patents

A patent grants the owner an exclusive monopoly on the ideas behind an invention for 20 years. The congressional intent behind patent law was to ensure that inventors of new machines, devices, or methods receive the full financial and other rewards of their labor and yet make widespread use of the invention possible by providing detailed diagrams for those wishing to use the idea under license from the patent's owner.

System Quality: Data Quality and System Errors

Three principle sources of poor system performance are (1) software bugs and errors (2) hardware or facility failures caused by natural or other causes and (3) poor input data quality. The software industry has not yet arrived at testing standards for producing software of acceptable but not perfect performance.

Quality of Life: Equity, Access, and Boundaries

Balancing Power: Center Versus Periphery

Lower level employees may be empowered to make minor decisions but the key policy decisions may be as centralized as in the past.

Rapidity of Change: Reduced Response Time to Competition

Information systems have helped to create much more efficient national and international market. The now-more-efficient global marketplace has reduced the normal social buffers that permitted businesses many years to adjust to competiton. We stand the risk of developing a "just-in-time society" with "just-in-time jobs" and "just-in-time" workplaces, families, and vacations.

Maintaining Boundaries: Family, Work, and Leisure

The danger to ubiquitous computing, telecommuting, nomad computing, and the "do anything anywhere" computing environment is that it is actually coming true. The traditional boundaries that separate work from family and just plain leisure have been weakened. The work umbrella now extends far beyond the eight-hour day.

Dependence and Vulnerability

Today our businesses, governments, schools, and private associations, such as churches are incredibly dependent on information systems and are, therefore, highly vulnerable if these systems fail. The absence of standards and the criticality of some system applications will probably call forth demands for national standards and perhaps regulatory oversight.

Computer Crime and Abuse

New technologies, including computers, create new opportunities for committing crimes by creating new valuable items to steal, new way to steal them, and new ways to harm others. **Computer crime** is the commission illegal acts through the use of a computer or against a computer system. Simply accessing a computer system without authorization or with intent to do harm, even by accident, is now a federal crime.

Computer abuse is the commission of acts involving a computer that may not illegal but that are considered unethical. The popularity of the Internet and e-mail has turned one form of computer abuse – spamming – into a serious problem for both individuals and businesses. **Spam** is junk e-mail sent by an organization or individual to a mass audience of Internet users who have expressed no interest in the product or service being marketed.

Employment: Trickle-Down Technology and Reengineering Job Loss

Reengineering work is typically hailed in the information systems community as a major benefit of new information technology. It is much less frequently noted that redesigning business processes could potentially cause millions of mid-level managers and clerical workers to lose their jobs. One economist has raised the possibility that we will create a society run by a small "high tech elite of corporate professionals…in a nation of permanently unemployed" (Rifkin, 1993). Careful planning and sensitivity to employee needs can help companies redesign work to minimize job losses.

Equity and Access: Increasing Racial and Social Class Cleavages

Several studies have found that certain ethnic and income groups in the United States are less likely to have computers or online Internet access even though computer ownership and Internet access have soared in the past five years. A similar digital divide exists in U.S. schools, with schools in high-poverty areas less likely to have computers, high-quality educational technology programs, or internet access availability for their students. Public interest groups want to narrow this digital divide by making digital information services – including the Internet – available to virtually everyone, just as basic telephone service is now.

Health Risks: RSI, CVS, and Technostress

The most common occupational disease today is repetitive stress injury (RSI). RSI occurs when muscle groups are forced through repetitive actions often with high-impact loads (such as tennis) or tens of thousands of repetitions under low-impact loads (such as working at a computer keyboard).

The single largest source of RSI is computer keyboards. The most common kind of computer-related RSI is carpal tunnel syndrome (CTS), in which pressure on the median nerve through the wrist's bony structure, called a carpal tunnel, produces pain. Millions of workers have been diagnosed with carpal tunnel syndrome. Computer vision syndrome (CVS) refers to any eyestrain condition related to display screen use in desktop computers, laptops, e-readers, smart-phones, and hand-held video games. Its symptoms, which are usually temporary, include headaches, blurred vision, and dry and irritated eyes.

The newest computer-related malady is technostress, which is stress induced by computer use. Its symptoms include aggravation, hostility toward humans, impatience, and fatigue. Technostress is thought to be related to high levels of job turnover in the computer industry, high levels of early retirement from computer-intense occupations, and elevated levels of drug and alcohol abuse.

Summary

Technology can be a double-edged sword. It can be the source of many benefits but it can also create new opportunities for invading your privacy, and enabling the reckless use of that information in a variety of decisions about you. The computer has become a part of our lives – personally as well as socially, culturally, and politically. It is unlikely that the issues and our choices will become easier as information technology continues to transform our world. The growth of the Internet and the information economy suggests that all the ethical and social issues we have described will be heightened further as we move into the first digital century.

CHAPTER3: INFORMATION TECHNOLOGY INFRASTRUCTURE

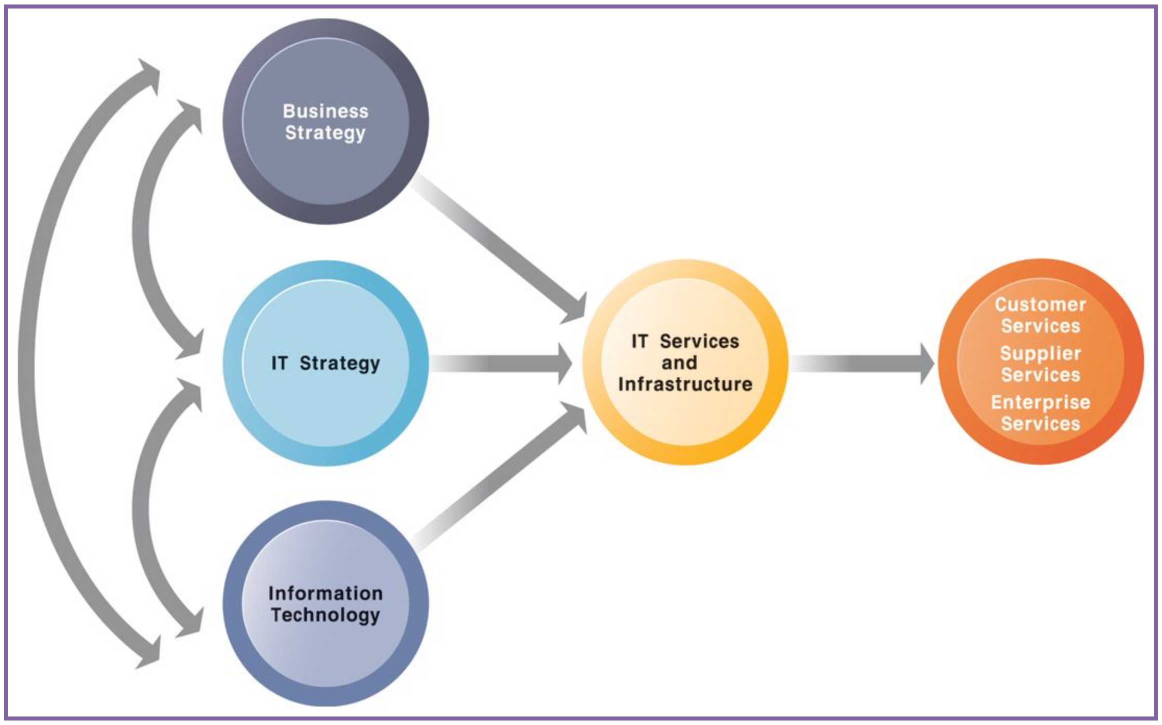
IT infrastructure is:

-Set of physical devices and software required to operate enterprise

-Set of firmwide services including: Computing platforms providing computing services; Telecommunications services; Data management services; Application software services, Physical facilities management services, IT management, standards, education, research and development services.

-Service platform" perspective more accurate view of value of investments

FIGURE 3-1 : CONNECTION BETWEEN THE FIRM, IT INFRASTRUCTURE, AND BUSINESS CAPABILITIES.



The services a firm is capable of providing to its customers, suppliers, and employees are a direct function of its IT infrastructure. Ideally, this infrastructure should support the firm's business and information systems strategy.

New information technologies have a powerful impact on business and IT strategies, as well as the services that can be provided to customers.

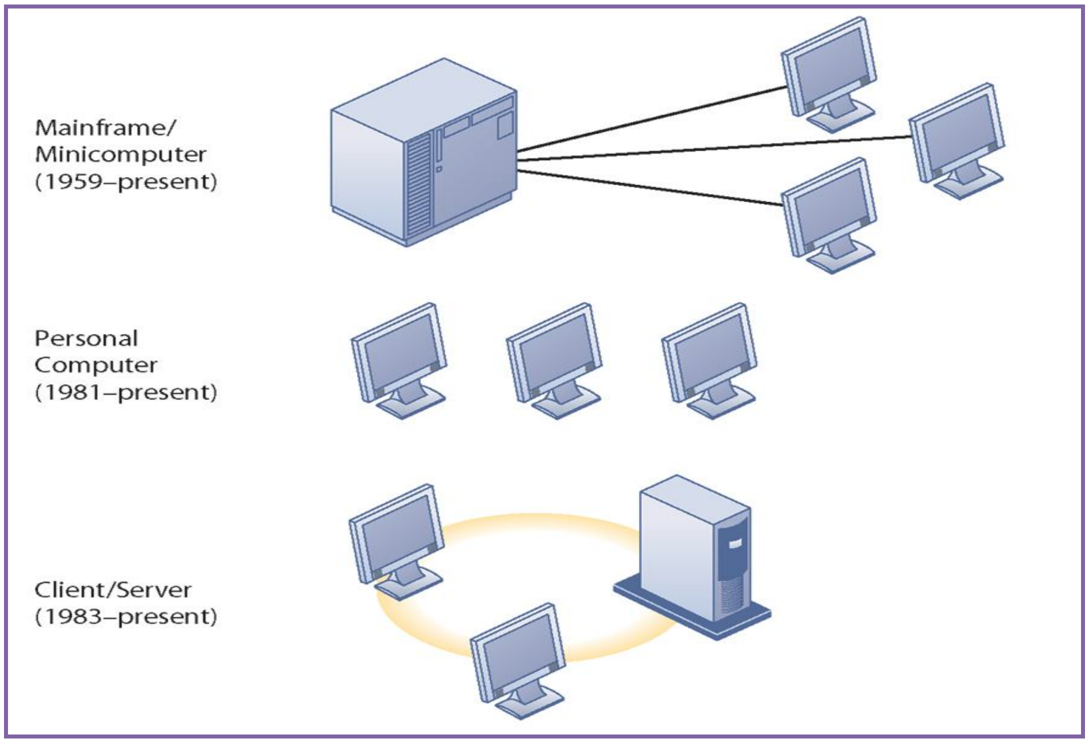
Evolution of IT infrastructure

- **General-purpose mainframe & minicomputer era: 1959 to present**: 1958 IBM first mainframes introduced, 1965 Less expensive DEC minicomputers introduced
- **Personal computer era: 1981 to present**: 1981 Introduction of IBM PC; Proliferation in 80s, 90s resulted in growth of personal software.
- **Client/server era: 1983 to present**
    - Desktop clients networked to servers, with processing work split between clients and servers
    - Network may be two-tiered or multitiered (N-tiered)
    - Various types of servers (network, application, Web)

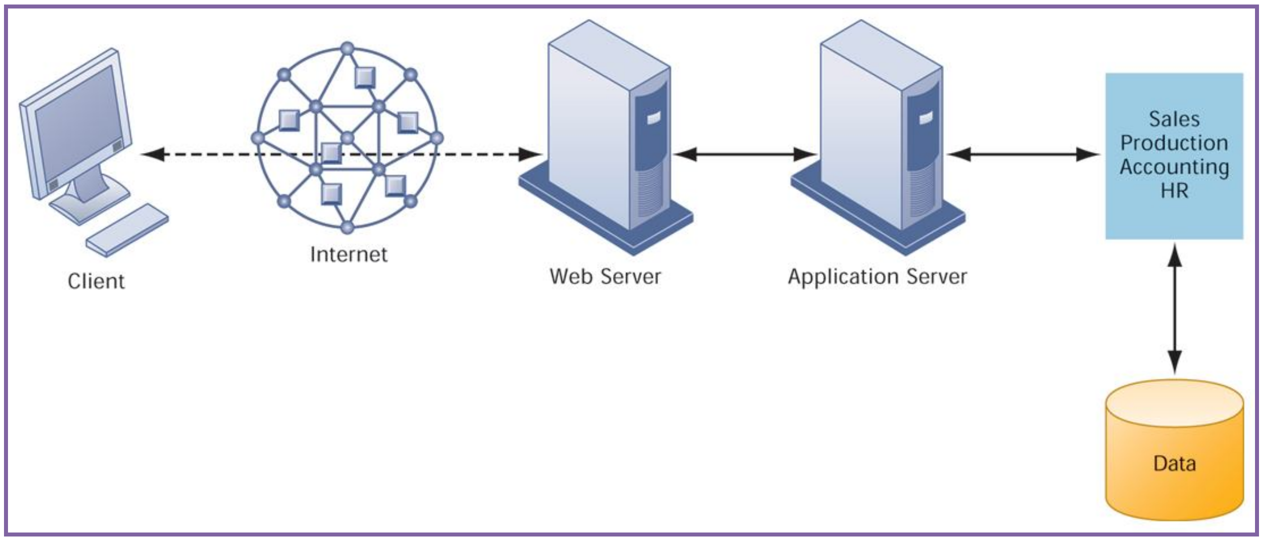
STAGES IN IT INFRASTRUCTURE EVOLUTION

Illustrated here are the typical computing configurations characterizing each of the five eras of IT infrastructure evolution

FIGURE 3-2



A MULTITIERED CLIENT/SERVER NETWORK (N-TIER)

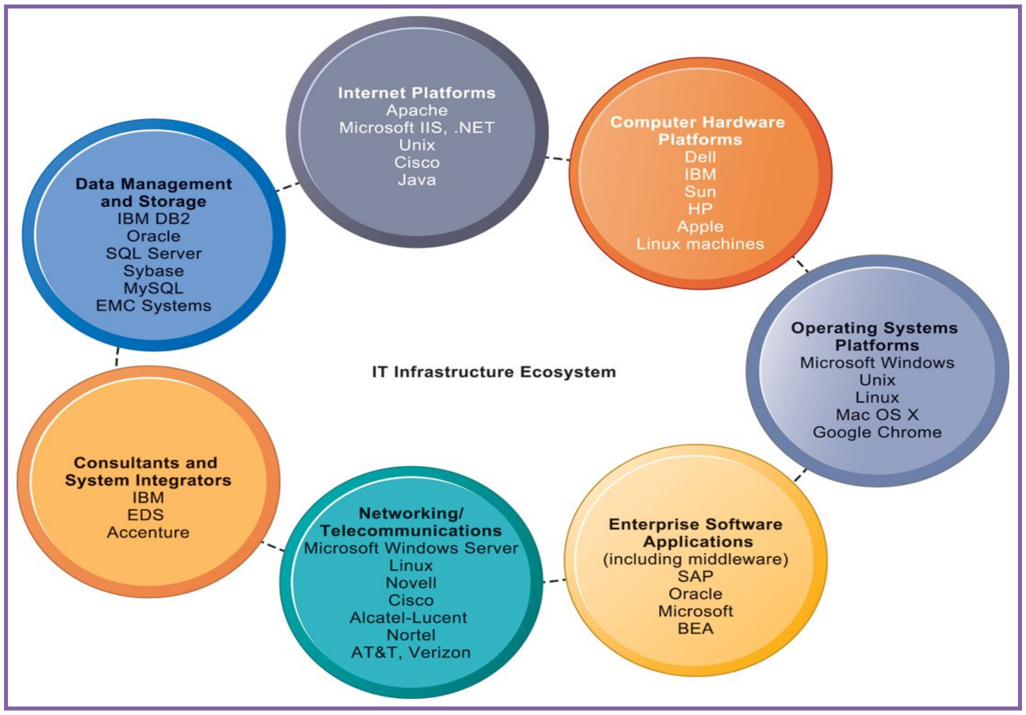


INFRASTRUCTURE COMPONENTS

- **IT Infrastructure has 7 main components**
    1. Computer hardware platforms
    2. Operating system platforms
    3. Enterprise software applications
    4. Data management and storage
    5. Networking/telecommunications platforms
    6. Internet platforms
    7. Consulting system integration services

THE IT INFRASTRUCTURE ECOSYSTEM

There are seven major components that must be coordinated to provide the firm with a coherent IT infrastructure. Listed here are major technologies and suppliers for each component.



- **Computer hardware platforms**
  - **Client machines**
    - Desktop PCs, mobile devices – PDAs, laptops
  - **Servers**
    - Blade servers: ultrathin computers stored in racks
  - **Mainframes:**
    - IBM mainframe equivalent to thousands of blade servers
  - **Top chip producers: AMD, Intel, IBM**
  - **Top firms: IBM, HP, Dell, Sun Microsystems**
- **Operating system platforms**
  - **Operating systems**
    - Server level: 75% run Windows; 25% run Unix or Linux
    - Client level:
    - 90% run Microsoft Windows (XP, 2000, CE, etc.)
    - Handheld device OS's (Android, iPhone OS)
    - Cloud computing OS's (Google's Chrome OS)

- **Enterprise software applications**
  - Enterprise application providers: SAP and Oracle
  - Middleware providers: BEA
- **Data management and storage**
  - **Database software:**
    - IBM (DB2), Oracle, Microsoft (SQL Server), Sybase (Adaptive Server Enterprise), MySQL
  - **Physical data storage:**
    - EMC Corp (large-scale systems), Seagate, Maxtor, Western Digital
  - **Storage area networks (SANs):**
    - Connect multiple storage devices on dedicated network

- **Networking/telecommunications platforms**
  - **Telecommunication services**
    - Telecommunications, cable, telephone company charges for voice lines and Internet access
    - AT&T, Verizon
  - **Network operating systems:**
    - Windows Server, Novell, Linux, Unix
  - **Network hardware providers:**

    Cisco, Alcatel-Lucent, Nortel, Juniper Networks

- **Internet platforms**
  - Hardware, software, management services to support company Web sites, (including Web hosting services) intranets, extranets
  - Internet hardware server market: Dell, HP/Compaq, IBM
  - Web development tools/suites: Microsoft (FrontPage, .NET) IBM (WebSphere) Sun (Java), independent software developers: Adobe, RealMedia
- **The emerging mobile digital platform**
  - **Cell phones, smartphones (BlackBerry, iPhone)**
    - Have assumed data transmission, Web surfing, e-mail and IM duties

  - **Netbooks:**
    - Small, low-cost lightweight notebooks optimized for wireless communication and core computing tasks
  - **Tables (iPad)**

    Networked e-readers (Kindle)

- **Grid computing**
  - Connects geographically remote computers into a single network to combine processing power and create virtual supercomputer
  - Provides cost savings, speed, agility
- **Virtualization**
  - Allows single physical resource to act as multiple resources (i.e., run multiple instances of OS)
  - Reduces hardware and power expenditures
  - Facilitates hardware centralization

CHAPTER 4: CLASSIFICATION OF INFORMATION SYSTEMS

4.1 Types of Information Systems

In the previous section, we defined “information system.” Many types of information systems exist on the market. To illustrate this, this section first provides a broad classification of information systems. We then narrow our view to enterprise information systems and present for this class of information systems an overview of existing types of software systems. Moreover, we provide examples of typical enterprise information systems in various industries.

4.1.1Classifying Information Systems

It is ambitious to classify the many types of information systems that have emerged in Practice. The problem is that classification is in flux; that is, a classification developed a few years ago is not necessarily current. As another and main limiting factor, the categories of a classification are typically not disjointed: one type of information system belongs to multiple categories. Given these problems, we present a high-level classification that distinguishes three classes of information systems.

The first class of information systems is *personal information systems*. Such an information

system can manage and store information for a private person. Examples are an address book or address database and an audio CD collection. *Enterprise* (or organizational) *information systems* are the second class of information systems. An enterprise information system is tailored toward the support of an organization. We distinguish between *generic* types and technologies of information systems and information systems for *certain* types of organizations. The former class of enterprise information systems supports functionality that can be used by a wide range

of organizations. Examples are workflow management systems, enterprise resource planning systems, data warehouse systems, and geographic information systems. In contrast, information systems for certain types of organizations offer functionality that is tailored toward certain industries or organizations. Examples are hospital information systems, airline reservation systems, and electronic learning systems.The third class of information systems is *public* *information systems*. Unlike personal information systems, public information systems can manage and store information that can be accessed by a community. Public libraries, information systems for museums, Web-based community information systems, and Web-based stock-portfolio information systems are examples of public information systems. These systems play a

crucial role in a wide variety of organizations and have an enormous economic value.The complexity and importance of such systems provide serious challenges for IT professionals ranging from software engineers to management consultants. Business processes and business process models play a dominant role in enterprise information systems. This explains why business process modeling is the focus of later chapters.

Types of Enterprise Information Systems

There are many types of enterprise information systems in practice. This section gives an overview of the most important types.

Enterprise Resource Planning Systems

An *enterprise resource planning* (ERP) *system* is an information system that supports the main business processes of an organization for example, human resource management, sales, marketing, management, financial accounting, controlling, and logistics. In the past, each business process was encapsulated in a separate information system. As most of these business processes use related data, much redundant data had to be stored within the respective information systems. The increasing number and complexity of information systems forced organizations to spend much effort in synchronizing the data of all information systems. An ERP system is a solution to overcome these synchronization efforts by integrating different information systems. It is a software system that is built on a distributed computing platform including one or more database management systems. The computing platform serves as an infrastructure on which the individual business processes are implemented. First-generation ERP systems now run the complete back office functions of the world’s largest corporations. ERP systems run typically in a three-tier client/server architecture consisting of a user interface (or presentation) tier, an application server tier, and a database server tier. ERP systems provide multi-instance database management, configuration management, and version (or customization) management for the underlying database schema, for the user interface, and for the many application programs associated with them. As ERP systems are typically designed for multinational companies, they have to support multiple languages, multiple currencies, and country-specific business practices.

Procurement Systems

A *procurement system* is an information system that helps an organization automate the purchasing process. The aim of a procurement system is to acquire what is needed to keep the business processes running at minimal cost. With the available inventory, the expected arrival of ordered goods, and forecasts based on sales and production plans, the procurement system determines the requirements and generates new orders. At the same time, it tracks whether ordered goods arrive. The key point is to order the right amount of material at the right time from the right source. If the material arrives too early, money for buying the material and warehouse space to store the material will be tied up. If, in contrast, the material arrives too late, then production is disrupted. Hence, the goal is to balance reducing inventory costs with reducing the risk of out-of-stock situations. Procurement is an important ingredient of *supply chain management* (SCM), in which coordination of the purchasing processes is not limited to two actors. Instead, SCM aims at closely coordinating an organization with its suppliers so that inefficiencies are avoided by optimizing the entire purchasing process. For example, by synchronizing the production process of an organization with its suppliers, all parties may reduce their inventories. The market leader in the SCM market is SAP with SAP SCM; competitors

are Oracle and JDA Software .

Procurement is related to *electronic data interchange* (EDI), the electronic exchange of information based on a standard set of messages. EDI can be used to avoid delays and errors in the procurement process as a result of rekeying information. In the classical (pre-EDI) situation, a purchase order is entered into the procurement system of one organization, it is printed, and the printed purchase order is sent to the order processing department or to another organization. The information on the printed purchase order is then reentered into the procurement system. By using EDI or technology such as Web services, organizations can automate these parts of the procurement process. The purchase order is electronically sent to the processing department or to the other organization. This automation makes the overall procurement process faster and less

Error-prone, thereby reducing the costs for each purchase order.

Manufacturing Systems

Manufacturing systems support the production processes in organizations. Driven by information, such as the bill of materials (BOM), inventory levels, and available capacity, they plan the production process. With increasing automation of production processes, manufacturing systems have become more and more important. For example, most steps in the production line of a car, such as welding the auto body, are performed by robots. This requires precise scheduling and material movement and, hence, a manufacturing system that supports these processes. *Material requirements planning* (MRP) is an approach to translate requirements (i.e.,

the number of products for each period), inventory status data, and the BOM into a production plan without considering capacities. Successors, such as *manufacturing resources planning* (MRP2), also take capacity information into account. Software based on MRP and MRP2 has been the starting point for many ERP systems. Consider an organization that produces different flavors of yogurt (e.g., strawberry, peach, and pear). The organization has several machines to produce yogurt; each machine can produce any flavor. Production planning means scheduling each machine for the flavor of yogurt it must produce. The production plan depends on the demand for each flavor and on the delivery of ingredients. Furthermore, each machine has to be cleaned at regular intervals and when the production changes to a new flavor. Calculating a production plan is a complex optimization problem, often depending on several thousand constraints. Consequently, the aim is to find a good solution rather than an optimum solution.

Sales and Marketing Systems

Sales and marketing systems need to process customer orders by taking into account issues such as availability. These systems are driven by software addressing the four *p*'s: product, price, place, and promotion. Organizations undertake promotional activities and offer their products at competitive prices to boost sales, but a product that is not available or not at the right location cannot be sold. One prominent example of a promotional activity is a bonus card in supermarkets.

Customers who register for a bonus card get a discount or a voucher. Bonus cards are an instrument for organizations to obtain personal data about their customers (e.g., age, address) and data about the buying behavior of customers (i.e., what they buy and when they buy it). These data are collected and processed by an information system. The information extracted from these data can help to improve marketing and to determine the range of products to offer.

Delivery Systems

A *delivery system* is an information system that supports the delivery of goods to customers. The task of these systems is to plan and schedule when and in what order customers receive their products. Consider, for example, a transportation company with hundreds of trucks. The planning of trips, the routing of these trucks, and reacting to on-the-fly changes require dedicated software. Creating an optimal schedule is a complex optimization problem. As circumstances for example, traffic jams and production problems—may force rescheduling, contemporary delivery systems aim to find a good solution rather than a theoretical optimum solution. More and more delivery systems offer tracking-and-tracing functionality; for example, customers of package delivery companies, such as UPS, can track down the location of a specific parcel via the Internet.

Finance Systems Among the oldest information systems are *finance systems*. These systems

support the flow of money within and between organizations. Finance systems typically provide accounting functionality to maintain a consistent and auditable set of books for reporting and management support. Another important application of finance systems is the stock market. At a stock market, dedicated information systems are essential to process the operations. Again, the functionality of finance systems is absorbed by ERP systems. The origin of the SAP system, for example, was in finance rather than production planning.

Product Design Systems

Enterprise information systems not only support the production of products, they also support the design of products. Examples are *computer-aided design* (CAD) *systems* and *product data management* (PDM) *systems*. CAD systems support the graphical representation and the design of product specifications. PDM systems support the design process in a broader sense by managing designs and their documentation. Typically, there are many versions of the same design, and designs of different components need to be integrated. To support such complex concurrent engineering processes, PDM systems offer versioning functionality.

Workflow Management Systems

Many organizations aim to automate their business processes. To this end, they have to specify in which order the activities of a business process must be executed and which person has to execute an activity at which time. A *workflow* refers to the automation of a business process, in whole or in part. Each activity of the workflow is implemented as software. The workflow logic specifies the order of the activities. A *workflow management system* (WfMS) is an information system that defines, manages, and executes workflows. The execution order of the workflow's activities is driven by a computer representation of the workflow logic. The ultimate goal of workflow management is to make sure that the proper activities are executed by the right people at the right time.

Data Warehouses

A *data warehouse* is a large database that stores historical and upto date information from a variety of sources. It is optimized for fast query answering. To allow this, there are three continuous processes: The first process extracts data at regular intervals from its information sources, loads the data into auxiliary tables, and then cleans and transforms the loaded data to make it suitable for the data warehouse schema. Processing queries from users and from data analysis applications is the task of the second process. The third process archives the information that is no longer needed by means of tertiary storage technology. Nowadays, most organizations employ information systems for financial accounting, purchasing, sales and inventory management, production planning, and management control. To efficiently use the vast amount of information that these operational systems have been collecting over the years for planning and decision-making purposes, the information from all relevant sources must be merged and consolidated in a data warehouse.

Business Intelligence Systems

A *business intelligence system* provides tools to analyze the performance that is, the efficiency and the effectiveness of *running* business processes.

These tools extract information on the business processes from the data available in an organization. Different tools and techniques exist, among them business performance management, business activity monitoring, querying and reporting, data mining, and process mining. *Business performance management* concentrates on improving the performance of business processes. The goal is to extract information from the history of running business processes and to display this information on a management dashboard. For example, one could monitor a credit approval process to get insight into the length of time required to make the decision. In contrast to business performance management, *business activity monitoring* aims

at providing real-time information on business processes and the activities in these business processes. The goal is to support decision making at runtime. Such a tool may monitor inventory levels, response times, or queues and take action whenever needed. *Querying and reporting* tools explore data (e.g., stored in a data warehouse) to provide insight into efficiency and effectiveness of business processes and trends in the environment. Typically, statistical analysis is applied to the data to distinguish between trends and isolated events.

REFERENCES

1. Beynon-Davies P., 2004. Database systems 3rd edition. Palgrave, Basingstoke, UK. ISBN 1-4039-1601-2.

2. DuBois P., 2001. MySQL Reference Manual, New Riders, ISBN0735709211, page 2

3. Jeffrey Ullman, 1997. First Course in Database systems, Prentice-Hall Inc., Simon & Schuster, Page 1, ISBN 0-13-861337-0.

4. Ramez E., et al. 2004: Fundamentals of Database Systems 3rd. ed. sect. 17.5, p. 582

- Database System Concepts 5th Edition

- http://www.w3schools.com/