CHAPTER ONE

BASIC DBMS CONCEPTS

WHY DATABASES?

Imagine trying to operate a business without knowing who your customers are, what products you are selling, who is working for you, who owes you money, and whom you owe money. All businesses have to keep this type of data and much more; and just as importantly, they must have those data available to decision makers when they need them.

The ultimate purpose of all business information systems is to help businesses use information as an organizational resource. At the heart of all of these systems, key activities involve collection, storage, aggregation, manipulation, dissemination, and management of data.

Depending on the type of information system, characteristics and size of the business, these data could vary in size.

It is estimated that the Google responds to over 91 million searches per day across a collection of data that is several terabytes in size. Impressively, the results of these searches are available nearly instantly. How does Google process this much data? How can they store it all, and then quickly retrieve just the facts that decision makers want to know, just when they want to know it? The answer is that they use databases.

Databases are specialized structures that allow computer-based systems to store, manage, and retrieve data very quickly.

Virtually all modern business systems rely on databases; therefore, a good understanding of how these structures are created and their proper use is vital for any information systems professional.

Even if your IT career does not take you down the amazing path of database design and development, databases will be a key component underpinning the systems that you work with. In any case, it is very likely that, in your career, you will be making decisions based on information generated from data. Thus, it is important that you know the difference between data and information.

DATA Vs INFORMATION

To understand what drives database design, you must understand the difference between data and information.

Data are raw facts.

The word *raw* indicates that the facts have not yet been processed to reveal their meaning.

Information is the result of processing raw data to reveal its meaning.

Data processing can be as simple as organizing data to reveal patterns or as complex as making forecasts or drawing inferences using statistical modeling.

In this "information age," production of accurate, relevant, and timely information is the key to good decision making.

In turn, *good decision making* is the key to business survival in the highly competitive global market. We are now said to be entering the "knowledge age."

Data are the foundation of information, which is the bedrock of *knowledge*—that is, the body of information and facts about a specific subject.

Let's summarize some key points:

_ Data constitute the building blocks of information.

_ Information is produced by processing data.

_ Information is used to reveal the meaning of data.

_ Accurate, relevant, and timely information is the key to good decision making.
_ Good decision making is the key to organizational survival in a global environment.
Timely and useful information requires accurate data. Such data must be properly generated and stored in a format that is easy to access and process. And, like any basic resource, the data environment must be managed carefully.
Data management is a discipline that focuses on the proper generation, storage, and retrieval of data. Given the crucial role that data play, data management is a core activity for any business, government, agency, service or organization

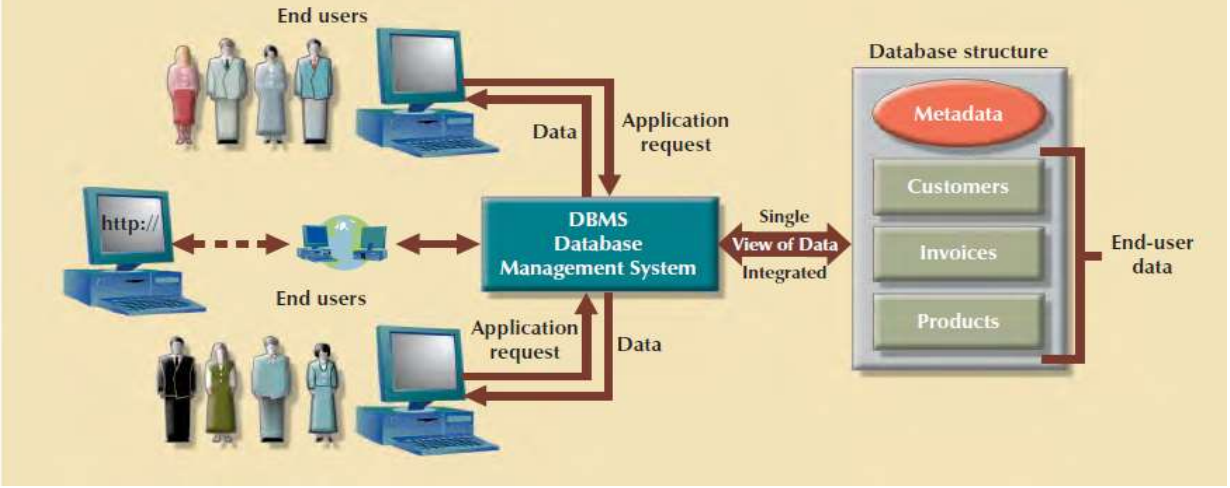
INTRODUCING THE DATABASE
Efficient data management typically requires the use of a computer database.
A **database** is a shared, integrated computer structure that stores a collection of:

- ✓ End-user data, that is, raw facts of interest to the end user.
- ✓ **Metadata**, or data about data, through which the end-user data are managed.

The metadata provide a description of the data characteristics and the set of relationships that links the data found within the database. For example, the metadata component stores information such as the name of each data element, the type of values (numeric, dates, or text) stored on each data element, whether or not the data element can be left empty, and so on.
The metadata present a more complete picture of the data in the database.
A **database management system** (**DBMS**) is a collection of programs that manages the database structure and controls access to the data stored in the database.
In a sense, a database resembles a very well-organized electronic filing cabinet in which powerful software, known as a *database management system*, helps manage the cabinet's contents.

Role and Advantages of the DBMS
The DBMS serves as the intermediary between the user and the database. The database structure itself is stored as a collection of files, and the only way to access the data in those files is through the DBMS. The DBMS presents the end user (or application program) with a single, integrated view of the data in the database. The DBMS receives all application requests and translates them into the complex operations required to fulfill those requests. The DBMS hides much of the database's internal complexity from the application programs and users.
The application program might be written by a programmer using a programming language such as Visual Basic, Java, or PhP.

FIGURE 1.2 The DBMS manages the interaction between the end user and the database



Having a DBMS between the end user's applications and the database offers some important advantages.

First, the DBMS enables the data in the database *to be shared* among multiple applications or users.

Second, the DBMS *integrates* the many different users' views of the data into a single all-encompassing data repository.

Because data are the crucial raw material from which information is derived, you must have a good method to manage such data. DBMS helps make data management more efficient and effective.

A DBMS provides advantages such as:

- ✓ *Improved data sharing* The DBMS helps create an environment in which end users have better access to more and better-managed data. Such access makes it possible for end users to respond quickly to changes in their environment.
- ✓ *Improved data security*. The more users access the data, the greater the risks of data security breaches. DBMS provides a framework for better enforcement of data privacy and security policies.
- ✓ *Better data integration*. Wider access to well-managed data promotes an integrated view of the organization's operations and a clearer view of the big picture. It becomes much easier to see how actions in one segment of the company affect other segments.
- ✓ *Minimized data inconsistency*. **Data inconsistency** exists when different versions of the same data appear in different places. For example, data inconsistency exists when a company's regional sales office shows the price of a product as KSh 450 and its national sales office shows the same product's price as KSh 400. The probability of data inconsistency is greatly reduced in a properly designed database.
- ✓ *Improved data access*. The DBMS makes it possible to produce quick answers to ad hoc queries. From a database perspective, a **query** is a specific request issued to the DBMS for data manipulation—for example, to read or update the data. Simply put, a query is a question, and an **ad hoc query** is a spur-of-the-moment question. The DBMS sends back an answer (called the **query result set**) to the application.

  For example, end users, when dealing with large amounts of sales data, might want quick answers to questions (ad hoc queries) such as:
  - What was the volume of sales by product during the past six months?
  - What is the sales commission for each sales people during the past three months?

- How many of our customers have credit balances of KSh 3,000 or more?

✓ *Improved decision making* Better-managed data and improved data access make it possible to generate better-quality information, on which better decisions are based.
The quality of the information generated depends on the quality of the underlying data.
Data quality is a measure of accuracy, validity, and timeliness of the data.
While the DBMS does not guarantee data quality, it provides a framework to facilitate data quality initiatives.

✓ *Increased end-user productivity* The availability of data, combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between success and failure in the global economy.

Types of Databases

A DBMS can support many different types of databases.
Databases can be classified according to:-

- *The number of users,*
- *The database location(s), and*
- *The expected type and extent of use.*

a) According to number of users
   - The number of users determines whether the database is classified as single-user or multiuser.
     i) **Single-user database** supports only one user at a time. In other words, if user A is using the database, users B and C must wait until user A is done.
     A single-user database that runs on a personal computer is called a **desktop database**.
     ii) **Multiuser database** supports multiple users at the same time.
     When the multiuser database supports a relatively small number of users (usually fewer than 50) or a specific department within an organization, it is called a **workgroup database**.
     When the database is used by the entire organization and supports many users (more than 50, usually hundreds) across many departments, the database is known as an **enterprise database**.

b) According to Location
   i) **Centralized database** is a database that supports data located at a single site
   ii) **Distributed database** is a database that supports data distributed across several different sites.

c) According to use
   - The most popular way of classifying databases today, however, is based on how they will be used and on the time sensitivity of the information gathered from them.
     - A database that is designed primarily to support a company's day-to-day operations is classified as an **operational database** (sometimes referred to as a **transactional** or **production database**).
     - **Data warehouse** focuses primarily on storing data used to generate information required to make tactical or strategic decisions. Such decisions typically require extensive "data massaging" (data manipulation) to extract information to formulate pricing decisions, sales forecasts, market positioning, and so on.
   - Most decision support data are based on data obtained from operational databases over time and stored in data warehouses.
   - Additionally, the data warehouse can store data derived from many sources. To make it easier to retrieve such data, the data warehouse structure is quite different from that of an operational or transactional database.

WHY DATABASE DESIGN IS IMPORTANT

Database design refers to the activities that focus on the design of the database structure that will be used to store and manage end-user data.

A database that meets all user requirements does not just happen; its structure must be designed carefully. In fact, database design is such a crucial aspect of working with databases.

Even a good DBMS will perform poorly with a badly designed database.

Proper database design requires the designer to identify precisely the database's expected use.

Designing a transactional database emphasizes accurate and consistent data and operational speed.

PROBLEMS WITH FILE SYSTEM DATA PROCESSING

The file system method of organizing and managing data was a definite improvement over the manual system, and the file system served a useful purpose in data management for over two decades Nonetheless, many problems and limitations became evident in this approach. A critique of the file system method serves two major purposes:

- Understanding the shortcomings of the file system enables you to understand the development of modern databases.
- Many of the problems are not unique to file systems. Failure to understand such problems is likely to lead to their duplication in a database environment.

The following problems associated with file systems, severely challenge the types of information that can be created from the data as well as the accuracy of the information:

- ✓ *Lengthy development times.* The most glaring problem with the file system approach is that even the simplest data-retrieval task requires extensive programming.
- ✓ *Difficulty of getting quick answers.* The need to write programs to produce even the simplest reports makes ad hoc queries impossible. If you need the information now, getting it next week or next month will not serve your information needs.
- ✓ *Complex system administration.* System administration becomes more difficult as the number of files in the system expands. Even a simple file system with a few files requires creating and maintaining several file management programs (each file must have its own file management programs that allow the user to add, modify, and delete records; to list the file contents; and to generate reports).
- ✓ *Lack of security and limited data sharing.* Another fault of a file system data repository is a lack of security and limited data sharing. Data sharing and security are closely related. Sharing data among multiple geographically dispersed users introduces a lot of security risks.
  Security and data-sharing features are difficult to program and are, therefore, often omitted in a file system environment.
- – *Extensive programming.* Making changes to an existing file structure can be difficult in a file system environment.
  Any change to a file structure, no matter how minor, forces modifications in all of the programs that use the data in that file. Modifications are likely to produce errors (bugs), and additional time is spent using a debugging process to find those errors. Those limitations, in turn, lead to problems of structural and data dependence

Structural and Data Dependence

- – A file system exhibits **structural dependence**, which means that access to a file is dependent on its structure.

For example, adding a customer date-of-birth field (if it didn't exist) to the CUSTOMER file would mean none of the previous programs will work with the new CUSTOMER file structure. Therefore, all of the file system programs must be modified to conform to the new file structure.
In short, because the file system application programs are affected by change in the file structure, they exhibit structural dependence.

- **Structural independence** exists when it is possible to make changes in the file structure without affecting the application program's ability to access the data.
- **Data dependence** exists when changes in the characteristics of data, require changes in all the programs that access the file.
  For example, when a change in the data type of admission number field (in the STUDENTS File) from integer to text, requires a change in all programs that accesses the students file
- **Data independence** exists when it is possible to make changes in the data storage characteristics without affecting the application program's ability to access the data.

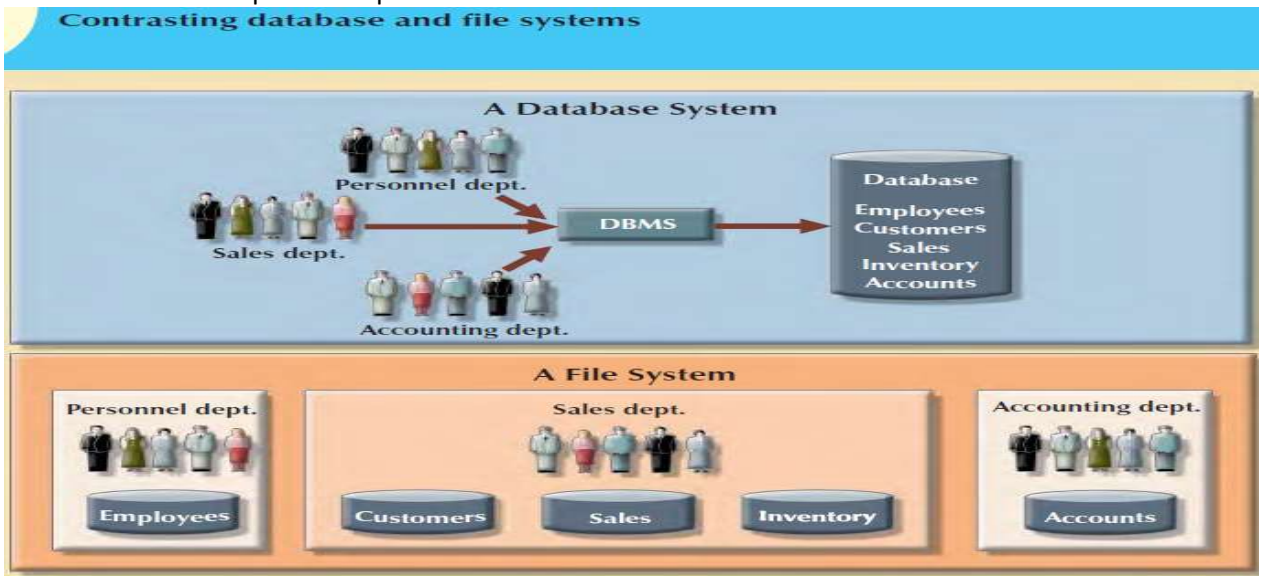
Data Redundancy

Data redundancy exists when the same data are stored unnecessarily at different places.
Uncontrolled data redundancy sets the stage for:

- ✓ *Poor data security*. Having multiple copies of data increases the chances for a copy of the data to be susceptible to unauthorized access.
- ✓ *Data inconsistency*. Data inconsistency exists when different and conflicting versions of the same data appear in different places.
  For example, suppose you change student's course code in the STUDENTS file. If you forget to make corresponding changes in the FEES PAYMENT file, the files contain different data for the same student. Reports will yield inconsistent results that depend on which version of the data is used.
  Note
  Data that display data inconsistency are also referred to as data that lack data integrity.
  Data integrity is the condition in which all of the data in the database are consistent with the real-world events and conditions.
  In other words, data integrity means that:
  • Data are *accurate*—there are no data inconsistencies.
  • Data are *verifiable*—the data will always yield consistent results.
- ✓ *Data anomalies*. The dictionary defines *anomaly* as "an abnormality."
  Ideally, a field value change should be made in only a single place. Data redundancy, however, fosters an abnormal condition by forcing field value changes in many different locations to maintain data integrity.
  A **data anomaly** develops when **not all** of the required changes in the redundant data are made successfully.
  The common data anomalies found are defined as follows:
  - *Update anomalies*. If student AMOS MWENDA has an additional name MUTWIRI, that name must be entered in each of the STUDENT file records in which AMOS MWENDA's name is shown. In a large file system, such a change might occur in hundreds or even thousands of records. Clearly, the potential for data inconsistencies is great.
  - *Insertion anomalies*. If only the CUSTOMER file existed, to add a new agent, you would also add a dummy customer data entry to reflect the new agent's addition. Again, the potential for creating data inconsistencies would be great.
  - *Deletion anomalies*. If you delete student AMOS MWENDA you may also delete have to delete all his related data, in FEES PAYMENT, ACDEMICS FILE etc. Clearly, this is not desirable.

DATABASE SYSTEMS

- The problems inherent in file systems make using a database system very desirable.
- Unlike the file system, with its many separate and unrelated files, the database system consists of logically related data stored in a single logical data repository.
- Because the database's data repository is a single logical unit, the database represents a major change in the way end-user data are stored, accessed, and managed.
- The database's DBMS provides numerous advantages over file system management, by making it possible to eliminate most of the file system's data inconsistency, data anomaly, data dependence, and structural dependence problems.



THE DATABASE SYSTEM ENVIRONMENT (COMPONENTS)

- DBMS is just one of several crucial components of a database system. The DBMS may even be referred to as the database system's heart, but it takes more than a DBMS to make a database system function.
- The term **database system** refers to an organization of components that define and regulate the collection, storage, management, and use of data within a database environment.
- The database system is composed of the five major
    - Hardware, software, people, procedures, and data.

1. *Hardware*
    - Hardware refers to all of the system's physical devices; for example, computers (PCs, workstations, servers, and supercomputers), storage devices, printers, network devices (hubs, switches, routers, fiber optics), and other devices (automated teller machines, ID readers, and so on).
2. *Software*.
    - For a database system to function fully, three types of software are needed:
        - *Operating system software, DBMS software, and application programs and utilities.*
            a) *Operating system software* manages all hardware components and makes it possible for all other software to run on the computers
            b) *DBMS software* manages the database within the database system.

c) *Application programs and utility software* are used to access and manipulate data in the DBMS and to manage the computer environment in which data access and manipulation take place.
Application programs are most commonly used to access data found within the database to generate reports, tabulations, and other information to facilitate decision making.
Utilities are the software tools used to help manage the database system's computer components. For example, all of the major DBMS vendors now provide graphical user interfaces (GUIs) to help create database structures, control database access, and monitor database operations.

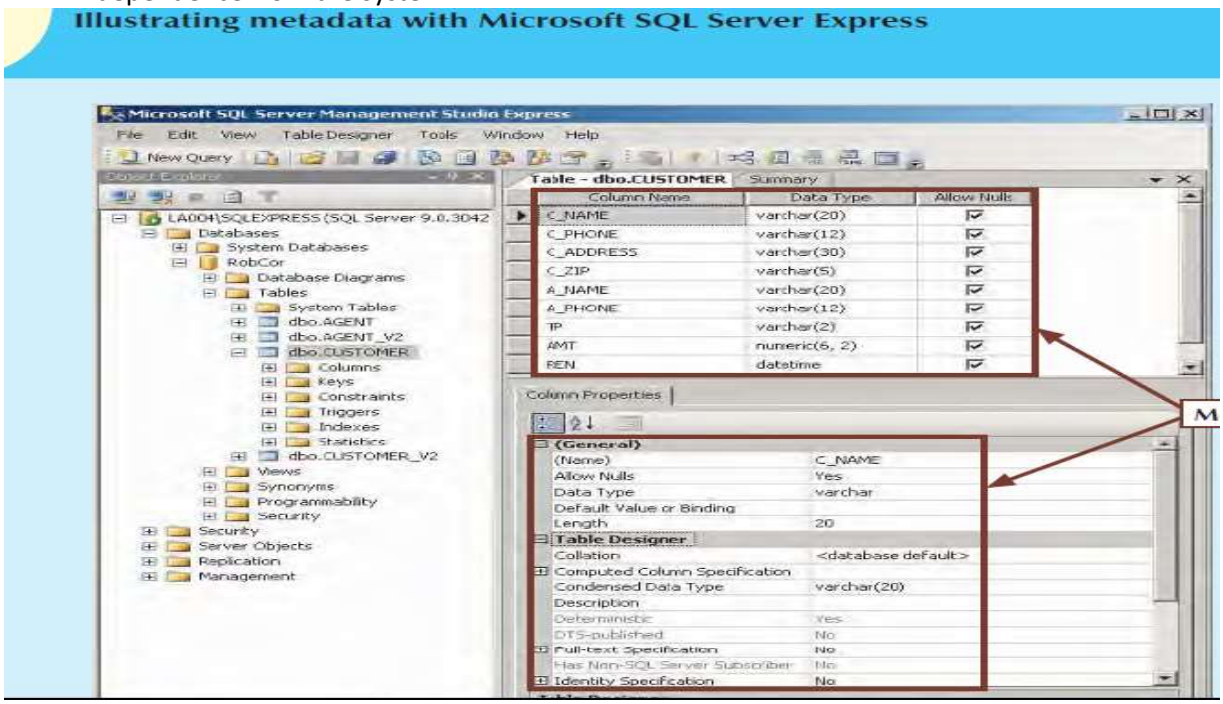
3. *People*
   - This component includes all users of the database system.
   - Primarily, FIVE types of users can be identified in a database system:
     - *System administrators, database administrators, database designers, system analysts and programmers, and end users.*
       Each user type, described below, performs both unique and complementary functions.
       a) *System administrators* oversee the database system's general operations.
       b) *Database administrators*, also known as DBAs, manage the DBMS and ensure that the database is functioning properly.
       c) *Database designers* design the database structure. They are, in effect, the database architects. If the database design is poor, even the best application programmers and the most dedicated DBAs cannot produce a useful database environment.
       d) *System analysts and programmers* design and implement the application programs. They design and create the data entry screens, reports, and procedures through which end users access and manipulate the database's data.
       e) *End users* are the people who use the application programs to run the organization's daily operations.
       For example, salesclerks, supervisors, managers, and directors are all classified as end users. High-level end users employ the information obtained from the database to make tactical and strategic business decisions.

4. *Procedures*.
   - Procedures are the instructions and rules that govern the design and use of the database system.
   - Procedures play an important role in a company because they enforce the standards by which business is conducted within the organization and with customers.
   - Procedures are also used to ensure that there is an organized way to monitor and audit both the data that enter the database and the information that is generated through the use of those data.
5. *Data*
   - The word *data* covers the collection of facts stored in the database.
   - Because data are the raw material from which information is generated, the determination of what data are to be entered into the database and how those data are to be organized is a vital part of the database designer's job.

DBMS Functions

A DBMS performs several important functions that guarantee the integrity and consistency of the data in the database.
Most of those functions are transparent to end users, and most can be achieved only through the use of a DBMS.

1. *Data dictionary management*
   - The DBMS stores definitions of the data elements and their relationships (metadata) in a **data dictionary**.
   - In turn, all programs that access the data in the database work through the DBMS.
   - The DBMS uses the data dictionary to look up the required data component structures and relationships, thus relieving programmers much strain in coding programs.
   - Any changes made in a database structure are automatically recorded in the data dictionary
   - In other words, the DBMS provides data abstraction, and it removes structural and data dependence from the system.



2. *Data storage management*
   - The DBMS creates and manages the complex structures required for data storage, thus relieving developers from the difficult task of defining and programming the physical data characteristics.
   - Data storage management is also important for database performance tuning. **Performance tuning** relates to the activities that make the database perform more efficiently in terms of storage and access speed.
   - Although the user sees the database as a single data storage unit, the DBMS actually stores the database in multiple physical data files.
   - The DBMS can fulfill database requests concurrently.
3. *Data transformation and presentation*.
   - The DBMS transforms entered data to conform to required data structures.
   - The DBMS formats the physically retrieved data to make it conform to the user's logical expectations.
     For example, imagine an enterprise database used by a multinational company. An end user in England would expect to enter data such as July 11, 2010, as "11/07/2010." In contrast, the

same date would be entered in the United States as "07/11/2010." Regardless of the data presentation format, the DBMS must manage the date in the proper format for each country.

4. *Security management*.
   - The DBMS creates a security system that enforces user security and data privacy.
   - Security rules determine which users can access the database, which data items each user can access, and which data operations (read, add, delete, or modify) the user can perform. This is especially important in multiuser database systems.
   - All database users may be authenticated to the DBMS through a username and password or through biometric authentication such as a fingerprint scan.
   - The DBMS uses this information to assign access privileges to various database components such as queries and reports.
5. *Multiuser access control*
   - To provide data integrity and data consistency, the DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently without compromising integrity of the database.
6. *Backup and recovery management*
   - The DBMS provides backup and data recovery tools to ensure data safety and integrity.
   - Recovery management deals with the recovery of the database after a failure, such as a bad sector in the disk or a power failure. Such capability is critical to preserving the database's integrity.
7. *Data integrity management*
   - The DBMS promotes and enforces integrity rules, thus minimizing data redundancy and maximizing data consistency.
   - The data relationships stored in the data dictionary are used to enforce data integrity.
8. *Database access languages and application programming interfaces*
   - The DBMS provides data access through a query language.
   - A **query language** is a nonprocedural language—one that lets the user specify what must be done without having to specify how it is to be done.
   - **Structured Query Language** (**SQL**) is the de facto query language and data access standard supported by the majority of DBMS vendors.
   - DBMS also provides administrative utilities used by the DBA and the database designer to create, implement, monitor, and maintain the database.
9. *Database communication interfaces*
   - Current-generation DBMSs accept end-user requests via multiple, different network environments.
     For example, the DBMS might provide access to the database via the Internet through the use of Web browsers such as Mozilla Firefox or Microsoft Internet Explorer.

DISADVANTAGES OF DATABASE SYSTEMS

Although the database system yields considerable advantages over previous data management approaches, database systems do carry significant disadvantages.
For example:

1. ***Increased costs.***
   - Database systems require sophisticated hardware and software and highly skilled personnel.
   - The cost of maintaining the hardware, software, and personnel required to operate and manage a database system can be substantial.
2. ***Management complexity***

- Database systems interface with many different technologies and have a significant impact on a company's resources and culture.
- The changes introduced by the adoption of a database system must be properly managed to ensure that they help advance the company's objectives.
- Given the fact that database systems hold crucial company data that are accessed from multiple sources, security issues must be assessed constantly.

3. ***Maintaining currency***
   - To maximize the efficiency of the database system, you must keep your system current.
   - Therefore, you must perform frequent updates and apply the latest patches and security measures to all components.
   - Because database technology advances rapidly, personnel training costs tend to be significant.
4. ***Vendor dependence***
   - Given the heavy investment in technology and personnel training, companies might be reluctant to change database vendors.
5. ***Frequent upgrade/replacement cycles***
   - DBMS vendors frequently upgrade their products by adding new functionality. Such new features often come bundled in new upgrade versions of the software.
   - Some of these versions require hardware upgrades.
   - Not only do the upgrades themselves cost money, but it also costs money to train database users and administrators to properly use and manage the new features.

SUMMARY

- Data are raw facts. Information is the result of processing data to reveal its meaning. Accurate, relevant, and timely information is the key to good decision making, and good decision making is the key to organizational survival in a global environment.
- Data are usually stored in a database. To implement a database and to manage its contents, you need a database management system (DBMS). The DBMS serves as the intermediary between the user and the database. The database contains the data you have collected and "data about data," known as metadata.
- Database design defines the database structure. A well-designed database facilitates data management and generates accurate and valuable information. A poorly designed database can lead to bad decision making, and bad decision making can lead to the failure of an organization.
- Databases evolved from manual and then computerized file systems. In a file system, data are stored in independent files, each requiring its own data management programs. Although this method of data management is largely outmoded, understanding its characteristics makes database design easier to comprehend.
- Some limitations of file system data management are that it requires extensive programming, system administration can be complex and difficult, making changes to existing structures is difficult, and security features are likely to be inadequate. Also, independent files tend to contain redundant data, leading to problems of structural and data dependence.
- Database management systems were developed to address the file system's inherent weaknesses. Rather than depositing data in independent files, a DBMS presents the database to the end user as a single data repository. This arrangement promotes data sharing, thus eliminating the potential problem of islands of information. In addition, the DBMS enforces data integrity, eliminates redundancy, and promotes data security.

REVIEW QUESTIONS

1. Define each of the following terms:
   a. data
   b. field
   c. record
   d. file
2. What is data redundancy, and which characteristics of the file system can lead to it?
3. What is data independence, and why is it lacking in file systems?
4. What is a DBMS, and what are its functions?
5. What is structural independence, and why is it important?
6. Explain the difference between data and information.
7. What is the role of a DBMS, and what are its advantages? What are its disadvantages?
8. List and describe the different types of databases.
9. What are the main components of a database system?
10. What are metadata?
11. Explain why database design is important.
12. What are the potential costs of implementing a database system?
13. Use examples to compare and contrast unstructured and structured data. Which type is more prevalent in a typical business environment?
14. What are some basic database functions that a spreadsheet cannot perform?
15. What common problems does a collection of spreadsheets created by end users share with the typical file system?
16. Explain the significance of the loss of direct, hands-on access to business data that end users experienced with the advent of computerized data repositories.

CHAPTER TWO
DATA MODELS

DATA MODELING AND DATA MODELS

- Database design focuses on how the database structure will be used to store and manage end-user data. Data modeling, the first step in designing a database, refers to the process of creating a specific data model for a determined problem domain.
- A *problem domain* is a clearly defined area within the real-world environment, with well-defined scope and boundaries that is to be systematically addressed.
- A **data model** is a relatively simple representation, usually graphical, of more complex real-world data structures.
- In general terms, a *model* is an abstraction of a more complex real-world object or event.
- A model helps understand the complexities of the real-world environment.
- Within the database environment, a data model represents data structures and their characteristics, relations, constraints, transformations, and other constructs with the purpose of supporting a specific problem domain.

THE IMPORTANCE OF DATA MODELS

- Data models can facilitate interaction among the designer, the applications programmer, and the end user.
- A well-developed data model can even foster improved understanding of the organization for which the database design is developed. In short, data models are a communication tool.
- Data constitute the most basic information units employed by a system. Applications are only created to manage data and to help transform data into information.
- But data are viewed in different ways by different people.
  For example, contrast the (data) view of a company manager with that of a company clerk. Although the manager and the clerk both work for the same company, the manager is more likely to have an enterprise-wide view of company data than the clerk.
  Applications programmers have yet another view of data, being more concerned with data location, formatting, and specific reporting requirements. Basically, applications programmers translate company policies and procedures from a variety of sources into reports.

DATA MODEL BASIC BUILDING BLOCKS

- The basic building blocks of all data models are entities, attributes, relationships, and constraints.

Entity

- An **entity** is anything (a person, a place, a thing, or an event) about which data are to be collected and stored.
- An entity represents a particular type of object in the real world.
- Because an entity represents a particular type of object, entities are “distinguishable”—that is, each entity occurrence is unique and distinct.
  For example, a STUDENT entity would have many distinguishable student occurrences, such as John Murithi, Sally Ndiritu, Dorcas Mutegi, Flomina Makena, etc.
- Entities may be physical objects, such as customers or products, but entities may also be abstractions, such as flight routes or school terms.

Attribute

- An **attribute** is a characteristic of an entity.
- For example, a CUSTOMER entity would be described by attributes such as customer last name, customer first name, customer phone, customer address, and customer credit limit.
- Attributes are the equivalent of fields in file systems.

Relationship

- A **relationship** describes an association among entities.
  For example, a relationship exists between students and subjects that can be described as follows: a student can be taking many subjects, and one subject may be taken by several students
- Data models use three types of relationships: ***one-to-many, many-to-many, and one-to-one.***
- Database designers usually use the shorthand notations ***1:M or 1..*, M:N or *..*, and 1:1 or 1..1,*** respectively.
- Although the M:N notation is a standard label for the many-to-many relationship, the label M:M may also be used.

Examples

1. **One-to-many (1:M or 1..*) relationship**.
   - A painter paints many different paintings, but each one of them is painted by only one painter. Thus, the painter (the "one") is related to the paintings (the "many"). Therefore, database designers label the relationship "PAINTER paints PAINTING" as 1:M.
   - Note that entity names are often capitalized as a convention, so they are easily identified.
   - Similarly, a customer (the "one") may generate many invoices, but each invoice (the "many") is generated by only a single customer. The "CUSTOMER generates INVOICE" relationship would also be labeled 1:M.
2. **Many-to-many (M:N or *..*) relationship**
   - An employee may learn many job skills, and each job skill may be learned by many employees. Database designers label the relationship "EMPLOYEE learns SKILL" as M:N.
   - Similarly, a student can take many subjects and each subject can be taken by many students, thus yielding the M:N relationship label for the relationship expressed by "STUDENT takes SUBJECT."
3. **One-to-one (1:1 or 1..1) relationship**
   - A retail company's management structure may require that each of its stores be managed by a single employee. In turn, each store manager, who is an employee, manages only a single store. Therefore, the relationship "EMPLOYEE manages STORE" is labeled 1:1.

Note

The preceding discussion identified each relationship in both directions; that is, relationships are bidirectional:

- *One* CUSTOMER can generate *many* INVOICEs.
- Each of the *many* INVOICEs is generated by only *one* CUSTOMER.

Constraint

- A **constraint** is a restriction placed on the data.
- Constraints are important because they help to ensure data integrity.
- Constraints are normally expressed in the form of rules.

- For example:
  - An employee's salary must have values that are between 6,000 and 350,000.
  - A student's score must be between 0 and 100
  - Each class must have one and only one teacher.

Question
How do you properly identify entities, attributes, relationships, and constraints? The first step is to clearly identify business rules for the problem domain you are modeling

Business rule
- *It is a brief, precise, and unambiguous description of a policy, procedure, or principle within a specific organization*
- Properly written business rules are used to define entities, attributes, relationships, and constraints.
- Any time you see relationship statements such as "an agent can serve many customers, and each customer can be served by only one agent," you are seeing business rules at work.

TYPES OF DATA MODELS

- Data models have evolved over time with each newer model overcoming the shortfalls of the preceding model(s)

a) Hierarchical Data Model
- The hierarchical model was developed in the 1960s to manage large amounts of data for complex manufacturing project.
- Its basic logical structure is represented by an upside-down tree.
- The hierarchical structure contains levels, or segments.
- A **segment** is the equivalent of a file system's record type.
- Within the hierarchy, a higher layer is perceived as the parent of the segment directly beneath it, which is called the child.
- The hierarchical model depicts a set of one-to-many (1:M) relationships between a parent and its children segments. (Each parent can have many children, but each child has only one parent.)

b) Network Data Model
- The **network model** was created to represent complex data relationships more effectively than the hierarchical model, to improve database performance, and to impose a database standard.
- In the network model, the user perceives the network database as a collection of records in 1:M relationships.
- However, unlike the hierarchical model, the network model allows a record to have more than one parent.
- While the network database model is generally not used today, the definitions of standard database *concepts* that emerged with the network model are still used by modern data models.
- Some important concepts that were defined at this time are:
  - The **schema**, which is the conceptual organization of the entire database as viewed by the database administrator.
  - The **subschema**, which defines the portion of the database "seen" by the application programs that actually produce the desired information from the data contained within the database.

- A **data management language** (**DML**), which defines the environment in which data can be managed and to work with the data in the database.
- A schema **data definition language** (**DDL**), which enables the database administrator to define the schema components.

Note

- As information needs grew and as more sophisticated databases and applications were required, the network model became too cumbersome.
- The lack of ad hoc query capability put heavy pressure on programmers to generate the code required to produce even the simplest reports. And although the existing databases provided limited data independence, any structural change in the database could still produce havoc in all application programs that drew data from the database.
- Because of the disadvantages of the hierarchical and network models, they were largely replaced by the relational data model

c) Relational Data Model

- **A relation** (sometimes called a **table**) is a matrix composed of intersecting rows and columns.
- Each row in a relation is called a **tuple**.
- Each column represents an **attribute**.
- The relational data model is implemented through a very sophisticated **relational database management system** (**RDBMS**).
- A RDBMS performs the same basic functions provided by the hierarchical and network DBMS systems, in addition to a host of other functions that make the relational data model easier to understand and implement.
- The most important advantage of the RDBMS is its ability to hide the complexities of the relational model from the user.
- The RDBMS manages all of the physical details, while the user sees the relational database as a collection of tables in which data are stored
- Tables are related to each other through the sharing of a common attribute (value in a column). *For example, the CUSTOMER table might contain a sales agent's number that is also contained in the AGENT table.*

FIGURE 2.1 Linking relational tables

Table name: AGENT (first six attributes) Database name: Ch02_InsureCo

AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

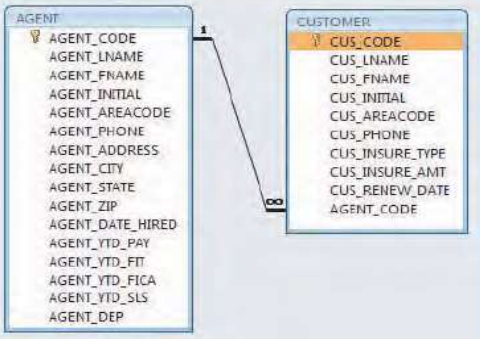
Link through AGENT_CODE

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_INSURE_TYPE	CUS_INSURE_AMT	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	T1	100.00	05-Apr-2010	502
10011	Dunne	Leona	K	713	894-1238	T1	250.00	16-Jun-2010	501
10012	Smith	Kathy	W	615	894-2285	S2	150.00	29-Jan-2011	502
10013	Olowski	Paul	F	615	894-2180	S1	300.00	14-Oct-2010	502
10014	Orlando	Myron		615	222-1672	T1	100.00	28-Dec-2010	501
10015	O'Brian	Amy	B	713	442-3381	T2	850.00	22-Sep-2010	503
10016	Brown	James	G	615	297-1228	S1	120.00	25-Mar-2011	502
10017	Williams	George		615	290-2556	S1	250.00	17-Jul-2010	503
10018	Farriss	Anne	G	713	382-7185	T2	100.00	03-Dec-2010	501
10019	Smith	Olette	K	615	297-3809	S2	500.00	14-Mar-2011	503

- The common link between the CUSTOMER and AGENT tables enables you to match the customer to his or her sales agent, even though the customer data are stored in one table and the sales representative data are stored in another table.
  For example, you can easily determine that customer Dunne's agent is Alex Alby because for customer Dunne, the CUSTOMER table's AGENT_CODE is 501, which matches the AGENT table's AGENT_CODE for Alex Alby.
- Although the tables are independent of one another, you can easily associate the data between tables.
- The relational model provides a minimum level of controlled redundancy to eliminate most of the redundancies commonly found in file systems.

FIGURE 2.2 A relational diagram



The relationship type (1:1, 1:M, or M:N) is often shown in a relational schema, an example of which is shown in Figure 2.2. A **relational diagram** is a representation of the relational database's entities, the attributes within those entities, and the relationships between those entities.

In Figure 2.2, the relational diagram shows the connecting fields (in this case, AGENT_CODE) and the relationship type, 1:M. Microsoft Access, the database software application used to generate Figure 2.2, employs the ∞ (infinity) symbol to indicate the "many" side. In this example, the CUSTOMER represents the "many" side because an AGENT can have many CUSTOMERs. The AGENT represents the "1" side because each CUSTOMER has only one AGENT.

Why popularity of the Relational Data Model grew

- A table yields complete data and structural independence because it is a purely logical structure. i.e. How the data are physically stored in the database is of no concern to the user or the designer; the perception is what counts.
- It supports powerful and flexible Structured Query Language (SQL) which allows the user to specify what must be done without specifying how it must be done. SQL makes it possible to retrieve data with far less effort than any other database or file environment.

Note

- From an end-user perspective, any SQL-based relational database application involves three parts: a user interface, a set of tables stored in the database, and the SQL "engine."
  i) The end-user interface.
    - This is the interface which allows end user to interact with the data (by auto-generating SQL code).
    - Each interface is a product of the software vendor's idea of meaningful interaction with the data.
  ii) A collection of tables stored in the database
    - In a relational database, all data are perceived to be stored in tables.
    - The tables simply "present" the data to the end user in a way that is easy to understand.
    - Each table is independent.
    - Rows in different tables are related by common values in common attributes.
  iii) SQL engine
    - Largely hidden from the end user, the SQL engine executes all queries, or data requests.

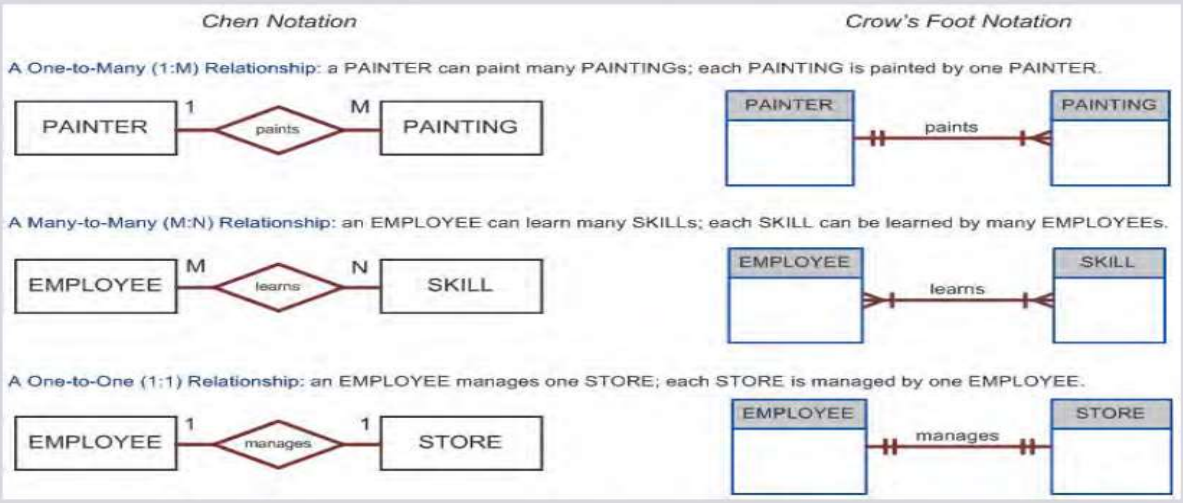
- SQL engine is part of the DBMS software.
- The end user uses SQL to create table structures and to perform data access and table maintenance. The SQL engine processes all user requests—largely behind the scenes and without the end user's knowledge. Hence, it's said that SQL is a declarative language that tells what must be done but not how it must be done.

d) The Entity Relationship Model

- Complex design activities require conceptual simplicity to yield successful results. Although the relational model was a vast improvement over the hierarchical and network models, it still lacked the features that would make it an effective database *design* tool.
- Because it is easier to examine structures graphically than to describe them in text, database designers prefer to use a graphical tool in which entities and their relationships are pictured. Thus, the **entity relationship** (**ER**) **model**, or **ERM**, has become a widely accepted standard for data modeling.
- ERM - is a graphical representation of entities and their relationships in a database structure that quickly became popular because it *complemented* the relational data model concepts.
- The relational data model and ERM combined to provide the foundation for tightly structured database design.
- ER models are normally represented in an **entity relationship diagram** (**ERD**), which uses graphical representations to model database components.

- The ER model is based on the following components:
  a) *Entit*y.
     - An entity is represented in the ERD by a rectangle, also known as an entity box.
     - The name of the entity, a noun, is written in the center of the rectangle. The entity name is generally written in capital letters and is written in the singular form: PAINTER rather than PAINTERS, and EMPLOYEE rather than EMPLOYEES.
     - Usually, when applying the ERD to the relational model, an entity is mapped to a relational table.
     - Each row in the relational table is known as an **entity instance** or **entity occurrence** in the ER model.
     - Each entity is described by a set of *attributes* that describes particular characteristics of the entity.
       For example, the entity EMPLOYEE will have attributes such as a Social Security number, a last name, and a first name.
  b) Relationships
     - Relationships describe associations among data.
     - Most relationships describe associations between two entities.
     - Three types of relationships among data exist: *one-to-many (1:M), many-to-many (M:N), and one-to-one (1:1).*
     - The ER model uses the term **connectivity** to label the relationship types.
     - The name of the relationship is usually an active or passive verb.
       For example, a PAINTER paints many PAINTINGs; an EMPLOYEE learns many SKILLs; an EMPLOYEE manages a STORE.

FIGURE 2.3 The Chen and Crow's Foot notations



e) The Object-Oriented (OO) Model

- Increasingly complex real-world problems demonstrated a need for a data model that more closely represented the real world.
- In the **object-oriented data model** (**OODM**), both data *and their relationships* are contained in a single structure known as an **object**. In turn, the OODM is the basis for the **object-oriented database management system** (**OODBMS**).
- Like the relational model's entity, an object is described by its factual content. But quite *unlike* an entity, an object includes information about relationships between the facts within the object, as well as information about its relationships with other objects. Therefore, the facts within the object are given greater *meaning*. The OODM is said to be a **semantic data model** because *semantic* indicates meaning.

- The OO data model is based on the following components:
  i) An object
     - This is an abstraction of a real-world entity.
     - In general terms, an object may be considered equivalent to an ER model's entity.
  ii) Attributes
     - These describe the properties of an object.
     - For example, a PERSON object includes the attributes Name, Social Security Number, and Date of Birth.
  iii) Objects that share similar characteristics are grouped in classes.
     - A **class** is a collection of similar objects with shared structure (attributes) and behavior (methods).
     - In a general sense, a class resembles the ER model's *entity set*. However, a class is different from an entity set in that it contains a set of procedures known as *methods*.
     - A class's **method** represents a real-world action such as *finding* a selected PERSON's name, *changing* a PERSON's name, or *printing* a PERSON's address. In other words, methods are the equivalent of *procedures*
     - In OO terms, methods define an object's *behavior*.
  iv) Classes are organized in a *class hierarchy*.

- The **class hierarchy** resembles an upside-down tree in which each class has only one parent.
  For example, the CUSTOMER class and the EMPLOYEE class share a parent PERSON class.

v) **Inheritance**

- This is the ability of an object within the class hierarchy to inherit the attributes and methods of the classes above it.
- For example, two classes, CUSTOMER and EMPLOYEE, can be created as subclasses from the class PERSON. In this case, CUSTOMER and EMPLOYEE will inherit all attributes and methods from PERSON.

REVIEW QUESTIONS

1. Discuss the importance of data modeling.
2. What is a business rule, and what is its purpose in data modeling?
3. How do you translate business rules into data model components?
4. What languages emerged to standardize the basic network data model, and why was such standardization important to users and designers?
5. Describe the basic features of the relational data model and discuss their importance to the end user and the designer.
6. Explain how the entity relationship (ER) model helped produce a more structured relational database design environment.
7. Use the scenario described by "A customer can make many payments, but each payment is made by only one customer" as the basis for an entity relationship diagram (ERD) representation.
8. Why is an object said to have greater semantic content than an entity?
9. What is the difference between an object and a class in the object-oriented data model (OODM)?
10. How would you model Question 7 with an OODM? (Use Figure 2.4 as your guide.)
11. What is an ERDM, and what role does it play in the modern (production) database environment?
12. In terms of data and structural independence, compare file system data management with the five data models discussed in this chapter.
13. What is a relationship, and what three types of relationships exist?
14. Give an example of each of the three types of relationships.
15. What is a table, and what role does it play in the relational model?
16. What is a relational diagram? Give an example.
17. What is logical independence?
18. What is physical independence?
19. What is connectivity? (Use a Crow's Foot ERD to illustrate connectivity.)

CHAPTER THREE
RELATIONAL DATABASE MODEL

Tables and Their Characteristics

- A table is perceived as a two-dimensional structure composed of rows and columns.
- A table is also called a *relation*
- You can think of a table as a *persistent* representation of a logical relation, that is, a relation whose contents can be permanently saved for future use.
- A *table contains a group of related entity occurrences*, that is, an entity set. *For example, a STUDENT table contains a collection of entity occurrences, each representing a student.* For that reason, the terms *entity set* and *table* are often used interchangeably.

TABLE 3.1 **Characteristics of a Relational Table**

1	A table is perceived as a two-dimensional structure composed of rows and columns.
2	Each table row (**tuple**) represents a single entity occurrence within the entity set.
3	Each table column represents an attribute, and each column has a distinct name.
4	Each row/column intersection represents a single data value.
5	All values in a column must conform to the same data format.
6	Each column has a specific range of values known as the **attribute domain**.
7	The order of the rows and columns is immaterial to the DBMS.
8	Each table must have an attribute or a combination of attributes that uniquely identifies each row.

CASE: Consider the Table below

FIGURE 3.1 **STUDENT table attribute values**

Table name: STUDENT — Database name: Ch03_TinyCollege

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT	STU_DOB	STU_HRS	STU_CLASS	STU_GPA	STU_TRANSFER	DEPT_CODE	STU_PHONE	PROF_NUM
321452	Bowser	William	C	12-Feb-1975	42	So	2.84	No	BIOL	2134	205
324257	Smithson	Anne	K	15-Nov-1981	81	Jr	3.27	Yes	CIS	2256	222
324258	Brewer	Juliette		23-Aug-1969	36	So	2.26	Yes	ACCT	2256	228
324269	Oblonski	Walter	H	16-Sep-1976	66	Jr	3.09	No	CIS	2114	222
324273	Smith	John	D	30-Dec-1958	102	Sr	2.11	Yes	ENGL	2231	199
324274	Katinga	Raphael	P	21-Oct-1979	114	Sr	3.15	No	ACCT	2267	228
324291	Robertson	Gerald	T	08-Apr-1973	120	Sr	3.87	No	EDU	2267	311
324299	Smith	John	B	30-Nov-1986	15	Fr	2.92	No	ACCT	2315	230

STU_NUM = Student number
STU_LNAME = Student last name
STU_FNAME = Student first name
STU_INIT = Student middle initial
STU_DOB = Student date of birth
STU_HRS = Credit hours earned
STU_CLASS = Student classification
STU_GPA = Grade point average
STU_TRANSFER = Student transferred from another institution
DEPT_CODE = Department code
STU_PHONE = 4-digit campus phone extension
PROF_NUM = Number of the professor who is the student's advisor

Using the STUDENT table shown in Figure 3.1, you can draw the following conclusions corresponding to the points in Table 3.1:

1. The STUDENT table is perceived to be a two-dimensional structure composed of eight rows (tuples) and twelve columns (attributes).
2. Each row in the STUDENT table describes a single entity occurrence within the entity set. (The entity set is represented by the STUDENT table.) For example, row 4 in Figure 3.1 describes a student named Walter H. Oblonski. Given the table contents, the STUDENT entity set includes eight distinct entities (rows), or students.
3. Each column represents an attribute, and each column has a distinct name.
4. All of the values in a column match the attribute's characteristics. For example, the grade point average (STU_GPA) column contains only STU_GPA entries for each of the table rows. Data must be classified according to their format and function. Although various DBMSs can support different data types, most support at least the following:
   a) *Numeric*. Numeric data are data on which you can perform meaningful arithmetic procedures. For example, in Figure 3.1, STU_HRS and STU_GPA are numeric attributes.
   b) *Character*. Character data, also known as text data or string data, can contain any character or symbol not intended for mathematical manipulation. In Figure 3.1, STU_CLASS and STU_PHONE are examples of character attributes.
   c) *Date.* Date attributes contain calendar dates stored in a special format known as the Julian date format. For example, STU_DOB in Figure 3.1 is a date attribute.
   d) *Logical.* Logical data can only have true or false (yes or no) values. In Figure 3.1, the STU_TRANSFER attribute uses a logical data format.
5. The column's range of permissible values is known as its **domain**. Because the STU_GPA values are limited to the range 0–4, inclusive, the domain is [0,4].
6. The order of rows and columns is immaterial to the user.
7. Each table must have a primary key. In general terms, the **primary key** (**PK**) is an attribute (or a combination of attributes) that uniquely identifies any given row. In this case, STU_NUM (the student number) is the primary key.

Using the data presented in Figure 3.1, observe that a student's last name (STU_LNAME) would not be a good primary key because it is possible to find several students whose last name is Smith. Even the combination of the last name and first name (STU_FNAME) would not be an appropriate primary key because, as Figure 3.1 shows, it is quite possible to find more than one student named John Smith.

KEYS

- Keys are used to ensure that each row in a table is uniquely identifiable.
- They are also used to establish relationships among tables and to ensure the integrity of the data.

- A **key** consists of one or more attributes that determine other attributes. For example, a student's admission number identifies all of the student attributes, such as the birth date, course, student name, fee status and gender.
- Some of relational table keys are:-
- Primary key, super key, candidate keys, and secondary keys.
- The key's role is based on a concept known as **determination**. In the context of a database table, the statement "A determines B" indicates that if you know the value of attribute A, you can look up (determine) the value of attribute B.

- For example, knowing the STU_NUM in the STUDENT table (see Figure 3.1) means that you are able to look up (determine) that student's last name, grade point average, phone number, and so on.
- The shorthand notation for "A determines B" is A → B. If A determines B, C, and D, you write A → B, C, D.
- Therefore, using the attributes of the STUDENT table in Figure 3.1, you can represent the statement *"STU_NUM determines STU_LNAME" by writing:*
  STU_NUM → STU_LNAME
  In fact, the STU_NUM value in the STUDENT table determines all of the student's attribute values. For example, you can write:
  STU_NUM → STU_LNAME, STU_FNAME, STU_INIT
- It is possible that more than a single attribute can be used to define functional dependence; that is, a key may be composed of more than one attribute. Such a multiattribute key is known as a **composite key**.
- Any attribute that is part of a key is known as a **key attribute**. For instance, in the STUDENT table, the student's last name would not be sufficient to serve as a key. On the other hand, the combination of last name, first name, initial, and phone is very likely to produce unique matches for the remaining attributes.
- A **superkey** is any key that uniquely identifies each row i.e. the superkey functionally determines all of a row's attributes.
  For example, in the STUDENT table, the superkey could be any of the following:
  STU_NUM
  STU_NUM, STU_LNAME
  STU_NUM, STU_LNAME, STU_INIT
  In fact, STU_NUM, with or without additional attributes, can be a superkey even when the additional attributes are redundant.
- A **candidate key** can be described as a superkey without unnecessary attributes, i.e. a minimal superkey. Using this distinction, note that the composite key STU_NUM, STU_LNAME is a superkey, but it is not a candidate key because STU_NUM by itself is a candidate key!
- Within a table, each primary key value must be unique to ensure that each row is uniquely identified by the primary key. In that case, the table is said to exhibit **entity integrity**. To maintain entity integrity, a **null** (that is, no data entry at all) is not permitted in the primary key.
- A **foreign key** (**FK**) is an attribute whose values match the primary key values in the related table.
  For example in the MARKS table, Student's Admission No is a Foreign key, since it is a Primary Key (PK) in the STUDENTS table
- A **secondary key** is a key that is used strictly for data retrieval purposes.
- **Referential integrity** means that if the foreign key contains a value, that value refers to an existing valid tuple (row) in another relation.
  For example, referential integrity is maintained between STUDENTS and MARKS table

TABLE 3.3 **Relational Database Keys**

KEY TYPE	DEFINITION
Superkey	An attribute (or combination of attributes) that uniquely identifies each row in a table.
Candidate key	A minimal (irreducible) superkey. A superkey that does not contain a subset of attributes that is itself a superkey.
Primary key	A candidate key selected to uniquely identify all other attribute values in any given row. Cannot contain null entries.
Secondary key	An attribute (or combination of attributes) used strictly for data retrieval purposes.
Foreign key	An attribute (or combination of attributes) in one table whose values must either match the primary key in another table or be null.

RELATIONAL SET OPERATORS

- The data in relational tables are of limited value unless the data can be manipulated to generate useful information.
- **Relational algebra** defines the theoretical way of manipulating table contents using the eight relational operators:
    - ***SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT,*** *and* ***DIVIDE***

a) SELECT, also known as RESTRICT

- It yields values for all rows found in a table that satisfy a given condition.
- SELECT can be used to list all of the row values, or it can yield only those row values that match a specified criterion.
- In other words, SELECT yields a horizontal subset of a table.
- The effect of a SELECT is shown in Figure 3.5 below

FIGURE 3.5 SELECT

Original table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

SELECT ALL yields

New table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

SELECT only PRICE less than $2.00 yields

P_CODE	P_DESCRIPT	PRICE
213345	9v battery	1.92
254467	100W bulb	1.47

SELECT only P_CODE = 311452 yields

P_CODE	P_DESCRIPT	PRICE
311452	Powerdrill	34.99

b) PROJECT

- This operator yields all values for selected attributes.
- In other words, PROJECT yields a vertical subset of a table.
- The effect of a PROJECT is shown in Figure 3.6 below

FIGURE 3.6 PROJECT

Original table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

PROJECT PRICE yields

New table

PRICE
5.26
25.15
10.99
1.92
1.47
34.99

PROJECT P_DESCRIPT and PRICE yields

P_DESCRIPT	PRICE
Flashlight	5.26
Lamp	25.15
Box Fan	10.99
9v battery	1.92
100W bulb	1.47
Powerdrill	34.99

PROJECT P_CODE and PRICE yields

P_CODE	PRICE
123456	5.26
123457	25.15
123458	10.99
213345	1.92
254467	1.47
311452	34.99

c) UNION

- It combines all rows from two tables, excluding duplicate rows, but the tables must have the same attribute characteristics (the columns and domains must be compatible) to be used in the UNION.
- When two or more tables share the same number of columns, and when their corresponding columns share the same (or compatible) domains, they are said to be **union-compatible**.
- The effect of a UNION is shown in Figure 3.7.

FIGURE 3.7 UNION

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

UNION

P_CODE	P_DESCRIPT	PRICE
345678	Microwave	160.00
345679	Dishwasher	500.00

yields

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99
345678	Microwave	160
345679	Dishwasher	500

d) INTERSECT

- It yields only the rows that appear in both tables. As was true in the case of UNION, the tables must be union-compatible to yield valid results.
- For example, you cannot use INTERSECT if one of the attributes is numeric and one is character-based.
- The effect of an INTERSECT is shown in Figure 3.8.

FIGURE 3.8 INTERSECT

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

INTERSECT

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

yields

STU_FNAME	STU_LNAME
Franklin	Johnson

e) DIFFERENCE

- It yields all rows in one table that are not found in the other table; that is, it subtracts one table from the other. As was true in the case of UNION, the tables must be union-compatible to yield valid results.
- The effect of a DIFFERENCE is shown in Figure 3.9. However, note that subtracting the first table from the second table is not the same as subtracting the second table from the first table.

FIGURE 3.9 DIFFERENCE

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

DIFFERENCE

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

yields

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Martin	Lopez

f) PRODUCT

- It yields all possible pairs of rows from two tables—also known as the Cartesian product. Therefore, if one table has six rows and the other table has three rows, the PRODUCT yields a list composed of 6 × 3 = 18 rows.
- The effect of a PRODUCT is shown in Figure 3.10.

FIGURE 3.10 PRODUCT

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

PRODUCT

STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

yields

P_CODE	P_DESCRIPT	PRICE	STORE	AISLE	SHELF
123456	Flashlight	5.26	23	W	5
123456	Flashlight	5.26	24	K	9
123456	Flashlight	5.26	25	Z	6
123457	Lamp	25.15	23	W	5
123457	Lamp	25.15	24	K	9
123457	Lamp	25.15	25	Z	6
123458	Box Fan	10.99	23	W	5
123458	Box Fan	10.99	24	K	9
123458	Box Fan	10.99	25	Z	6
213345	9v battery	1.92	23	W	5
213345	9v battery	1.92	24	K	9
213345	9v battery	1.92	25	Z	6
311452	Powerdrill	34.99	23	W	5
311452	Powerdrill	34.99	24	K	9
311452	Powerdrill	34.99	25	Z	6
254467	100W bulb	1.47	23	W	5
254467	100W bulb	1.47	24	K	9
254467	100W bulb	1.47	25	Z	6

g) JOIN

- It allows information to be combined from two or more tables.
- JOIN is the real power behind the relational database, allowing the use of independent tables linked by common attributes.
- The CUSTOMER and AGENT tables shown in Figure 3.11 will be used to illustrate several types of joins.

FIGURE 3.11 Two tables that will be used in join illustrations

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Table name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

1) NATURAL JOIN

- A **natural join** links tables by selecting only the rows with common values in their common attribute(s).
- A natural join is the result of a three-stage process:

  Step 1
  - A PRODUCT of the tables is created, yielding the results shown below

FIGURE 3.12 Natural join, Step 1: PRODUCT

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1132445	Walker	32145	231	125	6152439887
1132445	Walker	32145	231	167	6153426778
1132445	Walker	32145	231	231	6152431124
1132445	Walker	32145	231	333	9041234445
1217782	Adares	32145	125	125	6152439887
1217782	Adares	32145	125	167	6153426778
1217782	Adares	32145	125	231	6152431124
1217782	Adares	32145	125	333	9041234445
1312243	Rakowski	34129	167	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1312243	Rakowski	34129	167	231	6152431124
1312243	Rakowski	34129	167	333	9041234445
1321242	Rodriguez	37134	125	125	6152439887
1321242	Rodriguez	37134	125	167	6153426778
1321242	Rodriguez	37134	125	231	6152431124
1321242	Rodriguez	37134	125	333	9041234445
1542311	Smithson	37134	421	125	6152439887
1542311	Smithson	37134	421	167	6153426778
1542311	Smithson	37134	421	231	6152431124
1542311	Smithson	37134	421	333	9041234445
1657399	Vanloo	32145	231	125	6152439887
1657399	Vanloo	32145	231	167	6153426778
1657399	Vanloo	32145	231	231	6152431124
1657399	Vanloo	32145	231	333	9041234445

Step 2

Second, a SELECT is performed on the output of Step a to yield only the rows for which the AGENT_CODE values are equal. The common columns are referred to as the **join columns**.

Step 2 yields the results shown below

FIGURE 3.13 Natural join, Step 2: SELECT

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	125	6152439887
1321242	Rodriguez	37134	125	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1132445	Walker	32145	231	231	6152431124
1657399	Vanloo	32145	231	231	6152431124

Step 3

A PROJECT is performed on the results of Step b to yield a single copy of each attribute, thereby eliminating duplicate columns.

Step 3 yields the output shown below

FIGURE 3.14 Natural join, Step 3: PROJECT

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124

2) EQUIJOIN

- It links tables on the basis of an equality condition that compares specified columns of each table.
- The outcome of the equijoin does not eliminate duplicate columns, and the condition or criterion used to join the tables must be explicitly defined.
- The equijoin takes its name from the equality comparison operator (=) used in the condition. If any other comparison operator is used, the join is called a **theta join**.
- Each of the preceding joins is often classified as an inner join.
- An **inner join** is a join that only returns matched records from the tables that are being joined.
- In an **outer join**, the matched pairs would be retained, and any unmatched values in the other table would be left null.

  Note
- It is an easy mistake to think that an outer join is the opposite of an inner join, however the *outer join still returns all of the matched records that the inner join returns, plus it returns the unmatched records from one of the tables.*

Consider the relations below

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Table name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

Left outer join

- It yields all of the rows in the CUSTOMER table, including those that do not have a matching value in the AGENT table.
- Results are shown below

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124
1542311	Smithson	37134	421	

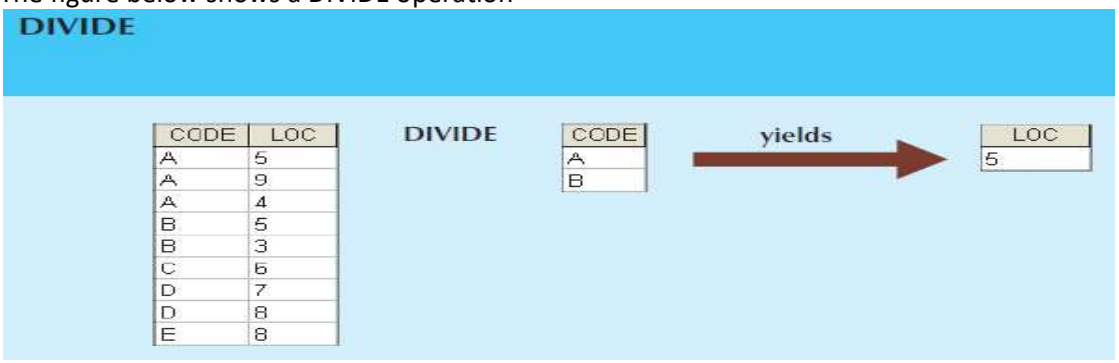
Right outer join

- It yields all of the rows in the AGENT table, including those that do not have matching values in the CUSTOMER table.
- Results are shown below

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124
			333	9041234445

h) DIVIDE

- The DIVIDE operation uses one single-column table (e.g., column "a") as the divisor and one 2-column table (i.e., columns "a" and "b") as the dividend.
- The tables must have a common column (e.g., column "a").
- The output of the DIVIDE operation is a single column with the values of column "a" from the dividend table rows where the value of the common column (i.e., column "a") in both tables matches.
- The figure below shows a DIVIDE operation



Using the example above, note that:

a) Table 1 is "divided" by Table 2 to produce Table 3. Tables 1 and 2 both contain the column CODE but do not share LOC.
b) To be included in the resulting Table 3, a value in the unshared column (LOC) must be associated (in the dividing Table 2) with every value in Table 1.
c) The only value associated with both A and B is 5.

THE DATA DICTIONARY AND THE SYSTEM CATALOG

Data Dictionary

- The **data dictionary** provides a detailed description of all tables found within the user/designer-created database.
- Thus, the data dictionary contains at least all of the attribute names and characteristics for each table in the system.
- In short, the data dictionary contains **metadata**—data about data.
- The data dictionary is sometimes described as "the database designer's database" because it records the design decisions about tables and their structures.

System catalog

- A **system catalog** contains metadata.
- It is a detailed system data dictionary that describes all objects within the database, including data about table names, the table's creator and creation date, the number of columns in each table, the data type corresponding to each column, authorized users, and access privileges.
- Because the system catalog contains all required data dictionary information, the terms *system catalog* and *data dictionary* are often used interchangeably.

SUMMARY

How is it different from a normal DBMS?

- DBMS stores data as files whereas RDBMS stores data in a tabular arrangement.
- RDBMS allows for normalization of data.
- RDBMS maintains a relation between the data stored in its tables. A normal DBMS does not provide any such link. It blankly stores data is its files.
- Structured approach of RDBMS supports a distributed database unlike a normal database management system.

Features of RDBMS

- Store data in tables. The fact that the very storage of data is in a structured form can significantly reduce iteration time.
- Data persists in the form of rows and columns and allows for a facility primary key to define unique identification of rows.
- It creates indexes for quicker data retrieval.
- Allows for various types of data integrity like
  (i) Entity Integrity; wherein no duplicate rows in a table exist,
  (ii) (ii)Domain Integrity; that enforces valid entries for a given column by filtering the type, the format, or the wide use of values,
  (iii)Referential Integrity; which disables the deletion of rows that are in use by other records and
  (iv)User Defined Integrity; providing some specific business rules that do not fall in the above three.
- Allows for the virtual table creation which provides a safe means to store and secure sensitive content.
- Common column implementation and also multi user accessibility is included in the RDBMS features.

Advantages of RDBMS

- Data is stored only once and hence multiple record changes are not required. Also deletion and modification of data becomes simpler and storage efficiency is very high.
- Complex queries can be carried out using the Structure Query Language. Terms like ‘Insert’, ‘Update’, ‘Delete’, ‘Create’ and ‘Drop’ are keywords in SQL that help in accessing a particular data of choice.
- Better security is offered by the creation of tables. Certain tables can be protected by this system. Users can set access barriers to limit access to the available content.
- Provision for future requirements as new data can easily be added and appended to the existing tables and can be made consistent with the previously available content.

Advantages of RDBMS

- The prime disadvantage of this effective system is its cost of execution. To set up a relational database management system, special software needs to be purchased. Once it is bought, setting up of data is a tedious task.
- Simple text data can be easily added and appended. However, newer forms of data can be confusing. Complex images, numbers, designs are not easy to be categorized into tables and presents a problem.
- Structure limits are another drawback. Certain fields in tables have a character limit.

- Isolated databases can be created if large chunks of information are separated from each other. Connecting such large volumes of data is not easy.

Uses of RDBMS

- They find application in disciplines like: Banking, airlines, universities, manufacturing and HR.
- Using RDBMS can bring a systematic view to raw data. It is easy to understand and execute and hence enables better decision making as well.
- It ensures effective running of an accounting system. Moreover, ticket service and passenger information documentation in airlines, student databases in universities and product details along with consumer demand

CHAPTER FOUR
ENTITY RELATIONSHIP (ER) IN DMS

(Components, Symbols, and Notations)

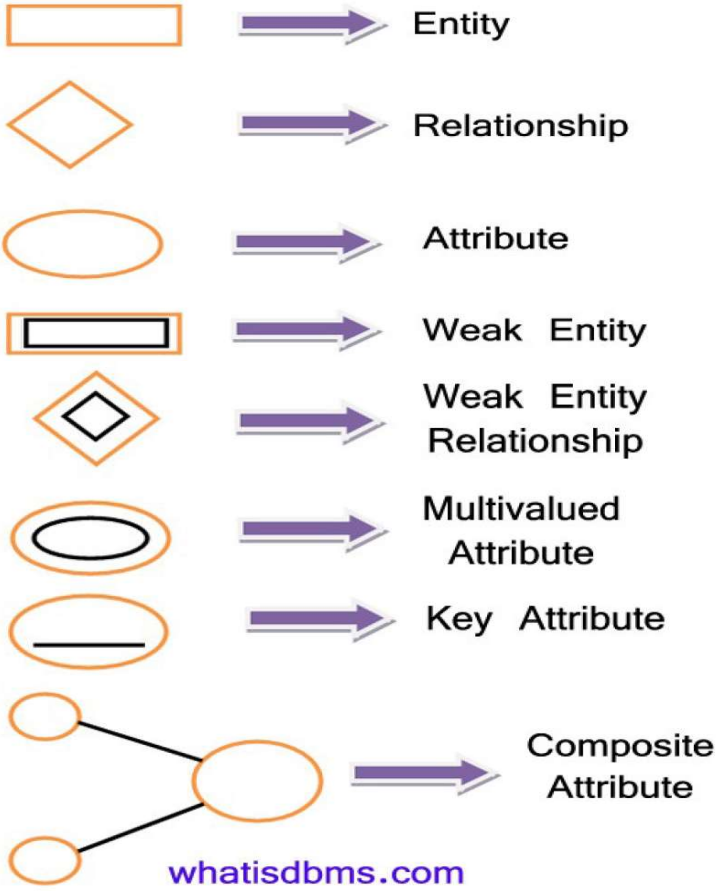
Introduction

- E-R diagram is the short form of "Entity-Relationship" diagram.
- An e-r diagram efficiently shows the relationships between various entities stored in a database
- E-R diagrams are used to model real-world objects like a person, a car, a company etc. and the relation between these real-world objects.

ER diagram features

- E-R diagrams are used to represent E-R model in a database, which makes them easy to be converted into relations (tables).
- E-R diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
- E-R diagrams require no technical knowledge & no hardware support.
- These diagrams are very easy to understand and easy to create even by a naïve user.
- It gives a standard solution of visualizing the data logically.

ER Diagram Symbols and Notations



COMPONENTS OF E-R DIAGRAM

- An E-R diagram constitutes of following Components

a) ENTITY

- Any real-world object can be represented as an entity about which data can be stored in a database.
- All the real world objects like a book, an organization, a product, a car, a person are the examples of an entity.
- Any living or non-living objects can be represented by an entity.
- An entity is symbolically represented by a rectangle enclosing its name.

Examples of Entity

STUDENT	**COURSE**
DEPARTMENT	**EMPLOYEE**

- Entities can be characterized into two types:

i. Strong entity

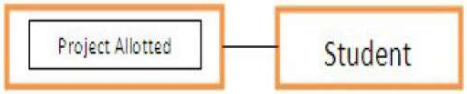
- A strong entity has a primary key attribute which uniquely identifies each entity.
- The symbol of strong entity is same as an entity.

Example of Strong Entity

ii. Weak entity

- A weak entity does not have a primary key attribute and depends on other entity via a foreign key attribute.

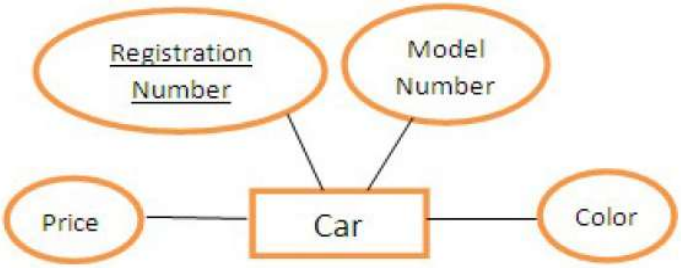
Example of Weak Entity



b) ATTRIBUTE

- Each entity has a set of properties or characteristics that describe it
- These properties of each entity are termed as attributes.
  For example, a car entity would be described by attributes such as price, registration number, model number, color etc.
- Attributes are indicated by ovals in an E-R diagram.

Example of Attribute



- A primary key attribute is depicted by an underline in the e-r diagram.
- An attribute can be characterized into following types:

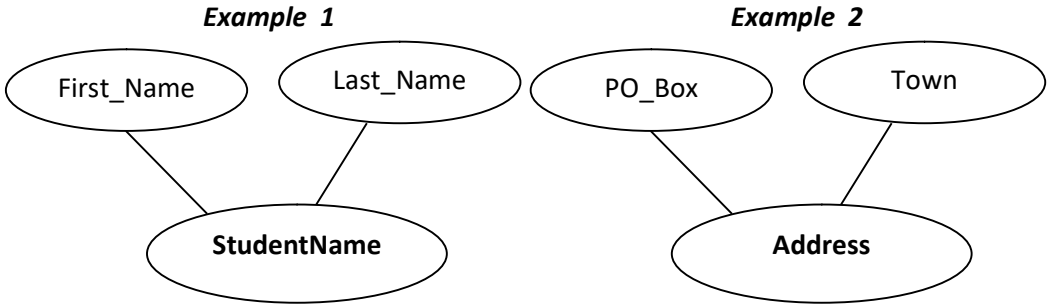
i. Simple attribute:

- An attribute is classified as a simple attribute if it cannot be partitioned into smaller components.
- For example, age and sex of a person.
- A simple attribute is represented by an oval.

Example

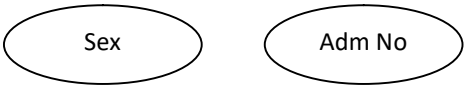
ii. Composite attribute

- A composite attribute can be subdivided into smaller components which further form attributes.
- For example, 'name' attribute of an entity "person" can be broken down into first name and last name which further form attributes.
- Grouping of these related attributes forms a composite attribute, and '*name'* is the composite attribute in this example.



iii. Single valued attribute

- If an attribute of a particular entity represents single value for each instance, then it is called a single-valued attribute.
- For example, Ramesh, Kamal and Suraj are the instances of entity 'student' and each of them is issued a separate roll number.
- A single oval is used to represent this attribute.



iv. Multi valued attribute

- An attribute which can hold more than one value, it is then termed as multi-valued attribute.
- For example, phone number of a person.
- The symbol of a multi-valued attribute is shown below

v. **Derived attribute**

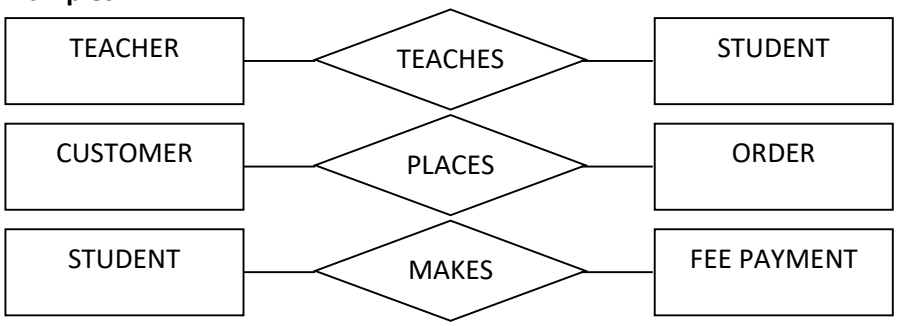
- A derived attribute calculate its value from another attribute.
- For example, 'age' is a derived attribute if it calculates its value from 'current date' & 'birth date' attributes.
- Also, 'Fee_Balance' could be a derived attribute if it calculates its value from 'Fee_due' and 'Fee_Paid' attribues
- A derived attribute is represented by a dashed oval.

Example

c) RELATIONSHIPS

- This is defined as bond or attachment between 2 or more entities.
- Normally, a verb in a sentence signifies a relationship.
- For example,
  - An employee assigned a project.
  - Teacher teaches a student.
  - Author writes a book.
- A diamond is used to symbolically represent a relationship in the ER diagram.

Examples



Terms related to relationships

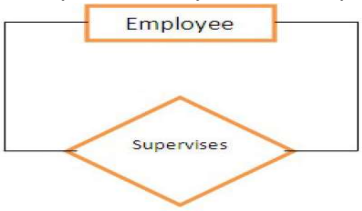
i. Degree of relationship

- It signifies the number of entities involved in a relationship.
- It indicates the number of entities or participants associated with a relationship
- Degree of a relationship can be classified into following types:

1. Unary relationship

- This is where only a single entity is involved in a relationship
- For example, An employee (manager) supervises another employee.

Example of Unary relationship



2. **Binary relationships**
   This is when two entities are associated to form a relation
   For example, A person works in a company.
   Most of the times we use only binary relationship in an ER diagram.
   The teacher-student example shown above signifies a binary relationship.

Note

- Other types of relationships are ternary and quaternary.
- As the name signifies, a ***ternary*** relationship is associated with three entities and a ***quaternary*** relationship is associated with four entities.

ii. Connectivity of a relationship

- Connectivity of a relationship describes how many instances of one entity type are linked to how many instances of another entity type.

- Various categories of connectivity of a relationship are;

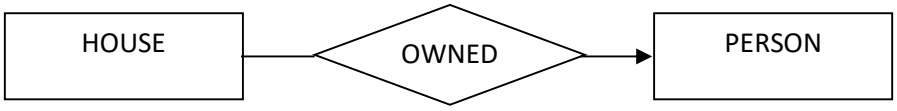
1. **One to One (1:1)**
   - "Student takes a course" signifies a one-to-one relationship because only one instance of an entity is related with exactly one instance of another entity type.

   Example

2. **One to Many (1:M)**
   - "A Customer Places Orders" is a one-to-many relationship because a customer can place more than one order, but an order is related to only one customer.

   Example

3. **Many to One (M:1)**
   - "Many houses are owned by a person" is a many-to one relationship because a person can own many houses but a particular house is owned only a person.
   Example



4. **Many to Many (M:N)**
   - "Student Sits Exams" is a many-to-many relationship because a student can take many exams, and an exam can be taken by many students.
   Example