

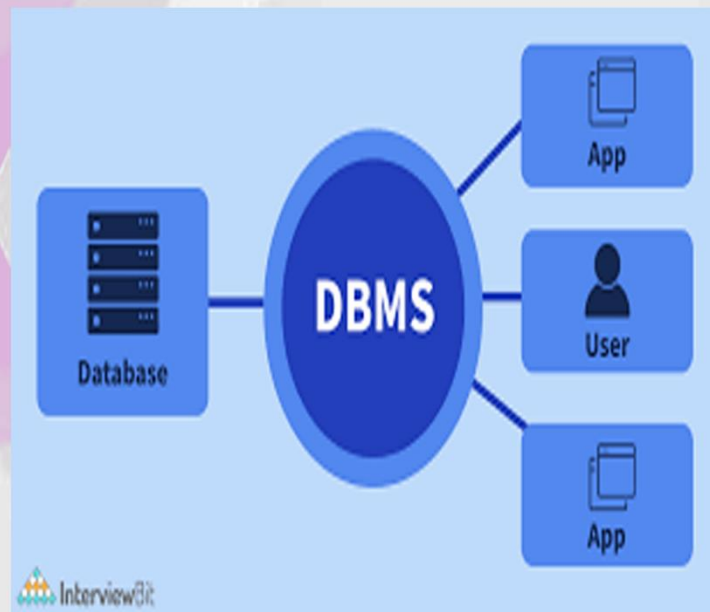
KARNATAKA STATE OPEN UNIVERSITY

Mukthagangothri, Mysuru



**Department of Studies and Research in
Information Technology**

**M.Sc. IN DATA SCIENCE
I SEMESTER**



**DATABASE MANAGEMENT SYSTEM
MDDSE-1.5**

BLOCK-1

CREDIT PAGE**Programme Name:** M.Sc. Data Science **Year/Semester:** I Semester **Block No :** 1**Course Name:** Database Management System **Credit:** 4 **Unit No :** 1-4**Course Design Expert Committee**

Dr. Sharanappa V. Halse. Chairman
Vice Chancellor,
Karnataka State Open University,
Mukthagangotri, Mysuru-570006.

Dr. Ashok Kamble Member
Dean (Academic),
Karnataka State Open University,
Mukthagangotri, Mysuru-570 006.

Dr. Rashmi B.S Member
Programme Co-ordinator
Assistant Professor & Chairperson,
DoS&R in Information Technology,
Karnataka State Open University, Mysuru-570 006.

Dr. Sanjay H A Member
BOS Chairman,
Professor,
Department of Information Science and Engineering,
MS Ramaiah Institute of Technology, Bengaluru-54.

Dr. Rashmi B.S Member Convener
Chairperson,
DoS&R in Information Technology,
Karnataka State Open University, Mysuru-570 006.

Editorial Committee

Dr. Sanjay H A Chairman
BOS Chairman
Professor,
Department of Information Science and Engineering,
MS Ramaiah Institute of Technology, Bengaluru-54.

Dr. Ashok Rao, Member
External Subject Expert,
Former Head, Network Project,
DESE, IISC Bangalore.

Dr. Nandini H M Internal Member
Assistant Professor,
DoS&R in Information Technology,
Karnataka State Open University, Mysuru-570 006.

Dr. Rashmi B S Member Convener Internal Subject Expert & Chairperson, DoS&R in Information Technology, Karnataka State Open University, Mysuru-570 006.	
Course Writer	Course Editor
Dr. Nandini H M Assistant Professor, DOS&R in Information Technology, Karnataka State Open University, Mysore.	Dr. Sudhamani M Assistant Professor Department of Computer Science, Amrita Vishwa Vidyapeetham, Mysore Campus.
Copy Right	
Registrar, Karnataka State Open University, Mukthagantoghri, Mysore 570 006.	
<p>Developed by the Department of Studies and Research in Information Technology, under the guidance of Dean (Academic), KSOU, Mysuru. Karnataka State Open University, February-2023. All rights reserved. No part of this work may be reproduced in any form or any other means, without permission in writing from the Karnataka State Open University. Further information on the Karnataka State Open University Programmes may be obtained from the University's Office at Mukthagangotri, Mysore – 570 006.</p>	
Printed and Published on behalf of Karnataka State Open University, Mysore-570 006 by the Registrar (Administration)-2023	

PREFACE

Any information system, corporate resource planning, research projects, and other operations requiring long-term data storage must have them as their foundation. Database management systems are more effective and efficient at managing data. The most typical DBMS is a relational database management system (RDBMS). A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases. This study material provides a new and comprehensive treatment of databases, dealing with the complete syllabuses for both an introductory course and an advanced course on databases. It offers a balanced view of concepts, languages and architectures, with reference to current technology on database management systems (DBMSs).

The material consists of total four modules. Each module is divided into 4 units. Therefore there will be total of sixteen units. The very first module covers the introduction to database systems, entity relationship, and data models. In second module database design and commercial query languages have been introduced. Module 3 covers File Organization, Transaction management, concurrency control and database backup in DBMS. In last module, i.e., module 4 introduces advanced database applications, distributed databases implementing security in databases and emerging database applications and case study.

In this course, every unit covers distinct objectives and includes a quick summary at the end. Every unit has an inducement such as keywords, 'check your progress' with answers which helps to test your knowledge and questions for self-study. A rich set of examples have been given for the better understanding. The learners are instructed to execute the queries and concepts given in the material for hands on experience.

Wish you happy reading!!!

KARNATAKA STATE



OPEN UNIVERSITY

MUKTHAGANGOTRI, MYSURU-06

Dept. of Studies and Research in Information Technology

M.Sc. in Data Science

I SEMESTER

MDDSE-1.5: DATABASE MANAGEMENT SYSTEM

**BLOCK 1: INTRODUCTION TO DATABASE SYSTEMS, ENTITY
RELATIONSHIP, AND DATA MODELS**

UNIT-1:	INTRODUCTION TO DBMS	1-17
UNIT-2:	DATABASE CORE CONCEPTS AND ARCHITECTURE	18-34
UNIT-3:	DATA MODELLING USING ER MODEL	35-56
UNIT-4:	EXTENDED ER MODEL	57-71

BLOCK 1 INTRODUCTION

This block is an introduction to database systems, entity relationship, and data models. The primary focus of this block is on the basic concepts of Database Management System, including characteristics, features, advantages, data models and architecture of database systems. Additionally, to demonstrate conceptual database design, the Entity-Relationship (ER) model and ER diagrams are shown. This block also demonstrates how to enhance the basic ER model to include new modelling ideas such as subclasses, specialization, generalization, union types (categories), and inheritance, which results in the enhanced-ER (EER) data model and EER diagrams.

This block is organized into four units and contents of the units are given in brief as follows:

Unit 1: Introduction to DBMS: An overview of Database Management System, Need for a DBMS, Traditional File Based System, Limitations of Traditional File Based System, Characteristics of database approach, Advantages of DBMS, Applications of DBMS Disadvantages of DBMS, Database Languages, The purpose of database applications and Database Users and Administrators.

Unit 2: Database core concepts and architecture: Data Model, Basic building blocks of data model, Types of Data Model, Hierarchical Model, Network Model, Relational Model, Object oriented Model, Schema and Instances, The logical DBMS architecture, Data Independence, DBMS Interfaces and Structure of DBMS.

Unit 3: Data modelling using ER model: Entity relationship model, ER diagram, Naming conventions of schema constructs, ER diagram notations, Entities, Strong entity, Weak entity, Attributes, Relationships, Roles in E-R diagrams, Relationship cardinality, Participation constraints and Constructing an ER model.

Unit 4: Extended ER model: The Enhanced Entity Relationship model, Features of EER Model, Subclasses and super classes, Specialization and generalization, Category or union, Aggregation, Use of extended ER features, Translation of ER schema into tables and Rules to transform ER diagram into tables.

UNIT-1: INTRODUCTION TO DBMS

STRUCTURE

- 1.0 Objectives
- 1.1 Introduction
- 1.2 An overview of Database Management System
- 1.3 Need for a DBMS
 - 1.3.1 Traditional File Based System
 - 1.3.2 Limitations of Traditional File Based System
- 1.4 Characteristics of database approach
- 1.5 Advantages of Database Management System
- 1.6 Applications of Database Management System
- 1.7 Disadvantages of Database Management System
- 1.8 Database Languages
- 1.9 The purpose of database applications
- 1.10 Database Users and Administrators
- 1.11 Check Your Progress
- 1.12 Summary
- 1.13 Key words
- 1.14 Questions for self-study
- 1.15 References

1.0 OBJECTIVES

After studying this unit, you will be able to:

- Define DBMS.
- Describe traditional file based system and its limitations.
- Explain how DBMS solves the problem of traditional file system
- Distinguish the database approach from the file-based system
- Identify database languages and its uses
- Define the need of application programs for database access.
- Explain the responsibilities of Database Administrator
- Differentiate between different types of database users.

1.1 INTRODUCTION

In this unit, we are going to discuss about the general overview and purpose of Database Management System. We discuss about the need of database management system and how it overcomes the limitations of a traditional file system. Databases and database systems have become an essential part of our everyday life. We encounter several activities that involve some interaction with a database almost daily. To familiarise with these activities some of the significant applications of DBMS in real life based on various fields and its uses are discussed. This unit also deals with different database languages available that are used to define and manipulate a database. At the end of this unit, different types of users based on their requirements and interaction with the database are discussed.

1.2 AN OVERVIEW OF DATABASE MANAGEMENT SYSTEM

The term Database Management System (DBMS) is a composite of two parts one is database and the other is management system.

“A database is a collection of related information stored so that it is available to many users for different purposes”

So, why do we need databases? In today’s world, data is all over the place and consistent. From birth to death, we generate and use data. The exploratory of data starts with the birth certificate and continues all the way to a death certificate (and beyond!). In between, every single person produces and consumes huge amounts of data. For that reason, databases are the best way to store and manage data.

To find out what motivates database design, we must understand the difference between data and information.

Data is nothing but raw facts. The word ‘raw’ implies that the facts have not yet been processed to reveal their meaning. Data on its own doesn’t carry any importance or purpose. Specifically, we have to interpret data for it to have meaning.

Information is the outcome of processing raw data to disclose its meaning. To disclose meaning, information needs context. For example, an average temperature reading of 105 degrees does not mean much unless we also know its context: Is this reading in degrees Fahrenheit or Celsius? Is this a machine temperature, a body temperature, or an outside air

temperature? So, data is the foundation or basic building block of information. Information can be used as the foundation for decision making.

“Management system is a collection of programs that enables users to create and maintain the database.” Or we can say management system is a set of programs that facilitates users to execute different kinds of operations on such a system for either to create database or maintain the database.

So “A DBMS is a collection of interrelated files and a set of programs that allow users to access and modify these files, while maintaining all the required features of data.”

Before we attempt to know about DBMS in detail, we must have a clear understanding about why DBMS.

A Database Management System gives a safe and deployable medium for storage and retrieval of data. The real world data have a structure and constraints associated with it and also different users of the data desire to create, manipulate and use the data. If the users want data to be shared, persistent, secure and easy to access and manipulate, then use of DBMS is must.

DBMS is general purpose system software facilitating each of the following (with respect to a database):

1. **Defining:** specifying data types, data organization, and constraints to which the data must conform
2. **Constructing:** the process of storing the data on some medium (e.g., magnetic disk) that is controlled by the DBMS
3. **Manipulating:** querying, updating, report generation
4. **Sharing:** a database allows multiple users and programs to database concurrently.
5. **Protection & security:** protection from hardware and software crashes and security from unauthorized access.

1.3 NEED FOR A DBMS

For any organization, database plays a major role in order to have persistent and organized data. The database of an enterprise is managed by the database management system. But why do we need the DBMS? To outline it, let us first understand the alternative to it, which is the traditional file-based system.

1.3.1 The Traditional File Based System

Manual file system and computerized file system are the two essential types of Traditional file processing systems. Let's understand each of them in detail:

Manual File System

A manual file system can be set up to hold all the correspondence relating to a particular matter as a project, product, task, client or employee of a company or organization. The manual files include papers containing the data which were organized through a system of file folders and filing cabinets. This system handled its function well as a data storage system as long as collection of data was relatively small. However, the rapid growth of organization and complexed data led to difficulty in keeping track of data in a manual file system. Whenever we have to do cross-reference or process the information in the manual files it breaks down. As a result of this, organizations sought the help of computer technology.

Computerized File Systems

File based systems are an early attempt to computerize the manual filing system. Here, information is stored in permanent files. Before the development of database and database management systems the most frequently used system to store digital data was File Processing System (FPS). So we can define "A file processing system is a collection of programs that store and manage files in computer hard-disk". As the name implies, FPS used to store the data in various files and these files were accessed by different applications. There are various formats in which data can be stored such as .txt, .jpg, .docx, or even structured datatypes such as .xml or .json.

In order to provide easy access each file is used to store in relevant folders. Consider an example of a KSOU student's file system. The KSOU student file will hold information regarding the student i.e., register_number, admission_cycle, student_name, Programme, course etc. Similarly, to store the subject details we have a separate subject file and the result file to store information about the results.

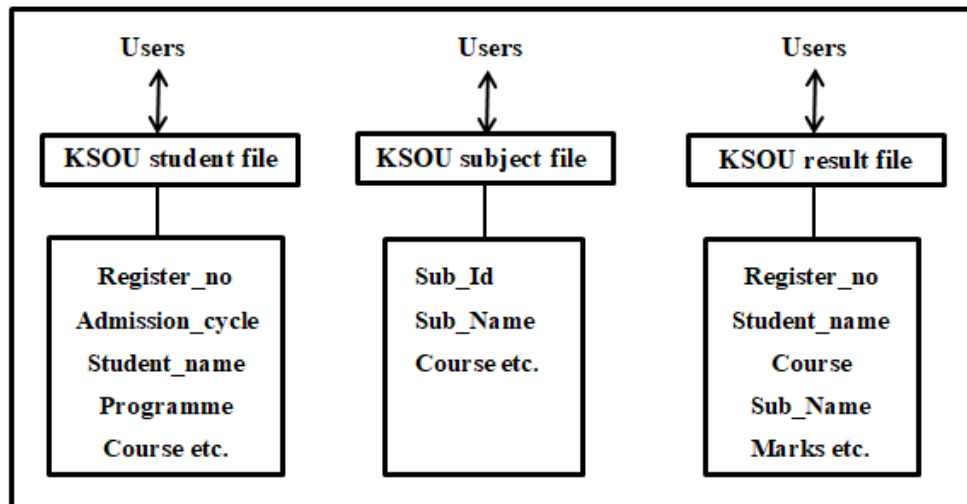


Fig. 1.1: KSOU student's file system

We can observe in the above diagram that some fields are repeated or duplicated in more than one file, which results in data redundancy. In order to overcome this issue, we have to use DBMS which is a centralized approach.

1.3.2 Limitations of Traditional File Based System

This file processing approach has many disadvantages or problems associated with it, including the following:

1. Data redundancy and inconsistency

The main issue with File processing system is that it keeps various versions of same file which leads to duplication of data at multiple places.

Also files are stored in multiple copies, if any one of the file is manipulated, the various versions of same file may not be updated which results in data inconsistency.

2. Data accessing issues

The traditional file-processing systems do not permit required data to be retrieved in a convenient and efficient manner. So, we need a flexible data-access approach for general use.

Consider an example of airline reservation system. Suppose, If the senior management of company needs to retrieve the data of its customers who are residents of Mysuru, it has to be done manually because current file processing system does not permit the user to get this information.

So we have two options to solve the above case:

Option 1: The application programmer has to write a new application program in order to get the required information. But it doesn't ensure that the same request will be asked and the **same application program will be helpful in future.**

Option 2: Manually we have to get the required information from the system.

3. Data isolation

In FPS data is scattered in various files and stored in different formats and also in multiple locations.

So, it is difficult to write new application program to fetch the required information from multiple location and various formats.

4. Concurrent access anomalies

In order to increase the overall performance of the system and accelerated response to applications, numerous systems permit the user to update the data simultaneously. In such an environment, interaction of simultaneous updates may result in inconsistent data.

Consider bank account K, containing rupees 1000. If two customers withdraw funds (say rupees 50 and rupees 100 respectively) from account k at about the same time, the outcome of the simultaneous executions may leave the account in a conflicting state (incorrect state). If the programs executing on behalf of every withdrawal take the old balance, deducts that value by the amount being withdrawn, and update the result back. If the two programs run concurrently, they may both read the value 1000, and write back rupees 950 and 900, respectively. Based on which one updated the value last, the account may have either rupees 950 and 900 instead of the correct value rupees 850. Therefore there should be some protection mechanism to avoid this concurrent updates.

5. Data Security problems

The database system should not permit accessibility of all the data to each and every user. For example, in KSOU, finance section staff required to view only the part of database where financial information has been stored. They do not need to view or access any information related to academic records. However, since application programmes are included to the file-processing system in an ad hoc manner, imposing such security constraint is a difficult task.

6. Integrity problems

Data integrity specifies that the data present in the database are accurate and consistent. In order to ensure that, data stored in the database must satisfy certain consistency constraints that is specified in the application programs.

However, in FPS it is very challenging task to enforce new constraints by making changes in the application program.

Consider an example; initially the minimum bank balance for account holder was 500 later it has been changed to 1000. The real problem arises when we try to apply new condition to the existing database. It is really a tough job to make those changes.

7. Lack of Atomicity

The database must be in a consistent state in spite of failures. Suppose, if there is any failure while updating, inserting or deleting the data in the file system there is no procedure to return back to the previous consistent state.

For example, if the customer is withdrawing some amount from the ATM machine from his account and due to some technical reasons if the system fails to dispense the cash. The database must go back to the previous consistent state i.e., withdrawn should be atomic- it must be happen in it's entirely or not at all.

It is difficult to uphold atomicity in file processing system because of data redundancy, data isolation etc. So to overcome these entire problems, we need to create a centralized system, i.e. DBMS approach.

1.4 CHARACTERISTICS OF DATABASE APPROACH

Knowing all the drawbacks of the traditional file system we can decide the required characteristics for a good database management system. Few of the important characteristics that distinguish the database approach from the file-based system are:

- **Self-describing nature of a database system:**

DBMS not only contains database but also the metadata which describes the data elements, data structures and relationship between the tables in the database. Hence it is referred as self-describing. Database users can use this information if required.

- **Insulation between programs and data:**

In the database approach, the data structure is stored in the system catalogue and not in the programs. Thus, it allows changing the data structure without altering the DBMS access programs.

- **Data Abstraction:**

”A data model is used to hide storage details and present the users with a conceptual view of the database. Programs refer to the data model constructs rather than data storage details”.

- **Support of multiple views of the data:**

Database approach provides multiple vies of data. A *view* is a subset of the database. Each user may see a different view and it may contain only the data of interest to that user.

- **Sharing of data and multi-user transaction processing:**

The database approach allows multiple users to access the same database at the same time. In order to achieve this access *concurrency control strategies* are used. These strategies guarantees that the data accessed are consistently accurate and integrity of data is maintained.

1.5 ADVANTAGES OF DATABASE MANAGEMENT SYSTEM (DBMS)

A DBMS approach is highly opted over traditional file processing system because of the following advantages:

- **Reduction of Redundancies**

In a traditional FPS, every department or group keeps its own private files which results in significant amount of redundancy in the stored data. This leads to wastage of storage space. Also, the files that represent the same data may become inconsistent as some may be updated whereas others may not be. This problem can be avoided by adapting to centralized database management system.

- **Shared Data**

A DBMS has the ability to share the data to any number of users or application programs under its control. The data can only be shared by the authorized users as DBMS centralized and follows different levels of authorization protocols.

- **Data Independence**

DBMS provides abstract view of the data by separating the data description from data. This mechanism hides the storage and representation details of data from application programs.

- **Improved Integrity**

Data Integrity refers to accuracy, consistency and completeness of data. In DBMS data integrity is enforced by providing some integrity constraints such as: entity integrity, referential integrity and domain integrity. Also, these consistency rules are not allowed to violate by the database. Constraints may apply to data items within a record or relationships between records. For example, the age of a KSOU student can be between 18 and 90 years only. So, the database should check this constraint while entering the data for the age of student.

- **Efficient Data Access**

DBMS makes it possible to provide quick answers to database queries thus making accurate and faster access of data.

- **Transaction processing**

DBMS contains a concurrency control mechanism which ensures that data remains consistent and valid during transaction processing even if many users update the same information.

- **Improved Data Security**

Since database allows more number of users to access and share the data among others, there exists a greater risk associated with data security. DBMS permits only authorized users to access the database. DBMS uses various strategies to authenticate identity of users; one such strategy is username and password. Since DBMS is centralized, access may be given to various types of data at different levels.

For example, in KSOU, admission section clerk may be given the authority to know the details of all the students such as name, age, DOB etc., but not the details of marks scored in each subject by the students.

- **Improved Backup and Recovery**

DBMS provides various techniques to recover from the hardware and software failures. A backup and recovery subsystem is responsible for this. In case a program fails, it restores the database to a state in which it was before the execution of the program.

1.6 APPLICATIONS OF DATABASE MANAGEMENT SYSTEM (DBMS)

Due to enormous advantages, DBMS is frequently used in various sectors, be it education, ecommerce, companies, telecommunications, defence etc. some of the significant applications of DBMS in real life based on various fields are as follows:

- **Railway Reservation System**

DBMS plays a major role in railway reservation system. Details of the customers such as name, aadhar number, age, mobile number, booking status, arrival and departure timings etc., can be stored and maintained in the database. Due to any reason if there is delay in the arrival time of train or cancelled, users are intimated through the database updates.

- **Education**

As we have come across the pandemic situation due to covid-19, students might have taken up online exams these days. This situation made universities, colleges, and examination boards take advantage of DBMS to keep or update data like registration details of students, fee payment, exam centres, exam results, etc. in the database.

DBMS is not only used for examination but to maintain databases to store the details of programmes offered, personal information of students, attendance, results, and many other details.

- **Library Management System**

As we are aware, there are thousands of books in the library. It is really difficult to keep information of all books manually in a register. DBMS helps library personnel to maintain all the information related to book issue dates, name of the book, author and availability of the book.

- **Banking**

Banking is viewed as one of the main applications of DBMS. DBMS assists to store millions of user account holder's records systematically. In addition to that, other banking information like loans, fixed deposits, interest credited etc., are maintained very efficiently.

- **Social Media Sites**

The era of smart phone made the users to get addicted to the use of social media. Millions of people have created account in Facebook, twitter, tiktok etc. Database contains all these records and manages it very systematically and securely.

- **Human Resource Management (HRM)**

Human Resource Management keeps and manages the records like name, designation, salary, tax, etc., of the employees. DBMS helps to manage all such information in databases securely.

These are very few applications; still there are plenty of applications in various fields where the DBMS is required to manage the data such as E-commerce, telecommunication, sales and marketing, medical field, agriculture sector etc.

1.7 DISADVANTAGES OF DATABASE MANAGEMENT SYSTEM (DBMS)

The advantages and applications outlined in the previous sections, makes DBMS a strong approach alternative to traditional file system. However, it does have some limitations. So, it is essential to have knowledge of it. Let's discuss some of the disadvantages of DBMS approach.

- **Database Complexity**

DBMS allows users to access many features such as creating a database, retrieving information, updating or deleting data etc. All these functionalities made DBMS highly complex software and it cannot be easily accessed by non-technical person. Without proper skill if one tries to use DBMS it can lead to loss of data or database functioning issues.

- **Increased Costs**

In order to implement DBMS, one has to invest huge amount as it requires sophisticated hardware, software, data conversion and highly skilled personnel. One has to spend additional costs for Software and hardware maintenance.

- **Application programs gets virtually affected if the database is corrupted**

Due to hardware or software failure if any part of the database gets corrupted or damaged, it implicitly affects the application programs which are dependent on the database. This is because centralised database doesn't allow having many versions of the file.

- **Difficult Backup and Recovery**

Since huge data is involved in DBMS, backing up of the whole data is really time consuming and difficult.

Database permits access to multiple users at same time, so it is extremely difficult to get the exact consistent state of database at the time of failure. This can cause loss of data to some extent.

1.8 DATABASE LANGUAGES

Database Language is a unique type of programming language which is used to define and manipulate a database. It is classified into four different types based upon the different operations performed by the language. These include: DDL, DML, DCL, and TCL.

1. Data definition language (DDL)

DDL contains a set of unique commands required to define and modify the structure and the metadata of the database. These commands define the implementation details of the database schemas, which are usually hidden from the users. Using DDL commands we can create, modify, and delete the database structures such as schema, tables, indexes, etc. DDL is also called as data description language in certain context, as it describes the fields and records in a database table.

These commands can modify the structure of the database hence all the modifications implemented by a DDL command is permanently auto saved in the database (auto-commit). Normally, DDL commands are not used by casual end users, who access the database via application program. We will discuss the DDL commands in detail in the forthcoming units.

2. Data Manipulation language (DML)

DML is a type of language that allows users to interact or manipulate the data as organized by the relevant data model. Using DML commands we can perform certain operations such as insertion, deletion, modification, and retrieval of the data from the database. It is categorised into two types: Procedural DML and Declarative DML.

- Procedural DML: User has to specify what data are needed and how to get those data.
- Declarative DML: User has to specify what data are needed without specifying how to get those data

We will discuss the DML commands in detail in the forthcoming units.

3. Data Control Language (DCL)

DCL contains a set of unique commands used to control the user privileges in the database system. Thus DCL is mainly related to security issues. It complements the data manipulation language (DML) and the data definition language (DDL).

It helps in controlling access to information stored in a database and we require data access permissions to execute any command in the database system. This user access is controlled using the DCL statements. These statements are used to grant and revoke user access to data or the database.

4. Transaction Control Language (TCL)

TCL contains a set of unique commands that deal with the transactions that take place in a database. A transaction is nothing but a set of related tasks that are considered as a single logical unit of execution by DBMS software. TCL commands manage all the modifications performed by the DML statements. Some of the TCL commands are: commit, savepoint, rollback and set transaction.

1.9 THE PURPOSE OF DATABASE APPLICATIONS

A database application is a program used to interact with the database. These programmes are developed using conventional programming languages such as C, C++, Java etc. In order to access or interact with the database, data manipulation language statements are used and executed from the conventional programming language. This can be done in two ways:

- a) In order to obtain the results, an application program interface is given which contains a set of procedures to send DML and DDL statements to the database.
- b) "By extending the conventional language syntax to embed DML calls within the conventional programming language. Usually, a special character prefaces DML calls, and a preprocessor, called the DML precompiler, converts the DML statements to normal procedure calls in the conventional programming language".

1.10 DATABASE USERS AND ADMINISTRATORS

The aim of DBMS is to store and manage the data in database for different types of users. There are varieties of users who use DBMS with different aspects. Based on the need DBMS users can access or retrieve the data using the application programmes and interfaces offered by the DBMS. Database users are categorised into different types of users based on their requirements and interaction with the database. Let's discuss each one of them in detail.

1. Database Administrator:

The IT professional who has central control over both the data and the programs that access those data in the database management system is called the Database Administrator (DBA). The following are the responsibilities of DBA:

- a) **Schema definition:** The DBA is responsible to create the original database schema by enforcing a set of data definition statements.
- b) **Storage structure and access-method definition:** The DBA is responsible to create suitable storage structure and interaction methods by writing a set of definitions.
- c) **Schema and physical-organization modification.** The DBA is responsible to carry out modifications to the schema and physical organization to implement the changes required by the organization, or to modify the physical organization to enhance performance.
- d) **Granting data access authorization:** DBA can control which parts of the database different users can access by providing various types of authorization. This authorization information is stored in a special system structure and consulted at the time of data access.
- e) **Integrity Constraint Specification:** In order to keep the data in the consistent form DBA designs and implements integrity rules like primary keys, referential keys, validation rules and access control.
- f) **Routine maintenance:** The DBA is also responsible for routine maintenance activities such as: Periodically backing up the database, ensuring that enough free disk space is available for normal operations, upgrading disk space as required etc.

2. Application programmers

They are software developers who interact with database by writing application programs or user interface. These professionals may use data manipulation queries written in the application programs like C, C++, JAVA etc.

3. Sophisticated users

Sophisticated users are those who use database query language to interact with the database. It indicates that, these users do not use any existing application program to get the desired results from the database. These users have a thorough knowledge about DBMS and Data manipulation languages and may use OLAP (Online Analytical Processing) and data mining tools.

4. Naive users

They are the unsophisticated users who do not have any skill about database. However, these users interact with the database by invoking one of the existing application programs to get the desired results. For example, when user visits the ATMs, he interacts with the existing application and gets the desired results such as withdrawal of amount, balance enquiry etc., without any knowledge about database.

5. Specialized users

They are also sophisticated users who develop special database application programs according to the requirement.

1.11 CHECK YOUR PROGRESS

1. Define database management system.
2. List any two applications of DBMS.
3. Using a database redundancy can be reduced. State true/false.
4. Data Integrity refers to _____, _____ and _____ of data.
5. A database allows multiple users and programs to database concurrently. State true/false.
6. A database application is a program used to interact with the _____.
7. What are the different types of languages that are available in the DBMS?
8. Mention the responsibilities of DBA.

Answers to check your progress.

1. “A DBMS is a collection of interrelated files and a set of programs that allow users to access and modify these files, while maintaining all the required features of data.”
2. The two applications are:
 - Reduction of Redundancies
 - Efficient Data Access
3. True.
4. Accuracy, consistency and completeness.
5. True.
6. Database.
7. Database languages classified into four different types based upon the different operations performed by the language. These include: Data definition language (DDL), Data Manipulation language (DML), Data Control Language (DCL) and Transaction Control Language (TCL)
8. The responsibilities of DBA are:
 - Schema definition
 - Storage structure and access-method definition
 - Schema and physical-organization modification
 - Granting data access authorization
 - Integrity Constraint Specification
 - Routine maintenance

1.12 SUMMARY

In this unit, we have discussed about DBMS and how it has become essential part of our everyday life. We have learnt about traditional file based system and its limitations. In order to overcome the limitations of file-based system, a new approach, a database approach, emerged. A database is a collection of logically related data. This has a large number of advantages over the file-based approach. DBMS system can be used in the fields such as transportation, education, banking, sales, manufacturing, human resource etc. We have discussed about database languages which are categorised as DDL, DML, DCL and TCL based upon the different operations performed by the language. At the end of the unit, we have also discussed about different types of users who use DBMS with different aspects.

1.13 KEYWORDS

- **Data:** Data is raw, unprocessed, unorganized facts that are seemingly random and do not yet carry any significance or meaning.
- **Information:** Information is the outcome of processing raw data to disclose its meaning
- **Database:** It is a collection of related information stored so that it is available to many users for different purposes.
- **Redundancy:** Redundancy in DBMS is having several copies of the same data in the database.
- **Data concurrency:** It is the ability to allow multiple users to affect multiple transactions within a database.
- **Atomicity:** It is a feature of databases systems dictating where a transaction must be all-or-nothing.

1.14 QUESTION FOR SELF STUDY

1. What is DBMS? Explain it with its advantages and disadvantages.
2. Describe the characteristics of DBMS.
3. Discuss the limitations of traditional File processing systems.
4. Differentiate traditional File systems and database systems.
5. What are the applications of DBMS and explain with any one example?
6. What are the types of database languages in DBMS? Difference between DDL and DML.
7. Explain the types of database users.
8. Discuss the responsibilities of DBA.

1.15 REFERENCES

1. Silberschatz Abraham, Henry F. Korth, and Shashank Sudarshan. Database system concepts. New York: McGraw-Hill, sixth edition, 2011.
2. Raghu Ramakrishnan, and Gehrke Johannes. "Database management systems." (2022).
3. Coronel, Carlos, and Steven Morris. *Database systems: design, implementation, & management*. Cengage Learning, 13th edition, 2016.
4. Leon, Alexis, and Mathews Leon. *SQL: A Complete Reference*. TaTa McGraw-Hill, 1999.
5. Gillenson, Mark L. *Fundamentals of database management systems*. John Wiley & Sons, 2008.

UNIT-2: DATABASE CORE CONCEPTS AND ARCHITECTURE

STRUCTURE

- 2.0 Objectives
- 2.1 Introduction
- 2.2 Data Model
- 2.3 Basic building blocks of data model
- 2.4 Types of Data Model
 - 2.4.1 Hierarchical Model
 - 2.4.2 Network Model
 - 2.4.3 Relational Model
 - 2.4.4 Object oriented Model
- 2.5 Schema and Instances
- 2.6 The logical DBMS architecture
- 2.7 Data Independence
- 2.8 DBMS Interfaces
- 2.9 Structure of DBMS
- 2.10 Check your progress
- 2.11 Summary
- 2.12 Key words
- 2.13 Questions for self-study
- 2.14 References

2.0 OBJECTIVES

After studying this unit, you will be able to:

- Define data model.
- Explain why data models are important.
- Describe the basic building blocks of data-modelling
- Discuss about various types of data models.
- Define schema and instances.
- Sketch and explain the 3 schema architecture of DBMS.
- Elucidate data independence.
- Describe overall structure of DBMS.

2.1 INTRODUCTION

In this unit, we are going to discuss about the basic data-modelling concepts and architecture of DBMS. In the beginning of this unit, we discuss about data models which give a structure to describe the design of a database and concepts of schemas and instances, which are fundamental to the study of database systems. Then, we discuss the three-schema DBMS architecture and data independence which provides a user's perspective on what a DBMS is supposed to do. At the end of this unit, we describe the types of interfaces and languages that are typically provided by a DBMS along with the components of DBMS.

2.2 DATA MODELS

In this section let's focus on a crucial question: how is the data organised in a database? In a database system, there are many different fundamental structures. Database models are the name given to them.

The Data Model provides us an idea of the system's final appearance after it has been fully deployed. It describes the data elements as well as their relationships. Data models are frequently used in database management systems to display how data is linked, stored, accessed, and modified. Here, the information is represented by a combination of symbols and texts.

The process of database design begins with data modelling. This stage, which is also known as conceptual design, is occasionally seen as a high-level and abstract design phase. This stage's goal is to describe:

- The data present in the database (example: entities such as students, professors, programmes, courses).
- The relationships between the data items (example: students are supervised by professors; professors teach programmes).
- Constraints on data (example: student register number has exactly ten digits; a course has four blocks, each block is of 4 units and follows credit system only).

2.3 BASIC BUILDING BLOCKS OF DATA MODEL

In the previous section, we have discussed about the fundamentals of data model and understood that data model gives a structure to describe the design of a database. In this section, we discuss basic building blocks of data model.

Entity, entity set, attributes, constraints and relationships are the basic building blocks of all data models. Let's understand each of them in detail:

Entity: "An entity represents the real world objects which are distinguishable from each other". It can be a person, object, place or event. For example: customer, student etc.

Entity Set: "Group of entities of the similar type that share the same properties are called as entity sets". For example: set of cars, set of *customer* (all customers of the bank) etc.

Attributes: An attribute is a characteristic of an entity. An entity may contain any number of attributes. For example, a KSOU student entity could be described by attributes like student name, student DOB, student phone number, address etc.

Constraints: constraints are the set of rules that placed on the data. It plays very essential role in DBMS as it ensures data integrity. For example, the age of a KSOU student can be between 18 and 90 years only.

Relationships: A relationship is an association between entities. For example, a relationship exists between students and teachers that can be described as follows: teacher can handle many students. Here handle is considered as relationship between teacher and student entity. Data models use four types of relationships: one-to-one (1:1), one-to-many (1: M), many-to-many (M: N) and many-to-one (M:1). We will discuss about each relationships with example.

- **One-to-many relationship:** This relationship is an association among instances of an entity with more than one instance of another entity. For example, one client may make a few buys, but each buy is made by a single client.
- **Many-to-many (M:N or M:M) relationship:** A M:M relationship could be a relationship among multiple instances of one entity and multiple instances of another entity, i.e. both entities can have numerous connections with each other. For example, in KSOU, a student might learn many open elective courses, and each open elective course might be learned by many students. Hence, yielding the M: M relationship.
- **One-to-one (1:1) relationship:** *This relationship* is an association among an instance of an entity with another. For example, in KSOU, each student has only one student register number and each student register number is allotted to only one student.
- **Many-to-one (M:1) relationship:** Many-to-one relationships are those in which entities in one entity set can participate in a relationship set only once, whereas entities in another entity can participate multiple times. For example: A relationship

between student and university, a student can only enroll in one university at a time while a university might have many students.

2.4 TYPES OF DATA MODEL

As we have discussed in the section 2.2, with the use of data models, the real world is abstracted into the digital format. From the perspective of the application, the storage and retrieval of information is facilitated by the various database models. The primary difference between the different database models lie in the methods of expressing relationships and constraints among the data elements. There are many kinds of data models. The most popular database models are discussed in this unit, which includes:

1. Hierarchical Model
2. Network Model
3. Relational Model
4. Object oriented Model

2.4.1 HIERARCHICAL MODEL

It is one of the first and oldest data model, dating from the late 1950s. In this model, the data are arranged in a hierarchical tree form. The hierarchy begins at the root, which contains root data, and grows as a tree by adding child nodes to the parent node. This model accurately captures some of the relationships that exist in the real world, such as food recipes, website sitemaps, etc.

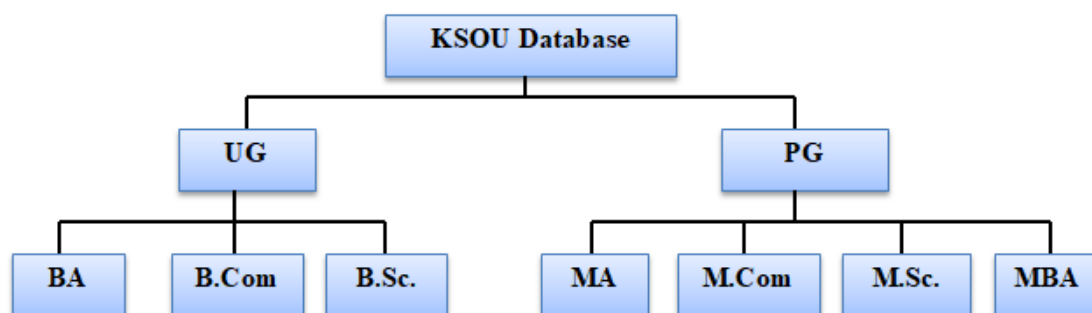


Fig 2.1: Hierarchical Model Representation

Features of Hierarchical Model:

1. **One-to-many relationship:** The data in this model are arranged in a tree-like structure, with the datatypes forming the one-to-many relationship. There can only be one path from

any parent to any node as well. Example: In the fig 2.1, there is only one way to go to the node B.Sc. through the UG node.

2. **Parent-Child relationship:** Each child node has a parent node, while a parent node may have more than one child node. No more than one parent may be present.
3. **Deletion Problem:** The issue with deletion is that when a parent node is removed, all the child nodes will automatically get deleted.
4. **Pointers:** Pointers are utilised to link between the stored data and to connect the parent node with the child node. Example: In the fig 2.1, the node "KSOU database" points to the nodes "UG" and "PG."

2.4.2 NETWORK MODEL

The network data model was established in the late 1960s by the Database Task Group of the Conference on Data System Language (DBTG/CODASYL). For some applications it is extremely necessary to have more than one parent. As a result, the network model made it possible to model many-to-many relationships in data. Thus, the network model was formally defined by the Conference on Data Systems Languages (CODASYL) in 1971. The only distinction between this model and the hierarchical model is that a record can have more than one parent. In this model, graph is used in place of the hierarchical tree.

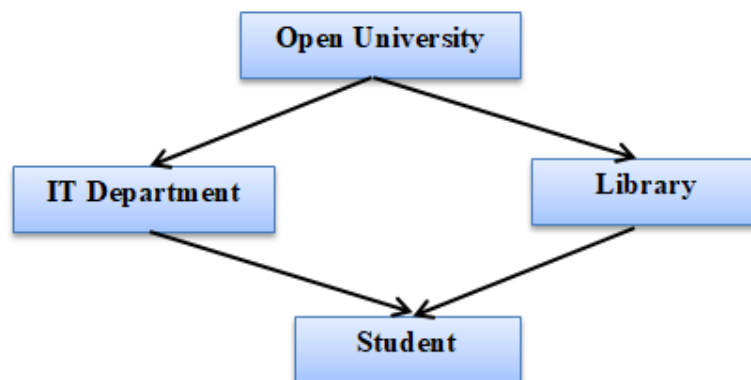


Fig 2.2: Network Model Representation

Features of Network Model:

1. **Ability to combine more relationships:** In this approach, as there are more relationships the data are more closely related. This model is capable of handling both many-to-many and one-to-one relationships.
2. **Many paths:** There may be more than one path to the same record since there are more relationships. This facilitates quick and easy data access.

3. **Circular Linked List:** The circular linked list is used to perform operations on the network model. A programme is used to maintain the present position, which navigates through the records according to relationships.

2.4.3 RELATIONAL MODEL

This approach uses two-dimensional tables to organise the data, and it preserves the relationship by keeping a common field. Since its introduction by E.F. Codd in 1970, this model has been the most popular and, in some cases, the only database model used globally. The basic structure of a relational model is tables. So, the tables are also called relations in the relational model. All the information related to a particular type is stored in rows of that table.

To make the database flexible, adaptive, and scalable, tables inside the database can be normalised, or brought into compliance with normalisation standards. Each piece of data is atomic, or divided into its smallest usable components, when it is normalised. A 6th unit in the second block of this course describes in detail the processing of normalization and its stages. At the moment it is sufficient to say that normalization is a technique, which helps to organise data in a database. This includes “creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency”.

Student Table

SL. NO	NAME OF THE STUDENTS	REGISTER NO.	MOBILE NO.	PROGRAMME	→ Attributes
1	GANESH	2010102500001	9986485564	M.Sc.	
2	ANANYA	2010102500002	9986147564	M.Sc.	
3	CHAUHAN	2010202500001	8123998648	M.Sc.	
4	RANJINI C	2010202500002	9035973051	M.Sc.	→ Tuple
5	RAGHU S	2010702500001	9123674122	M.Sc.	

Fig 2.3: Representation of (Relational model)

Features of Relational Model:

1. **Tuples:** Each row in a table is called tuple. All the information about any instance of the object is contained in a row. For example, in the fig 2.3 each row has all the details about any specific student, such as Ganesh’s information in the first row.

2. **Attribute or field:** An attribute is a property that specifies a relation or table. The attribute's values should be from the same domain. For example, in the fig 2.3 we have different attributes of the student, including register no., mobile no etc.

2.4.4 OBJECT ORIENTED MODEL

The object-oriented data model more accurately depicts the problems of the real world. In this model, both the data and relationship are present in a single structure known as an object. This model describes a database as a group of objects, or reusable software components, with related features and procedures. Various types of object-oriented databases exist, including:

- **Multimedia databases:** Images and other types of media that cannot be stored in a relational database are allowed to store in this (Although audio and video can be stored in relational databases, it is not recommended to do so).
- **Hypertext database:** Any object can link to any other object using a hypertext database. Although it's helpful for organising a lot of different data, it's not the best option for numerical analysis.

Generally, in this model, two or more objects are connected through links. We use this link to relate one object to other objects.

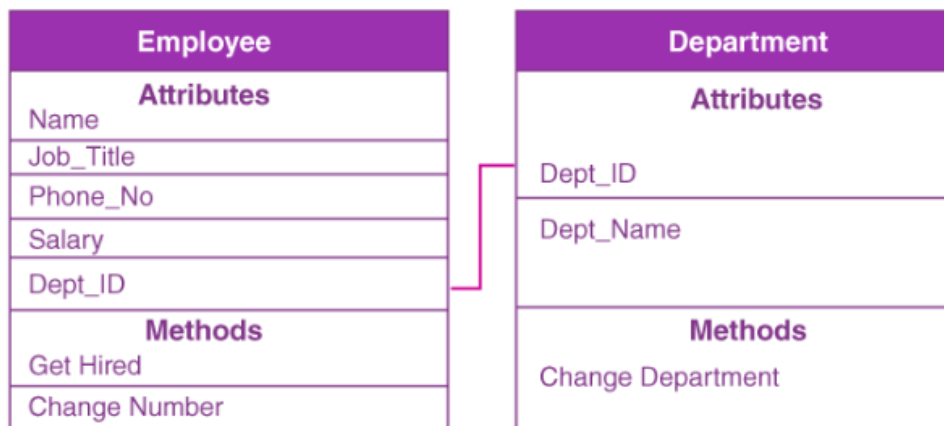


Fig 2.4: Object oriented model representation

For example: In the fig 2.4, we have two objects Employee and Department. Each object's data and relationships are all contained as a single unit. The attributes like Name, Job_title of the employee and the methods which will be performed by that object are stored as a single object. The Department id attribute links the two objects together, and it is through this attribute that they will communicate with one another.

2.5 SCHEMA AND INSTANCES

In DBMS, instance is the information stored in the database at a particular moment and schema is the overall design of the database. Let's understand schema and instances with suitable example.

INSTANCES: The data in the database at a particular moment of time is called an instance or a database state. In a given instance, each schema construct has its own current set of instances. Many instances or database states can be constructed to correspond to a particular database schema. Every time we update (i.e., insert, delete or modify) the value of a data item in a record, one state of the database changes into another state.

Let's use the KSOU student table as an example. The table currently contains 100 records, making the database instance 100 records at this particular moment of time. Let's assume that we will add 100 more records to this table by tomorrow, giving the database instance 200 records in the table. The data that is stored in a database at a specific time is referred to as an instance, and it varies over time as we add and remove data.

The table 2.1 shows an instance of the KSOU student table in a database schema. Here, the table currently contains 10 records, making the database instance 10 records at this particular moment of time.

Table 2.1: KSOU student table

SL.NO	NAME OF THE STUDENTS	REGISTER NO.
1	SHEETHAL N M	2010102500001
2	ANANYA	2010102500002
3	CHAUHAN SONALI	2010202500001
4	RANJINI C	2010202500002
5	RAGHU S	2010702500001

SCHEMA: Database schema refers to the overall design of the database. Schema won't be modified often. It is a database's logical structure that identifies the entities, properties, and relationships between them. A schema specifies the database name, the record type, and the elements that make up each record. It is also defined as a framework into which the values of the data items are fitted. The values incorporated into the framework vary frequently, while the schema's structure does not. It includes a description of the database that can be illustrated

using a schema diagram. The database designers create the schema to make the database easier for programmers to comprehend and use.

Typically, a schema can be divided into three types, namely:

1. **Logical database schema:** This schema provides a logical level description of the database design. It outlines all the logical restrictions that must be imposed to the data being stored. Tables, views, and integrity restrictions are also defined.
2. **Physical database schema:** The physical schema describes the database design at the physical level. It deals with how data is really stored, including the types of storage (files, indices, etc.). It outlines the procedures for data storage in a secondary storage.
3. **External schema:** It is schema at the view level. This is the highest level of a schema where the views for end users are defined.

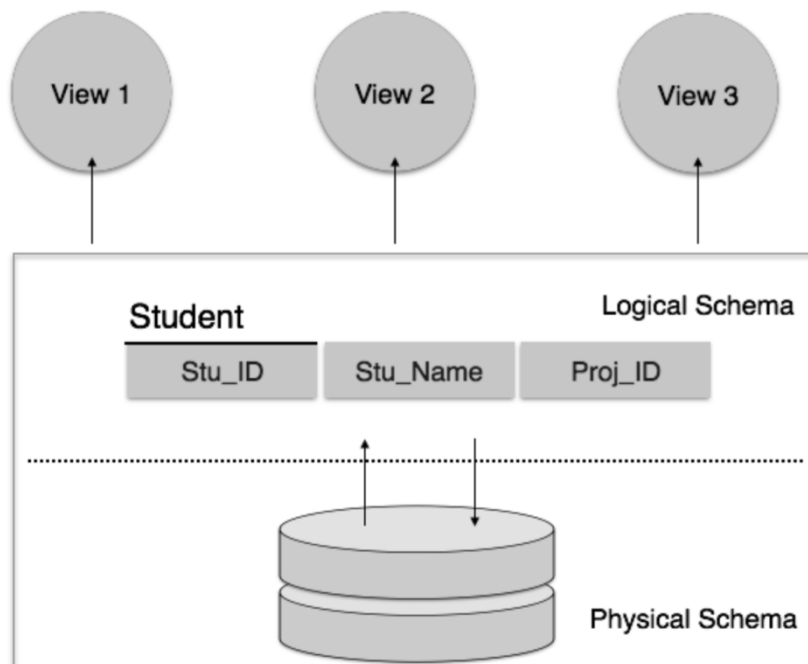


Fig. 2.5: Database Schemas

Subschema: It is a subset of the schema having the same properties that a schema has. It identifies a subset of areas, sets, records, and data names defined in the database schema available to user sessions. The subschema allows the user to view only that part of the database that is of interest to him. The subschema defines the portion of the database as seen by the application programs and the application programs can have different view of data stored in the database. The different application programs can change their respective subschema without affecting other's subschema or view.

2.6 THE LOGICAL DBMS ARCHITECTURE

Database Management Systems are exceptionally complex and advanced software applications that give secured management of huge amounts of data. In order to describe database concepts, database structure and capabilities in a better way, we must study the architecture of a typical DBMS. There are two distinctive ways to see at the architecture of a DBMS, one is logical DBMS architecture and the other is the physical DBMS architecture. The main concern of logical architecture is to present the way data is stored and displayed to users, whereas the physical architecture deals with components of the software which make up a DBMS.

This architecture describes how data within the database is seen by users. It does not deal with how the data is managed and processed by the database system, but only concerned about how it appears. Here, the technique of data storage is not revealed, thus users can easily modify the data without bothering about where it is stored or how it is stored. As a result of this, DBMS has different levels of abstraction.

In this section, we discuss about three tier architecture of database management systems, called the **three-schema architecture**. This architecture divides the system into three levels of abstraction: the internal or physical level, the conceptual level, and the external or view level. The Figure 2.6 below shows the three schema architecture of DBMS.

The External Level

External level is also called as view level. This is the highest level of abstraction of database and provides a number of external schema or views to users. This level is the one closest to the users. It permits the users to see only the part of the database that is interested to them and hides the remaining part of the database from users. Here, the storage and implementation details are hidden from the users.

The Conceptual Level

The conceptual level provides logical view, which describes the structure of the entire database for a community of users. It hides the details of physical storage structures and it is only concerned about the information content such as describing entities, data types, relationships, user operations, and constraints. DBMS provides a data definition language for defining the conceptual view of the database.

The Internal Level

Internal level is also called as physical level. This is the lowest level of abstraction and deals with how the data is stored physically and where it is stored in database. The internal level is the one closest to physical storage, and it acts as an interface between the operating systems file system and the record structures used in higher levels of abstraction.

We have to keep in mind that “the three schemas are only *descriptions* of data; the only data that *actually* exists is at the physical level. In a DBMS based on the three-schema architecture, each user group refers only to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is database retrieval, the data extracted from the stored database must be reformatted to match the user’s external view”.

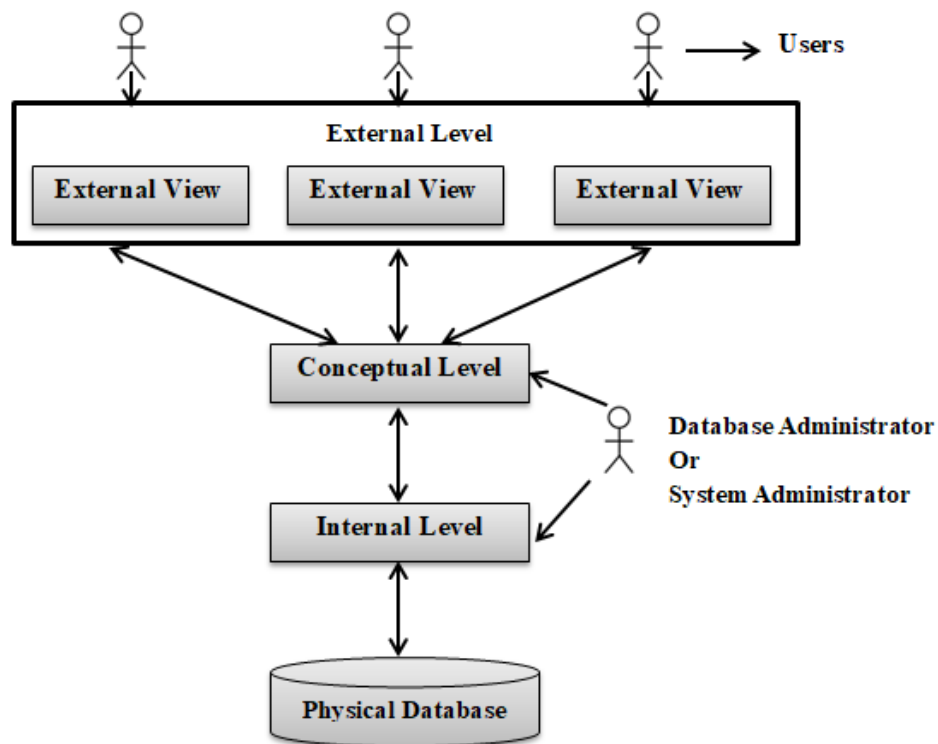


Fig. 2.6: The three schema architecture of DBMS

2.7 DATA INDEPENDENCE

The three-schema architecture discussed in the previous section will help us to easily understand the concept of data independence. “Data independence is defined as the ability to modify the schema at one level of a database system without having to modify the schema at

the next higher level”. Definition of Data Independence makes us clear that, it helps us to change data at one layer, without affecting the data at another level. We have two levels of data independence:

- **Logical Data Independence:**

Logical data independence is the ability to modify the conceptual schema without having to alter external schemas or application programs.

For example, suppose if we want to change the conceptual schema to add a record type, to change constraints etc. As DBMS supports logical data independence, we need to change only the view definition and the mappings in DBMS no need to worry about storage space or type.

- **Physical Data Independence:**

Physical data independence is the ability to modify the internal schema without having to alter conceptual schemas. Thus, there is no need to change external schema. However, some changes may be required in the internal schema due to the reorganization of some of the physical files.

For example, suppose if we want to upgrade the storage system itself, in some cases we may need to replace hard-disks with SSD. This upgradation should not have any impact on the logical data or schemas.

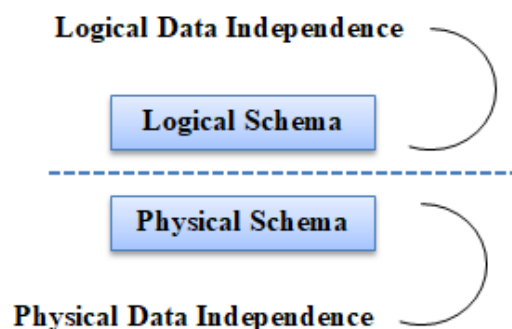


Fig. 2.7: Data Independence

2.8 DBMS INTERFACES

DBMS provides interface to users for interacting with the database. It has the ability which allows inputting the queries without utilising the database query languages. DBMS has given the following User-friendly interfaces:

- **Menu Based Interfaces for Browsing:** The lists of options called menus are presented to the users by these interfaces which leads the user over the formulation of request. Generally, these interfaces are also utilised in browsing interfaces that permits user to glance over the contents of database in unstructured manner.
- **Forms Based Interfaces:** This interface exhibits a form to every user. In order to insert a new data users have to fill the form entries or they have option to fill only certain form entries, in such cases the DBMS will fetch matching data for the remaining form entries.
- **Graphical User Interfaces:** This interface presents schema to the user in diagrammatic form. Especially Graphical User Interfaces utilize an indicating gadget, such as a mouse, to choose certain parts of the shown schema diagram.
- **Natural Language Interfaces:** Generally, these interfaces maintain its own schema, which is similar to the database conceptual schema, as well as a dictionary of important words. These interfaces accept the request from the user and try to interpret it.
- **Speech input and output interfaces:** These types of interfaces take speech as an input and output the speech as a result. These types of interfaces are commonly used in the inquiry for telephone directory or to get the flight information over the smart gadgets etc.
- **Interfaces for the Database Administrators:** The privileged commands such as creating accounts, granting account authorization etc., available in the database systems can be used only by the database administrator's staff.

2.9 STRUCTURE OF DBMS

The Structure of DBMS is also known as Overall System Structure or Database Architecture but it is different from the tier architecture of Database.

The DBMS serves as an interface between the user and the database in a database structure. The user sends request to the DBMS to carry out a different operations on the database, including insert, delete, update, and retrieve. The DBMS components carry out these requested operations on the database and give users with the information they require. Fig 2.8 illustrates the various DBMS components.

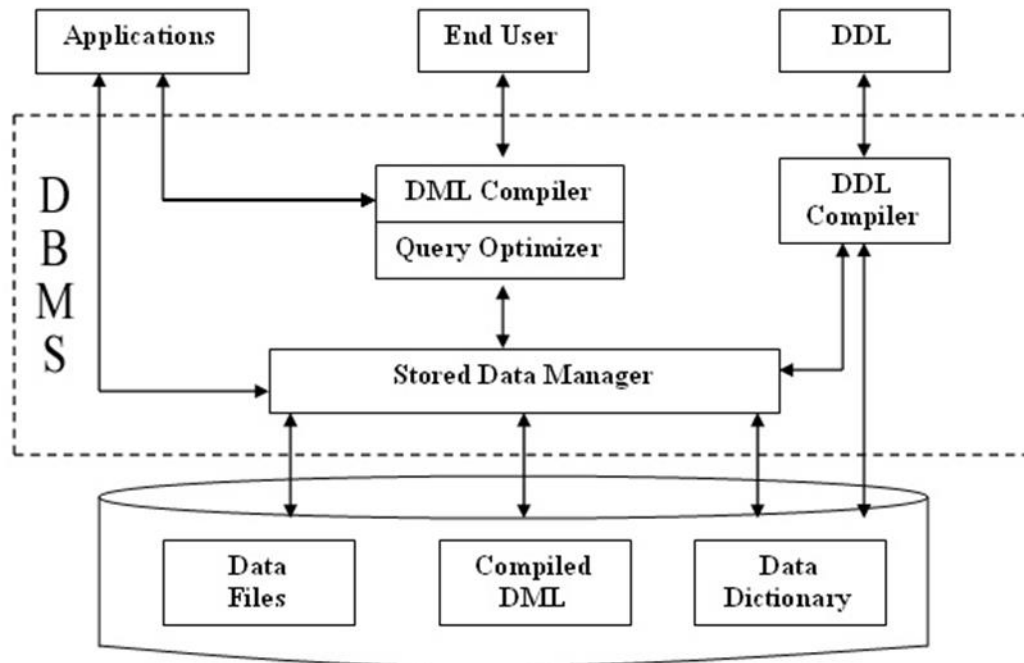


Fig 2.8: Structure of DBMS

DDL Compiler: Schema definitions given in the DDL are processed by the Data Description Language compiler. It contains metadata details such as file names, data items, storage information for each file, mapping details, constraints, etc.

DML Compiler and Query optimizer: The DML compiler receives the DML commands from the application programme and compiles them into object code for database access. These commands include insert, update, delete, and retrieve. The query optimizer then sends the object code to the data manager after optimising it for the best query execution.

Data Manager: The Data Manager, sometimes referred to as Database Control System, is the central software component of the DBMS.

The primary functions of a data manager include:

- One of the responsibilities of the data manager is to convert operations in user queries coming directly from the query processor or indirectly from application programmes from user's logical view to physical file system.
- Data manager is responsible for controlling DBMS information access that is stored on disk and also handling buffers in main memory;
- The data manager is responsible for enforcing constraints to keep the consistency and integrity of the data.

- One of its responsibilities is to provide synchronization in the simultaneous operations carried out by the concurrent users and also to manage backup and recovery procedures.

Data Dictionary: A database's data dictionary serves as a repository for data descriptions. A data dictionary includes a list of all database files, the number of records in each file, the names of all fields, and information about each field's type. To avoid users accidentally destroying the data dictionary's contents, the majority of database management systems keep the data dictionary hidden from users.

The Data Dictionary's functionalities are:

1. Defines the data element.
2. Supports scheduling.
3. Assists in management.
4. Allows the various users who are aware of the data's availability and acquisition methods.
5. Assists in locating organisational data irregularities.
6. Serves as a crucial tool for data management.
7. Offers an effective standardising method.
8. Serves as the corporate vocabulary for the ever growing information resource.
9. Provides the ability to create reports, control systems, and excerpts.

Data Files: Database is stored in the data files.

Compiled DML: The DML compiler converts the high level Queries into low level file access commands known as compiled DML.

End Users: End users are those who use apps or tools to interact with the database. We have discussed about end users in unit 1.

2.10 CHECK YOUR PROGRESS

1. What is an entity in DBMS?
2. Which of the following is not a Schema?
 - (A) Database Schema
 - (B) Physical Schema
 - (C) Critical Schema
 - (D) Logical Schema

3. The relational model was proposed by_____.
4. Define one to many relationships.
5. An attribute is a characteristic of _____.

Answers to check your progress

1. “An entity represents the real world objects which are distinguishable from each other”. It can be a person, object, place or event. For example: customer, student etc.
2. Critical Schema
3. E.F. Codd
4. One-to-many relationship is an association among instances of an entity with more than one instance of another entity.
5. An entity.

2.11 SUMMARY

In this unit, we have discussed about why data models are important and also the basic building blocks of data-modelling. Further, the most often used database models such as hierarchical, network, relational and object oriented were discussed. The concepts of schemas and instances, which act as fundamental to the study of database systems was discussed. We have also discussed the generic database architecture based on the 3-scheme-architecture followed by data independence and database interfaces. Finally, Overall structure of DBMS was discussed including the functional components of a database system.

2.12 KEYWORDS

- **Data Model:** A data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to the properties of real-world entities.
- **Schema:** The database schema is the structure of a database described in a formal language supported by DBMS.
- **Instance:** It is a collection of information that stored in a database at a particular moment.
- **Subschema:** A subschema provides a view of the database as seen by an application program.
- **Conceptual Level:** It describes how the database appears to the users conceptually and the relationships between various data tables.

2.13 QUESTION FOR SELF STUDY

1. What is a data model? Mention its types.
2. Explain the importance and basic building blocks of Database models.
3. Explain Hierarchical Data Model. Give example
4. Describe Network Data Model with example.
5. Explain Relational Database Model.
6. Define instance and schema. Give example.
7. Define subschema.
8. What is data independence? Why it is needed in database?
9. Discuss the different categories of Data independence.
10. Explain the different type of database interfaces.
11. Describe the structure of database with neat label diagram.
12. Describe the responsibilities of Database Manager.

2.14 REFERENCES

1. Silberschatz Abraham, Henry F. Korth, and Shashank Sudarshan. Database system concepts. New York: McGraw-Hill, sixth edition, 2011.
2. Raghuram Ramakrishnan, and Gehrke Johannes. "Database management systems." (2022).
3. Coronel, Carlos, and Steven Morris. *Database systems: design, implementation, & management*. Cengage Learning, 13th edition, 2016.
4. Leon, Alexis, and Mathews Leon. *SQL: A Complete Reference*. TaTa McGraw-Hill, 1999.
5. Gillenson, Mark L. *Fundamentals of database management systems*. John Wiley & Sons, 2008.

UNIT-3: DATA MODELLING USING ER MODEL

STRUCTURE

- 3.0 Objectives
- 3.1 Introduction
- 3.2 Entity relationship model
- 3.3 ER diagram
 - 3.3.1 Naming conventions of schema constructs
 - 3.3.2 ER diagram notations
- 3.4 Entities
- 3.5 Types of entity type
 - 3.5.1 Strong entity
 - 3.5.2 Weak entity
- 3.6 Attributes
 - 3.6.1 Types of attributes
- 3.7 Relationships
 - 3.7.1 Roles in E-R diagrams
 - 3.7.2 Relationship cardinality
- 3.8 Participation constraints
- 3.9 Constructing an ER model
- 3.10 Check your progress
- 3.11 Summary
- 3.12 Key words
- 3.13 Questions for self-study
- 3.14 References

3.0 OBJECTIVES

After studying this unit, you will be able to:

- Use Entity-Relationship (ER) modeling in database design.
- Explain basic concepts associated with ER diagram.
- Define Entity sets, Attributes, keys and Relationships.
- Identify and solve the problems associated with ER diagrams.
- Build ER model from requirement specification.

3.1 INTRODUCTION

In this unit, we are going to discuss about data-modelling using ER model. The ER model is important primarily for its role in database design. It provides useful concepts that allow us to move from an informal description of what users want from their database to a more detailed and precise, description that can be implemented in a DBMS. This unit attempts to provide detailed information about basic aspect of ER model which includes components such as: entities, attributes and relationships. Also, concepts such as weak entity, relationship cardinality and participation constraints are introduced along with suitable examples.

3.2 ENTITY RELATIONSHIP MODEL

Entity relationship model (ER Model) is a high-level conceptual data model developed by Dr. Peter Chen in 1976. The ER Model represents real-world entities and the relationships between them. This model is frequently used to create a database's initial design. Creating an ER Model in DBMS is considered as a best practice before implementing the database. It describes data as a collection of entities, relationships and attributes. Machine-related aspects cannot be described using the ER model.

This model is based on three basic concepts:

- Entities
- Attributes
- Relationships

To develop a database model based on the E-R technique involves identifying the entities and relationships of interest. An ER model uses an Entity Relationship diagram (ER Diagram) to illustrate how the structure of a database is described.

3.3 ER DIAGRAM

An ER Diagram is a visual representation of data that shows how data are related to one another using various ERD Symbols and Notations.

Another definition is “the structure of entities, attributes, and their relationships in a database are described in entity-relationship diagrams, often known as ERDs or ER diagrams”. Let us now discuss about proper naming of schema constructs.

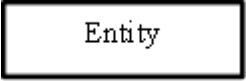


3.3.1 NAMING CONVENTIONS OF SCHEMA CONSTRUCTS

- Select names that accurately reflect the significance of the various schema constructs.
- Utilise singular names instead of plural ones for entity types.
- Use the convention that role names should be written in lowercase, attribute names with their first letter capitalised, and entity type and relationship type names in uppercase.
- The names of entity types are typically derived from nouns that appear in the narrative, and the names of relationship types are typically derived from verbs.
- Selecting binary relationship names to make the schema's ER diagram readable from left to right and from top to bottom.

3.3.2 ER DIAGRAM NOTATIONS

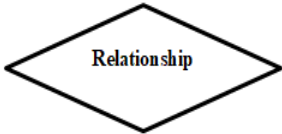

Now that the fundamentals of ER diagram have been discussed, let's discuss more about the typical ER diagram symbols that we have to use. As mentioned in the ER diagram definition, Entity, Relationship, and Attribute are the three main components of ER diagrams. Therefore, let's discuss several ER diagram notations based on these components.

ER Diagram Entity Symbols

SYMBOL	NAME	DESCRIPTION
	Entity	This is a basic entity that is represented by a rectangle with its name inside. Entities are also referred to as strong entities or parent entities since weak entities frequently depend on them. A primary key that distinguishes each instance of the entity will be present in the entity.
	Weak Entity	Weak entities are dependent on another entity since they lack primary key. Without their parent entity, they are meaningless in the diagram.
	Associative Entity	This unique entity is typically utilized in many-to-many relationships and refers to all of its relationships as being "many."

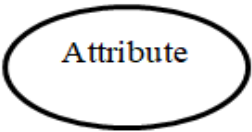

ER Diagram Relationship Symbols

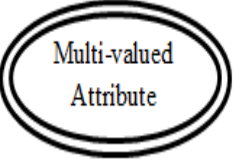

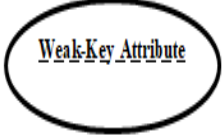
Relationships in an ER diagram would merely specify how two or more entities are related to one another.

SYMBOL	NAME	DESCRIPTION
	Relationship	Associations between or among entities are relationships. A single rhombus with its name inside represents a relationship also referred as strong relationship. In this case, a relationship where entity is existence-independent of other entities and Primary Key of Child doesn't contain Primary Key component of Parent Entity.
	Weak Relationship	A double rhombus with its name inside represents a weak relationship also referred as identifying relationship. In this case, the child entity is reliant on the parent entity since part of the parent entity's primary key is present in the child entity's primary key.

ER Diagram Attribute Symbols



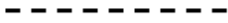
An attribute is a characteristic related to the entity. In any ER diagram symbols, we may locate all different types of attributes that would specify the value or property of any entity. To understand the database in greater depth and detail, these features are used. Attributes are divided into different categories. They are listed below.

SYMBOL	NAME	DESCRIPTION
	Attribute	The name of a basic attribute is written inside a single oval to represent it. They can either show a relation of either one-to-many or many-to-many.
	Key Attribute	"Key attribute" refers to an attribute that uniquely identifies a specific entity. The name of a key attribute is underscored.

	Multi-valued Attribute	This attribute can take multiple values and represented by a double oval.
	Derived Attribute	A derived attribute might not be physically present in the database and could be logically derived from any other attribute. It is represented by a dotted oval.
	Weak Key Attribute	It is an attribute that might be derived from any other attribute, but it would have unique identifiers for the entity. It is represented by a dotted oval with its name underlined.

ER Diagram Inheritance Symbols

The inheritance between child and parent entities must be taken into account when discussing Entity-Relationship Diagram Symbols. The below mentioned types of inheritance are possible in relationships between entities.

SYMBOL	NAME	DESCRIPTION
	Partial Participation	A single line is used to represent partial participation, which shows that not every entity in the collection is a part of the relationship.
	Total Participation	It denotes that all the entities in the set are in a relationship and are represented using a double line.
	Optional Participation	It denotes that the entities don't have a mandatory partition in the set and are depicted using dotted lines.

3.4 ENTITIES

Before discussing about ER diagrams, we need to understand some of the basic concepts. Let us first understand what an entity is?

Entity: “An entity represents the real world objects which are distinguishable from each other”. It can be a person, object, place or event. For example: customer, student etc.

Entity Set: “Group of entities of the similar type that share the same properties are called as entity sets”. For example: set of cars, set of *customer* (all customers of the bank) etc.

Entity Type: The entity type is a collection of the entity having similar attributes.

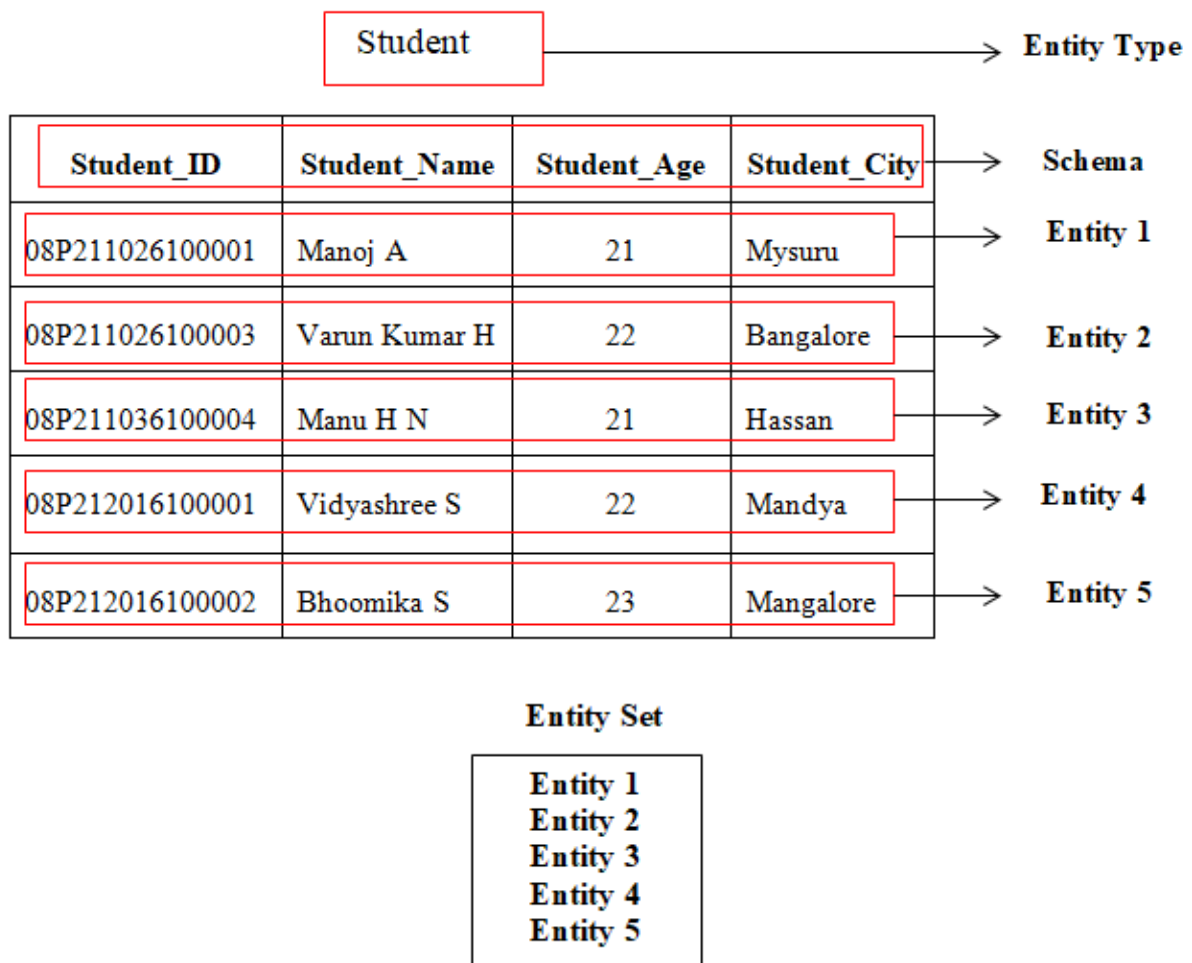


Fig 3.1: Example to illustrate data of different entities for entity type student

In order to make clear the concept entity, entity set and entity type the above given Student table is considered as an example. Here, we have each row as an entity and they are having common attributes i.e each row has its own value for attributes Student_ID, Student_name, Student_age and Email_ID. So, we can define the above STUDENT table as an entity type because it is a collection of entities having the same attributes. So, an entity type in an ER diagram is defined by a name(here, STUDENT) and a set of attributes(here, Student_ID, Student_name, Student_age and Email_ID). The entity set consists of all the records. In other

words, the records with ID08P211026100001, 08P211026100003, 08P211026100004, 08P212026100001 and 08P212026100002 represent the entire data set. It is the entity set. Therefore, entity set refers to all records in the Student table. Overall, the entity set denotes all records of a specific entity type.

3.5 TYPES OF ENTITY TYPE

So far we have discussed about the Entity, Entity type, and Entity set. Now, let's discuss about types of entity type and how they are uniquely different. They are of two types-

- Strong Entity
- Weak Entity

3.5.1 STRONG ENTITY

A strong entity is an entity type whose existence is independent of all other entities. Strong entity types have a key attribute and it aids in uniquely identifying each entity. A rectangle is used to represent it. A relationship between two strong entities is represented by a diamond shape. It is simply called a relationship.

For example: Let's say there are two entity types DOCTOR and PATIENT. The properties Doctor_id, Name, Salary, and Qualification belong to the entity DOCTOR. Likewise, entity PATIENT attributes include Patient_id, Age, Contact_no, Address, and P_name. Both entities are strong entity type as both of them have their own key attribute. The key attribute for the entity type DOCTOR is Doctor_id, and the key attribute for the entity type PATIENT is Patient id. A one-to-many relationship exists between these two entities. More than one patient may be treated by a single doctor. Also, it may be possible that all the doctors are related to some patients. Additionally, it's feasible that some doctors are not related to any patients. Hence, there may or may not be total participation of a strong entity in a relationship. The E-R diagram given in fig 3.2 illustrates the aforementioned relationship.

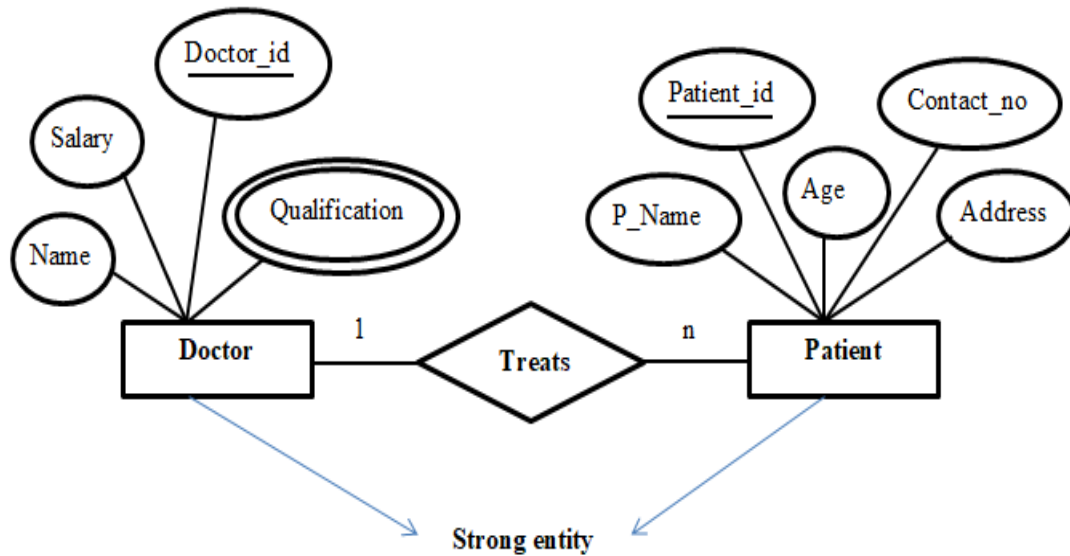


Fig 3.2: E-R diagram with a weak entity set

3.5.2 WEAK ENTITY

Weak entity type doesn't have a key attribute. Weak entity type can't be identified on their own. It is dependent on another strong entity for its unique identity. It is represented by a rectangle with two outlines. The relationship between a weak entity type and strong entity type is called an '**identifying relationship**' and represented by a double-outlined diamond rather than a single-outlined diamond. The dependent entity is sometimes referred to as a **subordinate entity**, and the strong entity is sometimes referred to as the **dominant entity**.

Since the weak entity lacks a key attribute, another attribute must be required in addition to the dominant entity's primary key in order for the subordinate entity to be uniquely identified. This attribute is called the discriminator of the weak entity. It is also called as the partial key.

The weak entity shows total participation in its relations and this is represented by a double line. Total participation indicates each entity in the weak entity type is related to one of the entities in the strong entity type. We will now use an example to try to comprehend the terms listed above.

For example: Let's say we have the entity types EMPLOYEE and DEPENDANT (children of the employee). The attributes of the EMPLOYEE entity include Emp_id, Emp_name, Job_title, Age, Salary, and Address. And, the DEPENDANT entity include the attributes Name, Age, and Relation. Here, Emp_id is the key attribute of the entity type EMPLOYEE so EMPLOYEE is a strong entity. Whereas DEPENDANT entity does not have a key

attribute so it is a weak entity. The 'name' attribute in this case acts as a discriminator or a partial key for the weak entity. This partial key, together with the employee's key attribute, aids in uniquely identifying each DEPENDANT entity. Additionally, every DEPENDANT entity is related to one of the EMPLOYEE entity.

All the EMPLOYEE entity may not be related to one or the other DEPENDANT entity but all the DEPENDANT entity is related to one or the other EMPLOYEE entity. This is called total participation. The weak entity DEPENDANT is identified by the strong entity EMPLOYEE. Consequently, there is a relationship between these two entity types. This relationship is called an identifying relationship. The following E-R diagram can be used to illustrate the relationship mentioned above.

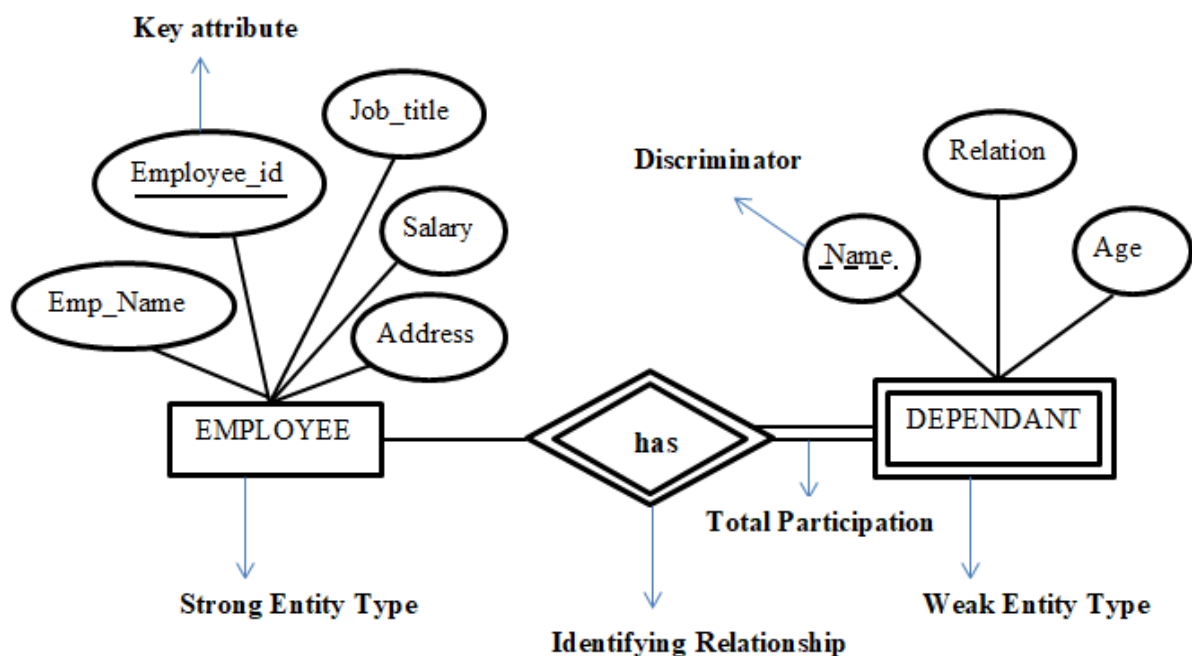


Fig 3.3: E-R diagram with a weak entity set

THE KEY DIFFERENCE BETWEEN STRONG AND WEAK ENTITY

Strong Entity Set	Weak Entity Set
Strong entity set always has a primary key.	It does not have enough attributes to build a primary key.
It is represented by a rectangle symbol.	It is represented by a double rectangle symbol.
It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.

The member of a strong entity set is called as dominant entity set.	The member of a weak entity set called as a subordinate entity set.
Primary Key is one of its attributes which helps to identify its member.	In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

3.6 ATTRIBUTES

An attribute is a characteristic of an entity. An entity may contain any number of attributes. For example, a KSOU_student entity could be described by attributes like student name, student DOB, student phone number, address etc.

For each of an entity's attributes, a value will be assigned. For example, for a particular KSOU_student, the following values could be assigned:

Register No. : 08P211055600092

Student name: Sanmathi

Age: 21

Address: Mysuru

Programme: M.Sc. IT

DOMAIN

To each basic attribute of an entity type there is a possible set of values that can be assigned. This is referred to as the attribute's domain. A value from a different domain cannot be contained in an attribute.

For example, in KSOU_student entity Register number has a specific domain, integer values and alphabets only, no other special symbols are allowed.

3.6.1 TYPES OF ATTRIBUTES

An entity may have different types of attributes associated with it. They are:

Simple attribute: A simple attribute is one that cannot be further subdivided into components and expresses the basic meaning. For example, in KSOU_student entity, Register number, Class and Age are simple attributes as they cannot be divided further.

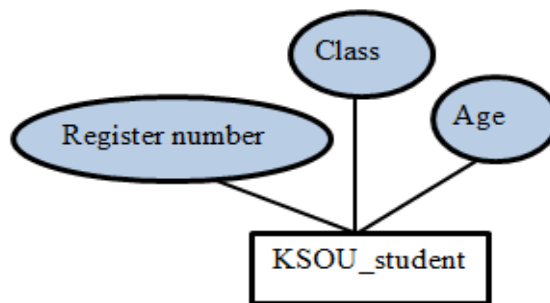


Fig 3.4: An example of simple attribute

Composite attribute: Attributes that can be further subdivided and are composed of many other simple attributes each of which has a distinct meaning are called composite attributes. The student address, for instance, is a composite attribute in the KSOU_student entity since an address is made up of other properties like the pin code, state, and country.

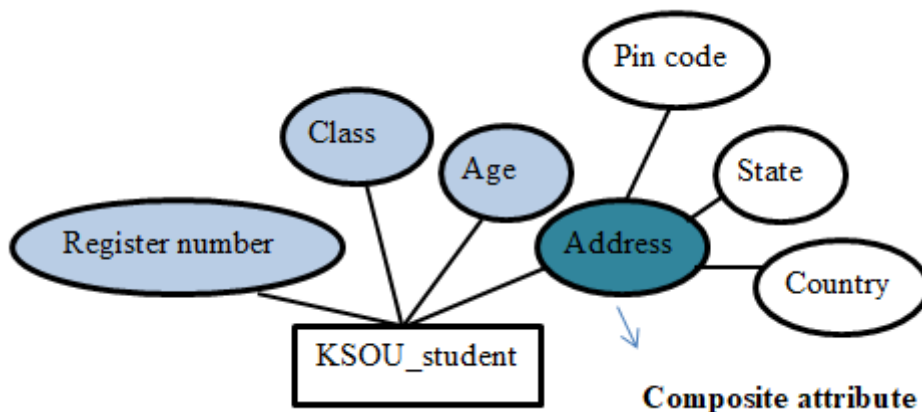


Fig 3.5: An example of composite attribute

Another example is KSOU_student entity's NAME attribute which can be further divided into First name, Last name, and Middle name.

Single valued attribute: Single valued attributes consist of a single value for each entity instance and can't store more than one value. For example, Age is a single-valued attribute of a KSOU_student entity.

Multivalued attributes: Multivalued attributes are those that have more than one value for a certain entity. For this attribute, distinct entities may have varying numbers of values. We must additionally specify the minimum and maximum number of values that can be attached for multivalued attributes. For example, the attributes "Mobile_no" and "Email_id" are multivalued attributes since they accept multiple values for a given entity.

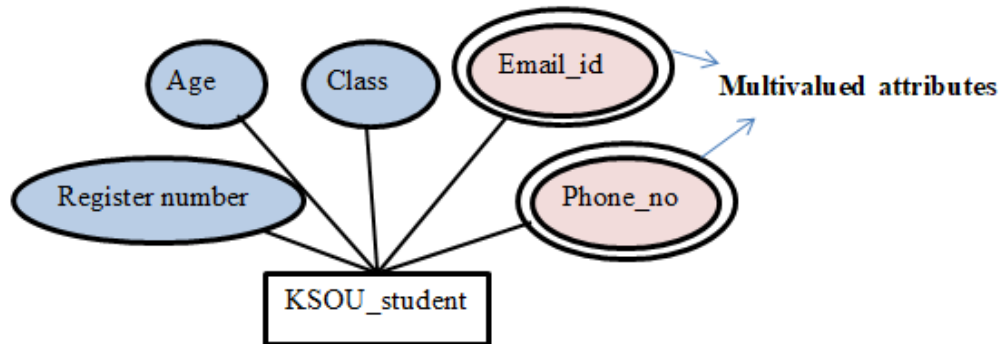


Fig 3.6: An example of multivalued attribute

Stored attribute: Attribute that is physically stored in the database and cannot be derived from other attributes is called stored attributes. Consider KSOU_ student entity with attributes register no., name and birth date. These attributes' value cannot be derived from other attributes. These attributes are therefore referred to as stored attributes.

Derived attribute: An attribute whose values are derived from other attributes is known as a derived attribute. If we have an attributes named date of birth and age in a student database. With the help of the date of birth attribute, we can get the value for age attribute.

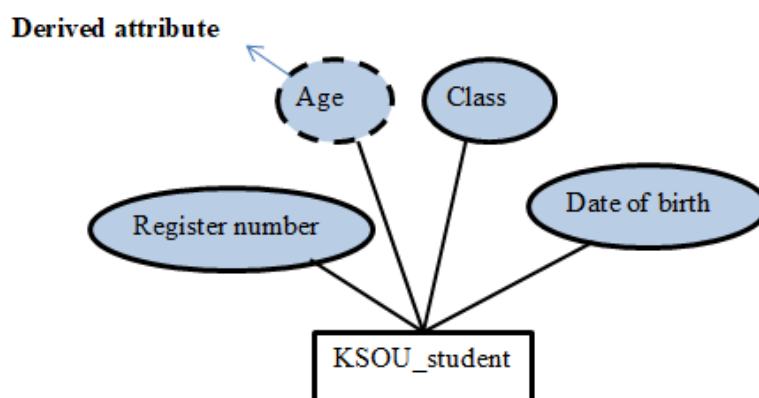


Fig 3.7: An example of derived attribute

Null attribute: When an entity does not have a value for an attribute, it will take NULL as its value. The Null property is one whose value is unknown, unassigned, and missing details.

Key attribute: This attribute is referred as key attribute of an entity since it has a unique value for an entity and is used to identify a specific row in the table. For example : In KSOU_student entity , register no. is an key attribute which has an unique value which is used to identifies given row in the table.

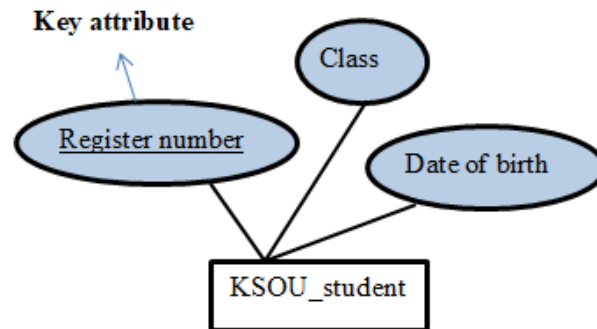


Fig 3.8: An example of key attribute

3.7 RELATIONSHIPS

A relationship is an association between entities. For example, a relationship exists between students and teachers that can be described as follows: teacher can handle many students. Here handle is considered as relationship between teacher and student entity.

Relationship sets

A set of relationships of the same type are referred to as a relationship set. For instance, think about the association between the two entity sets course and student. Collection of all the instances of relationship opts forms a relationship set called relationship type.

Degree

The number of participating entity types determines the degree of a relationship type.

- Binary relationships are those that exist between two entities.
- Ternary relationships are those between three entities.
- The term "n-ry relationship" refers to a relationship between n entities.

3.7.1 ROLES IN E-R DIAGRAMS

- The function that an entity plays in a relationship is called its role.
- Roles are normally explicit and not specified.
- They are helpful when a relationship set's meaning has to be clarified.

- Example, The relationship works-for define in employees as manager or worker. The labels “manager” and “worker” are called roles; they specify how employee entities interact via the works-for relationship set.
- In E-R diagrams, roles are denoted by labeling the lines joining diamonds and rectangles.
- Role labels are used to define the semantics of the relationship and are optional.

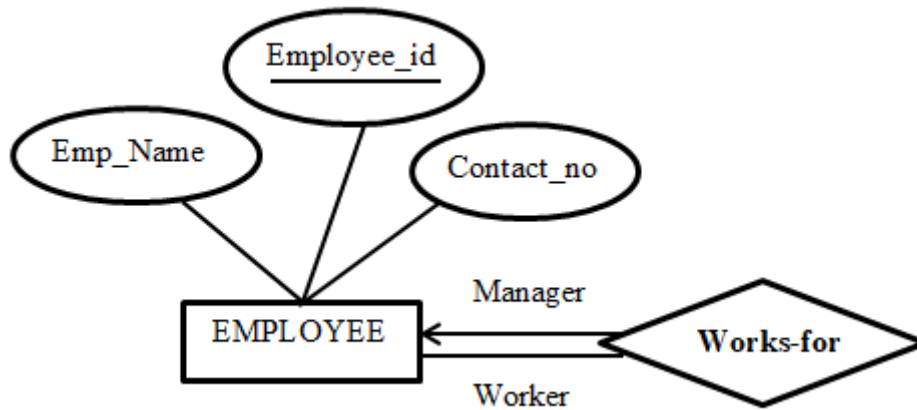


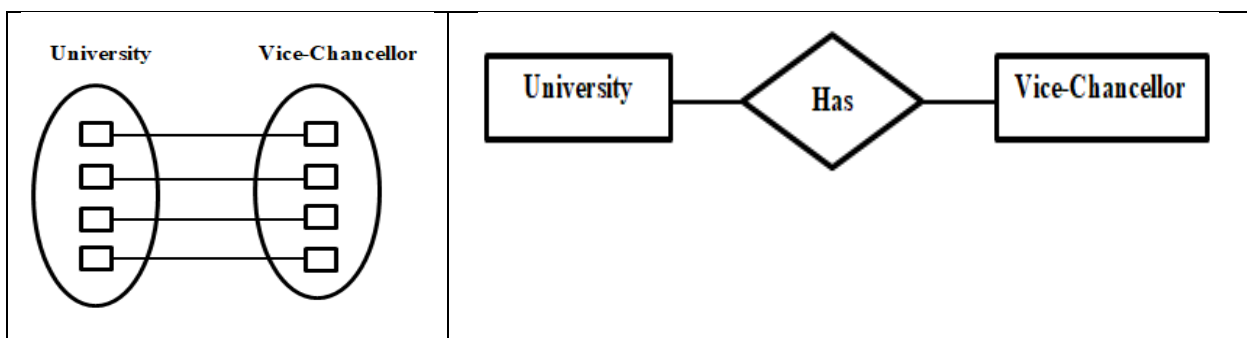
Fig 3.9: E-R diagram with role indicators

3.7.2 RELATIONSHIP CARDINALITY

Cardinality is a mathematical term that refers to the number of elements in a given set. “Cardinality specifies the number of instances of an entity associated with another entity participating in a relationship”. **Binary relationships can be further divided into the following groups based on cardinality:**

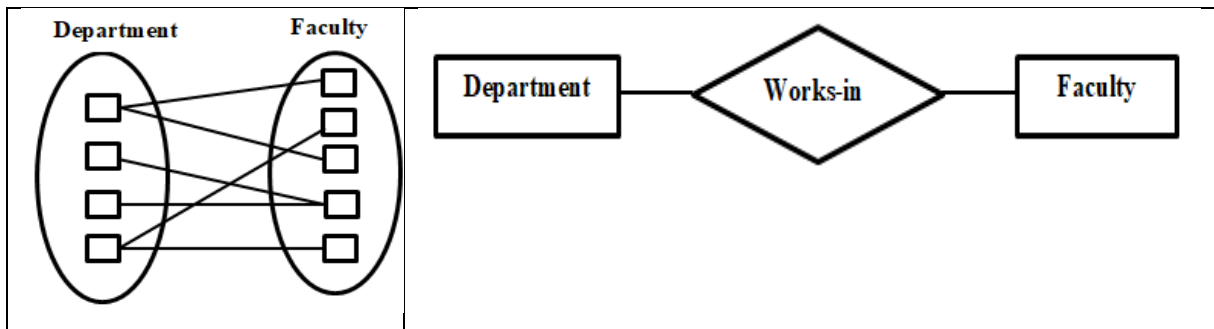
One-to-one: One to one cardinality refers to the relationship between a single instance of one entity and a single instance of another. In this case, each member of the entity set only takes part in the relationship once.

For example: Relationship between university and Vice Chancellor. One University can have at the most one Vice Chancellor and one Vice Chancellor can be assigned to only one University.



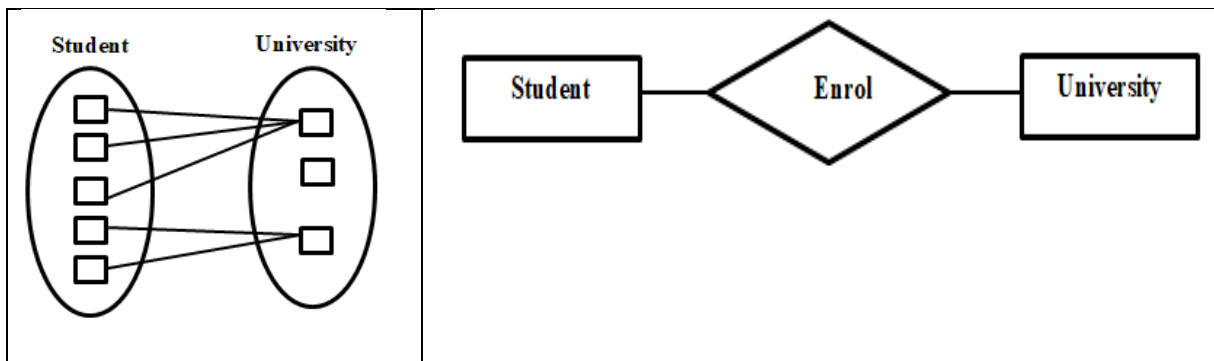
One-to-many: One-to-many relationships are those in which a single instance of one entity is linked to multiple instances of another entity. Here entities in one entity set can take participation in any number of times in relationships set and entities in another entity set can take participation only once in a relationship set.

For example: In the university, a faculty member is only allocated to one department, while a department may appoint any number of faculty members.



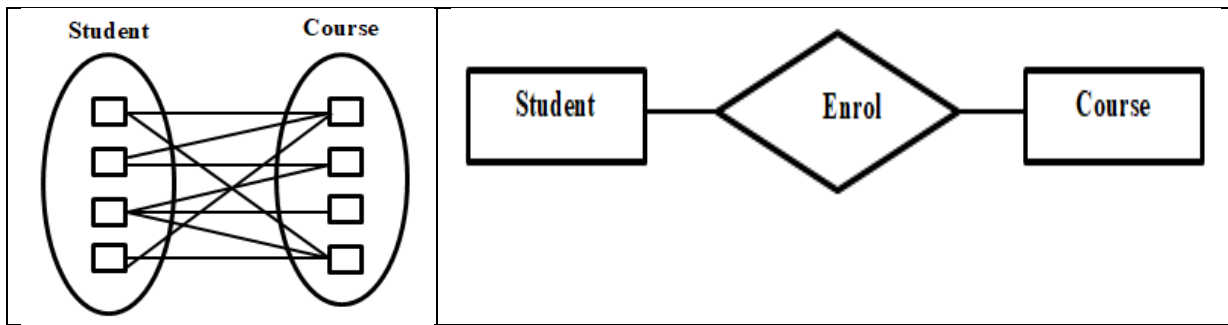
Many-to-one: Many-to-one relationships are those in which entities in one entity set can participate in a relationship set only once, whereas entities in another entity can participate multiple times.

For example: A relationship between student and university, a student can only enrol in one university at a time while a university might have many students.



Many-to-many: Many-to-many relationships are those in which there are several instances of one entity associated to multiple instances of another entity. In this cardinality, entities from all entity sets are able to participate in the many-to-many relationship any number of times.

For example: A relationship between student and course. Each course is for several students, and each student may take more than one course.



3.8 PARTICIPATION CONSTRAINTS

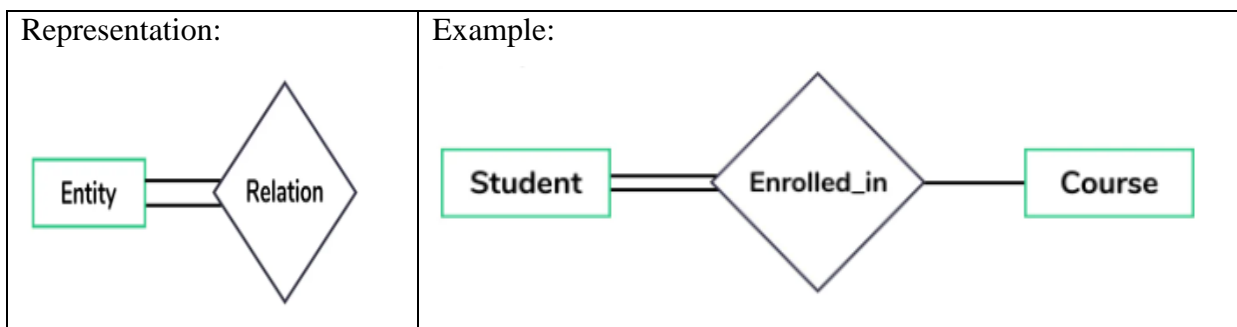
Participation constraints in relationships define the presence of an entity when it is related to another entity in a relationship type. The number of instances of an entity that are included in the relationship type is specified by this constraint.

Two categories of participation constraints exist, they are:

- Total participation
- Partial participation

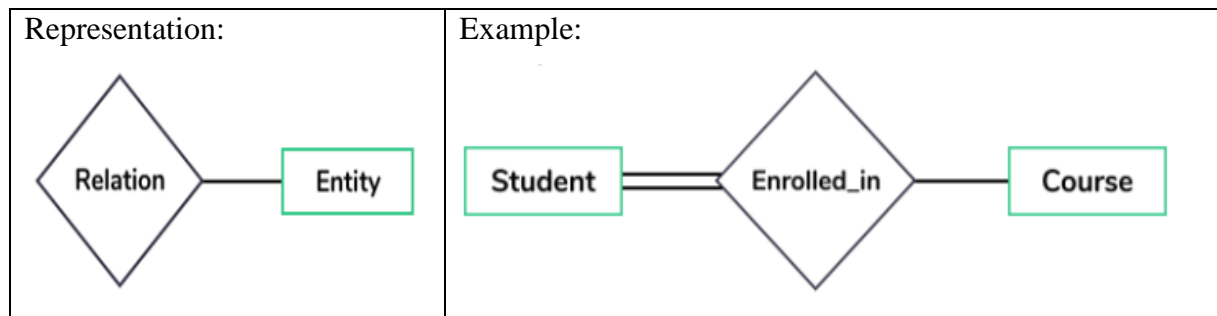
TOTAL PARTICIPATION

- It indicates that each entity present in the entity set must be required to participate in at least one relationship instance of that relationship set that is why it is also known as mandatory participation.
- A double line is used to depict total participation between the entity set and relationship set.
- **Example:** Double line between the entity set “Student” and relationship set “Enrolled in” denotes total participation. It specifies that each student must be enrolled in at least one course.



PARTIAL PARTICIPATION

- “It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set”.
- A single line is used to depict partial participation between the entity set and relationship set.
- **Example:** Single line between the entity set “Course” and relationship set “Enrolled in” denotes partial participation. It states that there may be some courses for which there are no enrollments.



3.9 CONSTRUCTING AN ER MODEL

It is important to thoroughly examine the requirement specifications before starting to draw the ER model. There are a few basic steps to take to draw an ER diagram.

Identify entities: List every possible entity type. These are the object of interest in the system. It is preferable to include too many entities at this stage and remove them later if necessary.

Remove duplicate entities: Make sure they truly distinguish across entity types and don't just two names for the same thing.

Also, the system shouldn't be considered an entity type. For example, when modelling a library Books, borrowers, and other entity kinds might be used. As the library is the system, it shouldn't be considered an entity.

List the attribute of each entity: Make sure the entity types are actually required. Are any of them merely attributes of a different entity type? If so keep them as attributes and remove them from the entity list accordingly. Don't use attributes of one entity as attributes of another.

Mark the primary keys: These are the attributes which uniquely identify instances of the entity type. For, some weak entities, this might not be possible.

Define the relationships: Analyze each entity type to determine its relationship to others.

Describe the cardinality and optionality of the relationships: Examine the constraints between participating entities.

Remove redundant relationships: Analyze the ER model for redundant relationships.

As discussed earlier, business processes and database architectures both use ER diagrams. Now that we have a basic idea of what ER diagrams are, Let us discuss a few scenarios where we can draw ER diagrams.

Example 1: The Student and University entities, as well as their relationships are considered and are shown in the diagram below. A university may have many students, but a student cannot get admitted to more than one university at once (assumption), hence the relationship between the two entities is many to one. University entity includes attributes like Uni_ID and Uni_Name, while Student entity has attributes like Stu_Id, Stu_Name and Stu_Addr.

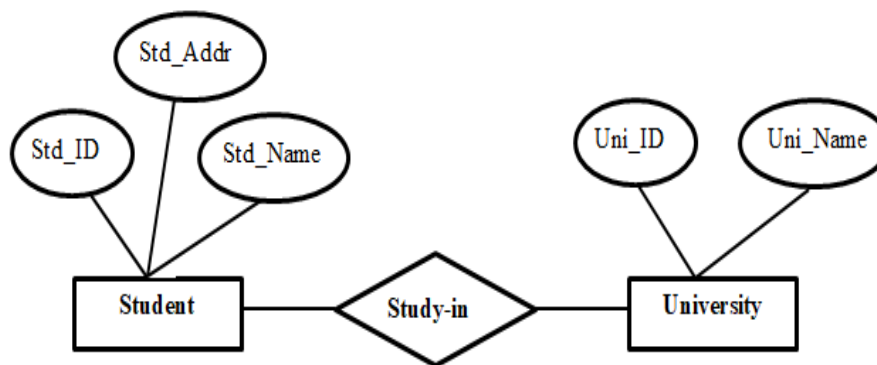


Fig 3.10: E-R diagram for student admission

Example 2: This ER diagram was developed according to the needs of loan management, for clients and loans. The technology has the ability to encrypt the debtors' data during the transaction. The administrator can view the transactions and status of the debtors. They can handle the data required for information management and record the customer's request. The ER diagram for clients and loans includes features for security and monitoring of lending/bank information as well as the status of debtors' information. The system's transaction history was documented in reports that had these aspects. As a result, the system is capable of managing and protecting the transactions and personal data of the debtors.

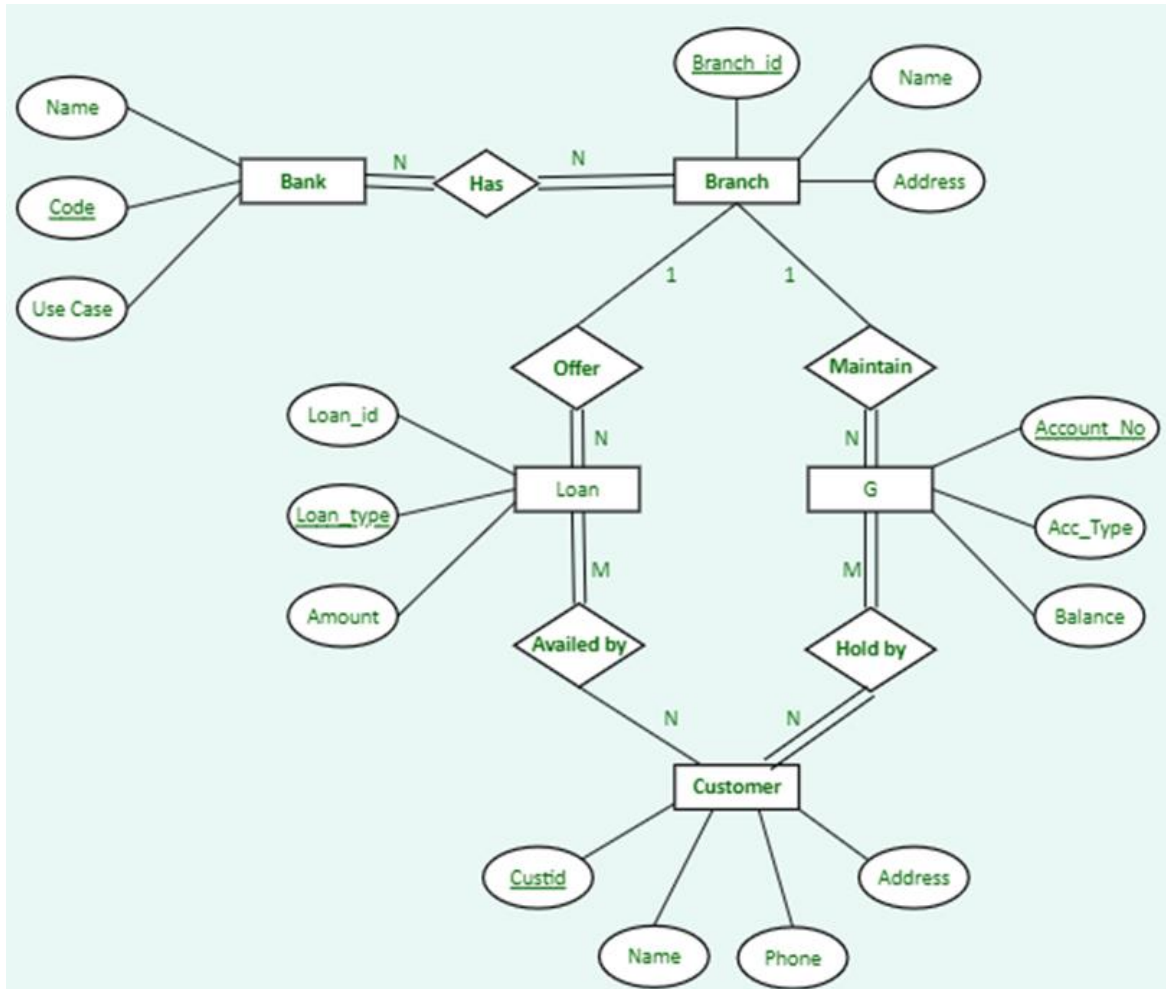


Fig 3.11: E-R diagram for customer and loan

Example 2: Consider a university database that would be used to schedule final exam rooms. This database could be modeled as the single entity set exam, with attributes course-name, section-number, room-number, and time. As an alternative, relationship sets and one or more additional entity sets could be created to replace some of the exam entity set's attributes, as

- course with the following attributes: name, department, and c-number.
- section with s-number and enrollment attributes as well as dependent as a weak entity set on course.
- room with the following attributes: building, capacity, and r-number.

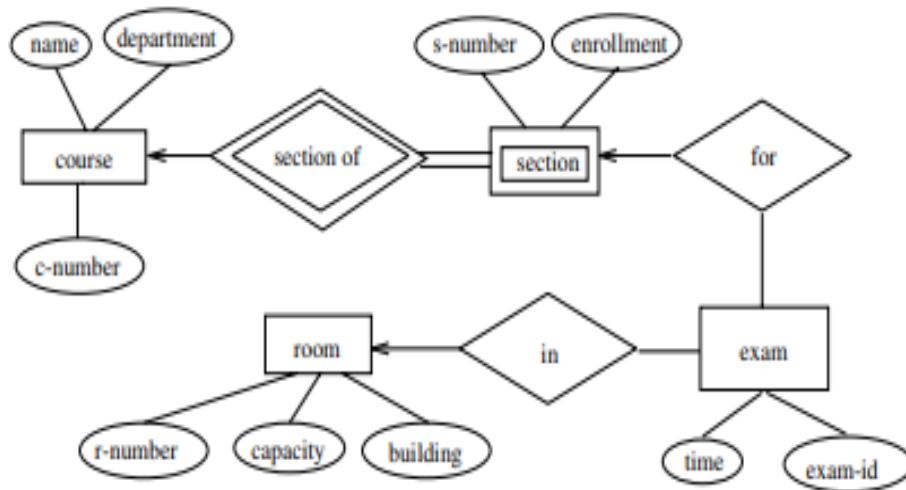


Fig 3.12: E-R diagram for exam scheduling

3.10 CHECK YOUR PROGRESS

1. ER model describes data as a collection of entities, relationships and attributes. State true/false.
2. Which of the following gives a logical structure of the database graphically?
 - a) Entity-Relationship diagram
 - b) Entity diagram
 - c) Database diagram
 - d) Data diagram
3. An entity set that does not have sufficient attributes to form a primary key is termed a _____.
4. For a weak entity set to be meaningful, it must be associated with another entity set, called the _____.
5. Multivalued attributes are those that have more than one value for a certain entity. State true/false.

Answers to check your progress

1. True
2. a) Entity-Relationship diagram.
3. Weak entity set.
4. Identifying set.
5. True.

3.11 SUMMARY

In this unit, we have discussed about basic aspects of E-R modeling. Data modeling in software engineering is the process of creating a data model by applying formal data model descriptions using data modelling techniques. This unit introduced data models and some related terminologies. The data requirements are recorded as a conceptual data model with associated data definitions. We also defined relationships, roles and structural constraints in this unit. E-R modeling is quite common to database design; therefore, we must attempt as many problems as possible from the further reading.

3.12 KEYWORDS

- **Data model:** A data model is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
- **Entity types:** An entity type defines a set of entities that have same attributes. A name and a list of attributes describe each entity type.
- **Entity set:** An entity set is a set of entities of the same type.
- **Attribute:** Attributes are the descriptive properties owned by each entity of an entity set.
- **Relationship:** Any association between two entities is known as a relationship between those two entities.
- **Identifying relationship:** The relationship that relates the weak entity type to an strong entity type is known as identifying relationship.
- **Cardinality:** It represents the number of times an entity of an entity set participates in a relationship set.

3.13 QUESTION FOR SELF STUDY

1. Why do we need an ER model in DBMS?
2. Define the terms: Entity, Entity set and Entity type. Give an example to each.
3. Discuss the different cardinality constraints with an example to each.
4. Explain various symbols used in ER diagrams.
5. Explain relationship set.
6. Define attribute. Discuss different types of attributes with an example to each.
7. Discuss the various E-R diagram conventions with a suitable example.
8. Differentiate strong and weak entity type.

9. Write a note on total and partial participation.
10. Construct an ER diagram for an Insurance company that has the set of customers each of whom owns one or more cars each car has associated with any number of accidents regarded.
11. Draw an ER diagram of registration process of the student in a particular course.

3.14 REFERENCES

1. Silberschatz Abraham, Henry F. Korth, and Shashank Sudarshan. Database system concepts. New York: McGraw-Hill, sixth edition, 2011.
2. Raghu Ramakrishnan, and Gehrke Johannes. "Database management systems." (2022).
3. Coronel, Carlos, and Steven Morris. *Database systems: design, implementation, & management*. Cengage Learning, 13th edition, 2016.
4. Leon, Alexis, and Mathews Leon. *SQL: A Complete Reference*. TaTa McGraw-Hill, 1999.
5. Gillenson, Mark L. *Fundamentals of database management systems*. John Wiley & Sons, 2008.

UNIT-4: EXTENDED ER MODEL

STRUCTURE

- 4.0 Objectives
- 4.1 Introduction
- 4.2 The Enhanced Entity Relationship model
 - 4.2.1 Features of EER Model
- 4.3 Subclasses and super classes.
- 4.4 Specialization and generalization
- 4.5 Category or union
- 4.6 Aggregation
- 4.7 Use of extended ER features
- 4.8 Translation of ER schema into tables
 - 4.8.1 Rules to transform ER diagram into tables
- 4.9 Check your progress
- 4.10 Summary
- 4.11 Key words
- 4.12 Questions for self-study
- 4.13 References

4.0 OBJECTIVES

After studying this unit, you will be able to:

- Describe Enhanced Entity-Relationship model features
- Define inheritance.
- Differentiate specialization and generalization.
- Specify reasons to use specialization.
- Discuss the need of category or union
- Explain why to use aggregation in DBMS
- Describe the use of extended ER features
- Translate an ER diagram to table.

4.1 INTRODUCTION

In this unit, we are going to discuss about Extended Entity-Relationship (EER) model concepts. We describe features that have been proposed for semantic data models and show how the ER model can be enhanced to include these concepts, which leads to the enhanced ER (EER) model. These concepts are subclasses and super classes, specialization, generalization, union and aggregation. This unit also provides the use of extended ER features and the rules to translate an ER diagram to table.

4.2 THE ENHANCE ENTITY RELATIONSHIP MODEL

Traditional database applications, which include many data-processing applications in business and industry, can be represented by a wide variety of database schemas using the ER modelling concepts discussed in unit 3. The entity-relationship (ER) model is a visual representation of the relationships between different entities of a domain. In the case of a large amount of data with multiple interrelated entities, the ER diagrams become complex and tough to interpret. Thus, ER model was enhanced to include the complex concepts and interpretations which lead to Enhanced Entity Relationship (EER) OR Extended Entity Relationship (EER) model.

In addition to ER model concepts discussed in unit 3 EER includes the below mentioned features –

- Subclasses and Super classes.
- Specialization and Generalization.
- Category or union type.
- Aggregation.

4.2.1 Features of EER Model

- EER creates a design that is more accurate to database schemas.
- It more accurately reflects the limitations and data properties.
- It incorporates all modelling concepts of ER model.
- The EER schema can be displayed more effectively using diagrammatic techniques.
- It includes specialization and generalization concepts.
- It is utilized to represent a collection of objects that is union of objects of different of different entity types.

4.3 SUBCLASSES AND SUPER CLASSES

As it has been discussed in unit 3, Section 3.4, an entity type is a set of entities of the same type that share the same properties or characteristics. Subclasses (or subtypes) and superclasses (or supertypes) are the special type of entities.

- Superclass is a top-level entity that can be classified into subclasses or subsets and it is also known as a parent class or base class.
- A subclass inherits all the attributes of its superclass.
- Sub class and Super class relationship leads the concept of Inheritance.
- Inheritance is a mechanism in which one class acquires the property of another class. The class whose members are inherited is called the base class, and the class that inherits those members is called the derived class.
- The relationship between a superclass and a subclass is called superclass/subclass or class/subclass relationship and it is indicated with \textcircled{d} symbol

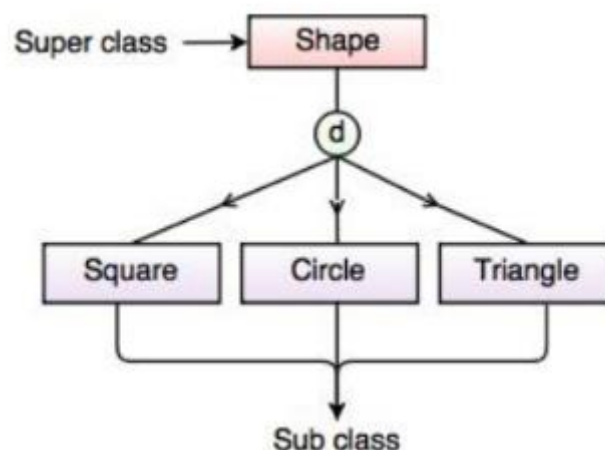


Fig 4.1: superclass/subclass relationship

For example: Shape super class is having sub groups as Square, Circle, Triangle. These Sub classes inherit properties and attributes from its super class.

4.4 SPECIALIZATION AND GENERALIZATION

The main principles of database modelling are specialization and generalization. Specialization is based on a refinement of classes to more specific ones. Generalization groups classes to more abstract or combined ones. In the following sub-sections let's understand generalization and specialization in detail.

4.4.1 SPECIALIZATION

Specialization is a top-to-bottom approach to designing the Enhanced ER model. The parent entity, or superclass, is defined first in this model. The super class is further categorized into one or more subclass and also forms the superclass/subclass relationship. EER diagrams using this data model contain expanded form.

The goal of specialization is to identify subsets of entities with few distinctive attributes. Now let's understand the two main reasons for including specializations in DBMS.

Reasons for Specialization

- Certain attributes may apply to some but not all entities of a super class. A subclass is defined in order to group the entities to which the attributes apply.
- The second reason for using subclasses is that some relationship types may be participated in only by entities that are members of the subclass.

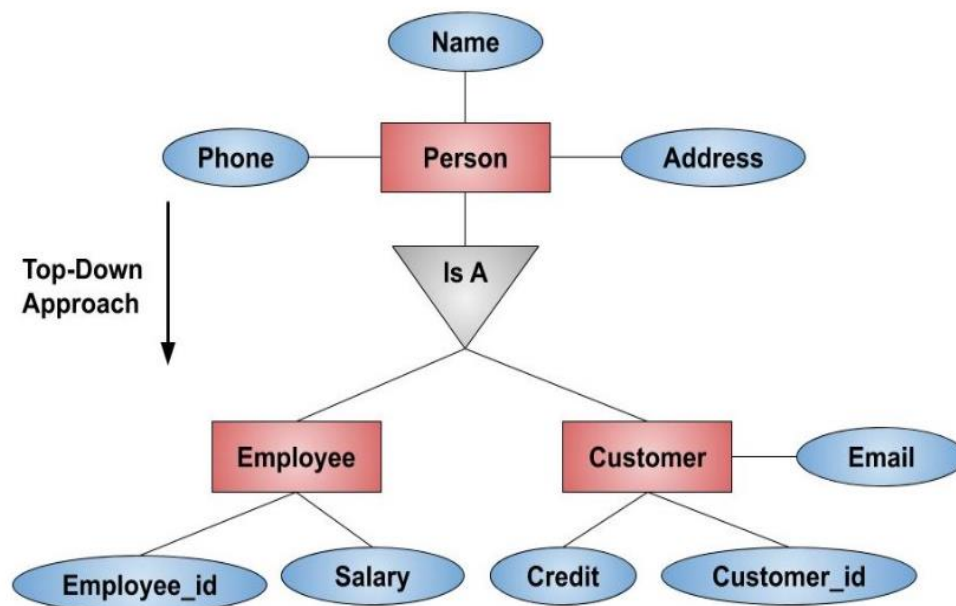


Fig 4.2: Specialization

For example, in the Fig. 4.2, we can see that person entity type with attributes like Name, Phone no, and Address. Now, if we are developing software for a store, there are two possible types of people. So, the Person entity is further subdivided into the Employee and Customer entities. How will it be specialized? By adding characteristics like Emp_no and Salary, we will specialize the Person entity type into the Employee entity type. Also, by including attributes like Customer_no, Email, and Credit, we can specialize the Person entity type to

the Customer entity type. The entire higher-level entity attribute's properties will be inherited by these lower-level entity attributes. The Name, Phone_no, and Address properties that were present in the higher-level entity will likewise be present in the Customer entity type.

4.4.2 GENERALIZATION

As the name implies, generalization is the process of combining two or more lower-level entity types into a single higher-level entity type. To represent a more generalized view, entities are combined or joined together. Through this method, a new entity type is created by combining the common attributes of two or more entities. The new entity type formed is called a generalized entity. This generalized entity could potentially merge with other entity types to create a new, higher-level entity type. It is a bottom-up approach and the reverse process of Specialization.

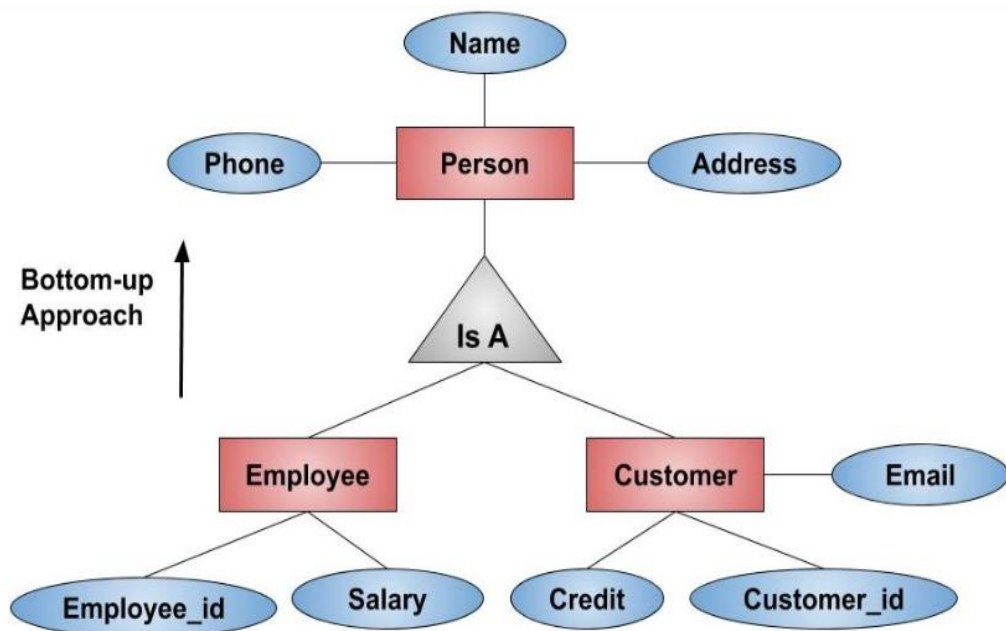


Fig 4.3: Generalization

For example: Consider that we have the entity types Customer and Employee. Name, Phone, Salary, Employee_id, and Address are the attributes of the Employee entity type. Name, Phone, Address, Credit, Customer_id, and Email are the attributes of the Customer entity type. As we can see, the three attributes—Name, Phone, and Address are common here. These two entities can be combined to create a new higher-level entity type called Person. A generalized entity is the new entity type that was created. As seen in the E-R diagram represented in the Fig.4.3, the new generalized entity Person only has the common attributes Name, Phone, and Address after generalization. Only specialized attributes like Employee_id

and Salary are included in the Employee entity. Similar to this, only specific attributes like Customer_id, Credit, and Email are present in the Customer entity type. As a result, it is very clear from this example that going from bottom to top is a Generalization and going from top to bottom is a Specialization. As a result, generalization might be viewed as the opposite of specialization.

4.5 CATEGORY OR UNION

It is sometimes necessary to represent a collection of entities from different entity types. In this case, a single super-class or sub-class relationship with numerous super-classes is represented by a category or union. Participation can be total or partial. In simpler terms, it represents an “either” type of relationship.

For instance, under the car booking model, the car owner may be a person, a bank, or a business. The three Super-classes Company, Bank, and Person are combined to form the category-subclass known as Owner. A member of a Category must exist in at least one of its super-classes.

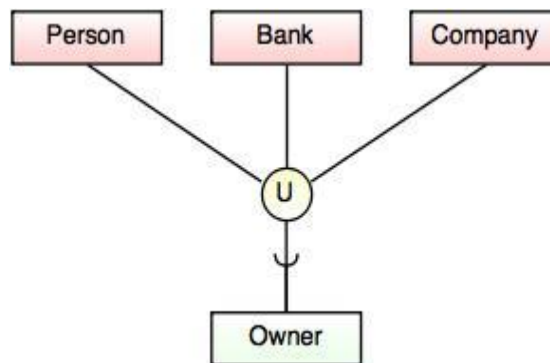


Fig 4.4: Categories (Union Type)

4.6 AGGREGATION

Aggregation is a process of combining two or more entities that cannot be used alone, to form a more meaningful entity. When a single entity in a relationship doesn't make sense on its own, then the aggregation process is done. As a result of this, the relationship between the two entities functions as one entity.

Let's consider a well-known example to understand aggregation in DBMS. The employee, a project, and a manager are the entities represented in the entity relationship diagram below (Fig 4.5). Additionally, there are two relationship "Builds" and "Manages" in the ER diagram.

Take into account a real-world instance where a firm employee works on a certain project (or builds certain project). Now since there may be many employees in the organization working on different projects, we must manually search through the files to go back to a project or find a reference to a project and its builder. But that's not how life really is. This highlights the requirement for an upper body that manages the project and the employee who is a new entity—"Manager." The manager manages the employee and the project and the data associated to the project with the employee is saved by the manager according to their preference. The term for this is aggregation. The Employee and the Project are linked to the "builds" operation in this diagram. The "manages" operation, which stands for the aggregation function, is linked to the "builds" operation. The new entity known as the Manager entity is then linked to the "manages" operation. By using the aggregation, this ER diagram appears to be more meaningful.

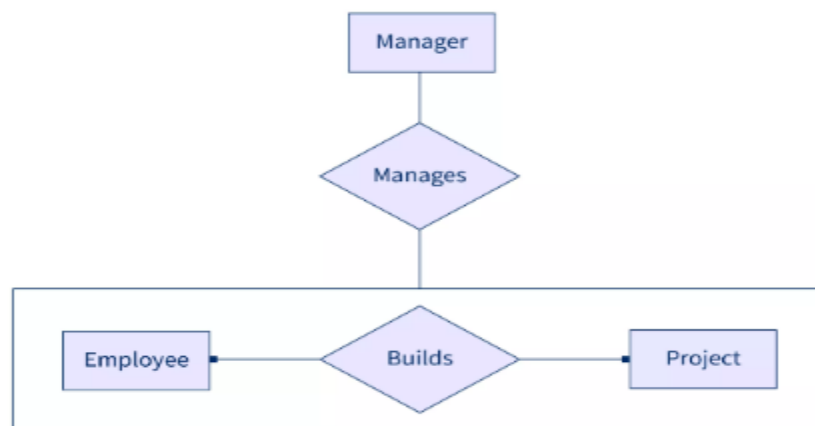


Fig 4.5: Aggregation

4.7 USE OF EXTENDED E-R FEATURES

We have seen weak entity sets, generalization and aggregation. Designers must decide when these features are appropriate.

- Strong entity sets and their dependent weak entity sets may be regarded as a single "object" in the database, as weak entities are existence-dependent on a strong entity.
- It is possible to treat an aggregated entity set as a single unit without concern for its inner structure details.
- Generalization contributes to modularity by allowing common attributes of similar entity sets to be represented in one place in an E-R diagram.

Excessive use of the features can introduce unnecessary complexity into the design.

4.8 TRANSLATION OF ER SCHEMA INTO TABLES

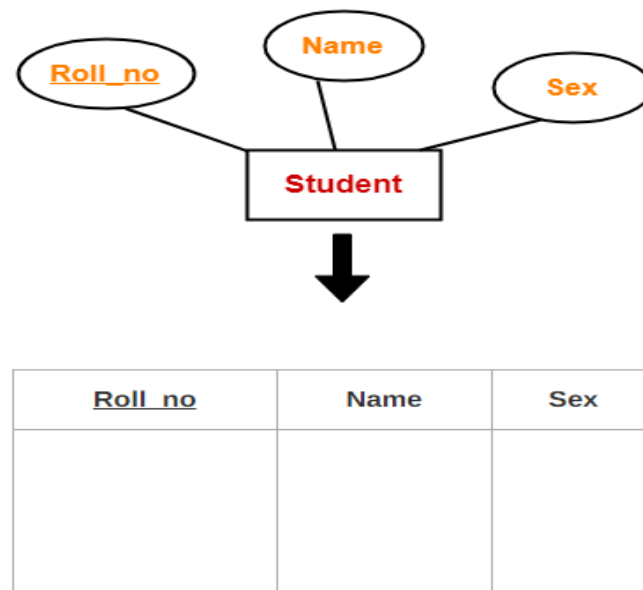
- An ER schema can be represented by a collection of tables which represent contents of the database (instance).
- Primary keys allow entity types and relationship types to be expressed uniformly as tables.
- For each entity and relationship type, a unique table can be derived which is assigned the name of the corresponding entity or relationship type.
- Each table has a number of columns that correspond to the attributes and which have unique names. An attribute of a table has the same domain as the attribute in the ER schema.
- Translating an ER schema into a collection of tables is the basis for deriving a relational database schema from an ER diagram.

4.8.1 RULES TO TRANSFORM ER DIAGRAM INTO TABLES

An ER diagram is transformed into tables according to the following rules:

Rule-01: For Strong Entity Set With Only Simple Attributes-

- In a relational model, a strong entity set with only simple attributes will require only one table. The entity set's attributes will be the attributes of the table.
- The entity set's key attribute will be the table's primary key.

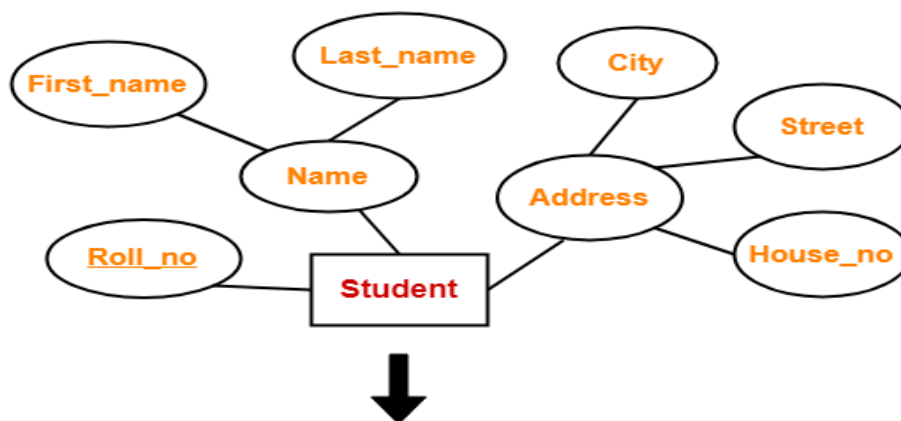


Schema : Student (Roll_no , Name , Sex)

Fig 4.6: Example for Rule-01

Rule-02: For Strong Entity Set With Composite Attributes-

- In a relational model, a strong entity set with any number of composite attributes only needs one table.
- While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.



<u>Roll_no</u>	First_name	Last_name	House_no	Street	City

Schema : Student (Roll_no , First_name , Last_name , House_no , Street , City)

Fig 4.7: Example for Rule-02

Rule-03: For Strong Entity Set With Multi Valued Attributes-

- Two tables will be needed in a relational model for a strong entity set with any number of multivalued attributes.
- All of the simple attributes will be contained in a single table with a primary key.
- The primary key and all the multi-valued attributes will be contained in another table.

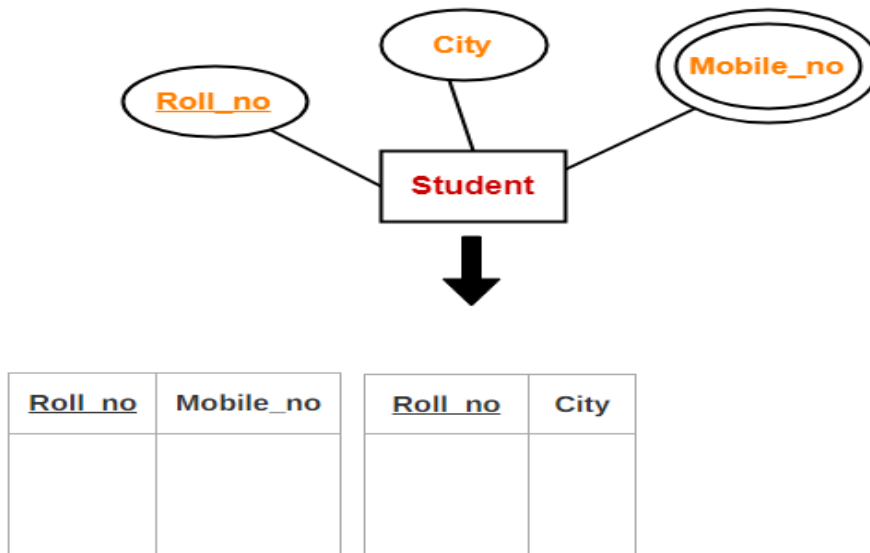


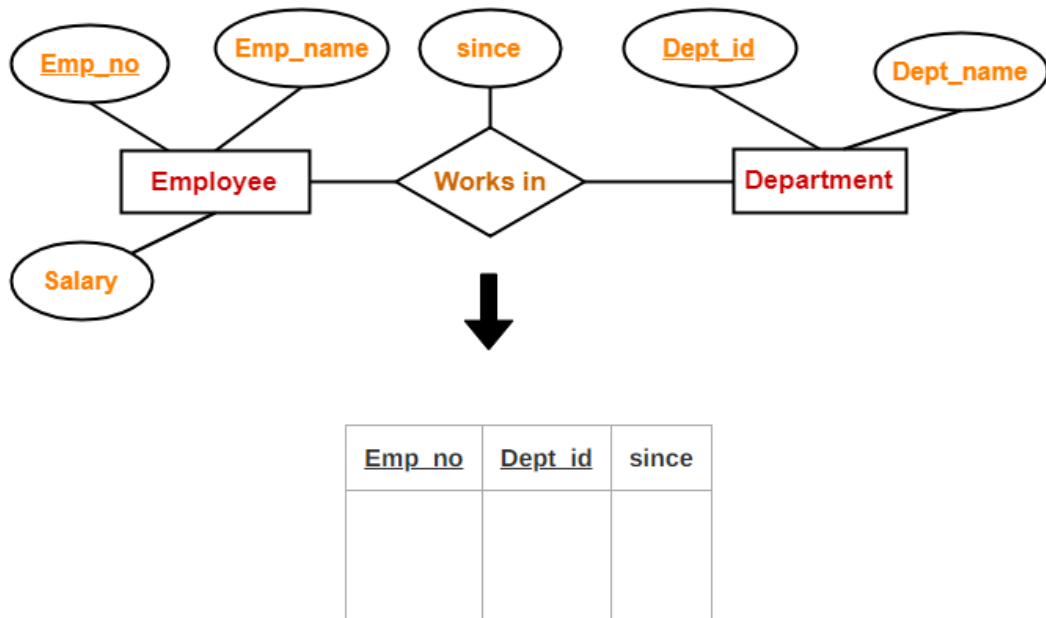
Fig 4.8: Example for Rule-03

Rule-04: Translating Relationship Set into a Table-

In the relational model, a relationship set necessitates a single table. These are the table's attributes:

- The participating entity sets' primary key attributes.
- Its own descriptive attributes if any.

The primary key will be a set of non-descriptive attributes.



Schema : Works in (Emp_no , Dept_id , since)

Fig 4.9: Example for Rule-04

Note: According to the relational model's overall ER diagram, three tables will be needed:

- "Employee" entity set's single table
- The entity set "Department" has just one table.
- For the relationship set "Works in," one table

Rule-05: For Binary Relationships with Cardinality Ratios-

The following four cases are possible-

- Case-01: Binary relationship with cardinality ratio m:n
- Case-02: Binary relationship with cardinality ratio 1:n
- Case-03: Binary relationship with cardinality ratio m:1
- Case-04: Binary relationship with cardinality ratio 1:1

• **Case-01: For Binary Relationship with Cardinality Ratio m:n**

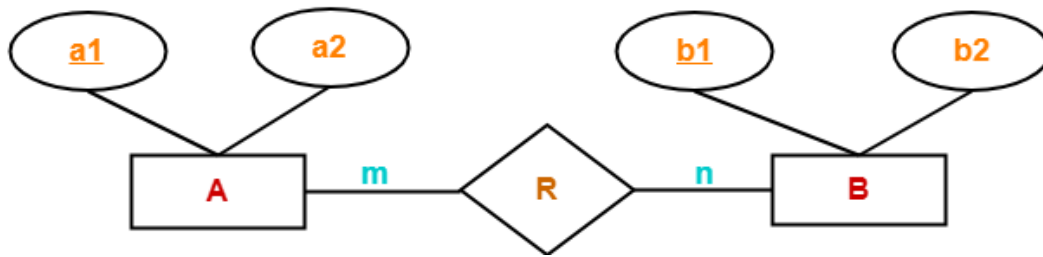


Fig 4.10: Example for Case-01

Three tables will be needed here:

1. A (a1 , a2)
2. R (a1 , b1)
3. B (b1 , b2)

• **Case-02: For Binary Relationship with Cardinality Ratio 1:n**

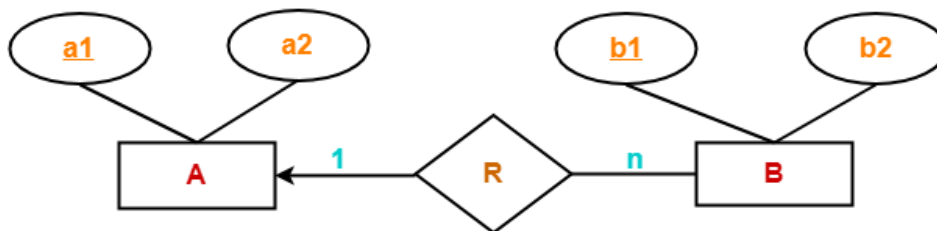


Fig 4.11: Example for Case-02

Two tables will be needed here:

1. A (a1 , a2)
2. BR (a1 , b1 , b2)

Note- Here, combined table will be drawn for the entity set B and relationship set R.

- **Case-03: For Binary Relationship with Cardinality Ratio m:1**

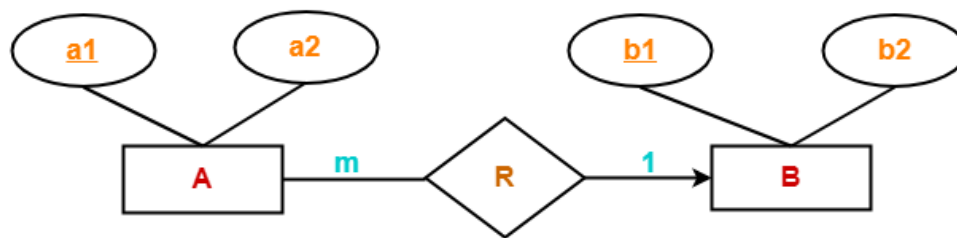


Fig 4.12: Example for Case-03

Two tables will be needed here:

1. AR (a1 , a2 , b1)
2. B (b1 , b2)

Here, combined table will be drawn for the entity set A and relationship set R.

- **Case-04: For Binary Relationship with Cardinality Ratio 1:1**

Here, two tables will be required. Either combine 'R' with 'A' or 'B'

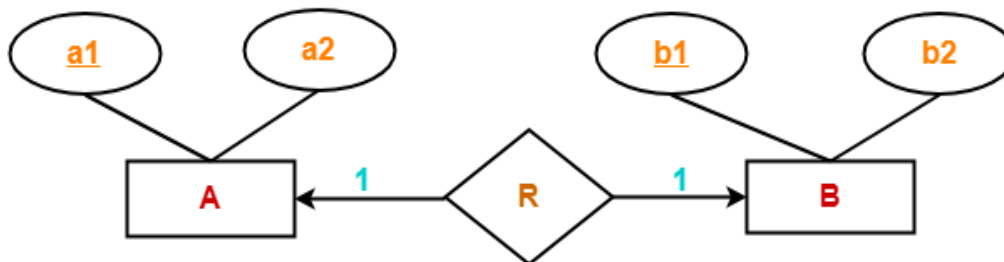


Fig 4.13: Example for Case-04

Way-01:

1. AR (a1 , a2 , b1)
2. B (b1 , b2)

Way-02:

1. A (a1 , a2)
2. BR (a1 , b1 , b2)

Rule-06: For Binary Relationship with Both Cardinality Constraints and Participation Constraints-

- Constraints on cardinality will be applied as described in Rule-05.
- Foreign key now acquires a NOT NULL constraint, meaning that it cannot be null because of the entire participation constraint.

Rule-07: For Binary Relationship with Weak Entity Set-

Always, an identifying relationship with a total participation requirement is associated with a weak entity set.

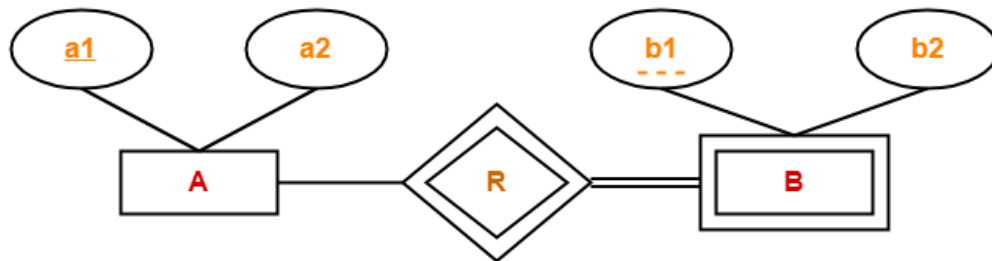


Fig 4.14: Example for Rule-07

Two tables will be needed here:

1. A (a1 , a2)
2. BR (a1 , b1 , b2)

4.9 CHECK YOUR PROGRESS

1. The generalization and specialization lattice are considered as :
 - a) Single inheritance
 - b) Multiple inheritance
 - c) Single aggregation
 - d) Multiple aggregation
2. The generalization process is similar to the bottom-up approach. State True/False.
3. _____ are defined first, followed by subclasses and their associated attributes, followed by the relationship set.
 - a. Subclasses
 - b. Super-classes
 - c. Classes
 - d. Fields
4. When entities are aggregated, their relationships with each other are combined into ____.
5. Higher level entity sets are designated by the term Super class. State True/False.

Answers to check your progress

1. b) Multiple inheritance.
2. True
3. b) Super-classes

4. Higher level entity.
5. True

4.10 SUMMARY

In this unit, we have discussed about basic aspects of Extended E-R modelling. Extended ER models help in modeling the enterprise more accurately and hence able to represent the information clearly. The concepts such as class/subclass relationships, type inheritance, specialization and generalization and how the union construct can be modelled were discussed in this unit. We have also discussed about the use of extended ER features And the various rules to translate an ER diagram to table.

4.11 KEYWORDS

- **Inheritance:** Inheritance is an important feature of Generalization and Specialization. It allows lower-level entities to inherit the attributes of higher-level entities.
- **Superclass:** The class from which the subclass is derived is called a superclass.
- **Subclass:** A class that is derived from another class is called a subclass.
- **Generalization:** It is a process in which a new entity is formed using the common attributes of two or more entities.
- **Specialization:** It is a top-down approach in which a higher-level entity is divided into multiple specialized lower-level entities.

4.12 QUESTIONS FOR SELF STUDY

1. What is the use of EER diagram?
2. Define subclass and superclass.
3. Explain the features of EER model.
4. Explain the concept of generalization and aggregation in ER diagrams. Give one example for each one of them.
5. Differentiate generalization and aggregation in DBMS.
6. Illustrate how aggregation and composition are represented in ER diagrams.
7. Discuss the Use of Extended E-R Features
8. Explain the rules to be followed to transform ER diagrams into tables.
9. Construct an ER diagram for a bank which has several branches at different places. Each branch maintains the account details of the customers. The customers may open joint as

well as single accounts. The bank also provides the loan to the customer for different purposes. Bank keeps record of each transaction by customer to this account. All of the branches have employees and some employees are managers.

4.13 REFERENCES

1. Elmasri, R., Sham Navathe., 2021. Fundamentals of database systems seventh edition, Pearson/Addison Wesley.
2. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, Database Systems, The Complete Book, Prentice Hall, 2002.
3. <https://www.gatevidyalay.com/er-diagrams-to-tables/>