ज्ञानगंगा घरोघरी

Yashwantrao Chavan Maharashtra Open University

P34

Diploma in Computer applications

CMP374 Introduction to RDBMS

Yashwantrao Chavan Maharashtra Open University

Vice-Chancellor: Dr. R. Krishnakumar

SCHOOL OF COMPUTER SCIENCE : SCHOOL COUNCIL

Dr. Ramchandra Tiwari Director School of Computer Science Y.C.M.Open University Nashik	Shri. Pramod Khandare Assistant Professor School of Computer Science Y.C.M.Open University, Nashik	**Prof. M.S. Karyakate** Associate Professor Computer Department VIIT, Pune
Dr. Manoj Killedar Director, School of Architecture, Science & Technology, Y.C. M. Open University, Nashik	**Shri. Surendra Patole** Assistant Professor School of Commerce & Management Y.C.M. Open University,Nashik	**Prof. M.N. Shelar** H.O.D., Computer Department K.K. Wagh College Pimpalgaon (B)
Dr. R.V. Vadnere Director School of Continuing Education Y.C.M.Open University, Nashik	**Shri. Madhav Palshikar** Associate Professor School of Computer Science Y.C.M. Open University, Nashik	**Shri. Mahesh Paradkar** IBM Pune
Prof. S.S. Sane Head of Department Computer Department K.K.Wagh College of Engineering Nashik	**Dr. Bharati Gawali** Associate Professor Dept. Of Computer Science Dr. Babasaheb Ambedkar Marathwada University, Aurangabad	**Mrs. Shubhangi Desle** Assistant Professor Student Services Division Y.C.M. Open University, Nashik

Course Development Team/Writer/s

Dr. R.S. Tiwari
Director
School of Computer Science
Y.C.M. Open University, Nashik

EDITOR & COURSE CO-ORDINATOR

Dr. Pramod Khandare
Assistant Professor
School of Computer Science
Y.C.M.Open University, Nashik 422 222

Production

Shri. Anand Yadav
Manager
Print Production Centre YCMOU, Nashik-422101

©2013, Yashwantrao Chavan Maharashtra Open University, Nashik 422 222

- **First Published by** : Nov2013 Reprint: Sept 2015, Oct2016, Sept 2017
- **Printed by :** Shri.Narendra shaligram, M/s. Replica Printers, 2, Chitco Centre, Wakilwadi, Nashik-422001
- **Published by:** Dr. Dinesh Bhonde, Registrar, Y.C.M. Open University, Nashik – 422 222.

DB/AG17-121*(CMP374: Introduction to RDBMS)*

INDEX

Unit No. & Name	**Details**	**Counseling Sessions**	**Weightage**
Unit 1 Data files and DBMS	• Introduction- Data, Files • Operations on file • Introduction to Database -Definition of database, Entity, • Attributes, Domain, Instance, Record/Tuple	3	10
Unit 2 Introduction to DBMS	• Definition to DMBS • WHY DMBS • Services provided by DMBS – Transaction Management, Concurrency Control, Recovery Management, Security Management, Language Interface • Applications of DMBS • Differences between File System and DMBS • Drawbacks of File system • Abstraction Levels (Three Levels of Abstraction) • Database Users • DDL and DML • Structure of DMBS • Metadata	3	10
Unit 3 Relational data models and relational algebra	• Introduction to DATA Models • Object-based Logical Models - E-R Model, Object-Oriented Model • Record-based Logical Models - Relational Model, Network Model, Hierarchical Model • Physical Data Models		10
Unit 4 Entity Relationship	• Overview • Modelling • Basic styles of data model • ER Model • Components of ER Model - Entity, Attributes, Entity Set, Domain • Entity Types – weak entity, Strong Entity, Recursive Entity, Composite Entities • Attributes Types – Simple, Composite, Single Valued, Multi Valued, Stored, Derived, Complex, Null Attributes. • Relation • Relationship – Relationship set, Connectivity in relationship • Types of Relationship – Unary, Binary, Ternary • Classifying Relationship – Degree of	3	10

	Relationship, Multiplicity, Existence • Mapping Cardinalities – One to One, One to Many, Many to One, Many to Many • Keys • Keys for Relationship set- Super key, Candidate key, Secondary key, Compound key, Alternate key, Primary key, Foreign key • E-R Diagrams – E-R Modelling Symbols • Cardinality Constraints related to E-R diagrams • Alternative Notations for cardinality limits • Weak Entity sets • Case Studies on E-R diagrams		
Unit 5 Normalizations	• Overview • Relational DB design • Decomposition (Small schema) • Lossy Decomposition • Loss less Decomposition • Functional Dependency – Full Dependency, Partial Dependency, Transitive Dependency • Normalized Forms – Un – Normalized form, 1NF, 2NF, 3NF • De-normalization	3	10
Unit 6 SQL	• Introduction • SQL Statements - DML, DDL, DCL • Data Types in SQL • Basic Types structure • SELECT- SQL SELECT DISTINCT Statement, SQL Where Clause, And, OR, In, Between, Like Operator, SQL Order by Keyword, Aggregate Functions, Group By, Having Clause. • CREATE – DROP TABLE, Constraints • INSERT, UPDATE, DELETE, ALTER • DATA Control Language (DCL) • Different operations on tables – Rename, Tuple Variables, Set Operations(UNION Operator, UNION ALL Operator, INTERSECT Operator, Minus Operator), String Operations • Null Values	4	10

Unit 7 Transaction Management	• Introduction • Transaction Concept • Properties of Transactions • Transaction Terminology • Transaction Terminology • Transaction States • Concurrent Execution of Transactions • Operations on a Transactions • Concurrency Control • Schedules • Recoverability	3	10
Unit 8 PL/SQL	Introduction to PL/SQL The Advantages of PL/SQL PL/SQL Architecture PL/SQL Data types Variable and Constants Using Built_in Functions Conditional and Unconditional Statements Simple if, if… else, nested if..else, if..else Ladder Selection Case, Simple Case, GOTO Label and EXIT Iterations in PL/SQL Procedures in PL/SQL EXCEPTIONS in PL/SQL Database Triggers in PL/SQL File Input/Output	4	10
	Examples and Revision	4	0
		30	80

Chapter 0

Data, Files and Databases

Topics To Learn

0.0 Overview
0.1 Introduction
0.2 Operations on file
0.3 Introduction to Database

0.0 Overview

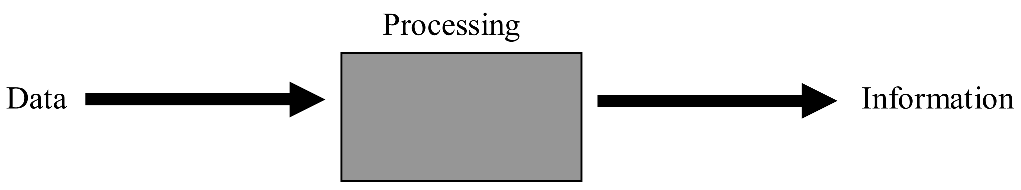
In this chapter we will study the concept of data, Files and then different operations that we can perform on the file. In the section 0.1 you will study Data and Files. Data is a collection of facts, such as values or measurements. It can be numbers, words, measurements, observations or even just descriptions of things. After understanding data we will study the concept of files. File is a container for data. We use files to keep our data safe and secured. For example, in office we keep all office information documents (office data) in different files. And each file has given a particular name. Similarly in computer we use files to keep our data. We will see these terms in section 0.2.

After understanding data and files we will study Database. Database is also a type of data storage like file but it has some higher facilities and features than a regular file which help us to manage the data easily. As in the market we are having different types of file folders to keep our paper documents similarly the database is also available in different structures and models to provide ease of retrieving the data.

0.1 Introduction

Data: In general we define data as; it is information in raw or unorganized form (such as alphabets, numbers or symbols) that represents conditions, ideas, or objects.

In short, Data is a collection of facts, Figures and statistics related to an object. Data can be processed to create useful information.
Hence we can say manipulated and processed form of data is called information, which is more meaningful than data. Data is used as input for processing and information is the output of this processing, as shown below.



For example: Suppose you are going to visit in a shop to purchase some items. So before you go, you make a list of items you want to purchase. This item list contains name and quantity of items (i.e. **data** about items) you are purchasing.

E.g.. Name and Quantity of items on item list are data about items to be purchased.

Item List	
Sugar	5 kg
Wheat	10 kg
Oil	5ltr
Rice	10 kg
Soap	4 qty

So here name and price of items is nothing but the data about that item.
Data can exist in a variety of forms such as numbers, text or symbols on pieces of paper, as bits and bytes stored in electronic memory, or as facts stored in a person's mind.

- In terms of computer data are symbols or signals that are input to the computer, stored in computer, and processed by a computer for output as usable information.
- We can say for example:

011000110011… can be represented as binary data in computer.

- Data is often distinguished from programs in computer. We define a program, as a set of instructions that perform a particular task for computer.

For example: we write C' Program for calculating sum of two numbers. This program contains set of C' language's instructions. Hence we cannot say those instructions as data. Yes but whatever numbers we are providing to that program to calculate sum are 'data'.
In this sense, data is everything that is not program code (instructions of programs).

Hence from above discussion we can differentiate Data, Instruction and Programs as-
Data: raw information.
Information: processed data.
Programs: set of instructions

Files: We use file to save the data. In general we keep our documents (papers on which something is written) in a folder file where they can be safe and secured.
For example after making a list of your purchasing items you may keep that list in your pocket or in envelope where it can be safe. With same manner, we store computer data in file (i.e. file of computer) where that data will be safe and secure.

We use file to save our data..

- In terms of computer we define a file as a collection of bytes stored as an individual unit/ entity. All data in disk are stored as a file with an assigned filename that is unique within the directory it resides in.
- Almost all information stored in a computer must be in a file.
- There are many different types of files: data files, text files, program files, directory files, and so on. Different types of files store different types of information.
- For example, program files store programs, whereas text files store text.
- To the computer a file is nothing more than a series of bytes.
- The software that manipulates it knows the structure of a file.

  E.g.. Database files are made up of a series of records where word processing file (documents) made up of a continuous flow of text.

0.2 Operations on file

There are mainly two types of operations on file –

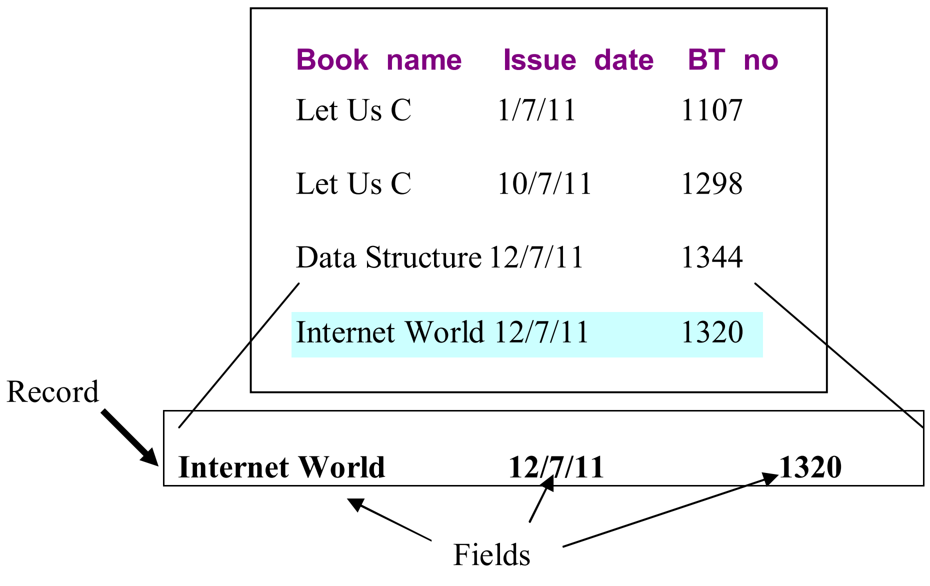
1. Retrieval Operation
2. Update Operation

When we want to extract the data from file for reading purpose then we perform retrieval operation on the file. Retrieval operations do not change the contents (data) of file; it only locates the records in file.

Before we go ahead, we will see what is record???

In database management systems, a complete set of information is called as a 'record'.

For Example: consider a below file, which contains library information.



Book name	Issue date	BT no
Let Us C	1/7/11	1107
Let Us C	10/7/11	1298
Data Structure	12/7/11	1344
Internet World	12/7/11	1320

The record is highlighted by color.

So in this example the records are: (Let Us C 1/7/11 1107),
(Let Us C 10/7/11 1298)
(Data Structure 12/7/11 1344)
(Internet World 12/7/11 1320)

We will see this term in more detail in further chapters.

Now let us continue with Update Operation….

Update operation on the other hand; change the file by modifying the records, deleting the records and inserting new records. That is we actually modify the contents in file in update operation.

The operations for locating, modifying, deleting and inserting records will vary from system to system, but there are most some operations that are used in most systems and those are given below:

- **Find (Locate):** The goal of this operation is to locate the record or records that satisfy the search criteria. A block that contains the records is transferred to the main memory and the records are searched. The first record that matches the search criteria is located and the search continues for other records until the end of file is reached.
- **Read:** In read operation the contents of the records are copied from memory to a program variable or work area. This command in some cases advances the pointer to next record in the result set.
- **Read Next:** Searches the next record that matches the selection condition and when found, the contents of record are copied to the program variable or work area.
- **Modify:** Also known as update. This command modifies the field values / data of the current record then writes modified record back to the disk.
- **Insert:** Inserts a new record to the file.
- **Delete:** Deletes the current record and updates the file on the disk to reflect the deletion.

0.3 Introduction to Database

Before we attempt to see what Data Base Management System is, we will see what is Database and what it contains.

0.3.1 Definition of Data Base:

- We use different files such as word file or Excel File to store text type of data or bmp files to store image type of data in computer. Similarly DBMS provides us another type of file known as database file to store our data. The difference between those ordinary text files and database file is that the database file (or simply, "Database") is more intelligent than those text files.
- A database is an application that manages data and allows fast storage and retrieval of that data. A database can organize, store, and retrieve large amount of data easily.
- And a DBMS (database management system) is a piece of software that is designed to make the different database operations easier.
- By storing data in a database, rather than in files, we can use the features of database to manage the data in a robust and efficient manner.
- For example: let consider a college contains a large collection of data (say, 30 GB) of student's information such as student's name, student's class, student's division, student's Roll no., marks, fees and so on. Several users access this data concurrently. Hence any questions that user has asked about the data must

be answered quickly, changes made to the data by different users must be applied consistently, and access to certain parts of the data (E.g.., fees) must be restricted.

- To maintain such large data in conventional file system is a time consuming and more critical job and we probably do not have 30 GB of main memory to hold all the data.
- Hence we can deal with this data management problem by storing the data in a database.
- There are different types of database but the most popular is a relational database that stores data in tables where each row in the table holds the same sort of information. In the early 1970s, Ted Codd, an IBM researcher devised 12 laws of normalization. These apply to how the data is stored and relations between different tables that we will see in further chapters.

0.3.2 Components of Database:

- A database consists of four elements as shown in Figure(0.1) below.

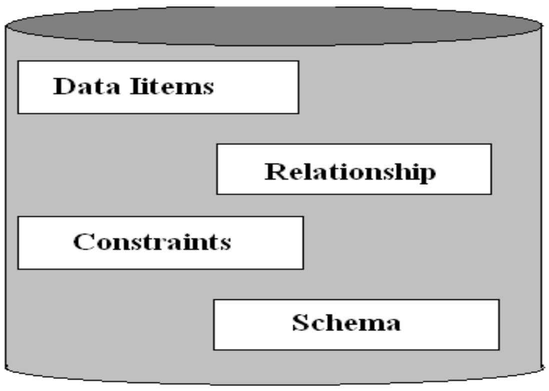


Figure 0.1 Components of Database

Data Item

- Each attribute of an entity is represented in storage by a **data item.**
- For example: Suppose there is a database of “customer account”. And this database contains attributes Cust_name, Cust_id, Cust_add, Balance.

Hence there is a data item for Cust_name, another data item for Cust_id etc.

- A data item is assigned a name in order to refer it in storage, retrieval and processing.

Relationships represent a correspondence between the various data elements / entities. In short it represents relations among the different entities.

Constraints are predicates that define correct database states.

In other words we can say that the Constraints within a database are rules, which control values allowed in columns and also enforce the integrity between columns and tables.

For example, consider a column constraint (i.e. constraint for column) “CHECK”. The “CHECK” constraint for column will limit the values allowed for that column.

For example, you could specify the data type of a column to be int, which can store values from 0-255, but then specify a CHECK constraint that limits values of 1-99 for that column. We will see these constraints in detail in further chapters.

Schema describes the organization of data and relationship within the database. In short schema is a logical structure (design) of database.

- A database schema is described in a formal language supported by the database management system (DBMS). In a relational database, the schema defines the tables, the fields in each table, and the relationships between fields and tables.
  Schemas are generally stored in a data dictionary.
  We will see examples of schema in section 1.6 in more detail.

0.3.3 Basic Terms related to Database:

Now we will see some basic terms related to database.

Entity:

- Entity is anything about which information is stored.
- For example: suppose we are going to maintain school level database. In this database student, teachers, books, subjects can be defined as different entities of that school. Or in a Bank database we can define customer, account etc as different entities of that particular bank.
- Each entity has its own set of properties or characteristics. For example, an student entity may have properties student’s name, student’s Roll no., student’s marks etc, or teacher entity may have properties teacher’s name, Salary, subject etc. In DBMS these properties of entity are called **attributes** of that entity.
- Hence as discussed above we can say that a person, concept, physical object or event all can be defined as entities.

Attribute:

Now we will discuss Attributes in more detail.

- In general, an attribute is a property or characteristic of something. For example, Color is an attribute of your hair. In programming computers, an attribute is a changeable property or characteristic of some component of a program that can be set to different values.
- The values of attributes describe a particular entity.

- For example: Suppose here is a relation or database of “Customer account” so we can define the attributes of this relation as Cust_name, Cust_add, Cust_id, balance.

Domain:

- In database, domain refers to the description of an attribute's allowed values.
- In short a domain is a set of values the attribute can have.
- For example: suppose there is a “student” database with attributes stud_name, stud_marks, stud_division. In this database, stud_marks may have allowed values of range 0 to 100 or stud_division may have allowed values A, B, C, D

etc. so these are the domains for the attributes sud_marks and stud_division respectively.

Instance:

- An instance of the entity represented by a set of specific values for each of the attributes at particular time period.
- For example; suppose there is a furniture shop where we are having variety of furniture so here furniture is the entity and the attributes of furniture entity could be **Furniture_type, item_color, item_price etc.**
- The attributes could be same for all kind of furniture in the shop. But the values of the attributes in each instance are different.
- Thus we have **chair, black, 2000** Rupees, as one instance and **bed, brown, 10000** Rupees as another instance.
- These two cases represent the attributes of two instances of the entity **"furniture".**

Record/ Tuple:

- The data representation in storage of each instance of an entity commonly called as record. And it is also called as tuple.
- In simple words, a row in a relation may represent a record.
- For example: Let consider below database/Relation of "Customer Details", where customer is an entity and cust_name, cust_id, cust_add, balance are attributes. And
  Jyoti A123, Nashik, 50000 is one record.

Relation "**Customer Details**"

Attributes →	**Cust_name**	**cust_id**	**Cust_add**	**Balance**
Tuple/ Record →	Jyoti	A123	Nashik	50000
	Sarika	B125	Pune	45000
	Manisha	C222	Nashik	30000

In above table / relation

Entity = Customer

Attributes= cust_name, cust_id, cust_add, balance.

Domains= cust_name (jyoti, sarika, manisha), cust_id (A123, B125, C222) etc.

Record / tuple= jyoti, A123, Nashik, 50000.

Instance= (Jyoti, A123, Nashik, 50000.) is one instance at a particular time period **T1**. And suppose jyoti has withdraw 10000 Rupees from her account then at a time period **T2**, then the instance at time T2 will be = (Jyoti, A123, Nashik, 40000).

Summary:

- Data are binary computer representation of stored logical entities.
- A file contains data that is needed for information processing.
- We use file to save these data.
- A file is a collection of bytes stored as an individual entity.
- There are mainly two types of operations on file –

    1. Retrieval Operation
    2. Update Operation

- **Definition of Data Base:** A database is an application that manages data and allows fast storage and retrieval of that data
- A database consists of four elements:
    1.Data item
    2.Relationships
    3.Constraints
    4.Schema

Objective Type Questions:

1. In Computer data are stored in ….
    a. Binary Format b. Decimal format
    c. Character Format d. Hexadecimal Format

2. Which of the followings is not an operation of file?
    a. Open b. Close c. Delete d. Hide

3. Which of the followings is a false statement?
I. A database is an application that manages data and allows fast storage and retrieval of that data.
II. Constraints are predicates that define correct database states.
III.In database, domain refers to the description of an attribute's allowed values.
IV. Schema describes the organization of data and relationship within the database.
    a. I b. I and III c. I and II d. None of the above

4. A database consists of four elements, these are…
    a. Entity, attribute, domain, relation
    b.Data item, Relationship, Constraint, Schema
    c.Schema, domain, Attributes, Constraints
    d.All of the above.

5. Which of the followings can be defined as an Entity?
    a. Person b. Bank c. Shop d. All of above

****

Chapter 1

Introduction to Database Management System

Topics to Learn

1.0 Overview
1.1 Definition to DBMS
1.2 WHY DBMS
1.2.1 Services provided by DBMS:
- Transaction Management:
- Concurrency Control**:**
- Recovery Management
- Security Management
- Language Interface

1.3 Applications of DBMS
1.4 Differences between File System and DBMS
1.5 Drawbacks of File system:
1.6 Abstraction Levels (Three Levels of Abstraction)
1.7 Database Users
1.8 DDL and DML
1.9 Structure of DBMS
1.10 Metadata

1.0 Overview

In previous chapter we studied data, files, and database. Now we may need to study such a tool (software tool), which will manage and handle the database efficiently.

In this chapter we will study the Database management system (DBMS), which is software package designed to store and manage databases. You will also study the different services that are provided by the DBMS, such as transaction management, concurrency control, recovery management etc. and different applications of the DBMS (i.e. where the DBMS used mostly) such as Banking, Airlines, Universities, and Telecommunication etc. After this we will learn the differences between conventional file systems and Database i.e. how the database is more convenient than the conventional file system.

The next topic is abstraction level in which you will study three levels of database abstraction (i.e. to hide specific data from specific user). Because sometimes you may need to hide some data in database from specific users so as to avoid complexity and to provide security to database. Hence we can use a database abstraction level which suggests what amount of data should be hidden from which users. Which are the database users and what are their roles will be covered in next topic i.e. “database users”.

To write the data into the database we must use some specific languages that DBMS can understand. The types of these languages are covered in next topic, which is “DDL and DML”. After that we will learn the structure of database, which describes how the data is to be stored in the database and how it is to be managed by the DBMS.

1.1 Definition to DBMS

As we studied that database can contain a huge amount of data. In short it is storage for data. If database contain large amount of data then there should be something to manage or handle it. As all of us know our mother manages all house related work, similarly in computer also we need to have such software or mechanism, which will handle all database related work and for that here we are going to study that mechanism which manages all database related work and that mechanism is nothing but Database Management System (DBMS).

A Database Management system (DBMS) is a collection of programs (or we can say it as a software) that enables you to store, modify and extract information from a database.DBMS is software designed to assist in maintaining and utilizing large collections of data in database.

1.2 WHY DBMS?

The Question is, why should we prefer DBMS over conventional File system…??

Below are some reasons which gives us clear idea that why to use DBMS?

- Data Base Management System stores data in database in an appropriate manner. I.e. it actually stores the data in a table format so that it will be easy to maintain and search the data.
- DBMS helps to control the organization, storage, management, and retrieval of data in a database.
- The DBMS accepts requests for data from the application program and instructs the operating system to transfer the appropriate data.
  That's why DBMS can be considered as system software and also can be said as Application Software.

Apart from these reasons DBMS provides following five services which a conventional File system does not provide.

1. Transaction Management:

- A transaction is sequence of database operations that represents a logical unit of work (delete record, update record, modify a set of records etc).
- DBMS manages these transactions. When the DBMS does a “commit” the changes made by the transaction are made permanent. If you don’t want to make changes permanent you can rollback the transaction and the database remain in its original state. You will study this Transaction Management process in detail in further chapter.

2. Concurrency Control:

- In a database, number of processes or actions (such as delete record, update record, insert new record etc) may execute simultaneously i.e. concurrently. So this simultaneous or concurrent execution of actions must be managed somehow.
- DBMS provides concurrency control mechanism by coordinating the actions of database manipulation processes.

3. Recovery Management:

- A simple meaning of recovery is to return to a normal/stable condition.
- Recovery mechanism in DBMS ensures that the database returned to a consistent state after a transaction fails or abort i.e. database should not be in inconsistent state.
- Inconsistency means that two files may contain different data of the same entity. For example, let consider a "student Admission" database with attributes stud_name and stud_address. And another database "student account" with attributes stud_id and stud_address. Now if the address of a student is changed in "student Admission" database, it must be changed in "student account" database. Because there is a possibility that it is changed in the "Students Admission" database and not from "student account" database. In this case, the data of the student becomes inconsistent. But the recovery Management service of the DBMS maintains consistency in database.

4. Security Management:

- Security mechanism of DBMS ensures that only authorized users are given access to the data in database.

5. Language Interface:

- To handle data in database a user has to use some kind of language, which will be understandable for database.
- And that's why DBMS provides languages support, which is used to define and manipulate the data in database.
- Basically there are two languages used by DBMS to interact between user and database and those are DDL and DML.
- The data structures are created by the data definition language (DDL) commands.
- The data manipulation is done using the data manipulation language (DML) commands.

We will see these languages in detail in further topics.

1.3 Difference between file system and DBMS

Now let see some differences between file system and DBMS .

1) Representing complex relationship among data is possible in DBMS not in file system.
2) Recovery and back up can be done in DBMS not in file system
3) Multiple user interfaces is allowed in DBMS
4) The changes made in one database will be applicable to all tables which is not possible in file system.
5) Data redundancy is removed in DBMS.
6) Unauthorized access in DBMS can easily be restricted and therefore data in databases are more secure compared to data in files!!
7) Enforcing integrity constraint in DBMS.
8) Provide storage structures for efficient query processing.

1.4 Drawbacks of File system

Here we are going to see that why the conventional File system is not so effective as compared to database.

1) Data redundancy and inconsistency

In file processing system, the same data may be duplicated in several files. For example, there are two files "Students" and "Library". The file "Students" contains the Roll No, name, address and telephone number and other details of all students in a college. The file "Library" contains the Roll No and name of those students who get a book from the library along with the information about the rented books. The data of one student appears in two files. This is known as data redundancy. This redundancy causes higher storage.

This situation can also result in data inconsistency. Inconsistency means that two files may contain different data of the same student. For example, if the address of a student is changed, it must be changed in both files. There is a possibility that it is changed in the "Students" file and not from "Library" file. In this case, the data of the student becomes inconsistent.

2) **Difficulty in Accessing Data**

It is not easy to retrieve information using a conventional file processing system. Convenient and efficient information retrieval is almost impossible using conventional file processing system.

3) **Data Isolation:**

Data are scattered in various files, and the files may be in different format, writing new application program to retrieve data is difficult.

4) Integrity Problems

The data values may need to satisfy some integrity constraints. For example: let consider a "student info" database, which contains all information of students in a college. Now suppose there is a field 'Stud_marks' in that database. Then value of this field must be greater than or equal to 0, because student's marks never been less than 0.

If we are using conventional file system then we have to handle this integrity constraint through programming (program code). But in database we can declare the integrity constraints along with definition itself (that you will study in SQL chap).

5) Atomicity Problem

It is difficult to ensure atomicity in file processing system. For example; in online banking system suppose you are transferring $1000 from Account A to account B. And if a failure occurs during transaction there could be situation like $1000 is deducted from Account A and not credited in Account B.

6) Concurrent Access anomalies

If multiple users are updating the same data simultaneously it will result in inconsistent data state. In file processing system it is very difficult to handle **this** using program code. This results in concurrent access anomalies.

7) Security Problems

Enforcing Security Constraints in file processing system is very difficult.

1.5 Applications of DBMS

Database is widely used all around the world in different sectors such as:

1. **Banking**: For customer information, accounts loans and banking transactions.

2. **Airlines**: For reservations and schedule information.

3. **Universities**: For maintaining student information, course registrations and grades etc.

4. **Credit card transactions**: For purchases on credit cards and generation of monthly statements.

5. **Telecommunications**: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards and storing information about the communication networks.

6. **Finance**: For storing information about holdings, sales and purchase of financial instruments such as stocks and bonds.

7. **Sales**: For maintaining customer, product and purchase information.

8. **Manufacturing**: For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores and orders for items.

9. **Human Resources**: For information about employees, salaries, payroll taxes and benefits and for generation of paychecks.

10. **Web based services**: For taking web users feedback, responses, resource sharing online shopping etc.

1.6 Abstraction levels (Three level architecture)

Data Abstraction:

In our day-to-day life many times we need to hide specific information from different persons for their convenience.
For example: suppose you are interested in purchasing a new car. At that time a car manufacturing company hides some details from you (such as where the car is manufactured, how many engineers made the design of this car, or who did manufacture the car etc.), to remove the complexity in your mind. Similarly DBMS hides certain data in the database from different users.

So the abstraction means to hide the specific data from different database users.

- Major aim of a DBMS is to provide users with an abstract view of data.
- Data abstraction hides certain details of how the data are stored & maintained in database.
- Most DB users are not computer trained, developers hide complexity through several levels of abstraction to simplify user's interaction with the systems

Three levels of data abstraction are:

1) Physical or Internal Level (Physical Schema):
2) Logical or Conceptual Level (Conceptual Schema):
3) View or External Level (External schema):

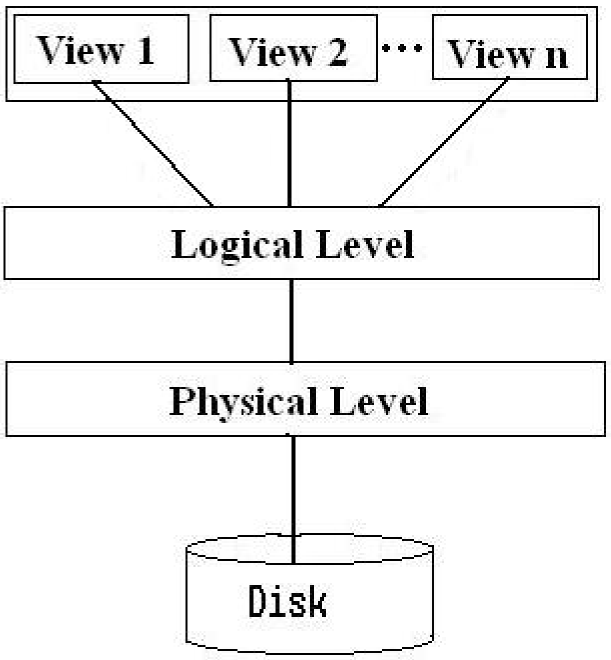


Figure 1.1 DBMS Levels of abstraction

1) Physical or Internal Level (Physical Schema):

- This is the lowest level of abstraction which describes how data are actually stored
- It also describes complex low-level data structures in detail

The physical schema (physical level) specifies additional storage details for data in database. Essentially, the physical schema summarizes how the relations described in the conceptual schema are actually stored on secondary storage devices such as disks and tapes.

The process of arriving at a good physical schema is called physical database design.

2) Logical or Conceptual Level (Conceptual Schema):

- Describes what data are stored in the database & what relationships exist among those data

- Describes the entire database in terms of relatively simpler structures

The conceptual schema (sometimes called the logical schema) describes the stored data in terms of the data model of the DBMS. In a relational DBMS, the conceptual schema describes all relations that are stored in the database.

- For Example: Let consider there is a University database, which contains relations/tables such as “student”, “faculty”, and “students’ enrollment in courses”, these relations contain information about entities, such as “students” and “Faculty”, and about relationships, such as “student’s enrollment in courses”. We can define the following conceptual schema for these relations:

  Students (sid:string, name:string, login:string, age:integer)
  Faculty (fid:string, fname:string, sal:real)
  Courses (cid:string, cname:string, credits:integer)
  Enrolled (sid:string, cid:string, grade:string)

In this example we can say that “Student” contains sid as type string, then age as integer, fname as string etc.

3) View or External Level (External schema):

- Highest level of abstraction which describes only a part of the DB
- User’s view of the DB. This level describes that part of the DB that is relevant to each user.

External schemas, allow data access to be customized (and authorized) at the level of individual users or groups of users. Any given database has exactly one conceptual schema and one physical schema because it has just one set of stored relations, but it may have several external schemas, each tailored to a particular group of users. Each external schema consists of a collection of one or more views and relations from the conceptual schema.

A view is conceptually a relation, but the records in a view are not stored in the DBMS. The external schema design is guided by end user requirements.
For example, we might want to allow students to find out the names of faculty members teaching courses, as well as course enrollments. This can be done by defining the following view:

Courseinfo (cid: string, fname: string, enrollment: integer)

1.7 Database users

There are different types of user that play different roles in a database environment. Following is a brief description of these users:

1) Application Programmers:

Application programmer is the person who is responsible for implementing the required functionality of database for the end user. Application programmer works according to the specification provided by the system analyst.

(2) End Users:

End users are those persons who interact with the application directly. They are responsible to insert, delete and update data in the database. They get information from the system as and when required. Different types of end users are as follows:

(3) Naive Users:

Naive users are those users who do not have any technical knowledge about the DBMS. They use the database through application programs by using simple user interface. They perform all operations by using simple commands (menus and buttons) provided in the user interface.

Example: The data entry operator in an office is responsible for entering records in the database. He performs this task by using menus and buttons etc. He does not know anything about database or DBMS. He interacts with the database through the application program.

(4) Sophisticated Users:
Sophisticated users are the users who are familiar with the structure of database and facilities of DBMS. Such users can use a query language such as SQL to perform the required operations on databases. Some sophisticated users can also write application programs.

(5) Database Administrator:
Database administrator is responsible for, managing the whole database system. He designs creates and maintains the database. He manages the users who can access this database, and controls integrity issues. He also monitors the performance of the system and makes changes in the system as and when required.

1.8 DDL and DML:

DDL: Data Definition Language

- The Data Definition Language (DDL) is used to create and destroy databases and database objects. Database administrators will primarily use these commands during the setup and removal phases, of a database project.

- DDL statements are used to define the database structure or schema.
  In below given example there are some DDL commands:
  CREATE - to create objects in the database

  ALTER - alters the structure of the database

  DROP - delete objects from the database

  TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed

  COMMENT - add comments to the data dictionary

  RENAME - rename an object

DML: Data Manipulation Language

- DML is used to change the data in the database tables. Instructions of DML are well known for everyone: insert, update, delete.

- DML statements are used for managing data within schema objects i.e. Data within database.

Some examples:

SELECT - retrieve data from the a database

INSERT - insert data into a table

UPDATE - updates existing data within a table

DELETE - deletes all records from a table, the space for the records remain

CALL - call a PL/SQL or Java subprogram.

➢ There are two main types of DML :
1. Procedural DML
2. Nonprocedural DML

- **Procedural DML:**

A low-level or procedural DML allows the user, i.e. programmer to specify what data is needed and how to obtain it. This type of DML typically retrieves individual records from the database and processes each separately. The programmers use the low-level DML.
Example of procedural DML is “Relational Algebra”.

- **Nonprocedural DML:**

A high-level or non-procedural DML allows the user to specify what data is required without specifying how it is to be obtained. Many DBMSs allow high-level DML statements either to be entered interactively from a terminal or to be embedded in a general-purpose programming language.

The end-users use a high-level DML to specify their requests to DBMS to retrieve data. Usually a single statement is given to the DBMS to retrieve or update multiple records. The DBMS translates a DML statement into a procedure that manipulates the set of records. The examples of non-procedural DMLs are SQL and QBE (Query-By-Example) that are used by relational database systems. These languages are easier to learn and use. The part of a non-procedural DML, which is related to data retrieval from database, is known as query language.

1.9 STRUCTURE OF DBMS

DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations (insert, delete, update and retrieval) on the database. The components of DBMS perform these requested operations on the database and provide necessary data to the users. The various components of DBMS are shown below:

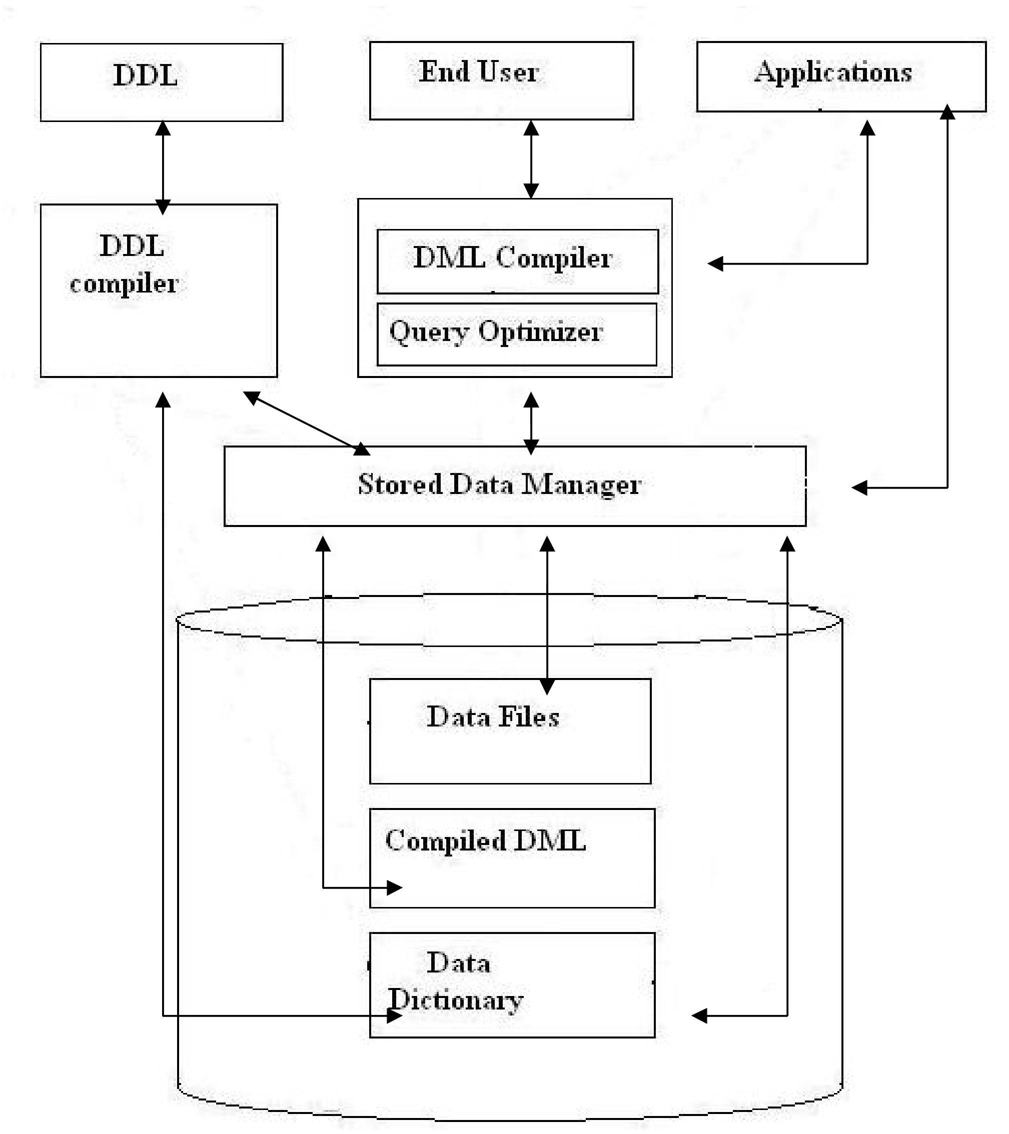


Figure 1.2 Structure of DBMS

1. DDL Compiler - Data Description Language compiler processes schema definitions specified in the DDL. It includes metadata (data about data) information such as the name of the files, data items, storage details of each file, mapping information and constraints etc.

2. DML Compiler and Query optimizer - The DML commands such as insert, update, delete, retrieve from the application program are sent to the DML compiler for compilation into object code for database access. The object code is then optimized in the best way to execute a query by the query optimizer and then send to the data manager.

3. Data Manager - The Data Manager is the central software component of the DBMS also knows as Database Control System.

➢ **The Main Functions Of Data Manager Are: –**

• Convert operations in user's Queries coming from the application programs or combination of DML Compiler and Query optimizer which is known as Query Processor from user's logical view to physical file system.
• Controls DBMS information access that is stored on disk.
• It also controls handling buffers in main memory.
• It also enforces constraints to maintain consistency and integrity of the data.
• It also synchronizes the simultaneous operations performed by the concurrent users.
• It also controls the backup and recovery operations.

4. Data Dictionary –
Data Dictionary is a repository of description of data in the database. It contains information about
• **Data** - names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.
• Relationships between database transactions and data items referenced by them, which is useful in determining which transactions are affected when certain data definitions are changed.
• Constraints on data i.e. range of values permitted.
• Detailed information on physical database design such as storage structure, access paths, files and record sizes.
• Access Authorization - is the Description of database users their responsibilities and their access rights.
• Usage statistics such as frequency of query and transactions.
Data dictionary is used to actually control the data integrity, database operation and accuracy. It may be used as a important part of the DBMS.

- **Importance of Data Dictionary** -

Data Dictionary is necessary in the databases due to following reasons:
• It improves the control of DBA over the information system and user's understanding of use of the system.
• It helps in documenting the database design process by storing documentation of the result of every design phase and design decisions.
• It helps in searching the views on the database definitions of those views.
• It provides great assistance in producing a report of which data elements (i.e. data values) are used in all the programs.
• It promotes data independence i.e. by addition or modifications of structures in the database application program are not effected.

5. Data Files - It contains the data portion of the database.

6. Compiled DML - The DML complier converts the high level Queries into low level file access commands known as compiled DML.

7. End Users - They are already discussed in previous section.

1.10 Metadata:

- **Metadata** is loosely defined as data about data.
  For example: a web page may include metadata specifying what language it's written in, what tools were used to create it, and where to go for more on the subject, allowing browsers to automatically improve the experience of users.

- Metadata is defined as data providing information about one or more aspects of the data, such as:

  1) Means of creation of the data
  2) Purpose of the data
  3) Time and date of creation
  4) Creator or author of data
  5) Placement on a computer network where the data was created
  6) Standards used

- **For example**, a digital image may include metadata that describes how large the picture is, the color depth, the image resolution, when the image was created, and other data. A text document's metadata may contain information about how long the document is, who the author is, when the document was written, and a short summary of the document.
- Metadata is data about data.
- As such, metadata can be stored and managed in a database, often called a registry or repository. However, it is impossible to identify metadata just by looking at it because a user would not know when data is metadata or just data.

Summary

- DBMS is a collection of programs that enables you to store, modify and extract information from a database.

- **DBMS provides below five services:**
  1) Transaction Management:
  2) Concurrency Control:
  3) Recovery Management:
  4) Security Management:
  5) Language Interface:

- **Applications of DBMS:**

Following are the applications of DBMS
-Banking**:**
-Airlines**:**
-Universities**:**
-Credit card transactions**:**
-Telecommunications**:**
-Finance
-Sales
-Manufacturing
-Human Resources
-Web based services

Drawbacks of File system:

-Data redundancy and inconsistency:
-Difficulty in Accessing Data
-Data Isolation:
-Integrity Problems
-Atomicity Problem
-Concurrent Access anomalies
-Security Problems

Abstraction levels (Three level architecture):

-Three levels of data abstraction are:

1. Physical or Internal Level (Physical Schema): the lowest level of abstraction, which describes how data are actually stored
2. Logical or Conceptual Level (Conceptual Schema): Describes the entire DB in terms of relatively simpler structures.
3. View or External Level (External schema): Highest level of abstraction, which describes only a part of the DB

Database users:

1. Application Programmers
2. End Users:
   I. Naive Users:
   II. Sophisticated Users:
3. Database Administrator:

DDL: The Data Definition Language (DDL) is used to create and destroy databases and database objects.

DML: DML is used to change the data in the database tables.

Metadata: Metadata is defined as data providing information about one or more aspects of the data

Multiple Choice Questions

1. Three levels of data abstraction are:
   a. Logical, Physical, View
   b. Logical, abstract, conceptual
   c. Physical, hidden, view
   d. None of the above.
2. Which of the followings is not in the category of database users
   a. Naïve User b. Sophisticated User
   c.End User d.Home User

3. ----------- is a repository of description of data in the database.
   a. Data dictionary b. DML compiler
   c. DDL compiler d. Query Optimizer

4. …… and …… are two main types of DML.

a. Procedural DML and Nonprocedural DML
a. Procedural and Transactional DML
b. Nonprocedural and Transactional DML
c. None of the above

5. Data Files:
a. It contains the data portion of the database.
b. It contains Metadata
c. It contains DML commands
d. It contains DDL commands

6. DDL stands for
a. Data Dictionary Language b.Data Directory Language
c.Directory Definition Language d. Data Definition Language

7. DML stands for:
a. Data Maintenance Language b. Data Manipulation language
c.Data Mediation Language d. None of the above

8. Logical Level of data abstraction describes-
a. What data are stored in the DB & what relationships exist among those data.
b. Complex low-level data structures
c. Only a part of the DB
d. Complex high-level data structure.

9. Metadata is….
a. Complex data item in database b. Data about data
c. Data about entity d. Hidden data in database.

10. State whether the following statements true or false?
a. DBMS suffers from redundancy and inconsistency
b. End Users are responsible for managing the whole database system.
c. Banking is one of the applications of DBMS.
d. Metadata can be defined as ‘data about data’.

****

Chapter 2

Relational Data Models and Relational Algebra

Topics to Learn

2.0 Overview
2.1 Introduction to Data Models
1. Object-based Logical Models
    - The E-R Model
    - The Object-Oriented Model
2. Record-based Logical Models
    - The Relational Model
    - The Network Model
    - The Hierarchical Model
3. Physical Data Models

2.2 Integrity Constraints
- NOT NULL Integrity Constraints
- Entity Integrity constraint
- Referential Integrity or foreign key constraint
- Key constraint or unique constraint

2.3 Relational Algebra/ Relational Data Manipulation
2.4 Relational Algebra Operations
2.4.1 Selection and Projection
2.4.2 Set Operations
- Union (U),
- intersection (∩),
- set-difference (−),
- cross-product (×).

2.4.3 Joins

2.0 Overview

In this chapter we are going to study different data models such as object based data models, Record based data models and their subtypes. A data model is sometimes called a database model, which describes how a database is structured and used. That means different data models give us different designs of database and their corresponding uses. We are going to see those design types that is data models in this chapter.

When we design database we should provide some restrictions and rules for that database. These rules and restrictions are known as Constraints. In DBMS we call these constraints as Integrity Constraint. Integrity constraints that we are going to see in this chapter are NOT NULL, Entity integrity, Referential integrity and Unique key constraints.

We will also cover Relational Algebra, which is nothing but the collection of different operations that are used to modify the contents of data in database.

2.1 Introduction to Database

As we know database is basically used to store the data. And it represents the data with different views; sometimes in table format or by using some graphical structures. In short a database may have different structures and each structure represents data with different view.

A data model is a collection of concepts, which describes the structure of database. Similarly it provides the necessary mean to achieve abstraction of data in database. What is Data Abstraction and why it is required that we already have seen in last chapter.

A data model also defines the method of storing and retrieving data in database.
Three types of data models are available:

1. Object-based Logical Models
   - i. The E-R Model
   - ii. The Object-Oriented Model
2. Record-based Logical Models
   - i. The Relational Model
   - ii. The Network Model
   - iii. The Hierarchical Model

3. Physical Data Models

Let see these models one by one.

1. Object-based logical models:

This model describes data at the conceptual and view levels. What is there in conceptual and view level that we already have seen in "Data Abstraction" topic. Databases designed under this model are having more flexible structure.

By using this model one can specify data constraints explicitly. Data or domain constraints means: Each column contains same type of data thus when you select a data type for a particular domain then DBMS will not accept any value of other data type.

Object based logical model includes following sub-data models:

i. The entity relationship (E-R Model) model
ii. The object-oriented model

i. The entity relationship model

The E-R model consists of a collection of basic **objects** (entities) and **relationships** among these objects.

We have studied that an **entity** is a distinguishable object that exists, and each entity has a set of **attributes** that describes it.

For ex. *number* and *balance* attributes for an "account" entity.
And a **relationship** is an association among these several entities.

For ex: if there are two relations (databases), say, “cust_details” and “acc_details”. Then the third relation, say, *“cust_acc”* specifies relationship associated with a customer and each account he or she has.

- The set of all entities or relationships of the same type is called the **entity set** or **relationship set**. (These terms we will study in detail in chap: E-R diagram)
- Another essential element of the E-R diagram is the **mapping cardinalities**, which express the number of entities to which another entity can be associated via a relationship set. This topic too will be covered in chap Entity Relational Model.

Now let’s see some basic terms that Entity Relational Model uses while representing data in database.

An E-R diagram can express the overall logical structure of a database graphically and in E-R model-

- **Rectangles**: represent entity sets.
- **Ellipses**: represent attributes.
- **Diamonds**: represent relationships among entity sets.
- **Lines**: link attributes to entity sets and entity sets to relationships.

For example: let consider an entity “student” which have attributes Roll_no, name, class etc and “book” is another entity which has attributes book_code, book_name, price, author etc. And a Relation called “Buy” shows the relationship between these two entities. This is because the student can buy book or a book can be bought by the student.
Here ‘M’ indicates Many to Many cardinality. I.e. one student can purchase more than one book and more than one student can purchase one book.

This model can also be shown by graphical method, as follows.

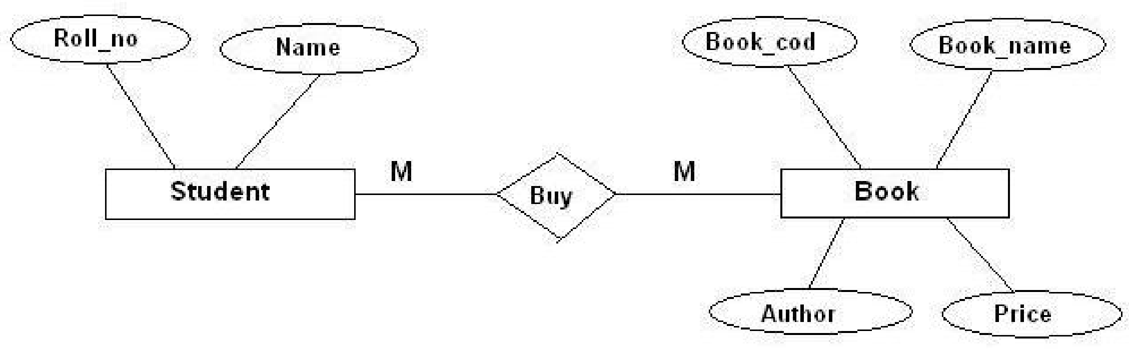


Figure2.1: A sample E-R diagram.

ii. **Object Oriented Model**
-An object database (also object-oriented database) is a model in which information is represented in the form of objects as used in object-oriented programming.

-Object databases have been considered since the early 1980s and 1990s but they have made little impact on mainstream commercial data processing, though there is some usage in specialized areas.

-When database capabilities are combined with object-oriented
programming language capabilities, the result is an object-oriented database
management system (OODBMS).

-Today's trend in programming languages is to utilize objects, thereby making OODBMS ideal for object-oriented programmers because they can develop the product, store them as objects, and can replicate or modify existing objects to make new objects within the OODBMS. Information today includes not only data but also video, audio, graphs and photos which are considered complex data types. Relational DBMS are not natively capable of supporting these complex data types. By being integrated with the programming language, the programmer can maintain consistency within one environment because both the OODBMS and the programming language will use the same model of representation

- An object contains values stored in **instance variables** within the object.
  <u>Instance Variable:</u> In object-oriented programming with classes, an instance variable is a variable defined in a class (i.e. a member variable), for which each object of the class has a separate copy. They live in memory for the life of the object.)

Unlike the record-oriented models, these values are themselves objects.

- Thus objects contain objects to an arbitrarily deep level of nesting.
- An object also contains bodies of code that operate on the object.
- These bodies of code are called **methods**.
- Objects that contain the same types of values and the same methods are grouped into **classes**.
- A class may be viewed as a type definition for objects.
- Analogy: the programming language concept of an abstract data type.
- The only way in which one object can access the data of another object is by invoking the method of that other object.
- This is called **sending a message** to the object.
- Internal parts of the object, the instance variables and method code, are not visible externally.
- Result is two levels of data abstraction.

For example, consider an object representing a bank account.

- The object contains instance variables *number* and *balance*.
- The object contains a method *pay-interest*, which adds interest to the balance.
- Under most data models, changing the interest rate mean changing code in application programs.
- In the object-oriented model, this only means a change within the *pay-interest* method.

2. Unlike entities in the E-R model, each object has its own unique identity, independent of the values it contains:

- Two objects containing the same values are distinct.
- Distinction is created and maintained in physical level by assigning distinct object identifiers.

2. Record based Logical Models

- Record based logical model describe data at the conceptual and view levels.
- Unlike object-oriented models, these models are used to
    1. Specify overall logical structure of the database, **and**
    2. Provide a higher-level description of the implementation.

- It is named so because in this model the database is structured in fixed-format records of several types where here each record type defines a fixed number of fields, or attributes.
- Each field is usually of a fixed length (this simplifies the implementation).
- Record-based models do not include a mechanism for direct representation of code in the database.
- Separate languages associated with the model are used to express database queries and updates.

We are familiar with the ER model, which is an example of an object based logical model. There are other data models one of those is Record based logical model. Record based model consists of three types.

The three types of record based models are:
- Hierarchical model
- Network model
- Relational model

In a Hierarchical model, data is represented in the form of a tree. Data in a hierarchical model is represented by a collection of records, and relationships between the data are represented by links.

A network model is similar to a hierarchical model in the way that data and the relationships among them are represented in the form of records and links. However, records in a database are represented graphically.

In the relational model, the table in a database has fixed record length with fixed number of attributes or fields.

Of these three models, the relational model is the most popular.

i. Relational Model (relational database)

Relational model is also known as relational database (RDB/RDBMS). A Relational Database Management System (RDBMS) provides a comprehensive (complete) and integrated approach to information management.

A relational model provides the basis for a relational database. A relational model has three aspects:

- Structures
- Operations
- Integrity rules

Structures consist of a collection of objects or relations that store data. An example of structure is a table (relation). You can store information in a table and use the table to retrieve and modify data.

Operations are used to manipulate data and structures in a database. When using operations, you must hold a predefined set of integrity rules.

Integrity rules are laws that govern the operations allowed on data in a database. This ensures data accuracy and consistency.

Relational database components include:

- Table
- Row
- Column
- Field
- Primary key
- Foreign key

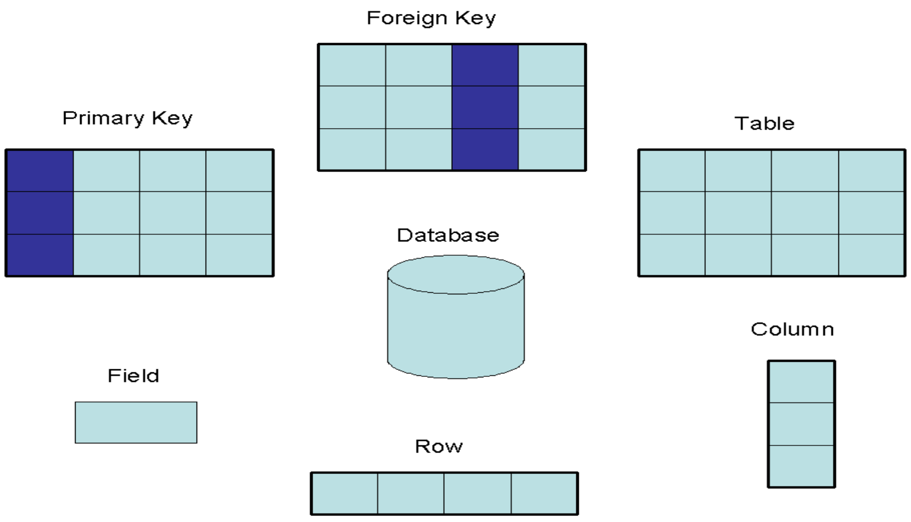


Figure2.2. Relational database components

A Table is a basic storage structure of an RDBMS and consists of columns and rows. A table represents an entity. For example, suppose customer in a bank is an entity then "cust_account_details" table shown in Figure 1.3: stores information about all customers of that bank.

A Row is a combination of column values in a table and is identified by a primary key (you will study primary key concept in detail in chap 4). Rows are also known as records. For example, a row in the table "cust_account_details" contains information about one customer in a bank.

A Column is a collection of one type of data in a table. Columns represent the attributes of an object. Each column has a column name and contains values that are bound by the same type and size. For example, a column in the "cust_account_details" table specifies the names of the customers, account number of the customers, balance of the customers etc in the bank.

A Field is an intersection of a row and a column. A field contains one data value. If there is no data in the field, the field is said to contain a NULL value.

Cust_name	**Acc_no**	**Balance**
Pravin	A123	10000
Shital	A221	25000
Anuja	A423	60000

Figure 2.3: cust_account_details

A Primary key is a column or a combination of columns that is used to uniquely identify each row in a table. For example, the column containing account numbers in the "cust_account_details" table is created as a primary key and therefore every account number is different. I.e. a column in a table can be considered as a primary key if and only if its values are not repeated. For example we cannot consider "cust_name" as a primary key because name of customers may repeat in a same column

A primary key must contain a value i.e. it cannot contain a NULL value.
(We will see "key" concept in more detail in chap 4)

A Foreign key is a column or set of columns that refers to a primary key in the same table or another table. You use foreign keys to establish connections between, or within, tables. A foreign key must either match a primary key or else be NULL.

- Data and relationships are represented by a collection of **tables**.
- Each **table** has a number of columns with unique names, E.g.. *customer, account.*
- A relational data model represents the database as a collection of relations.
- The fundamental assumption of the relational model is that all data is represented in relations also called as tables.
- A relational database allows the definition of data structures, storage and retrieval operations and integrity constraints. In such a database the data and relations between them are organized in the form of tables. (A table is a collection of records).
- The purpose of the relational model is to provide a declarative method for specifying data and queries: we directly state what information the database contains and what information we want from it, and let the database management system software take care of describing data structures for storing the data and retrieval procedures for getting queries answered.

Table: It describes the overall relation.
Tuple: The row in table is called a tuple.
Attribute: A column header of table is called attribute.
Degree: A degree of relation is the number of attribute of its relational schema.

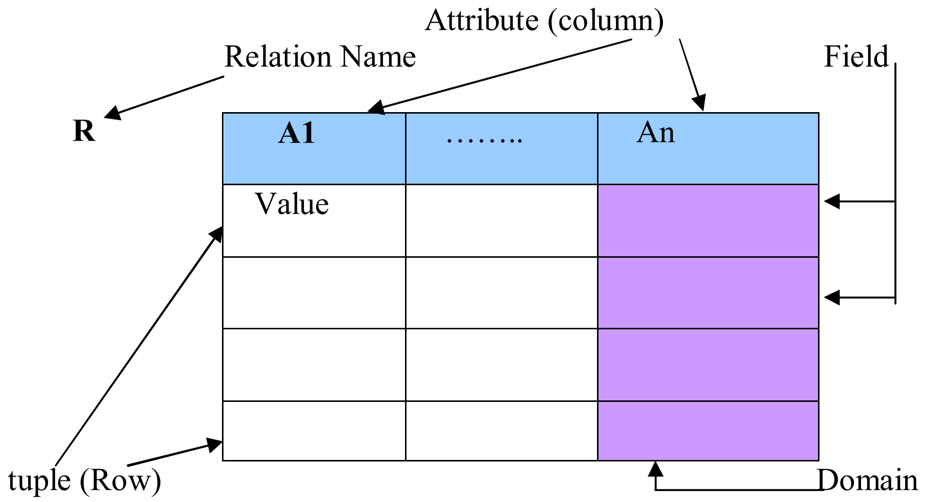


Figure2.4: A sample relational database.

Example: The E-R diagram shown in Figure 2.1 can be represented in relational model as shown in Figure2.5 below.

" Student"

Roll no	Name
13	Atul
22	Manisha
23	Shital

"Book"

Book_name	**Book_cod**	**Author**	**Price**
DBMS	A123	Korth	450
OS	A223	stalling	400
JAVA	A144	Balguru	350

"Buy"

Roll	**Book_name**
13	DBMS
22	OS

Figure 2.5: sample Relational model

- The relational model does not use pointers or links, but relates records by the values they contain. This allows a formal mathematical foundation to be defined.

ii. The Hierarchical Model

- Hierarchical data files permits records to be grouped together. This allows a parent-child relationship (a single one-to-many relationship) to be defined between records. In simple forms, the parent records are used to collect information that is common to all child records of the same group. This effect reduces redundancy with the database.
- Therefore here organization of the records is as a collection of **trees.**
- Figure2.7 shows a sample hierarchical database that is the equivalent of the relational database of Figure2.6.

Name	Street	City	Member
Jiten	Maple	Mumbai	990
Jolly	West	Nashik	336
Sudha	Sidehill	Pune	800
Jolly	West	Nashik	644
Sudha	Sidehill	Pune	644

Member	Balance
990	10000
336	25000
644	15000
800	40000

Figure: 2.6 Relational database

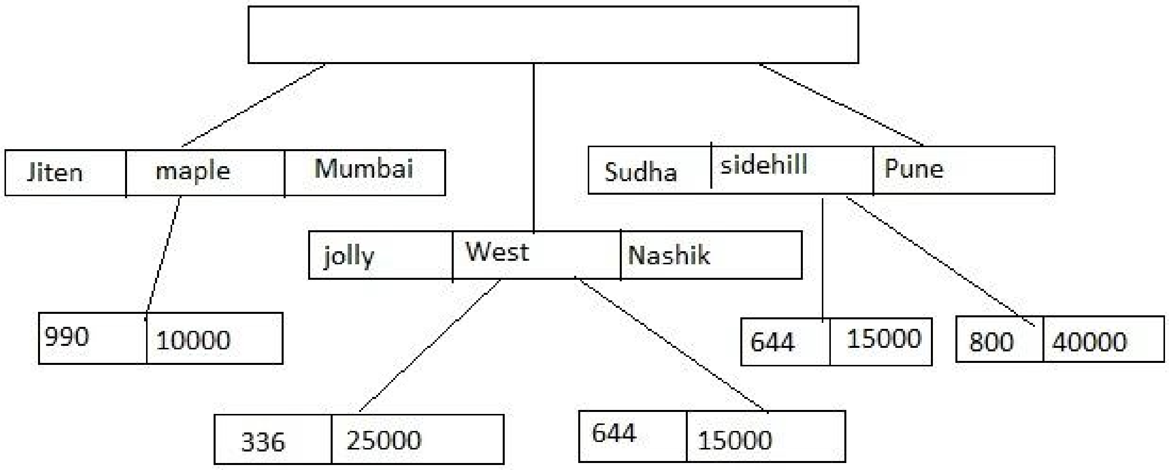


Figure2.7: A sample hierarchical database

- In this example the data is sorted hierarchically, using a downward tree.
- This model uses pointers to navigate between stored data. It was the first DBMS model.

iii. The Network Model

- While hierarchical databases emphasize the use of a single "path" to access all records, network databases may provide multiple paths to locate individual records and sets of records.
- In this model data are represented by collections of records.
- Network databases allow records to participate in multiple relationships or sets.
- And the relationships among data are represented by links.
- In network model each record is treated as a collection of fields:
- For ex: let us see how data is to be defined in network database.

There are two databases, say, cust_detail and acc_detail. Where record of cust_detail contains fields cust_name, cust_addr, cust_city. And record of acc_detail contains fields acc_no, bal.

```
type cust_detail = record
                   customer-name: string;
                   customer-street: string;
                   customer-city: string;
              end
```

```
type acc_detail = record
                        account-number: string;
                        balance: integer;
                  end
```

- Figure2.8 shows a sample network database that is the equivalent of the relational database of Figure2.6.

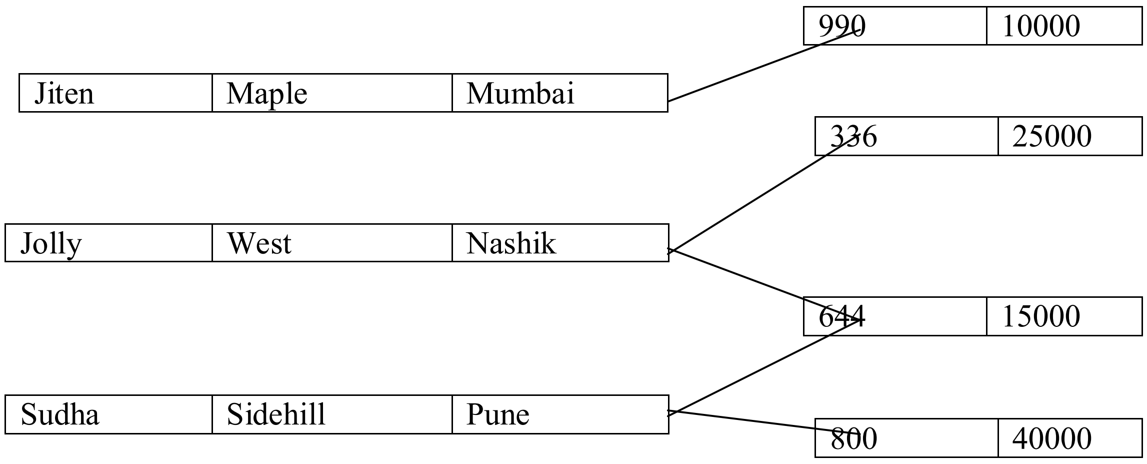


Figure 2.8:A sample network database

3. Physical Data Models

A physical database model is similar to other data models which show all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships between tables.

Features of a physical data model include:

- This model specifies all tables and columns in database.
- This model uses foreign keys to identify relationships between relations.
- Because of some Physical considerations the physical data model is quite different from the logical data model.
- Physical data model will be different for different databases.

For example, In MySQL and SQL Server the data type we use for column may be different.

This model is used to describe data at the lowest level. This model contains few models, E.g. Unifying model and Frame memory.

But we will not cover physical models here.

2.2 Integrity constraints

- An integrity constraint is a condition that is enforced automatically by the DBMS and that prevents the data from being stored in the database. The DBMS enforces integrity constraints in that it only permits legal instances to be stored in the database. The **key constraint** and the **referential**

integrity constraints are identified as the two minimum constraints that must be enforced by the DBMS.

- Integrity constraints are used to ensure accuracy and consistency of data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity. There are many types of integrity constraints that play a role in referential integrity.

1. NOT NULL Integrity Constraints
2. Entity Integrity constraint
3. Referential Integrity or foreign key constraint
4. Key constraint or unique constraint:

NOT NULL Integrity Constraints

By default, all columns in a relation allow null values. **Null** means the absence of a value. A **NOT NULL** constraint requires a column of a table contain no null values.For ex: Consider below relation of "Stud _info" in which NOT NULL constraint is applied on column "Marks" hence no row may contain NULL value for this column. Similarly NOT NULL constraint is not applied for "ID" hence any row can contain NULL value for this column.

"Stud_info"

ID	Name	Marks
1	Pooja	89
2	Vivek	90
	Mohini	79
4	Vivek	86

Figure 2.9: Example NOT NULL constraint

Entity integrity

- Entity Integrity ensures that for each row in a table, the value of the primary key is unique and is not null.

The entity integrity constraint states that no primary key value can be null. This is because the primary key value is used to identify individual tuples in a relation . Having null value for the primary key implies that we cannot identify some tuples.

ex: consider below relation "stud_info" with "ID" as a primary key.

ID	Name	Marks
1	Pooja	89
2	Vivek	90
Null	Mohini	79
2	Vivek	86

Figure: 2.10 Relation "stud_info"

In above relation the primary key “ID” contains null value as well as repeated value ‘2’ . Hence here we cannot consider ID as a primary key for the relation “Stud_info” because it is not following entity integrity constraints.

Now see below example of relation “stud_info1”. Let’s again consider primary key as “ID”. Here primary key “ID” follows the entity integrity constraints. i.e. here primary key contains unique values and it doesn’t have any null value.

ID	Name	Marks
1	Pooja	89
2	Vivek	90
3	Mohini	79
4	Vivek	86

Figure: 2.11 example relation “stud_info1”

Referential Integrity **or foreign key constraint**

- Often we wish to ensure that a value appearing in a relation for a given set of attributes also appears for another set of attributes in another relation. This is called **referential integrity**.
- Different tables in a relational database can be related by common columns, and the rules that govern the relationship of the columns must be maintained. Referential integrity rules guarantee that these relationships are preserved.

The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.

This constraint asserts that a reference in one data item indeed leads to another data item. A foreign key is a field that is a primary key in another table.

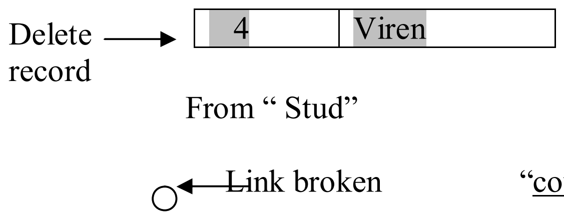
Referential integrity consists of:

1) Not inserting a record if the value of the foreign key being inserted does not match an existing record with the primary key having the same value in another related table.
2) Not deleting a record whose primary key is defined as a foreign key in another table and
3) Not modifying the value of primary keys.

- Consider below example of a database that has not enforced referential integrity.

“Stud”

Stud_id	Stud_name
1	Pooja
2	Jaya
3	Akash
4	Viren



"course"

Stud_id	Course_id	Course_name
3	A1	Java
4	A2	.Net
3	A3	PHP

Figure: 2.12 Example of referential integrity

In this example, there is a foreign key (**Stud_id**) value '4' in the course table. This value references a non-existent student — in other words there is a foreign key value with no corresponding primary key value in the referenced table.This anomaly came about when the record for a student called "Viren", with an **Stud_id** of "4", was deleted from the Stud table, even though the course ".Net" referred to this student. If referential integrity had been enforced, the deletion of the main record i.e. (4, Viren) in "Stud" would have been possible, but its associated record in "course" would have been deleted as well.

Unique Key Constraint:

A UNIQUE key integrity constraint states that every value in a column or set of columns (key) is unique—that is, no two rows of a table have duplicate values in a specified column or set of columns.

For example, in below Figure:2.13 a UNIQUE key constraint is defined on the "Course_name" column of the "course" table to disallow rows with duplicate course names.

Unique key Constraint
(no row may duplicate a value in the constraints column)
"Course"

Stud_id	Course_id	Course_name
1	A1	Java
2	A2	.Net
3	A3	PHP

Figure: 2.13 Example of Unique key constraints

Suppose we insert a new row (4, A3, PHP). This row violates the UNIQUE key constraint, because "A3" is already present in another row; therefore this insertion is not allowed in course relation.

Now if we insert a new row, say, (4, null, PHP). Then this row is allowed because the null value is entered for "Course_id" column. However if a NOT NULL constraint is also applied on "Course_id" column, this row is not allowed.

2.3 Relational Algebra or Relational Data Manipulation

Collection of different types of operations to manipulate relation is known as relational algebra. Relational algebra is a procedural language associated with the relational model. Or we can say that it is one of the examples of procedural DML. In general, a basic set of relational model operation constitutes the relational algebra. Thus the algebra operations produce new relations, which can be further, manipulated using operations of the same algebra.

Queries in algebra are composed using a collection of operators. A fundamental property is that every operator in the algebra accepts (one or two) relation instances as arguments and returns a relation instance as the result. This property makes it easy to compose operators to form a complex query—a relational algebra expression is recursively defined to be a relation, a unary algebra operator applied to a single expression, or a binary algebra operator applied to two expressions.

We describe the basic operators of the algebra (selection, projection, union, cross-product, and difference), as well as some additional operators that can be defined in terms of the basic operators.

2.4. Relational algebra operations

- These operations are categorized into unary and binary operations.
- The operations that need single source (one table) of relation is known as unary operations.
- The operation that need dual source (two tables) of relations is known binary operations.

2.4.1 Selection and Projection

We will use the Sailors table shown below for the next concepts:

Id	Name	Rating	Age
28	John	9	40
55	Smith	10	35
44	Angel	7	25
50	Grek	9	35

Figure: 2.14 Sailors Table

We will consider two instances of Sailors table for below operations:

Id	Name	Rating	Age
28	John	9	40
55	Smith	10	35
44	Angel	7	25
50	Grek	9	35

Figure 2.15 Instance S1

Id	Name	Rating	Age
28	John	9	40
55	Smith	10	35
44	Angel	7	25
50	Grek	9	35

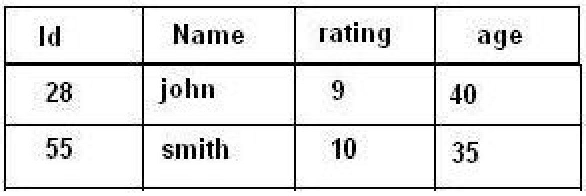
Figure 2.16 Instance S2

- Relational algebra includes operators to **select rows** from a relation **(σ)** and to **project columns (π).**
- The Select operation is used to select a particular column from a table. That means the select operation selects tuples that satisfy a given predicate (condition).
- These operations allow us to manipulate data in a single relation.
- Consider the instance of the Sailors relation shown in Figure2.16, denoted as S2. We can retrieve rows corresponding to expert sailors by using the σ operator. The expression

$$\sigma_{rating>8}(S2)$$

evaluates to the relation shown in Figure2.17.

- The subscript rating>8 specifies the selection criterion to be applied while retrieving tuples.



Id	Name	rating	age
28	john	9	40
55	smith	10	35

Figure 2.17 $\boldsymbol{\sigma_{rating>8}(S2)}$

- The subscript rating > 8 specifies the selection criterion to be applied while retrieving tuples.
- The **projection** operator π allows us to extract desired columns from a relation and remaining columns are left out.
- We list those column names (attributes) that we wish to appear in the result as a subscript to **π.**
- for example, we can find out all sailor names and ratings by using π.
- The expression

$$\boldsymbol{\pi_{name,\ rating}(S2)}$$

Evaluates to the relation shown in Figure 2.18. The subscript name, rating specifies the fields to be retained; the other fields are 'projected out.'

Name	rating
john	9
smith	10
angel	7

Figure 2.18 $\pi_{name,rating}(S2)$

- The schema of the result of a projection is determined by the fields that are projected in the obvious way.
- Suppose that we want to find out only the ages of sailors.
- The expression

$$\pi_{age}(S2)$$

evaluates to the relation shown in Figure2.19.

age
40
35

Figure 2.19 $\pi_{age}(S2)$

2.4.2 Set Operations

The following standard operations on sets are also available in relational algebra:

Union (U),
intersection (∩),
set-difference (−),
cross-product (×).

Union:

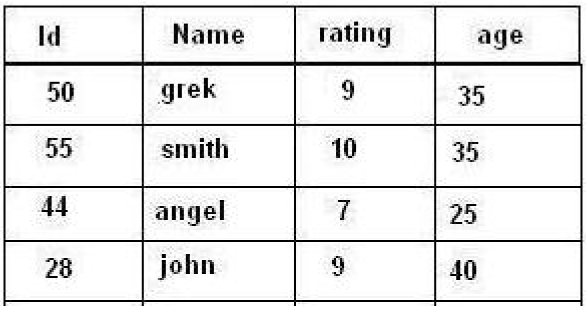
- The union is one of the binary set operation.
- The union operation produces a relation that includes all the tuples in relation R or relation S or it contains all the tuples in both relations R and S.
- Suppose R and S are the two relations, then "S U R " denotes union operation between two relations.
- The union symbol (U) is used for this purpose.
- Thus R U S returns a relation instance containing all tuples that occur in either relation instance R or relation instance S or both R and S.

R and S must be union compatible, and the schema of the result is defined to be identical to the schema of R.

- Two relation instances are said to be union-compatible if the following conditions hold:

  – They have the same number of the fields (attributes are referred here as field), and
  – Corresponding fields, taken in order from left to right, have the same domains

- **Example:**
  - The union of S1 and S2 is shown in Figure2.20.
  - Fields are listed in order.
  - field names are also inherited from S1.
  - S2 has the same field names, of course, since it is also an instance of Sailors.
  - In general, fields of S2 may have different names; recall that we require only domains to match.
  - Note that the result is a set of tuples. Tuples that appear in both S1 and S2 appear only once in S1 U S2.



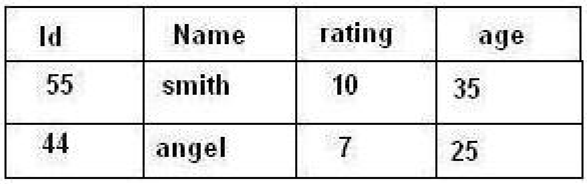
Id	Name	rating	age
50	grek	9	35
55	smith	10	35
44	angel	7	25
28	john	9	40

Figure 2.20 **S1 U S2.**

Intersection:

- S1 ∩ S2 returns a relation instance containing all tuples that are present in both S1 and S2.
- The relations S1 and S2 must be union-compatible, and the schema of the result is defined to be identical to the schema of S1.

Example:
The intersection of S1and S2 is shown in Figure2.21.



Id	Name	rating	age
55	smith	10	35
44	angel	7	25

Figure 2.21 S1 ∩ S2

Set-difference:

- R−S returns a relation instance containing all tuples that occur in R but not in S.
- The relations R and S must be union-compatible, and the schema of the result is defined to be identical to the schema of R.

Example:

- The set-difference S1−S2 is shown in Figure 2.22.

Id	Name	rating	age
50	grek	9	35

Figure 2.22 S1−S2

Cross-product:

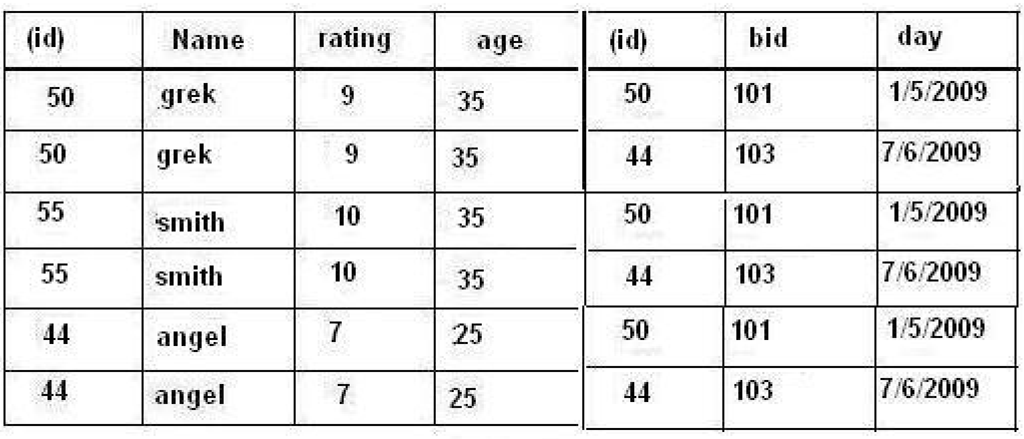
- S1×S2 returns a relation instance whose schema contains all the fields of S1 (in the same order as they appear in S1) followed by all the fields of S2 (in the same order as they appear in S2).
- The cross-product operation is sometimes called Cartesian product.
- We will use the convention that the fields of S1×S2 inherit names from the corresponding fields of S1 and S2.
- It is possible for both S1 and S2 to contain one or more fields having the same name; this situation creates a naming conflict.
- The corresponding fields in S1×S2 are unnamed and are referred to solely by position.

▪ The result of the cross-product S1 × R1 is shown in Figure2.24.

▪ Because R1and S1 both have a field named id, by our convention on field names, the corresponding two fields in S1×R1 are unnamed, and referred to solely by the position in which they appear in Figure2.24.
▪ The fields in S1×R1 have the same domains as the corresponding fields in R1 and S1.
▪ In Figure2.21 id is listed in parentheses to emphasize that it is not an inherited field name; only the corresponding domain is inherited.

Id	bid	day
50	101	1/5/2009
44	103	7/6/2009

Figure 2.23 **instance R1 of reserves**



(id)	Name	rating	age	(id)	bid	day
50	grek	9	35	50	101	1/5/2009
50	grek	9	35	44	103	7/6/2009
55	smith	10	35	50	101	1/5/2009
55	smith	10	35	44	103	7/6/2009
44	angel	7	25	50	101	1/5/2009
44	angel	7	25	44	103	7/6/2009

Figure 2.24 **S1 × R1**

2.4.3 Joins

- The join operation is one of the most useful operations in relational algebra and is the most commonly used way to combine information from two or more relations.
- The join operation, as the name suggests, allows the combining of two relations to form a single new relation.
- It is denoted by symbol $\bowtie$.

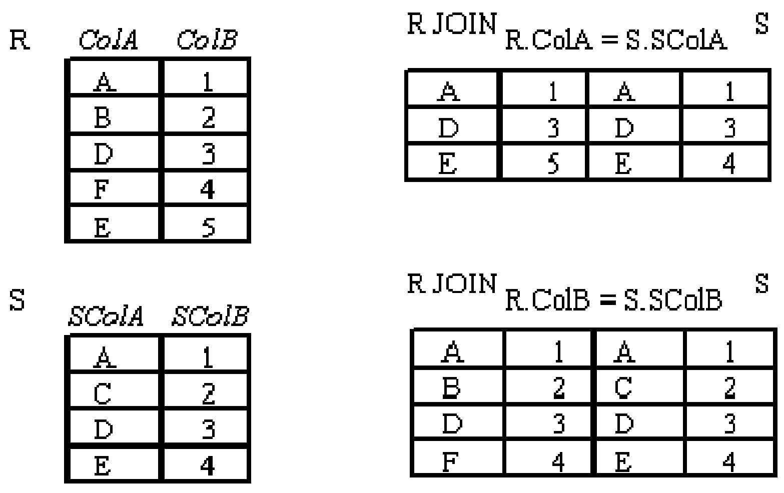
- The join is very important for any relational database with more than a single relation, because it allows us to process relationships among relations.
- In its simplest form the JOIN operator is just the cross product of the two relations.
- As the join becomes more complex, tuples are removed within the cross product to make the result of the join more meaningful.
- JOIN allows you to evaluate a join condition between the attributes of the relations on which the join is undertaken.

The notation used is

$$R\ \text{JOIN}_{\text{join condition}}\ S$$

- Where R and S are two relations
- Example:

Let consider below two relations (tables) R and S.



R

ColA	*ColB*
A	1
B	2
D	3
F	4
E	5

R JOIN $_{R.ColA = S.SColA}$ S

A	1	A	1
D	3	D	3
E	5	E	4

S

SColA	*SColB*
A	1
C	2
D	3
E	4

R JOIN $_{R.ColB = S.SColB}$ S

A	1	A	1
B	2	C	2
D	3	D	3
F	4	E	4

- There is one difference between JOIN operation and CARTESIAN PRODUCT that, JOIN is only combination of tuples satisfying the join condition (i.e. they should be equal on their common attribute names) appear in the result where as CARTESIAN PRODUCT takes all combination of tuples that are included in result.

- **Natural join:**
- Natural join (⋈) is a binary operator that is written as (R ⋈ S) where R and S are relations.
- The result of the natural join is the set of all combinations of tuples in R and S that are equal on their common attribute names.
- For an example consider the tables Employee and Dept and their natural join:

Name	**Id**	**Dept**
Archana	123	Finance
Pramod	124	Sales
Pooja	125	Finance
Vivek	126	Sales

Figure 2.25: Relation for “Employee”

Dept	**Manager**
Finance	Shah
Sales	Verma
Production	Mitali

Figure 2.26: Relation for “Department”

Name	**Id**	**Dept**	**Manager**
Archana	123	Finance	Shah
Pramod	124	Sales	Verma
Pooja	125	Finance	Shah
Vivek	126	Sales	Verma

Figure 2.27: Employee ⋈ Department

The relation Employee contains name, id, and department of the employee. And the relation “Department” contains name of departments and name of manager of that particular department. Then we performed natural join operation on these two relations and the resulting relation is given in Figure 2.27 Employee ⋈ Department. In "Employee" relation the first record contains name "Archana" whose id is 123 and she belongs to the department "Finance". And in the relation "Department" the first entry (i.e. dept name) is Finance and the manager of Finance is Shah. Hence when we perform natural join then in **Employee ⋈ Department** we will have Archana whose id is 123 and dept is Finance. And her manager is Shah.

- Invariably the JOIN involves an equality test, and thus is often described as an equi-join. Such joins result in two attributes in the resulting relation having exactly the same value.
- A `natural join' will remove the duplicate attribute(s).

There are different types of Join Operations such as Outer Join, Inner Join etc that we will cover in Chapter "SQL"

Summary:

➢ Three types of data models are available:

1. Object-based Logical Models: This model describes data at the conceptual and view levels. The data models come under this model are-
    - The E-R Model
    - The Object-Oriented Model

2. Record-based Logical Models: Record based logical model describe data at the conceptual and view levels. The data models come under this model are-
    i. The Relational Model
    ii. The Network Model
    iii. The Hierarchical Model

3. Physical Data Models: This model is used to describe data at the lowest level.

➢ **Integrity constraints**

An integrity constraint is a condition that is enforced automatically by the DBMS and that prevents the data from being stored in the database.

- NOT NULL Integrity Constraints: A NOT NULL constraint requires a column of a table contain no null values.
- Entity integrity: Entity Integrity ensures that for each row in a table, the value of the primary key is unique and is not null.

Referential Integrity or foreign key constraint: the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.

4. Unique Key Constraint: A UNIQUE key integrity constraint states that every value in a column or set of columns (key) is unique—that is, no two rows of a table have duplicate values in a specified column or set of columns

➢ **Relational algebra operations**

- **Selection operation:** The Select operation is used to select a particular column from a table. That means the select operation selects tuples that satisfy a given predicate (condition).

- **Projection Operation:** The projection operator π allows us to extract desired columns from a relation and remaining columns are left out.
- **Set Operations**
    - ❖ Union (U),
    - ❖ Intersection (∩),
    - ❖ Set-difference (−),
    - ❖ Cross-product (×).
- **Joins: Join operation** is the most commonly used to combine information from two or more relations.

Multiple Choice Questions

1. Which of the followings is not a type of data model?
    a. Object Oriented data Model
    b. Static Data Model
    c. Network Model
    d. Relational Data Model

2. In an an E-R diagram rectangle represents:
    a. Entity Set
    b. Attributes
    c. Relationship
    d. None of the above

3. In an E-R diagram Ellipse represents:
    a. Entity Set
    b. Attributes
    c. Relationship
    d. None of the above

4. The number of entities to which another entity can be associated via a relationship set is called...
    a. The mapping cardinalities
    b. The entity Relationship
    c. Integrity Constraint
    d. None of the above

5. In which of the following model database is represented graphically.
    a. Relational Model
    b. Object oriented data Model
    c. Entity Relationship Model
    d. Physical Model

6. Which of the followings is not an Entity Integrity Constraint.
    a. Not Null constraint
    b. Foreign key Constraint
    c. Unique Key Constraint
    d. Prime Constraint

7. Selection Operation in relational algebra is denoted by a symbol

   a. σ
   b. π
   c. >
   d. <!

8. Which of the followings is not a set operation

   a. Union
   b. Intersection
   c. Cross Product
   d. Join

9. A join operation is dented by the symbol

   a. ∑
   b. ∞
   c. Δ
   d. ⋈

10. S1 ∩ S2 returns-

a. A relation instance containing all tuples that are present in both S1 and S2.
b. A relation instance containing all tuples that are not present in S1.
c. A relation instance containing all tuples that are not present in S2.
d. A relation instance containing all tuples that are not present in both S1 and S2.

****

Chapter 3

Structured Query Language (SQL)

Topics To Learn

3.0. Introduction
3.1. SQL Statements
3.2. Data Types in SQL
3.3.Basic Query structure
3.4.SELECT
3.5.CREATE
3.6.INSERT
3.7 UPDATE
3.8 DELETE
3.9.ALTER
3.10.Data Control Language (DCL)
3.11.Different operations on tables
3.12.Null Values

Overview: -

In any organization we need to maintain data related to our organization in systematic manner, so it can be maintained easily using SQL (Structural Query Language). For managing, updating, accessing and to perform various operations on databases we required languages, which will maintain all theses functionality so SQL was developed. Fundamentally SQL is collection of a number of different elements: queries, statements, expressions, predicates, and clauses .By using different statements, expression, constraints, queries we can easily preserve & perform operations. It is very effective & easy by using SQL.

In this chapter we learn many things related to deletion, retrieval & modification of data, how to create, delete tables, how to define constraints on tables, how constraints helps for restricting data, how joins of multiple table is useful etc. We will see how SQL helps in managing and accessing of data efficiently. Let's start with introduction to SQL.

3.0. Introduction: -

Structured Query Language (SQL) is standardized by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). SQL consist queries, Expression, Statements, which required for giving instructions to the database. Let's see how SQL works.

How SQL Works: -

SQL is helpful for all types of users, including application programmers, database administrators, managers and end-users. It provides an interface to relational database. All SQL statements are like instructions to the database. Query is used to give instruction to database.

Example: -Suppose you want to filter the rows and for that purpose you use condition for filtering. All rows satisfying the condition are retrieving in a single step. All SQL statements use the optimizer. Optimizer it is component of database management system which consider query plans for a given input query, and try to decide which of those plans will be the better for accessing the specified data.

SQL use for: -
- Querying data
- Inserting, updating, and deleting rows in a table
- Creating, replacing, altering, and dropping objects
- Controlling access to the database and its objects
- Provide database consistency and integrity

3.1. SQL Statements: -

1. Database Manipulation Language (DML): -

When we want to retrieve, store, modify, delete, insert and update data in database we use DML Statements. DML statements are:
- SELECT
- INSERT
- UPDATE
- DELETE

2. Data Definition Language (DDL): -

When we want to create & modify structure of objects in a database, we use DDL Statements. DDL statements are:
- CREATE
- ALTER
- DROP

3. Data Control Language (DCL): -

DCL statements are used for database administration.
- GRANT
- DENY (SQL Server Only)
- REVOKE

3.2. Data Types in SQL: -

Definition: - A data type is a set of data with values having predefined characteristics.

A data type is a type of data. In SQL a wide range of data types is present. By using different data types we can store data in different forms, including character strings, numbers, file and date.
Example: -For name of student, we may use a string data type for the student's first and last name.
Following table shows data types with their syntax and explanation .

Data Type	**Syntax**	**Storage Size**	**Explanation**
Integer	integer	4 bytes	Integer numerical
Numeric	numeric (p, s)	5-17 bytes	Where p is a precision value; s is a scale value.
decimal	decimal (p,s)	5-17 bytes	Where p is a precision value; s is a scale value.
float	float(p)	4 Or 8 bytes	Where p is a precision value.

char(n)	char(n)	Max 8,000 Char	Fixed-length character string.
varchar(n)	varchar(n)	Max 8,000 Char	Variable-length character string.
text	text	Max 2GB of text data	Variable-length character string.
bit	bit(x)		Allows 0, 1, or NULL
date	date	8 bytes	Stores year, month, and day values.
datetime	datetime	8 bytes	Store Date & Time in numeric format Eg. 2007-10-08 12:35:30.123
Time	time	3-5 bytes	Stores the hour, minute, and second values.
Image	Binary Value	Maximum 2GB	Image Variables store up to 2 gigabytes of data and used to store any type of data file (not just images).
Currency	Currency	8 bytes	Use for currency.
AutoNumber	AutoNumber	4 bytes	AutoNumber fields mechanically give each record its own number, usually starting at 1
Ole Object	Ole Object	Up to 1GB	Can store pictures, audio, video, or other BLOBs (Binary Large Objects)

3.3. Basic Query structure: -

SQL is not case sensitive language. Like other languages all reserved words are written in all capital letters.

SQL Query is divided into 3 parts: - Select, From & Where.

- In Select part, we use to select attributes of tables i.e. Columns.
- In From part, we use table's name from which we want to select data.
- In where part we apply condition for filter the data from tables.

Syntax:

Select A1, A2…An From R1…Rn Where C	In Above Query: - {R1, R2...Rn} –Tuples, {A1, A2...An} –Tuples, C-Condition

In select we can use asterisk (*) for selection of all rows from tables.

Example: -

```
SELECT RollNo, Studname, Class
FROM Student WHERE Studname='paresh'
```

Above query obtain only those rows where student name is paresh.

In general, attributes are referenced in the form **R.A Means Relation. Attributes**

We can write above query as

SELECT stud. RollNo, stud.studname, stud. class
FROM Student stud WHERE stud.studname='paresh'

3.4. SELECT: -

It is used to retrieve data from an SQL database. This process is also known as a query. The asterisk (*) can be used to SELECT all columns of a table. The SQL SELECT command is used as follows

Syntax: -SELECT <ColumnList> FROM <TableName>

Example: -Consider table Student. The following example retrieves the columns RollNo and Name from all rows of the Student table.

RollNo	Name
101	Sunita
201	Ramesh
202	Poonam
203	Trupta

Figure.3.4. Student Table

SELECT RollNo, Name from Student
Output of Query: - all records of the Student table

Similarly if we use asterisk Symbol in place of column names then query will retrieve all rows, columns from Student table.

SELECT * From Student

Output of Query: - all records of the Student table.

3.4.1. SQL SELECT DISTINCT Statement

Sometimes some columns in tables have duplicate values and we want to avoid duplication .We want to access distinct values, at that time we use Select Distinct statement. The DISTINCT keyword can be used to return only **distinct** values.

Syntax: - SELECT DISTINCT column_name(s) FROM table name

Example: - The "Students" table: -

RollNo	Name	Address	City	Fees
1	Suresh	Cidco	Nashik	1000
2	Ramesh	Bytco	Nashik	2500
3	Kavita	Shivaginagar	Pune	2500

4	Ravi	Pune Station	Pune	1000
5	Nayana	Nashik Road	Nashik	2600
6	Sachin	Juhu	Mumbai	2600

Figure.3.4.1. Students Table

In above table, values for ‘City’ columns are duplicated, so if we want to select distinct city we used query as below: -

SELECT DISTINCT City From Students
Output of Query is as follows: -

City
Nashik
Pune
Mumbai

3.4.2. SQL Where Clause: -

We have seen that where clause is part of SQL select Statement, which is used to state any condition for filtering data access. We use where clause when we want to access certain rows within a table. We can use where clause if we want to update, delete based on certain condition or only for fetching certain data that satisfy our given conditions.

The SQL WHERE clause is used as follows.

SELECT * From TableName WHERE <Condition>

We can use comparison operator(s) in condition of **where** clause. List of Comparison operators that are use in where is given below:-

Operator	**Description**
=	Equal
<>	Not Equal
>	Greater Than
<	Less Than
>=	Greater Than Or Equal
<=	Less Than Or Equal

Example: -Select all rows from Student table given in Figure. 3.4. Where Student Name is Trupta

Select * From Student Where Name=’ Trupta’

Output of Query is as follows: -

RollNo	Name
203	Trupta

3.4.3. Where Clause with Logical Operator: -

And, OR, In, Between, Like these are logical operators whoch we can use with Where clause.

1. And, OR Operator:

In Where we can use condition but for specify multiple criteria & multiple condition we used And, Or, In, Between, Like Operator.

The syntax for a compound condition is as follows: -

```
SELECT "column_name" FROM "table_name"
WHERE "Condition" {[AND|OR] "Condition"}+
```

- In case of AND operator both condition must be satisfy. If out of two, one condition is not satisfy then query didn't retrieve any row.
- In case of OR operator if one condition is satisfy out of both condition, query retrieves all rows for which condition is satisfy.

Example: -Select all rows from Student table given in Figure. 3.4. Where RollNo greater than equal to 101 and less than 201.

Query: -

```
Select * From Students
Where RollNo>=101 and RollNo<202
```

Output of Query is as follows: -

RollNo	Name
101	Sunita
201	Ramesh

Select all rows from Student table given in Figure. 3.4. Where Name is Sunita or Poonam.

```
Select * From Student
Where Name=' Sunita' Or Name =' Poonam'
```

Output of Query is as follows: -

RollNo	Name
101	Sunita
202	Poonam

2. In Operator: -

If you want to specify multiple values in a WHERE Clause, we used In operator. If you want to exclude the rows in your list, you used NOT IN at the place of IN operator

The syntax for using the IN keyword is as follows: -

```
SELECT "column_name" FROM "table_name"
WHERE "column_name" IN ('value1', ‘value2’...)
```

You can use one or more values in the parenthesis in where clause of select query. Each value is separated by comma. We can use numerical or characters values. If only one value is present inside the parenthesis, it is equivalent to check for only one value as shown below.

```
WHERE "column_name" = 'value1'
```

Example: -Select all rows from Student table given in Figure. 3.4. Where Name is Sunita, Poonam.

```
Select * from Student
Where Name In (‘Sunita’, ‘Poonam’)
```

Output of Query is as follows: -

RollNo	Name
101	Sunita
202	Poonam

3. BETWEEN Operator: -

In Operator help for selection from one or more distinct values, but if we want collection from a range, we used BETWEEN operator. It used to verify whether we access data between the values range we mention.

The syntax for the BETWEEN clause is as follows: -

```
SELECT "column_name" FROM "table_name"
WHERE "column_name" BETWEEN 'value1' AND 'value2'
```

This will select all rows whose column has a value between 'value1' and 'value2'.

Example: - Select all rows from Student table given in Figure. 3.4 where RollNo is between 101 and 202.

```
Select * from Student
Where RollNo Between 101 And 202
```

Output of Query is as follows: -

RollNo	Name
101	Sunita
201	Ramesh
202	Poonam

4. LIKE Operator: -

Sometimes we want to access data from tables that matches specified pattern in columns, in that case we use like operator. It is a method to check for matching strings. We can use wildcards ahead of the pattern as well as after the pattern.

Two Wildcards are used (%) & (_).

- The Percent Sign (%) is used to match zero or more characters.
- Underscore (_) used to match a single character.

SQL LIKE Syntax: -

SELECT column_name(s) FROM table_name
WHERE column_name LIKE **pattern**

When we want to match any single character within the particular range or set we use wild character []. We can identify range in square brackets []. If we want to use opposite condition means any single character not within the specified range or set we use [^] .

Example: - [a-f] any single character within the specified range
([^a-f]) any single character not within the specified range [^abcdef]).

Example: -Consider Students Table as show below. Select Students Name that is starts with 'S' from Students table.

FName	LName	Address	City
Ramesh	Pawar	Cidco	Nashik
Sachin	Patil	Dadar	Mumbai
Nayana	Shah	CBS	Nashik
Zakir	Hussein	CBS	Nashik
Kavita	Jain	ShivajiNagar	Pune
Ravi	Mishra	ShivajiNagar	Pune

SELECT * FROM Students WHERE FName LIKE 'S%

Output of Query is as follows: -

FName	LName	Address	City
Sachin	Patil	Dadar	Mumbai

It is also possible to select the Students living in the city that NOT contains the pattern "**shik**" from the "Students" table, by using the NOT keyword.
SELECT * FROM Students WHERE City NOT LIKE '%shik%'

Output of Query is as follows: -

FName	LName	Address	City
Sachin	Patil	Dadar	Mumbai
Kavita	Jain	ShivajiNagar	Pune
Ravi	Mishra	ShivajiNagar	Pune

3.4.4. SQL Order by Keyword: -

Sometime we want to sort the result-set by a specified column, at that time we used ORDER BY keyword. In Order By Keyword by default it sort the records in ascending order. If you want to sort in a descending order, you can use the **DESC** keyword. It gives choices for sorting result set ascending or descending.

SQL ORDER BY Syntax: -

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

Example: -Consider Students Table as shown in Figure.3.4.4.

Sort the Students by their City.

RollNo	Name	Address	City	Fees
1	Suresh	Cidco	Nashik	1000
2	Ramesh	Bytco	Nashik	2500
3	Kavita	Shivaginagar	Pune	2500
4	Ravi	Pune Station	Pune	1000
5	Nayana	Nashik Road	Nashik	2600
6	Sachin	Juhu	Mumbai	2600

Figure.3.4.4. Student Table

Query: -Select Name, City from Students Order By City
Output of Query is as follows: -

Name	City
Sachin	Mumbai
Nayana	Nashik
Suresh	Nashik
Ramesh	Nashik
Kavita	Pune
Ravi	Pune

Example: -Consider Students Table as shown in Figure.3.4.4. Sort the Students by their City descending.

Query:-SELECT * FROM Students ORDER BY City DESC
Output of Query is as follows: -

Name	City
Kavita	Pune
Ravi	Pune
Ramesh	Nashik

Nayana	Nashik
Suresh	Nashik
Sachin	Mumbai

3.4.5. Aggregate Functions: -

Sometimes we need summarization of large volumes of data, at that time aggregate functions are used. SQL aggregate functions always return a single value that is calculated from values in a column. When mathematical operations must be performed on all or a group of values aggregate functions is useful.

If a group of values is needed also include the **GROUP BY** clause. In SQL column name or expression is use as the parameter to the Aggregate Function. The parameter '*' represents all rows.

Syntax: -SELECT <column_name1>, <column_name2>
<Aggregate function(s)> FROM <table_name>

Useful aggregate functions:

- AVG () - Returns the average value
- COUNT () - Returns the number of rows
- FIRST () - Returns the first value
- LAST () - Returns the last value
- MAX () - Returns the largest value
- MIN () - Returns the smallest value
- SUM () - Returns the sum
- NOW () -Returns the current system date and time.
- ROUND () -Used to round a numeric field to the number of decimals specified.
- FORMAT () -Used to format how a field is to be displayed.

Example: -Consider Table Student as follows:

RollNo	Name	Amount	Percentage
1021	Sachin	15000	55.50
1022	Sandesh	20000	73.20
1023	Kavita	15000	75.00
1024	Ravi	25000	60.80
1025	Kunal	18000	63.45

Figure.3.4.5. Student

Select Count (Amount) From Student
Or Output: - 5
Select Count (*) from Student

Select First (Amount) From Student Output: - 15000

Select Last (Amount) From Student Output: - 18000

Select Max (Amount) From Student Output: - 25000

Select Min (Amount) From Student Output: - 15000

Syntax: -Now ()
SELECT NOW () FROM table_name

Example: -
SELECT RollNo, Amount, Now () as Date
FROM Student where RollNo=1021
Output: -

RollNo	Amount	Date
1021	15000	20/1/2010 1:35:02 AM

Syntax: -Round ()
SELECT ROUND (column_name, decimals) FROM table_name

Example: -
SELECT RollNo, ROUND (Percentage, 0) as Percentage
FROM Student Where RollNo=1021

Output: -

RollNo	Percentage
1021	56

Syntax: -Format ()
SELECT FORMAT (column_name, format) FROM table_name

Example: -
SELECT RollNo, Amount, FORMAT (Now (),'YYYY-MM-DD') as Date
From Student Where RollNo =1021

Output: -

RollNo	Amount	Date
1021	150000	2010-01-20

3.4.6. Group By: -

When we want grouping the result dataset by certain database table columns ,we use GROUP BY statement along with the SQL aggregate functions.

Syntax: - SELECT ColumnName, aggregate function (ColumnName)
FROM table WHERE ColumnName operator value
GROUP BY ColumnName

Example: -Consider Following table Figure. 3.4.6 if we want to select student records whose fees is maximum among all students & group student by there name, we use group by statement as follows.

ChallenNo	Fees	Date	Studname
1021	10000	20/1/2010	Ramesh
1022	3000	21/1/2010	Suresh
1023	18000	18/1/2010	Deepak
1022	10000	21/1/2010	Suresh
1025	2000	20/1/2010	Ramesh

Figure. 3.4.6 Fees Table

Query: -

```
Select Studname, max(fees) as Maxfees
from Fees where fees >10000  group by Studname
```

After Group by Studname we will get result as follows: -

Studname	Fees
Deepak	18000

3.4.7.Having Clause: -

We can't use where clause with aggregate functions so SQL HAVING clause is used to perform action on groups. It is used with the SELECT clause to specify a search condition for a group or aggregate. Having clause is applicable only for groups whereas where is limited to individual rows, not for groups.

Syntax: -

```
SELECT column1, column2, ... column_n, aggregate_function (expression)
FROM tables WHERE predicates
GROUP BY column1, column2, ... column_n
HAVING condition1 ... condition_n;
```

Aggregate function can be a function such as SUM, COUNT, MIN, or MAX.

Example: -Consider following table Feeinfo, if we want to retrieve total count of record where Studname is Suresh & check count > 1 or not. i.e. more than one record having Studname as 'Suresh' is exist .
In such case we will use Having clause as follows:

ChallenNo	Fees	Date	Studname
1021	10000	20/1/2010	Ramesh
1022	3000	21/1/2010	Suresh
1023	18000	18/1/2010	Deepak
1022	10000	21/1/2010	Suresh
1025	2000	20/1/2010	Ramesh

Figure.3.4.7 FeeInfo table

Query :-SELECT Studname, count (*) as "Total count of name"
```
FROM FeeInfo
WHERE Studname ='Suresh'
group by studname
HAVING count (*)>1
```

Output of Query as follow: -

Studname	Total count of name
Suresh	2

3.5. CREATE: -

Data is stored in basic structure like Table in SQL. For creating new table in SQL we used Create Table Statements, it adds a new table to an SQL database. Tables consist of rows and columns. Create Table have following

Syntax: -
```
CREATE TABLE "table_name"
("column 1" "data_type_for_column_1",
"column 2" "data_type_for_column_2",...)
```

Example: - if we have a table for recording Students information, then the columns may include information such as RollNo, Name, Address, City, Birth Date, and so on. We use create statement as follows:-

Example: -CREATE TABLE Students

```
CREATE TABLE Students
        (RollNo Number (2),
        Name char (50),
        Address char (50),
        City char (50),
        Birth_Date date)
```

3.5.1. DROP: -Drop Table: -

Drop table remove the table(s) from the database.

Syntax: -DROP TABLE [IF EXISTS] table_name1, table_name2, ...
Example: -Drop Students table

Drop Table Students

If we want to check that before dropping any table it is exist or not we use IF EXISTS clause .This clause will drop the table only if it exists. If the table does not exist, it will create error.

Example: -DROP TABLE IF EXISTS Student

Before actually dropping student table it check for existences of student table.

3.5.2. Constraints: -

In general Constraints are like rules. Sometime we want to restrict for doing some operation on database so constraint are used to enforce the integrity between columns and tables & for controlling values allowed in columns. By using constraint we can avoid user from inserting certain types of mismatched data values in the column(s). Constraint ensures correctness and reliability of the data in the database.

Different type of the data integrity: -

- Entity Integrity
- Domain Integrity
- Referential integrity
- User-Defined Integrity

Entity Integrity: When we want to check for no duplicate rows exist in a table we used entity integrity.

Domain Integrity: -It allows only legal entries for a specified column by check on type, format, or the range of possible values.

Referential integrity: -it checks that rows cannot be deleted, which are used by other records.

User-Defined Integrity: -When we want to check that various specific business rules that must be follow, no one can crash that rules at that time we use user defined integrity. For enforce these categories of the data integrity we used appropriate constraints.

SQL supports the following constraints:

- ❑ PRIMARY KEY
- ❑ UNIQUE
- ❑ FOREIGN KEY
- ❑ CHECK
- ❑ NOT NULL

1. Not Null: -

Many times user didn't enter data for field in table and we want to check column not to accept NULL values, we used NOT NULL constraint. NOT NULL Constraint enforces the field to accept a value. We Specify Not Null for Column where we want that user must enter data for that field. This means that you cannot insert a new record, or update a record without adding a value to this field.

Example: -Following Example shows Student table with Not Null Constraint. Here user can't enter RollNo as Null. We restrict user to enter Students RollNo.i.e. RollNo is required for our tables.

```
CREATE TABLE Student
        (RollNo int NOT NULL,
        Name varchar (255),
         Address varchar (255))
```

2. Primary Key: -

A primary key is used to uniquely identify each row in a table. It can be used either when the table is created using create table statement or by changing the existing table structure using alter table. It consists of one or more fields on a table. When multiple fields are used as a primary key, they are called a composite key.

Example: -

```
CREATE TABLE Student
(RollNo integer PRIMARY KEY,
FirstName varchar (30),
LastName varchar (30));
```

3.Foreign Key: -

A FOREIGN KEY in one table point to a PRIMARY KEY in another table. It is used to make sure referential integrity of the data. When we want to generate foreign key we must have two tables one is parent table & child table. Both tables have relationship with each other based on common column. A common column in both tables relates parent table & child table.

Syntax: -

```
CREATE TABLE ChildTableName
(Column_Name Size,
Foreign Key (Common Key)
references ParentTableName (Common_Key));
```

Example: -Consider two tables Student & FeeInfo having RollNo as a common Field. As shown in following example student has RollNo as Primary Key which have references to RollNo field of FeeInfo table .In FeeInfo table ChallenNo is Primary Key & RollNo is Foreign Key.

```
CREATE TABLE Student
(RollNo integer PRIMARY KEY,
First_Name varchar (30),
Last_Name varchar (30));

CREATE TABLE FeeInfo
(ChallenNo integer PRIMARY KEY,
Studname varchar (30),
Date date,
Amount Numeric (5),
RollNo integer references Student (RollNo));
Or
CREATE TABLE FeeInfo
(ChallenNo integer,
Studname varchar (30),
Date date,
Amount Numeric (5),
RollNo integer,
Primary Key (ChallenNo),
Foreign Key (RollNo) references Student (RollNo))
```

4. UNIQUE constraint: -

It used to check the uniqueness of the values in a set of columns, so no duplicate values are entered. It is used to enforce primary key constraints.

One Major difference between Unique & Primary Key Constraint is that primary key & unique both can't allow repeated value, but in case of primary key we can't enter Null for that column, while in unique constraint we can enter Null Value.

Example: -

```
CREATE TABLE DEPARTMENT
(DEPTNO integer UNIQUE,
DEPTName varchar (30))
```

Above Query creates unique constraint on DEPTNO. Means DEPTNO must be unique. No duplicates are allowed.

5. CHECK constraint: -

A CHECK constraint is used to limit the values that can be placed in a column. It is used to enforce domain integrity.

Example: - The following example creates a checkFee CHECK constraint on a Students table to check that Fees >1500 or not. We can disable & enable the check constraint with alter statement.

```
CREATE TABLE Students (RollNo INT NOT NULL,
Name VARCHAR (30) NOT NULL,
Address VARCHAR (100),
Fees Numeric (5),
Fees NOT NULL CONSTRAINT checkFee CHECK (Fees > 1500))
```

For disable constraint we used NOCHECK & for enabling Check constraint we used CHECK CONSTRAINT

Example: -Disable the checkFee constraint in the Students table

```
ALTER TABLE Students NOCHECK CONSTRAINT checkFee
```

Example: -Enable the checkFee constraint in the Students table

```
ALTER TABLE Students CHECK CONSTRAINT checkFee
```

3.6. INSERT: -

When we want to insert records into a table in the database we used INSERT Command. This statement comes in three forms.

The first inserts data from a VALUES clause.

Syntax: -

```
INSERT INTO table_name [(Col1,Col2,....)]
 VALUES (Expression1,Experssion2, .... )
```

Example of first forms: -Consider Students table in Figure. 3.4. Insert RollNo 105 & name Satish in it.

Query as follows: -
INSERT INTO Students (RollNo, Name) VALUES (105,'Satish')

The second form is used to copy information from a SELECT query into the table specified in the INSERT statement.

```
Syntax: -
     INSERT INTO table_name1 [(Col1, Col2, .... )]
     SELECT Col1, Col2, ... ColN From Table_Name2
```

Example of second form:-Insert records into Students from StudInfo table where Students Name is Soham.

INSERT INTO **Students** (RollNo, Name)
SELECT RollNo, Name FROM **StudInfo** WHERE Name = 'Soham'

The third form uses a list of column SET assignments.

```
Syntax: -
     INSERT INTO table_name
     SET Col1 = expression1, Col2 = expression2, ....
```

Example:-Insert RollNo 205 into student's tables where students name is Satish .

INSERT INTO Students SET RollNo=205 where Name='Satish'

If a column of the table is not specified in an INSERT the default value declared for the column is used. If no default value was declared a NULL value is inserted in the column. If the column is declared as NOT NULL the insert operation fails.

3.7. UPDATE: -

When we want to update information in a table we use update query. It is used to set multiple expression value to multiple columns also used to update query with where clause for obtaining conditionality in query.

```
Syntax: - UPDATE table_name
          SET Col1 = expression1, Col2 = expression2, ....
          [WHERE expression]
```

RollNo	Name	Address	Fees
1	Ramesh	Pune	12000
2	Trupta	Mumbai	15000
3	Nayana	Nashik	22000

Figure. 3.7 Students Table

Examples: -Consider table Students increases students 's Fees by 100.
UPDATE Students SET Fees = Fees+100

Examples: -update Students table set name to Sumit where Fess is 15000

```
UPDATE Students SET Name = 'Sumit'
WHERE Fees = 15000
```

3.8. DELETE: -

The DELETE statement is used to delete records in a table. By default delete Statement delete all records from table .By using where clause we can restrict query by using condition in it.

Syntax: -DELETE FROM table_name [WHERE expression]

Example: -Consider table in Figure.3.7 Delete record from Students table where student's name is Trupta and RollNo is 202

```
DELETE FROM Students
WHERE Name=Trupta' and RollNo=2
```

Output of above Query: -

RollNo	Name	Address	Fees
1	Ramesh	Pune	12000
3	Nayana	Nashik	22000

3.9. ALTER: -

Drop & ALTER Tables, Columns, and Constraints: -

Sometime it may happen that you create table and after some time you want to modify its column name, Size, data type OR you may add, remove columns of tables at that time you use Alter command. It used to add, remove and modify columns & constraints and also to drop columns from table or you can use to set default expression to default value of column.

Syntax: -ALTER TABLE table_name ADD [COLUMN] column declare

ALTER TABLE table_name ADD constraint declare

ALTER TABLE table_name DROP [COLUMN] column_name

ALTER TABLE table_name DROP CONSTRAINT constraint_name

ALTER TABLE table_name DROP PRIMARY KEY

ALTER TABLE table_name ALTER [COLUMN] column_name
SET default_expr

ALTER TABLE table_name ALTER [COLUMN] column_name DROP DEFAULT

- **ALTER Column size:-**

Example: -We want to alter Students table to change Name column such that varchar field Name should be converted to accept no of chars up to 40.

ALTER TABLE Students ALTER COLUMN Name (40)

- **ALTER Constraint:**

Example1: -Suppose you create student table but after creation you want to modify Students tables such that RollNo field should be primary key of table. We alter Students Table as Follow:-

ALTER TABLE Students ADD PRIMARY KEY (RollNo);

Example2:-If We created Employee table having Primary key & we created another table Emp having Field EmpId. But we want to make relation between these two tables based on their similar Field. So we give references of Parent Table to Child table EmpINFO. We alter structure of Child table as follow.

ALTER TABLE Emp ADD FOREIGN KEY (EmpId)
REFERENCES EmpInfo (EmpId);

SQL table columns can be altered and changed using the **MODIFY** COLUMN command.

Example2.For adding Check Constraint to students table

ALTER TABLE Students ADD CHECK (Fees>0)

- **ALTER Table & Drop Column: -**

Example: -we alter students table to drop Birth_Date column from table. Above Query will delete column Birth_Date from students Table

ALTER TABLE students DROP Birth_Date

- **ALTER Table & Drop Constraint: -**

Example1: -Consider we create Order Table having not null constraint for Order's Quantity & OrderID is Primary key for order table as follow:

```
Create Table Order
(OrderID int Constraint CK1_ OrderID Primary Key,
Quantity int Constraint CK2_Qty Not Null,
Order_Desc Varchar (30))
```

After creating this Order table if we want to delete Constraint for Order's Quantity we drop it as follow:

ALTER TABLE Order DROP CONSTRAINT CK2_Qty

Example2: - If you have check constraint and you want to drop Check Constraint for salary.

```
Create Table Employee
 (EmpId int Primary Key,
 Name Varchar (20),
 Salary CONSTRAINT chk_Salary  Check  (Salary>0))

ALTER TABLE Employee
DROP CONSTRAINT chk_Salary
```

3.10. Data Control Language (DCL):-

1. ROLE: -

SQL offer Create Role statement to create role. Role is set of privileges that can be granted to users or to other roles. For adding privileges to a role we use the GRANT statement. Privileges express the access rights that are provided to a user on a database object. When you firstly create any role, it is empty, after creating it you add privileges (rights) to a role.

There are two types of privileges.

- System privileges – It allows the user to CREATE, ALTER, or DROP database objects.
- Object privileges - It allow the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects.

2. DROP ROLE: -

To remove the specified role(s) we used DROP ROLE.

Syntax: -DROP ROLE [IF EXISTS] *name* [, ...]

Example: -For dropping Role Admin query is as follow

DROP ROLE Admins

3. GRANT: -

For offering access or rights on the database objects to the users, we use GRANT Statement.

Syntax: -

```
GRANT privilege_name
ON object_name To {user_name | Public role_name}
```

- Privilege_name :-It is the right or access right like ALL, EXECUTE, and SELECT granted to the user.
- Object_name :-It is the name of a database object like TABLE, VIEW, STORED PROC and SEQUENCE.
- user_name: -name of the user to whom an access right is being granted.
- PUBLIC: - used to grant access rights to all users.

- ROLES :-It is a set of privileges grouped together.

Example: -Grant inserts privilege to all users on table Students:

```
GRANT INSERT ON Students TO PUBLIC;
```

Above query will grant Insert operation for all users .So all users can insert in Student Table.

Example: -Grant all available privileges to user John on Students: -

```
GRANT ALL PRIVILEGES ON Students TO John
```

Above Query will assign all right to user john .So John user can do any operation like insert, update, and delete on student table.

4. REVOKE: -

To remove user access rights or privileges from the database objects, we use REVOKE command.

Syntax: -REVOKE privilege_name ON object_name

FROM {user_name | PUBLIC | role_name}

Example: Revoke SELECT privilege on Students.

```
REVOKE SELECT ON Students FROM user1
```

This command will REVOKE a SELECT privilege on Students table from user1. When you REVOKE SELECT privilege on a table from a user, the user will not be able to SELECT data from that table.

Example: -Revoke inserts privilege for the John on table Students:

```
REVOKE INSERT ON Students FROM John;
```

3.11. Different operations on tables

1. Rename: -

We can rename any table by using RENAME SQL command. If we use rename command, the data will not be lost only the table name will be changed to new name.

Syntax for renaming tables name is as follow: -

Syntax: -**Rename table Old_table_name to new_table_name**

Example: -we will change the name of our student table name to StudInfo table.
Rename Table Student To StudInfo

We can also change the name of column to new name as follows:

Syntax: -**ALTER TABLE table_name**
RENAME COLUMN old_name to new_name;

Example: -Change eno column name to empno.

```
ALTER TABLE employee RENAME COLUMN eno to empno;
```

2. Tuple Variables: -

Set of one or more attributes is known as tuples. Tuple variables are used in the SQL clause known as FROM.

3. Set Operations: -

SQL set operators allows to join (combine) results from two or more SELECT statements. It combines rows from different queries. The four set operators union, union all, intersect and minus allows to serially combine more than one select statement.

- **UNION Operator: -**

UNION combines the results of two SQL queries into a single table of all matching rows. The two queries must result in the same number of columns and compatible data types in order to unite.

Syntax: -

```
SELECT * FROM Table1
    UNION
SELECT * FROM Table2
```

Example: -Consider two table T1 & T2 as given below:

T1

RollNo	Name
1	Hussein
2	Jain
3	Sameer

T2

RollNo	Name
1	Sanjay
2	Ravi
3	Radha
4	Sumit
5	Poonam

Union of T1 & T2 as follow: -

```
SELECT RollNo, fName
FROM T1 Where RollNo in (1,2)
        UNION
SELECT RollNo, fName
FROM T2  Where RollNo <=2 ORDER BY Name1
```

Output of Query is as follows: -

RollNo	Name
1	Hussein
2	Jain
2	Ravi
1	Sanjay

- **UNION ALL Operator: -**
  The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL.

Syntax: -

```
SELECT Col1, Col2, Col3 From Table1
  Union ALL
SELECT Col1, Col2, Col3 From Table2
```

Example: -Consider two table T1 & T2 as given below:

RollNo	Name
01	Ravi
02	Vinod
03	Sudhir

T1

RollNo	Name
01	Jayesh
02	Deepak
03	Sudhir
04	Kalpesh

T2

Union ALL Of T1 & T2 as follow: -

```
Select ID, Name from T1
UNION ALL
Select ID, Name from T2
```

Output of Query is as follows: -

RollNo	Name
01	Ravi
02	Vinod
03	Sudhir
01	Jayesh
02	Deepak
03	Sudhir
04	Kalpesh

- **INTERSECT Operator: -**
  The Intersect operator takes the results of two queries and returns only rows that appear in both result sets. This operator does not distinguish between Nulls. It removes duplicate rows from the final result set. The INTERSECT ALL operator does not remove duplicate rows from the final result set.

Syntax:-

```
SELECT Col1, Col2, Col3 From Table1
   Intersect
SELECT Col1, Col2, Col3 From Table2
```

Example: -Consider two table T1 & T2 as given below:

Roll_No	Name
101	Sachin
102	Trupta
103	Nayan
104	Tanvi

T1

Roll_No	Name
101	Sachin
103	Nayan
105	Disha
106	Deepak

T2

Intersect of T1 & T2 as follow: -

```
Select * from T1
Intersect
Select * from T2
```

Output of Query is as follows: -

RollNo	Name
101	Sachin
103	Nayan

- **Minus Operator: -**

EXCEPT operator also called MINUS .It returns rows that appear in one input but not in the other, so it also eliminates duplicates. The EXCEPT ALL operator does not remove duplicates. The EXCEPT operator can't distinguish between Nulls in case if you are doing row removal and duplicate data removal from table.

Syntax: -SELECT Col1, Col2, Col3 From Table1
Minus
SELECT Col1, Col2, Col3 From Table2

Example: -Consider two table T1 & T2 as given below
Select all rows from table1 that is not present in table 2:

RollNo	Name
101	Sachin
102	Trupta
103	Nayan
104	Tanvi

T1

RollNo	Name
101	Sachin
103	Nayan
105	Disha
106	Deepak

T2

Query :- Select * from T1
Minus
Select * from T2

Output of Query is as follows: -

RollNo	Name
102	Trupta
104	Tanvi

4. String Operations: -

A string is a finite sequence of symbols that are chosen from a set or alphabet.

Example: -Computer, Nashik, 123string etc.

If string consist no characters then it is known as empty string. String functions are mainly used to modify the case of strings, concatenate strings, reverse strings, extract different parts of strings and perform lots of other types of string manipulation.

Assume we have the following table for string operation:

RollNo	Last_Name	First_Name	Address	City
1	Pawar	Ramesh	Bytco	Nashik
2	Patil	Sachin	Juhu	Mumbai
3	Shah	Nayana	Nashik Road	Nashik
4	Hussein	Zakir	Cidco	Nashik
5	Jain	Kavita	Shivaginagar	Pune
6	Mishra	Ravi	Pune Station	Pune

Figure.3.11 Students Table

1. **Substring: -**

The Substring function in SQL is used to grab a piece of the stored data.

• SUBSTR (string, position): Select all characters from < string > starting with position < position >. Note that this syntax is not supported in SQL Server.

• SUBSTR (string, position, length): This function will returns number of characters that mentioned in length parameter from specified position of the string mentioned in function.

Example: -SELECT SUBSTR (Last_Name, 3)
FROM Persons WHERE Last_Name = ' Hussein'

Output: -ssein

SELECT SUBSTR (Last_Name, 2,6)
FROM Students WHERE Last_Name = ' Mishra'

Output :-Ishra

2. TRIM: -

The TRIM function in SQL is used to remove particular prefix or suffix from a string. The most common pattern being removed is white spaces.

TRIM ([[LOCATION] [remstr] FROM] str): -

[LOCATION] can be either LEADING, TRAILING, or BOTH. This function gets rid of the [remstr] pattern from either the beginning of the string or the end of the string, or both. If no [remstr] is specified, white spaces are removed.

- LTRIM (str): Removes all white spaces from the beginning of the string.
- RTRIM (str): Removes all white spaces at the end of the string.

Example: -**SELECT TRIM (‘ Sample   ')**
Output: -Sample

Example: - SELECT LTRIM (‘   Sample')
Output: -Sample

Example: -SELECT RTRIM (‘Sample   ')
Output: -Sample

3.LENGTH: -

The Length function in SQL is used to get the length of a string.
Length (str): Find the length of the string *STR*.

```
Example: -SELECT Length (Last_Name)
            FROM Students
            WHERE Last_Name = ' Hussein’
Output: -7
```

4. REPLACE: -

Replaces all occurrences of the string2 in the string1 with string3.

Syntax - REPLACE (‘string1’, ‘string2’, ‘string3’)

```
Example: -SELECT REPLACE (‘Last_Name ‘),’til’,’tel’)
         FROM Students WHERE Last_Name = ' Patil
```

Output: -Patel

5. CONCATENATE: -

Sometimes it is required to combine together (concatenate) the results from several different fields.

CONCAT (str1, str2, str3,...): Concatenate str1, str2, str3, and any other strings together.

Example: -
```
SELECT CONCAT (Last_Name, First_Name)
FROM Students WHERE Last_Name = ' Hussein’
```

Output: -HusseinZakir

Example: -
```
SELECT Last_Name + ‘  ‘  + First_Name
   FROM Students WHERE Last_Name = ' Hussein’
```

Output: -Hussein Zakir

Example: -
```
SELECT Last_Name || ‘ ‘ || First_Name
  FROM Students   WHERE Last_Name = ' Hussein’
```

Output: -Hussein Zakir

6.LEFT: -

Returns left part of a string with the specified number of characters.

Syntax - LEFT (string, integer)

Example: -
```
SELECT LEFT (‘Last_Name ‘, 3)
  FROM Students WHERE Last_Name = ' Shah’
```
Output: - Sha

7. RIGHT: -

Returns right part of a string with the specified number of characters.

Syntax - RIGHT (string, integer)

Example: -
```
SELECT RIGHT (‘Last_Name ‘, 3)
   FROM Students WHERE Last_Name = ' Shah
```

Output: -hah

8. REPLICATE: -

Repeats string for a specified number of times.

Syntax - REPLICATE (string, integer)

Example: -
```
SELECT REPLICATE (‘Last_Name ‘, 2)
  FROM Students WHERE Last_Name = ' Shah’
```

Output: -ShahShah

9.REVERSE: -

Returns reverse of a string.

Syntax - REVERSE (string)

Example: -SELECT REVERSE (‘Last_Name ‘)
FROM Students WHERE Last_Name = ' Shah’

Output: -hahS

10.UPPER: -
Convert string to Uppercase.

Syntax - UPPER (string)

Example: -SELECT UPPER (‘Last_Name ‘)
FROM Students WHERE Last_Name = ' Shah’
Output: -SHAH

11.LOWER: -
Convert string to lowercase.

Syntax - LOWER (string)

Example: -SELECT LOWER (UPPER (‘Last_Name ‘))
FROM Students WHERE Last_Name = ' Shah’

Output: -Shan

12. UNICODE: -
Returns Unicode standard integer value.

Syntax - UNICODE (string)

Example: -SELECT UNICODE (‘Last_Name ‘))
FROM Students WHERE Last_Name = ' Shah’

Output: -83(it take First char Unicode)

13. STUFF: -
Deletes a specified length of characters and inserts string at a specified starting index.

Syntax- STUFF (string1, start index, length, string2)

Example: -SELECT STUFF (‘Last_Name’, 3, 3,’tel’)
FROM Students WHERE Last_Name = ' Patil

Output: -Patel

15. MID: - Function is used to extract characters from a text field.

Syntax: -SELECT MID (column_name, start [, length]) FROM table_name

Example: - SELECT MID (First_Name, 1,3) FROM Students
WHERE First_Name = ' Ramesh'

Output: -Ram

3.12.Null Values: -

NULL values represent missing unknown data. The NULL value is different from all valid values for a given data type. Null Value is unknown in the sense that the value is: missing from the system. By default, a column of table can hold NULL values. It can be used as a placeholder for unknown values.

Example: -Consider table Students tables as follow where Email id is optional, in that case "EmailID " Column will be saved as Null.

Studname	Class	EmailID
Sumit	FYBSC	Sumit@gmail.com
Ravi	TYBSC	
Smith	SYBSC	Smith@gmail.com
Disha	FYBSC	

We can check that data in particular column is null or not by using **IS Null** Operator.

Syntax: -ISNULL (check expression, replacement value)

Check expression: -It checks expression for null.
replacement value: -It is expression to be returned when check expression is null.

Example: -Check Whether EmailID is Null or not in Students Table

Query :-Select * From Students Where EmailID IS NULL

Output of query: -

Studname	Class	EmailID
Ravi	TYBSC	
Disha	FYBSC	

Summary: -

In this chapter we learnt different statement that are supported by SQL DML, DCL, DDL.We learnt different syntax of queries supported by SQL. We have learned how to use queries, how to retrieve data, insert new records, delete records and update records in a database with SQL.We have also learned how to create databases, tables, and indexes, views with SQL, and how to alter tables, how to drop constraints, tables, view & indexes. We have learned the different aggregate functions in SQL.

Multiple Choice Questions: -

1. In SQL to create & modify structure of objects in a database, we use ____ statement.
a) DDL b) DCL c) DML d) None of above

2. To sort the result-set by a specified column, at that time we use _________ keyword.
a) Group by b)Order by c)Having d)None of above

3. ______ function is used to remove specified prefix or suffix from a string.
a) Length b)Trim c)SUBSTR d)Right

4. For providing access or privileges on the database objects to the users, we use __________ statement.
a) Revoke
b) Role
c) Grant
d) None of above

5. __________ is used to match zero or more characters.
a) The Percent Sign (%)
b) UnderScore(_)
c) Double UnderScore(_ _)
d) None of above

6. When we want to select duplicate values from two tables we use _________ operator.
a) Union
b) Intersect
c) Union All
d) Minus

7) When we want to check for no duplicate rows exist in a table we used _________.
a) Foreign Key
b) Primary Key
c) Null Value
d) Distinct

8) A _________ constraint is used to limit the values that can be placed in a column.
a) Check
b) Unique
c) Not Null
d) Foreign key

9) The _________operator takes the results of two queries and returns only rows that appear in both result sets.
a) Union
b) Union All
c) Intersect
d) Minus

10) _______ function repeats string for a specified number of times.
a) Replicate
b) Substring
c) Reverse
d) Trim

****

Chapter 4

Entity Relationship Model

Topics To Learn: -

4.0 Overview
4.1 Modeling
4.2 ER Model
4.3 Entity Types
4.4 Attributes Type
4.5 Relationship
4.6 Relationship Set
4.7. Case Studies on E-R diagrams

4.0 Overview: -

When you are trying to solve any difficult problem, we first try to visualize that problem. After that we plan for the solution. When anyone wants to design any database it is necessary to represent all data such that user can easily visualize database and relationship that exists between the tables, this can be done with help of ER (Entity –Relationship) model. This Chapter will introduce you to a new technique that is used for representing data requirements and also to organize data in various views.

ER model is a graphical picture of the data for a particular database. To represent real world object that is distinguishable from other object we use entities & for describing properties of entities we use attribute. In this chapter you will learn how to represent any database in terms of entities & attributes as well as how to represent relationships between these entities. We will also learn other extended features of ER model such as specialization, generalization, attribute inheritances, aggregation. So as to design any scenario effectively in ER model, let's start with modeling first.

4.1 Modeling: -

Modeling languages is used to define schema (data, data relationships, data explanation, and constraints on the data). For designing database it is necessary to design database structure correctly so we use data modeling.
Let's learn about Data Modeling.

➢ **Data Modeling**: -

It is Process, which is used to produce data model for describing data, its relationships, and its constraints. It also defines data elements, their structures and relationships between them. We use data modeling to define and analyze data requirements that are needed. It is a technique that is used to model data in a standard, reliable, expected manner so that it manages resources efficiently. It is generally used in business and information technology to give a suggestion about how information is, how it should be, how it is stored and used inside a business system.

Importance of Data models: -

- It defines and analyzes data requirements.
- It's make communication easy for application programmer, end user, and designer.

- To define data elements, their structures and relationships between them.
- It provides different view for data for end-users
- It organizes data for various users.

4.1.1 Basic styles of data model:-

Three basic styles of data model:

1. **Conceptual data models (CDM): -**

   Conceptual data model also called as domain models. It is a primary step in constructing data model. It visualizes data structure of group.CDM is created by collecting all business requirement, it is created .It contains all main entities & relationship but does not consists of the complete information about attributes.

Eg. ER model

2. **Logical data models (LDMs)**: -

   LDMs means logical data model, which is used to give information of logical entity types, the data attributes describing those entities, and the relationships between the entities. It includes all entities and relationships between them and also specifies all attributes for each entity .It also state Primary key & foreign key to identify relationship in entities.

Eg. Relational model

3. **Physical data models (PDMs): -**

   PDMs means physical data model which explains all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships among tables. In this data model entities are transformed into tables. Relationship is transformed by using foreign key. Attribute is transformed into columns.

   Eg. Table, Columns, Keys, Trigger

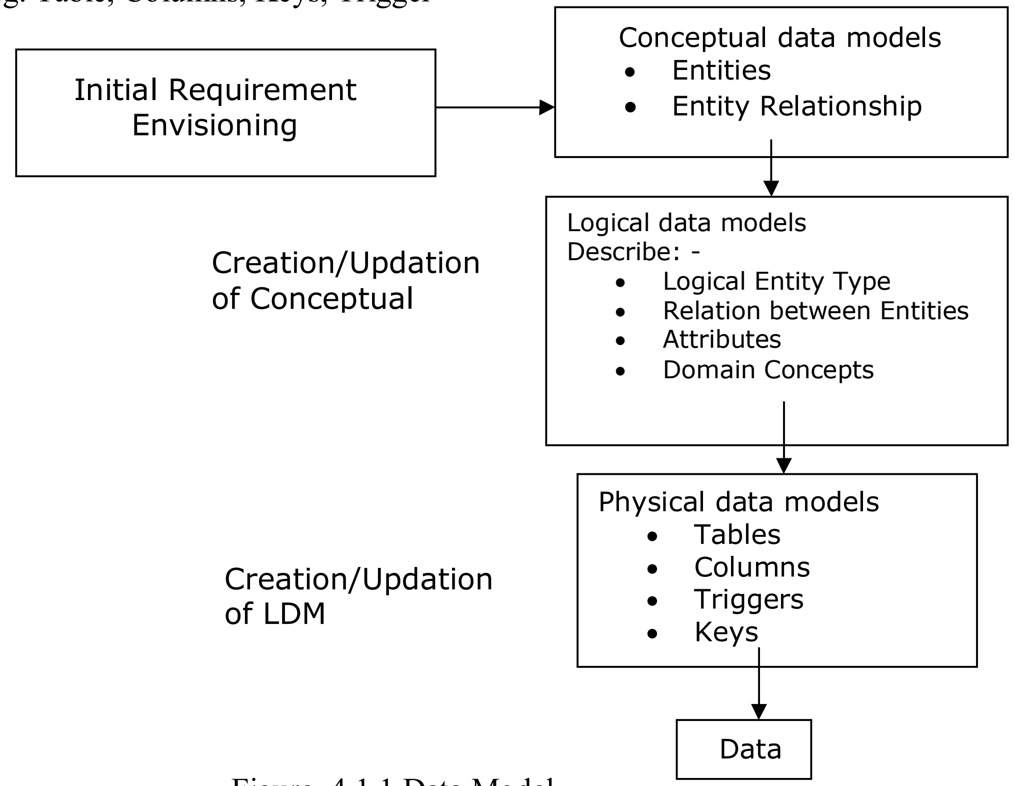


Figure. 4.1.1 Data Model

4.2 ER Model: -

The Entity-Relationship model is developed by Chen in 1976 to design database more effectively. A Conceptual data model explains the structure of a database. The fundamental concept of ER model includes entity types, relationship type and attributes. Entity type represents a set of object in actual world with similar properties. Let's see all components of ER Model.

4.2.1 Components of ER Model: -

a. Entity: -

An entity is a thing or object in the actual world that is distinguishable from all other objects. An entity is represented by rectangle containing the entity name.

Example: Student is entity having many properties name, address, roll no etc.
Example: In Bank System Customer withdraw money from bank, in terms of cash withdrawal process entities are Account & Customer so process is related with Customer & Account only.

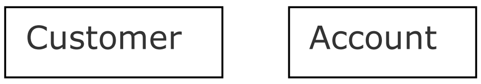


Figure 4.2.1. Entity

b. Attributes: -

An attribute is a property or characteristic. An entity is represented by a set of attributes. It corresponds to a field in a table. For each attribute there is a set of allowable values called the domain or value set of the attribute. Attributes are represented by ovals and are linked to the entity with a line. Each oval contains the name of the attribute it represents.

Example: -Professor= (Name, Address, salary, Age)

Course= (CourseFees, CourseName, Duration)

Example: - Course Entity has a CourseFees, CourseName and Duration as attribute.

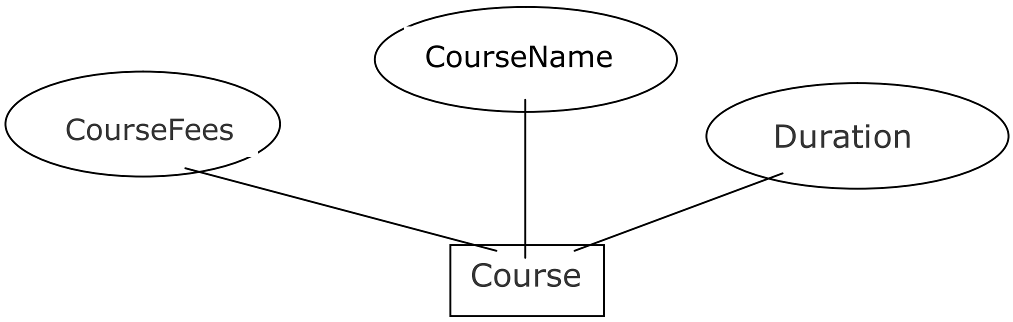


Figure.4.2.2 Attribute CourseFees, Duration, CourseName of course entity.

c. Entity Set: -

In ER model a specific table row is referred as an entity instance or entity occurrence. Each entity set has a key. All entities in an entity set have the same set of attributes. Thus entity set is a set of entities of the same type that share the same properties or attributes. Set of entities of particular type is known as Entity Set.

Example: In Bank System we deal with five entity sets as follows:

1. Customer entity set that includes set of all people having an account at the bank. Attributes are customer-name, address, Phoneno.

2. Employee entity set that have attributes employee-name, EmpId and phone-number.

3. Branch entity set that has all branches of a particular bank. Each attributes branch_name, branch-city and assets describe each branch.

4. Account, the set of all accounts created and maintained in the bank. Attributes are account_number,balance.

5. Transaction entity set that have set of all account transactions executed in the bank. Attributes are transaction_number, date and amount.

d. Domain: -

Attribute is **characteristic** or property used to correspond field in table. Each attribute is connected with a set of values that is known as domain. The domain defines the possible value that an attribute may grasp.

Example: Consider Result grading system, in which for first class grade that student should obtain 60% or more than that. So domain in this case is number from 60 to 100.

Attributes may share a domain.

Example: The name attribute for Student & Teacher shares same domain. Means name is compose of set of characters.

Domain can compose of more than one domain.

Example: EmailID is compose of Username & Address of website.

4.3 Entity Types: -

The ER model consists of different types of entities. The existence of an entity may depend on the existence of one or more other entities, such an entity is said to be **existence dependent**.

Example: If we consider Furniture the entity Chair is **existence dependent** on entity Wood.

Entities that are not depending on any other entities is termed as **not existence dependent**.

Example: In College entity elective subject does not depend upon entity student. Because many students study more than one subject .It may happen that if it is elective subject & if no student selects that elective subject for exam, though any student in exam does not select that subject but it exists for examination. Only no student is appearing for that subject. Means entity elective subject is not existence dependent on student.

Entities based on their characteristics are classified as follows.

- Strong Entities
- Weak Entities
- Recursive Entities
- Composite Entities

4.3.1. Weak entity: -

An entity, which is dependent on one or more entities for its existence, is known as Weak entity. A weak entity is existence dependent because entity existence depends on one or more entities. The existence of a weak entity is indicated by a **double rectangle** in the ER diagram.

An entity set that does not contain enough attributes to form a primary key is known as a **weak entity set**. In weak Entity set it is vital to identify attributes of weak entity set so we require a key it is known as **discriminator or partial Key**. The discriminator (or partial key) is used to categorize other attributes of a weak entity set.

- **What is Discriminator or partial Key?**
  Discriminator (or partial Key): -

It is a set of attributes that uniquely identify weak entities and are connected to same owner entity. It is called as **Discriminator or partial key.** Primary key of identifying entity set and the discriminator of weak entity set form the primary key of a weak entity set.
For example, a child entity is a weak entity because it relies on the parent entity in order for it to exist. We underline the discriminator of a weak entity set by a dashed line in the ER diagram.

Example:-If we consider cash withdraw system in Bank, we observe that different transaction is executed on same account. But in this case we can’t judge transaction amount as a primary key because its may happened that more than one transaction executed on same account having same amount. To form primary key we must consider **Account_no**.
Weak Entity is denoted as below: -

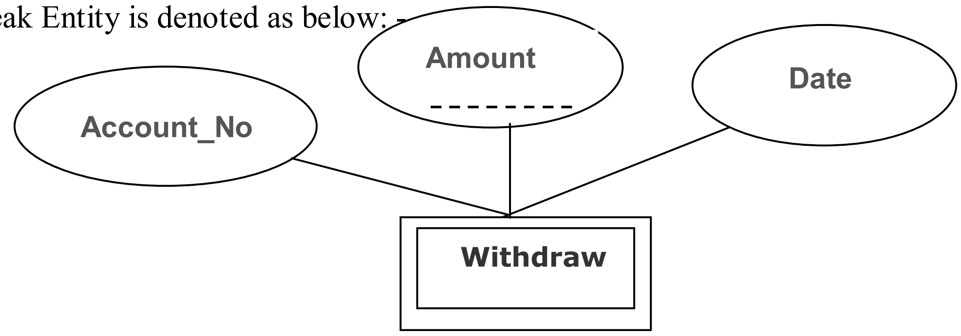


Figure.4.3.1 Weak Entity

We underline the discriminator of weak entity .In above example we underline discriminator of weak entity i.e. Amount.

4.3.2. Strong entity: -

An entity set that has a primary key is known as strong entity set. A strong entity is independent of other entities and can exist on its own.

Example: Student is strong entity type because it associates with Primary key Roll_No.

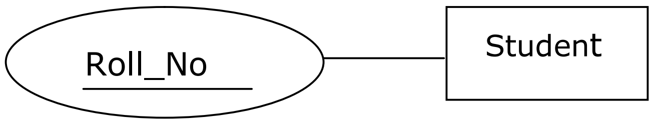


Figure. 4.3.2 Strong Entity set

4.3.3. Recursive Entity

A recursive entity is one in which a relation can exist between occurrences of the same entity set. This occurs in a unary relationship.
Example: In Organization many Employees managed by other employee's. Recursive entity is employee.

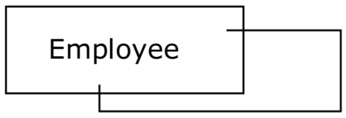


Figure.4.3.3.Recursive Entity

4.3.4. Composite Entities: -

Composite means thing composed of other things. Similar to that composite entity is composed of many parts. It is used to link the relationships that exist between many relationships. A composite entity is represented by a diamond shape within a rectangle in an ER Diagram.

Example: A Student is identified by a RollNo and a Class.
A Book is identified by a name of book, Author, Version, subject.
Address is identified by Street name, City name.

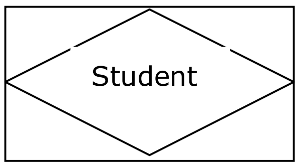


Figure. 4.3.4 Composite Entities

4.4 Attributes Type: -

In ER Model attributes can be classified into the following types.

- Simple and Composite Attribute
- Single Valued and Multi Valued attribute
- Stored and Derived Attributes
- Complex Attribute

4.4.1. Simple Attribute: -

When attribute consist of a single atomic value it is Simple attribute.
i.e. Simple attribute cannot be subdivided.
Example: - The attributes age, Gender etc is simple attributes.

Figure. 4.4.1. Simple Attribute

4.4.2. Composite Attribute: -

When attribute value is not atomic it is composite attribute. A composite attribute is an attribute that can be further subdivided.

Example: The attribute ADDRESS can be subdivided into street, city, state, and zip code.

Address: Street: City: State: Zip Code
Name: 'First Name: Middle Name: Last Name'

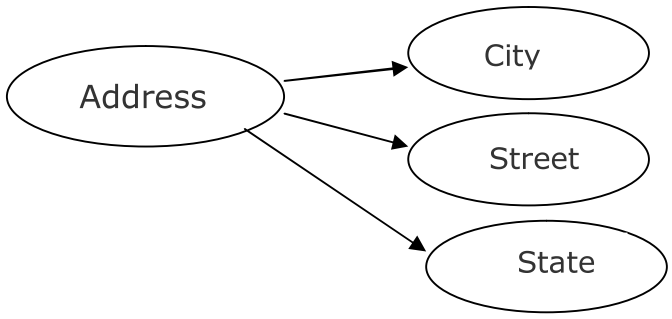


Figure. 4.4.2.Composite Attribute

4.4.3. Single Valued Attribute: -

Attributes that have a single value are known as single valued attribute.

Example:-A Person has 'age'. Age have fixed value means multiple value is not possible for Age so Age is Single valued attribute .Single valued attribute can be simple or composite.

Example:-Consider 'Percentage' in result system. It is a composite attribute .It is composed of main percentage part as well as decimal parts. It has fixed value. It is single valued attributes .Similarly 'Customer Id 'is a simple single valued attribute.

Example: Age, City, Customer id.

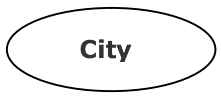


Figure. 4.4.3. Single valued attribute

4.4.4. Multi Valued Attribute: -

Multivalued attributes can have multiple values. Multivalued attributes are shown by a double line connecting to the entity in the ER diagram.

Example: A customer can have multiple phone numbers, email id's etc
A person may have several college degrees.

Multi valued Entity is denoted as below: -

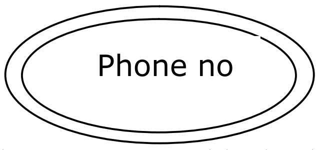


Figure. 4.4.4. Multivalued attribute

4.4.5. Stored Attributes: -

An attribute that supplies a value to the related attribute is called as Stored Attribute. The value for the other attribute is derived from the stored attribute.

Example: when calculating result Percentage that student obtain in examination is depends on the Grand total marks .In this case total marks is a stored attribute in which we store sum of all subject's marks.

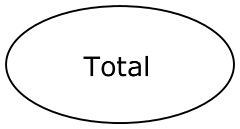


Figure. 4.4.5. Stored Attribute

4.4.6. Derived Attributes: -

If **value** of attribute is derived from a stored attribute such type of attribute is called as derived attributes. We shown derived attributes name in Dotted oval.

Example: In Result system First rank student is derived by the highest percentage i.e. student who obtained highest percentage is first ranker.

Derived Attribute is shown below: -

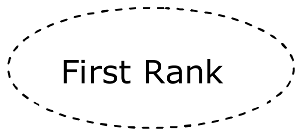


Figure. 4.4.6. Derived Attribute

4.4 7. Complex Attribute

A complex attribute is both composite and multi valued.i.e. Attribute have more than one value as well as it is composed of more than one attributes.

Example: - Person Entity has Phone_No attribute, which is composite & mulitvalued. Means One person may have more than one phone number & Phone_No is may be composed of STD Code & Telephone number.

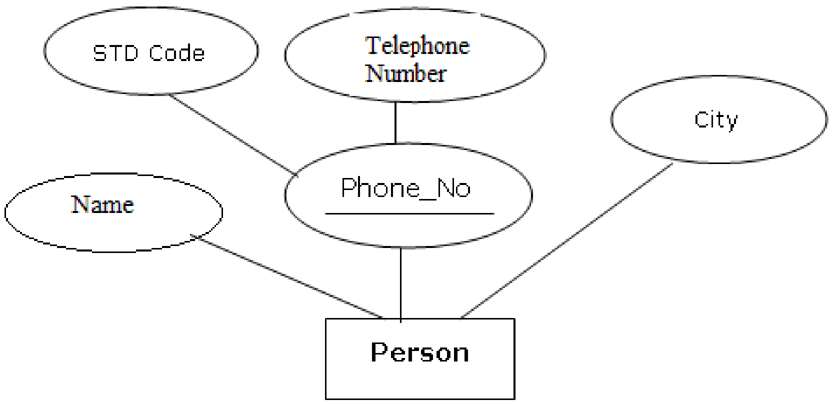


Figure. 4.4.7. Complex Attribute

4.4 8. Null Attribute: -

Some attributes are sometimes not necessary in future. In that case they may not have an applicable value for that attribute. The attributes having no value are called null.

Example: EmailID. This attribute is not applicable every time.

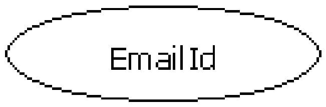


Figure. 4.4.8. Null Attribute

4.5 Relationship: -

A Relationship is an association among several entities. The association or relationship that exists between the entities relates data item to each other in a meaningful way. The existence of data association defines that, the relationship exist between two or more entities. So to represent relationship an additional entity is used called as **relation entity**.

A Relation may be expressed using notation R (A, B, C,...) where
R=name of the relation
A, B, C=the attributes within the relation
A=the attributes which forms the primary key in a relation.

Example: - Relationship R between Entity Types E1 and E2

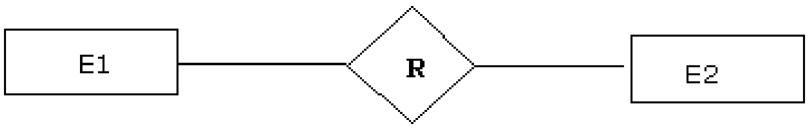
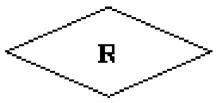


Figure.4.5 Relationship R in Entity type E1 & E2

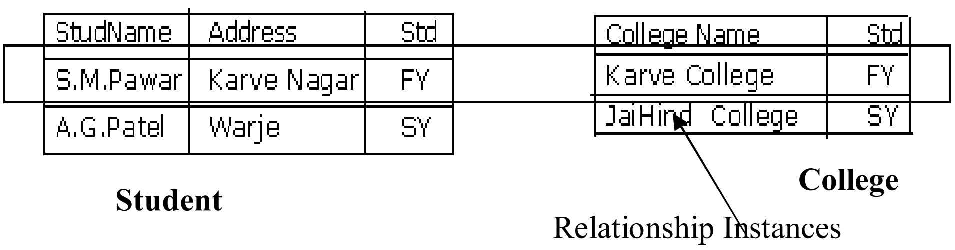
4.6 Relationship Set: -

Relationships: - A relationship is a combination (association) among the instance of one or more entity type.
Relationship set is shown below: -



Relationship instances in ER represent an association between entities. Each instance of the relationship between members of these entities is called a relationship instance.

Example: - Following Figure shows Student table have relation with College table. Student S.M.Pawar is associated with Karve College. In this case it is called as relationship instances.



StudName	Address	Std
S.M.Pawar	Karve Nagar	FY
A.G.Patel	Warje	SY

College Name	Std
Karve College	FY
JaiHind College	SY

Figure.4.6 Relationship Instances

Relationship is represented by diamond symbol. The two side of diamond are connected to the entities it relates.

Example: - Professor teaches Students.

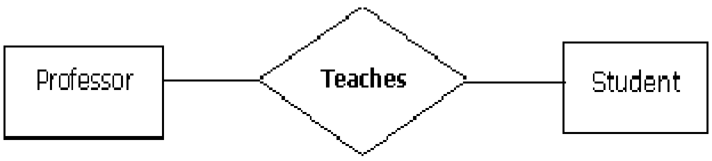


Figure.4.6 Relationship between the entities Professor and students

4.6.1 Connectivity in relationship: -

Relationship can be classified as one-to-one, one- to -many and many- to-many. The terms connectivity is used to describe this relationship classification. The ER Diagram indicates the relationship's connectivity by placing 1, M or N near the related entities as shown.

1. One-to-one Relationship: -

Example: One Manager in company manages a single department (a one to one connectivity).

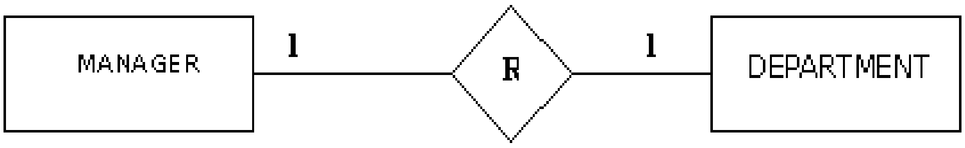


Figure: One-to-one Relationship

2. One-to-Many Relationship: -

Example: One Professor teaches in many classes. (a one- to- many connectivity)

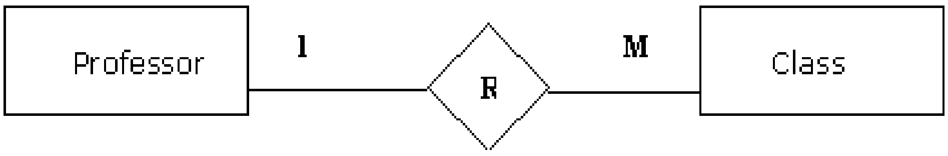


Figure: One-to-Many Relationship

3. Many-to-one Relationship: -

Example: Many Employees working in one Company. (a One –to-many connectivity).

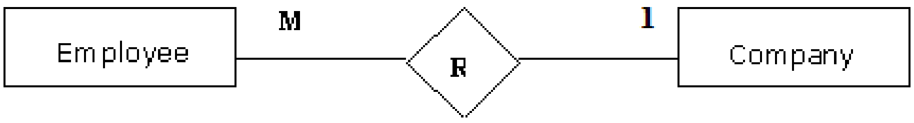


Figure: **Many**-to-one Relationship

4. Many-to-Many Relationship: -

Example: The many Colleges can offer courses to many students as a part of student development program. (a many–to–many connectivity).

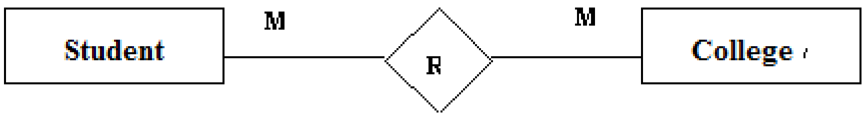


Figure: Many-to-Many Relationship

4.6.2 Types of Relationship:-

a. Unary Relationship: -

An Entity can be related to itself, such kind of relationship is known as **Unary** or **Recursive** Relationship. A unary relationship exists when an association is maintained within a single entity.

Example: One Person at a time marries with only one person.

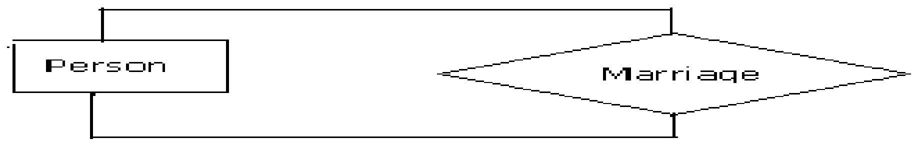


Figure.4.6.2.a.Unary Relationship

b. Binary Relationship: -

A binary relationship exists when two entities are associated.

Example: Is Assigned is Relation between two entities Person & Account.
Relation may be one to one or one to many or many to many.

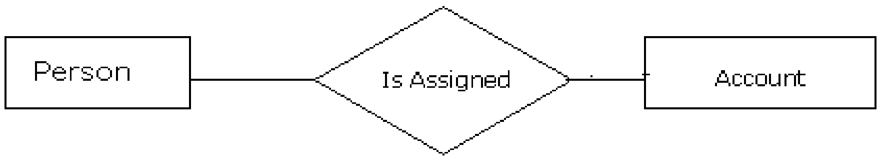


Figure.4.6.2.b.One-to-One

Many-to-many: - Many Student Studying in many Colleges.

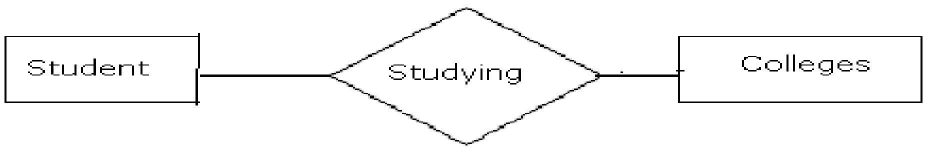


Figure.4.6.2.c. Many-to-many

C. Ternary Relationship: - A ternary relationship exists when three entities are associated. Example: -Many students learning in one colleges as well as many professor teaches in that same college.

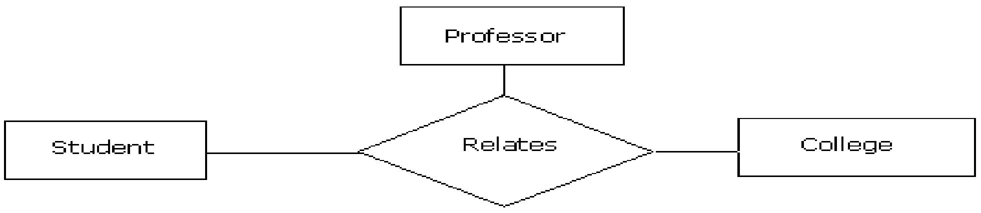


Figure.4.6.2.d. Ternary Relationship

The degree of the relationship in each diagram is represented by the number of rectangles i.e. **Entities**. Relationships are represented by diamond-shaped Figures.

The lines between the diamonds and the rectangles represent the **multiplicity** of the relationships. A single line represents a one-to-one relationship. A line that branches into three segments where it connects to the entity represents the one-to-many or many-to-many relationships.

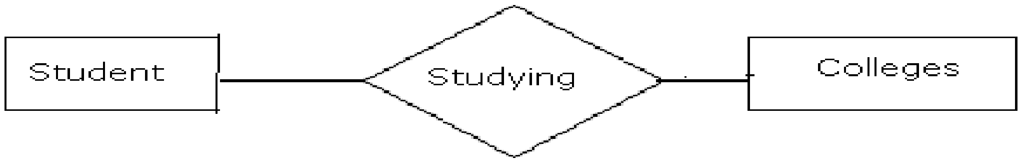
4.6.3 Classifying Relationships: -

Relationships are classified by degree, multiplicity and existence. In data modeling, relationships have degree (unary, binary, ternary, or n-ary), and multiplicity (one-to-one, one-to-many, or many-to-many). i.e. the association is between entities of the same type.

a. Degree of Relationship: -

The degree of a relationship is the number of entities among which the relationship exists. The most common degree of relationship is binary, which exists between two entities. The Degree of binary relationship is two (number of entities in binary relation is two).

Example: - In following Figure student entity is related with College entity. Many students studying in one college, this is binary relationship. Here two entities are present so degree of this binary relationship is two.



b. Multiplicity: -

Multiplicity is the number of instances of entities that are related.

Example: -A Binary relationship exists between student & colleges. Many students are studying in one college or it may happen that one student studying many degrees from many colleges. So multiplicity of this relationship is many-to-one or many-to-many.

c. Existence: -

Existence means whether the existence of an entity instance is dependent upon the existence of another related entity instance.

The existence of an entity in a relationship may be either mandatory or optional.

If an instance of an entity must always occur for an entity to be included in a relationship, then it is mandatory. An example of mandatory existence is that "Student must select at least **one** subject from optional subject List". It's mandatory for students.

If the instance of the entity is not required, it is optional. An example of optional existence is that "Some activities in colleges are not compulsory to students ". Means for example to participate in annual gathering is not compulsory to all students.

4.6.4 Mapping Cardinalities: -

Before learning about mapping cardinalities we will learn about what is cardinality and its type.

Cardinality:-

It is use in database relations to indicate the occurrences of data on either side of the relation. Let's see Mapping cardinalities or cardinality ratios.

In Relationship one entity is associated with another entity. Mapping cardinalities or cardinality ratios is act to express the number of entities to which another entity can be connected (associated) via a relationship set. Mapping cardinalities are most useful in describing relationship sets.

- **One to one**: -

  An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.

Example: Every University has exactly one Vice Chancellor.
Every Country has only one Prime minister.

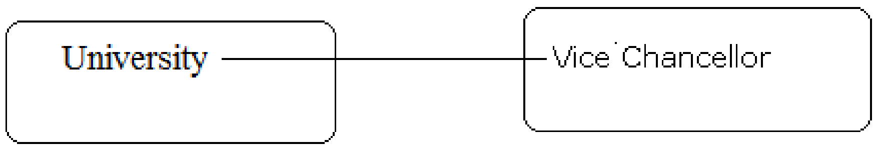


Figure. One to One relationship

- **One to many: -**

  An entity in A is associated with any number (zero or more) of entities in B. An entity in B, however, can be associated with at most one entity in A.

Example: One student takes many Subjects.
One Person may have bank Accounts in more than one bank.

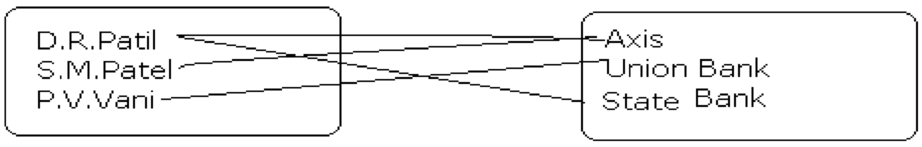


Figure. One-to-many Relationship.

- **Many to one: -**

  An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number (zero or more) of entities in A.

Example: Many city groups under one state

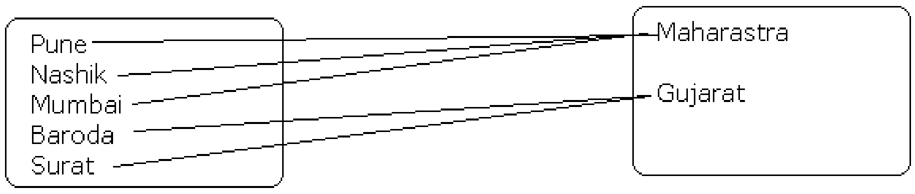


Figure. Many-to-one Relationship.

- **Many to many**: -

  An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.

Example: Many Students studying in many colleges.
Many students can participate in many conferences.
Many Universities are spread across Many Cities.

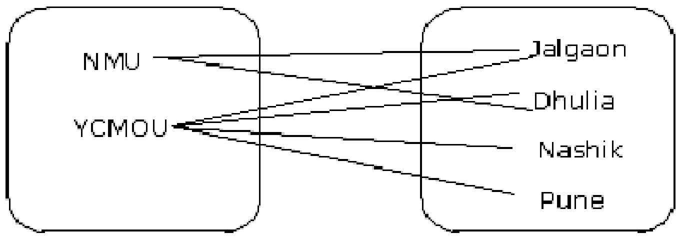


Figure. Many-to-many Relationship.

4.6.5 Keys: -

A key is an attribute or a combination of attributes that is used to identify records. Sometimes we might have to retrieve data from more than one table, in those cases it is required to join tables with the help of keys. The purpose of the key is to bind data together across tables without repeating all of the data in table.

4.6.5.1 Keys for Relationship set: -

a. Super Key: -

A Super Key is a set of one or more attributes that are taken collectively and can identify all other attributes uniquely. Super key stands for superset of a key.

Example: Student (RollNo, Name, Class, Address, Phone_No)
<RollNo, Name>
<RollNo, Name, Address> all these are super Key.

b. Candidate Key: -

Candidate Keys are super keys for **which no proper subset is a super key**.
Example: let suppose we have a Student Table with attributes
('RollNo', 'Name',' Address', 'Class','Phone_No')

In above table RollNo Key can identify the record uniquely and similarly combination of Name and Address can identify the record uniquely, but neither Name nor Address can be used to identify the records uniquely as it might be possible that we have two Students with similar name or two Students from the same Address. So for above table we have only two Candidate Keys used to identify the records from the table uniquely.

1. RollNo
2. Name, Address. These are Candidate Key

c.Secondary key: - alternate of primary key. Means we can use that key to identify record but they might not be unique.

Example: Consider Student Table with attributes
Student ('RollNo', 'Name',' Address', 'Class', 'Phone_No').
Secondary Key can be Name, Address, and Class etc. as they can identify the records but they might not be unique.

d. Compound key: –

Compound key is known as composite key or concatenated key. It **consists of 2 or more attributes**. If in a table, no single data element is identified uniquely, then we can form compound key by combining multiple elements to identify the data element.

Example: -Student ('RollNo', 'Name',' Address', 'Class', 'Phone_No').
Attributes Name and Phone_No is compound key in Student Table. i.e. Name is composed of First Name, Last Name & Middle Name as well as Phone_No is also Compound Key as it may be a combination of STD Code & phone no.

e. Alternate Key: -

Any of the candidate keys **that are not part of the primary key** is called Alternate Key.

Example: Consider Student Table with attributes ‘RollNo’, ‘Name’,’ Address’, ‘Class,’ Phone_No’.In above example Name, Address is Candidate Key, which is not a Primary Key.

h. Primary Key: -

Primary key is an attribute in table that uniquely identifies each record in table and we cannot put primary key as null as well as duplicate. In the ER diagram, primary key is represented by underlining the primary key attribute. Usually primary key is composed of a single attribute

Example: Order (OrderID, Quantity, Price, Date).

In this case we will consider OrderID as a Primary Key .By using OrderID we can easily identify individual details of Orders uniquely.

Primary Key is Shown Below: -

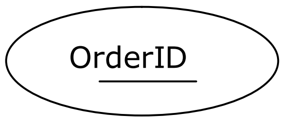


Figure. Primary Key OrderID

g. Foreign key: -

A foreign key (FK) is a field in a database record that point to a key field forming a key of another database record in different table. A foreign key in one table points to the primary key (PK) of another table. This way reference can be made to bind the information together.

Example: Consider two tables Order and Purchase_Info
Order (OrderId’,’Quantity’,’OrderDate’,’Purches_No’)
Purchase_Info (‘Purches_no’,’Purches_date’,’Amount’)

In this case ‘Purches_no’ is primary key for Purchase_Info .It is foreign key for table Order. Order table is Parent table having references to child table for Purchase_Info. Means a Foreign Key value must match an existing value in the parent table or be NULL.

It cannot happen that if foreign key value is not present in the parent table though that value is present in Child table. If you want to delete values from child table then that value should also be deleted from parent table.

4.6.5.2. E-R Diagrams: -

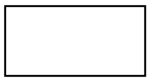
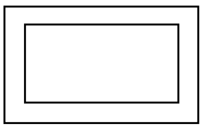
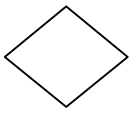
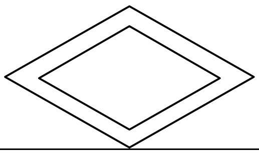
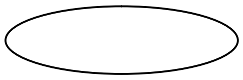
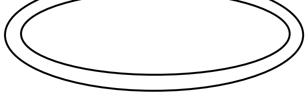
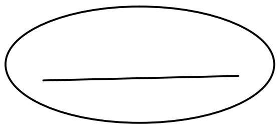
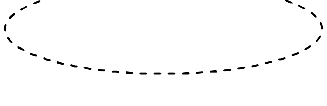
ER diagram (ERD) is well known as data schema map or data map .It is use to capture the data. The ERD captures data about the entities, their attributes and the relationships between the entities. The ERD pictorially represent the interrelationship of the database schema. The major components of ERD are entities, attributes, relationship and cardinality of association.

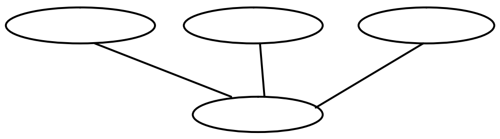
Steps in ER Modeling: -

1. Identify the Entities.
2. Find relationships.
3. Identify the key attributes for every Entity.
4. Identify other relevant attributes.
5. Draw complete E-R diagram with all attributes including Primary Key

4.6.5.3.E-R Modeling Symbols: -

The database structure adopting the entity relationship model is shown pictorially using E-R diagram. The entities and the relationship between them are drawn using the following convention:

Sr.No	**Symbol**	**Meaning**
1		Entity Set
2		Weak Entity Set
3		Relationship
4		Identifying Relationship
5		Attribute
6		Multivalued Attribute
7		Primary Key
8		Derived Attribute

9		Composite Attribute
10		Connector

4.6.5.4 Cardinality Constraints related to E-r diagrams: -

Cardinality means how many instances of an entity related to one instance of another entity.Maximum number of relationships is specify as **Cardinality** and minimum number of relationships is specify as ordinarily. Relationship is frequently called **optional**, when the minimum number is zero and relationship is called **mandatory,** when the minimum number is one or more.

We can use different notation for cardinality by drawing line such as directed line ($\rightarrow$) for showing 'one' and undirected line **(**——**)** for 'many' between relationship set & entity set.

For example, a Professor teaches to more than one student.
This is described by the **cardinality** of the relationship, for which there are four possible categories.

- One-to-one (1:1) relationship
- One-to-many (1:m) relationship
- Many-to-one (m: 1) relationship
- Many-to-many (m: n) relationship

I. One to One: -

A one to one relationship – One College having one principle only, so it is a many to one (1: 1) relationship

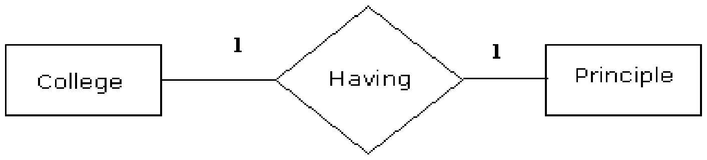


Figure: - one to one relationship

II. Many to one: -

A many to one relationship - many Study Centers are grouped under One University, so it is a many to one (m: 1) relationship

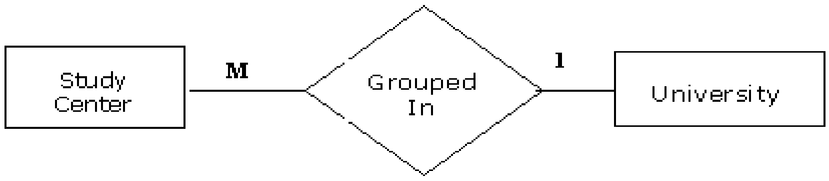


Figure: - Many to one relationship

III. One to Many: -

A one- to-many relationship- Many Colleges Placed in One City, so it is a One- to-many (1: M) relationship.
Example: Many Colleges Placed in One City.
One state consist many cities.

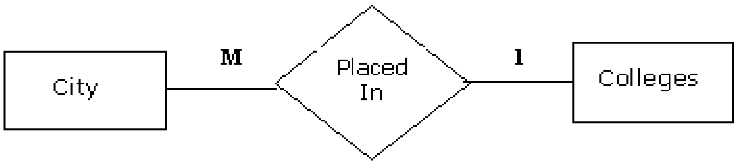


Figure: - One to many relationship

IV.Many to Many: -

Many student is taught by many lecturers, so it is a many to many (m: n) relationship
Example: Many Lecturers taught many students.

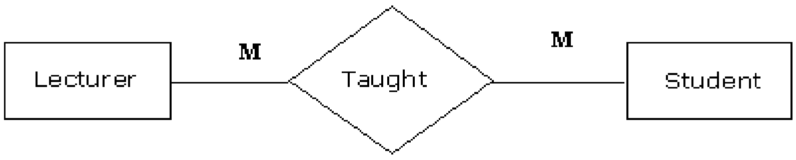


Figure: - Many to many relationships

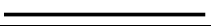
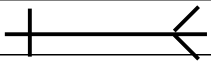
4.6.5.5 Alternative Notations for cardinality limits: -

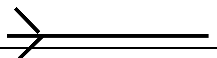
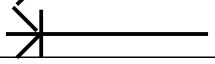
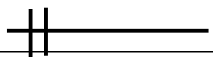
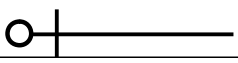
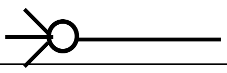
Many different type of notation is available for cardinality.

- •Information Style
- •Chen Style
- •Bachman Style
- •Martin Style

We will See Information Engineering Style & Martin Style only which is widely used.

Information Engineering Style Notation: -

Symbol	**Meaning**
	One-to-One
	One-to-many (mandatory)

	Many
	One-to-more (mandatory)
	One-and only one
	Zero or one (Optional)
	Zero or Many (Optional)

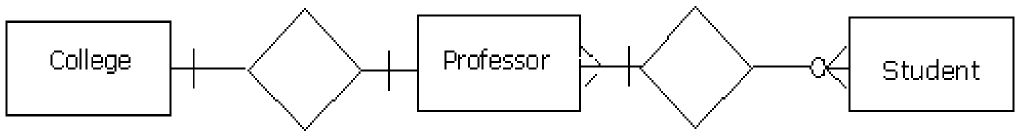
Martin Style Notation for Cardinality: -

Symbol	**Meaning**
1	One and only one (mandatory)
*	Many (Zero Or More-Optional)
1…*	One or more (Mandatory)
0…1	Zero or more (Optional)
(0,1)	Zero or more (Optional)
(1,n)	One or more (Mandatory)
(0,n)	Zero or more (Optional)
(1,1)	Zero or more (Mandatory)

Let's see an example by using cardinality notation.

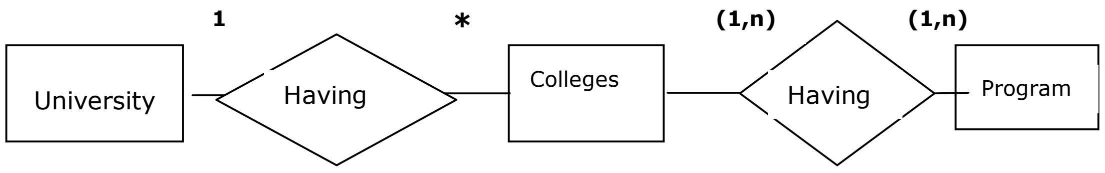
Example: -One College has many Professors and many Professors taught many Students.

Using Information Engineering Style Notation: -



Example: -One University has many Colleges and many Colleges have many Programs.

Using Martin Style Notation: -



4.6.5.6 Weak Entity Sets: -

An entity set that does not contain sufficient attributes to form a primary key is called as weak entity set and an entity set that contains a primary key is called a strong entity set.

Example: -Consider entity student having attribute name, Roll_No. Student is part of university means many students are connected to university. Many Student from different colleges may have same name .so in this case entity set Students does not have sufficient attribute to form primary key. It is weak entity set.

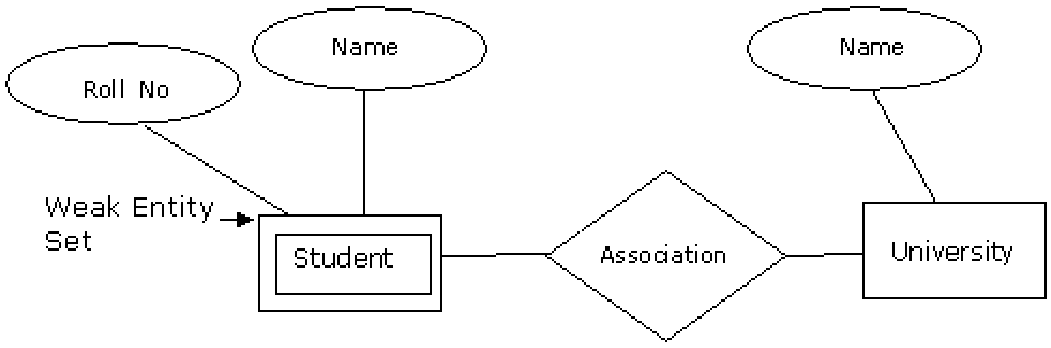


Figure. 4.6.6.5 Weak Entity set

4.7. Case Studies on E-R diagrams: -

Case Study 1: - Design a database for Students management system that has a set of students. Each Student is associated with one or more colleges. Colleges are Private or Government.

A database should provide following details to user:

1. Identify all entities
2. Identify all relations
3. E-R Diagram
4. Relational model

Solution: -

ER Diagram

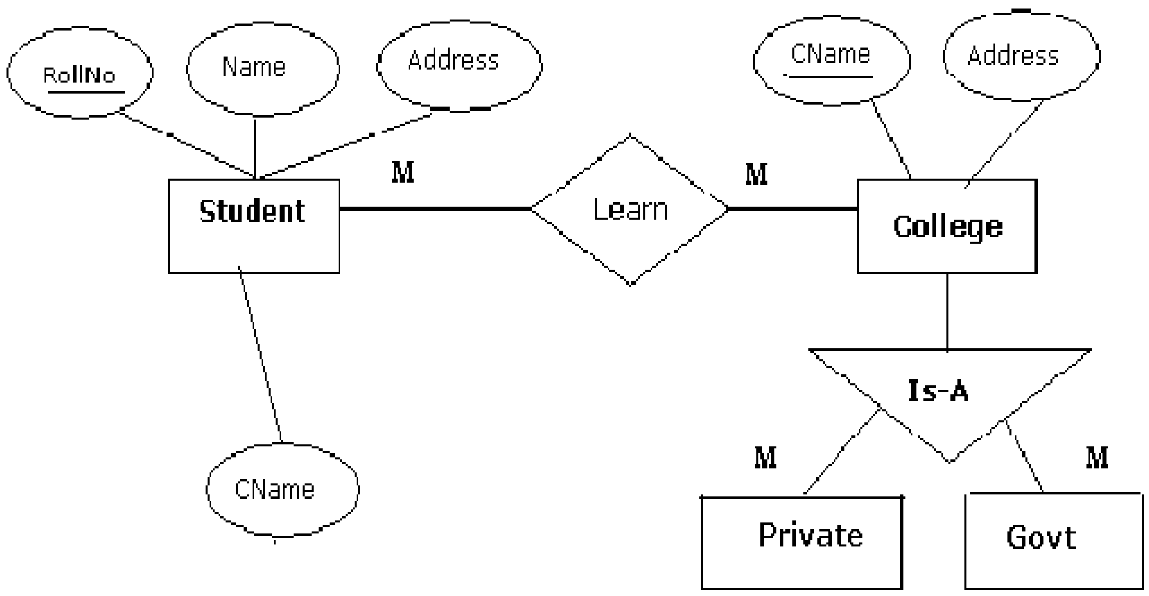


Figure.4.8. ER Diagram for case Study Diagram

Identified with Primary key/Candidate key

Entity	**Primary Key/Candidate Key**
Student	RollNo
College	CName

Relational model with all entity: -

Student (RollNo, name, address, Name)
College (CName, Address)

Relationship (As Per shown in ER Diagram)

1. Many Students has many Colleges (Many-to-Many)

Tabular representation

Table Student

RollNo	Name	Address	CName

Primary Key Foreign Key

Table College

CName	Name	RollNo

Case study 2:-Consider Courier services system. In which Administrator is person who handles administrations of system. Administrator is person who is actually owner of Courier Services shop. Client is person who courier the documents or things. Workers are person who works in courier office to handle enquiry, dispatching process of courier's etc. Payment _Mode option is available for Client. Client can do Payments by using different Payment modes. Draw ER diagram for Courier Service System.

For drawing ER Diagram we follow steps: -

Step 1: Identify the Entities that exist in this relation.

- Courier_Office
- Administrator
- Payment_Mode
- Client
- Worker

Step 2: Find the relationships that are present in Courier services system

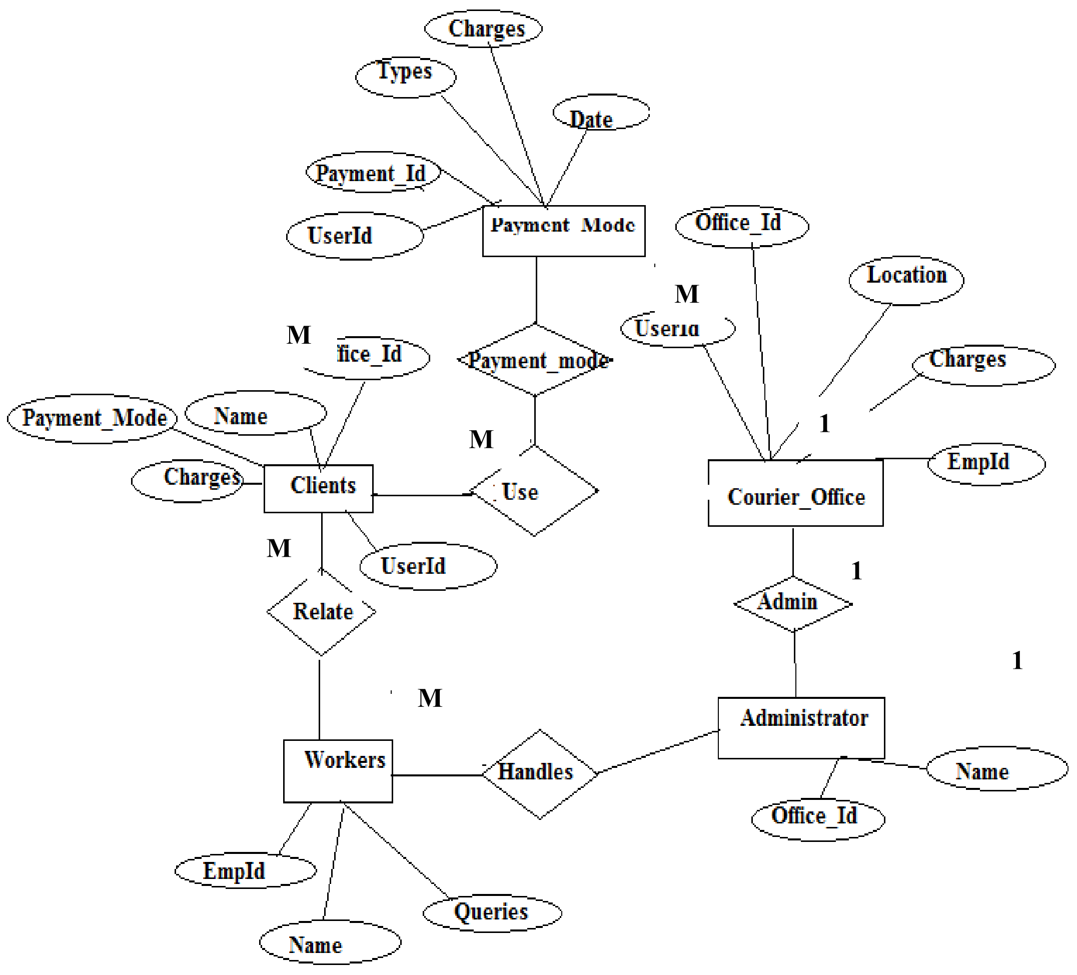
- One Courier_Office have one administrator who handles all activities of administration. So one to one relationship exists between Courier_Office and Administrator.
- Many Workers' works under One Administrator. Many to one relationship exist between worker and administrator.
- Many Clients uses different modes of Payment by cash or Credit Cards. So many to many relationships exists.

Step 3: Identify the key attributes.

- Courier_Office entity has attributes like Office_Id, Name, Location, Payment mode, UserId, Charges.
- Client has attributes like UserId, Name, Location, Charges, Date, Payment_Mode,
- Payment_Mode have attributes like Payment_ModeId, Charges, Payment_Type, Date.
- Administrator has attributes like Office_Id, Workerid, Name, Salary
- Worker has attributes like Office_Id, Workerid, Name, Salary

Step 4: Draw ER diagram.

Following Figure shows ER Diagram for Courier System.



Summary: -
First we learnt in this chapter that ERD means Entity Relationship diagram that is use for easily visualize database and relationship that exists between the tables.

- Data modeling is used to explain structure of database.
- The ER model refers to a specific table row as an **entity instance** or **entity occurrence.**
- An attribute is a **property** or **characteristic**. An entity is represented by a set of attributes.
- A weak entity is entity, which is **dependent** on one or more other entities for its existence.
- **Discriminator or partial key** is set of attribute that uniquely identify weak entities and that are connected to same owner entity.
- Simple attribute consist of a **single** atomic value
- A composite attribute is an attribute that can be further subdivided. i.e. **not** atomic Value.
- A single valued attribute can have only a **single** value.
- Multivalued attribute can have **multiple** values.
- **Stored** attribute is an attribute that supply values to other attribute.
- An attribute whose value is derived from a stored attribute is known as **derived** Attribute.
- The attributes having no value are called **null** attributes.

- An Entity can be related to itself is called **Unary** or **Recursive** Relationship
- When two entities are associated then relationship is **binary** relationship.
- When three entities are associated, it is **ternary** relationship.
- Cardinality means how many instances of an entity relate to one instance of another entity.
- A key is an attribute or a combination of attribute that is used to identify records.

Multiple Choice Questions: -

1>Data Modeling is used for _____________.

a) To define and analyze data requirements.
b) To defines data elements, their structures and relationships between them.
c) How to represent Data, How it should be, How To Stored.
d) All of Above.

2>________ transform conceptual Schema into logical model.

a) Logical data Model b) Network Model
c) Conceptual Model d) None of above.

3> The ER model refers to a specific table ________as an **entity instance** or **entity occurrence.**

a) Column b) Group of Rows
c) Row d) Group of Fields

4> __________attribute is an attribute that can be further subdivided.

a) Multivalued attribute
b) Simple Attribute
c) Composite Attribute
d) Null Attribute.

5> An Entity can be related to itself is called ________________.

a) Recursive Relationship
b) Binary Relationship
c) Ternary Relationship
d) None of above

6> An attribute that's value is derived from a stored attribute is known as _______ Attribute.

a) Simple attribute
b) Derived Attribute
c) Composite Attribute
d) Null Attribute

****

Chapter 5

Normalization

Topics To Learn

5.1 Overview:
5.2 Relational DB design
5.3 Decomposition (Small Schema)
5.4 Lossy Decomposition
5.5 Loss less Decomposition
5.6. Functional Dependency
5.7 Normalized Forms
5.8. De-normalization

5.1 Overview:-

SQL offers to use different types of queries but sometime it may happen that we get redundant, inconsistent data, which is not sufficient. To avoid problems regarding to database, it necessary to convert database into normalize form means normal, that concern to standard or norm. So we need normalization of database.

This chapter explains you that what is normalization, why we use it, different types of normalization that helps you to maintain normalize forms in different ways. We also learn denormalization process, which is also useful sometimes to avoid drawbacks of normalization. Let's start from Relational database design first.

5.2 Relational Database design:-

While designing relational database it is necessary to design "good" schemas. A bad design may create many problems such as repetition of data or it may produce data, which is not sufficient to shows all relations etc.

If we want to create good relational database design we must remember things like avoiding redundant data, ensuring the relationships among attributes and managing the consistency of the database.

Example: Consider following table student:
Student = (College_name, City, Fees, Stud_Name, Roll_No, Grade)

College_name	City	Fees	Stud _name	Roll_No	Grade
S.S.V.P.S	Nashik	150000	Sumit	11	A
D.Y.Patil	Pune	200000	Smith	21	B
Vivekanad	Nashik	150000	John	17	A+
D.Y.Patil	Pune	300000	Kalpesh	15	B+

In this schema we found redundancy in field College_name, City, Fees that is data for College_name, City, Fees are repeated for each student .It may create difficulty in updation of data. Thus the above design of table consist certain unwanted properties, which should be overcome.

5.3 Decomposition (Small Schema): -

In general decomposition means to break down particular thing into many pieces. Similar to that we decompose relation into several relations to avoid redundancy. When we decompose relation it is required to check that decomposition does not lead to bad design.

Definition: -Let R be a relation schema

A set of relation schemas {R1, R2, …, Rn} is a decomposition of R if

- R = R1 U R2 U …..U Rn
- Each Ri is a subset of R (for i = 1,2…n)

Example: Consider relation R (x, y, z) :
We decompose R into two subset R1 and R2 as below

R1 (x, z) and R2 (y, z)

If we union R1 and R2, we get R

R = R1 U R2

In decomposition, if we decompose tables then it may happen that, we may not be able to reconstruct the corresponding instance of the original relation it means **information loss** in decomposition appear**.**

Example: -Consider Following Table for decomposition
Give Relation R we decompose it into two subset R1 and R2 .If we try to reconstruct R from R1 U R2 ,we cannot get original R so there is information loss in decomposition.

RollNo	Fees	Grade
A11	15000	A+
S20	4000	B
A70	1500	A+

⇩

Decomposition

RollNo	Grade
A11	A+
S20	B
A70	A+

R1

Fees	Grade
15000	A+
4000	B
25000	A+

R2

5.4 Lossy Decomposition: -

Sometimes we decompose relation into many relations, it may happen that decomposition of relations may not produce original relation .So at that time we loss data for original relation. Such type of decomposition is known as Lossy decomposition.

➢ Definition: - Loss less decomposition: -When we decompose relation of R into {R1, R2… Rn} and if the natural join of R1, R2… Rn cannot produce original relation R it is called as Loss less decomposition.

Example: -Given two relation R1 and R2 , We combine them R1 ∪ R2.Combine both relations into single relation as follow :-

Roll_No	Grade
A11	A+
S20	B
A70	A+

R1

Fees	Grade
15000	A+
400	B
1500	A+

R2

R1 ∪ R2

Roll_No	Fees	Grade
A11	15000	A+
A11	**1500**	**A+**
S20	400	B
A70	**15000**	**A+**
A70	1500	A+

In previous example, additional tuples are obtained along with original tuples. Although there are more tuples, it produces not as much of information. Due to the loss of information, decomposition for previous example is called **Lossy decomposition or Lossy-join decomposition.**

5.5 Loss less Decomposition: -

Consider Relation R (A, B, C) and we decompose it into R1 (A, B) and R2 (A, C). After decomposition try to recover it as R' (A, B, C).Finally if you get R'=R then you can say that it is Loss less decomposition.

- Definition: -When we decompose relation of R into {R1, R2,…, Rn} and if the natural join of R1, R2… Rn produces exactly the relation R it is called as Loss less decomposition.

Example: Consider Student-schema = (RollNo, Name, Address, Fees, Class, Grade)

In this schema, we found that Name, Address, Grade depend upon RollNo and RollNo, Class together determine fees.

Decompose Student -schema into two schemas:-

Personal-schema = (RollNo, Name, Address, Grade)
Fees-schema = (RollNo, Class, fees)

Show that decomposition is Loss less Decomposition.

Roll_No	Name	Address	Fees	Class	Grade
11	Sumit	Nashik	150000	FY	A
21	Smith	Pune	200000	SY	B
17	John	Nashik	150000	FY	A+
15	Sonam	Pune	300000	TY	B+

R1

Roll_No	Name	Address	Grade
11	Sumit	Nashik	A
21	Smith	Pune	B
17	John	Nashik	A+
15	Sonam	Pune	B+

R2

Roll_No	Fees	Class
11	150000	FY
21	200000	SY
17	150000	FY
15	300000	TY

R1 ∪ R2 ⇒

Roll_No	Name	Address	Fees	Class	Grade
11	Sumit	Nashik	150000	FY	A
21	Smith	Pune	200000	SY	B
17	John	Nashik	150000	FY	A+
15	Sonam	Pune	300000	TY	B+

5.6. Functional Dependency: -

We see that relation exist between different attributes. First that what is dependency? When one entity depends on another entity it is called as **Dependency**. This is also applicable for the function too. We represent functional dependency by ->.

If we write **A->B, it** means that

-A is determinant of B

-The value of A uniquely determine value of B

-The functional dependences of B on A.

What is a Functional dependency?

Functional dependencies (FD) are used to express constraints between attributes. FD exists when value of one attribute in a table is determined completely by the value of another.

If Given relation R and A->B then each value of B in R is associated with exactly one value of A.

Example: -Determine Functional Dependency notation and primary key.

RollNo	Student_Name	City
1	P.R.Patil	Pune
2	S.P.Pawar	Nashik

Solution: -For determine primary key in table, we use column which uniquely identify each record .For identifying FD's we identify relationship between two attributes which allows us to uniquely determine the corresponding attribute's value. By using RollNo we can uniquely determine name and city in above relation.

Student: RollNo -> Student_Name, City

Primary Key Functional Dependency

In above example for particular value of RollNo, there is exactly one corresponding value for NAME. For example, for RollNo 1, there is exactly one value of NAME, P.R.Patil. Hence, NAME is functionally dependent on RollNo. Similarly, there is exactly one value of CITY for each value of RollNo. Hence, the attribute CITY is functionally dependent on the attribute RollNo. The attribute RollNo is the determinant. We can also say that RollNo determines CITY and NAME.

• **Functional Dependency**: -

It is divided into subtypes as follows:

1. Full Functional Dependency
2. Partial Dependency
3. Transitive Dependency

1. Full Functional Dependency:-

If we have Functional Dependency P->Q then it can be Full Functional Dependency only when we cannot remove any attribute of Q from P.In other words Q is fully functionally dependent on P and there should not be any R->Q, where R is a proper subset of P.

Example: -Determine functional dependency in Mark sheet table given below.

RollNo	Subject_Code	Marks
101	C110	82
101	C112	45
102	C122	65
103	C123	70

Solution: -Mark Sheet (RollNo, Subject_Code, Marks).

In this case (**RollNo, Subject_Code)** both form composite key (primary key with more than one attribute) means if we want to find marks for particular subject we need RollNo and Subject_ Code.

I.e. Functional Dependency exists in this relation as follows:-

Marks->RollNo, Subject_Code

2. Partial Dependency: -

If relations have functional dependency like A -> B, we remove some attribute from A and yet the dependency stills hold, this type of dependency is known as **Partial dependency**. It is type of functional dependency where an attribute is functionally dependent on only part of the primary key (primary key must be a composite key). Partial dependency exists when the value in a non-key attribute of a table is dependent on the value of some part of the table's primary key

Lets see why we need to remove partial dependency.
Consider Student Schema as given below: -

Student (RollNo, Name, Course, Fee)
There are 3 anomalies.

1. Insertion:-If you want to enter new record having only student's basic information like RollNo, name but if we don't want to enter Fee, Course details we cannot do like this. It is must for user to enter information about Fees and Course.

2. Deletion:-If we want to delete particular basic information of student's and we didn't want to delete other information like fees, Course at that time problem arises.

3. Modification:-If we want to update any information many times at that time anomaly may appear.

All these anomalies must be avoided so it is needed to remove partial dependency.

Example: Determine partial Dependency in Student Table
Student (RollNo, Name, Course, Fee)

Solution: RollNo->$\underleftrightarrow{\text{Name, Course}}$

RollNo->$\underleftrightarrow{\text{Course, fees}}$

Partial Dependency

To remove partial functional dependency split table into new relation according to Functional dependencies.

Personal (RollNo, Name, Course)
Fees (RollNo, Course, Fee)

3. Transitive Dependency: -

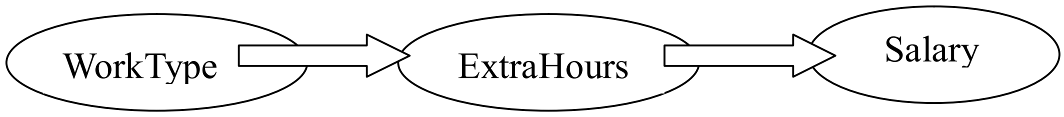
A Functional dependency between two (or more) non-key attributes in relation is known as Transitive functional dependency.

Example:-Let's see example.

EMP (EmpId, WorkType, ExtraHours, Salary)

If Employee works in company then it has EmpId that is unique for each employee. Suppose company offer two types of WorkType Full Time work and Part Time works and also offer extra charges on **hour's basis** depends on employees worktype.Lets consider for Full Time work company offer 150 rupees per hour and for Part Time work company offer only 75 rupees per hour .If employee works for more hours than actual time limit per day then that amount will be added in salary at the end of month based on WorkType of employee. So in this case transitive dependency appears such as follows:-

EmpId->WorkType, ExtraHours
WorkType->ExtraHours->Salary



Example: Remove Transitive dependency.
Student (RollNo, Name, SubjectCode, SubjectName, Marks)

RollNo	Name	Class	SubjectCode	SubjectName	Marks
101	P.M.Pathak	FY	C101	C	57
102	S.M.Patil	SY	C201	DBMS	72
103	A.J.Pawar	FY	C102	C++	60
104	A.D.Despande	SY	C202	Oracle	75

Solution: -For removing transitive dependency, you have to first remove partial dependencies then transitive dependency.

RollNo ->Name, Class
SubjectCode->SubjectName-> Marks.

Partial Dependency: -Name and Class is partially dependent on RollNo.
Transitive Dependency: -SubjectName is depend on SubjectCode of subject.
Marks obtain by student is depend on SubjectName.So transitive functional dependency exists as follows:-

Non-Key Attribute **SubjectCode -> SubjectName->Marks**
⬇
Transitive dependent

To Remove Transitive dependency split the relation into new relations.

Personal (RollNo, Name, Class)

SubjectInfo (RollNo, SubjectCode, SubjectName, Marks)

RollNo	Name	Class	SubjectCode	SubjectName	Marks
101	P.M.Pathak	FY	C101	C	57
102	S.M.Patil	SY	C201	DBMS	72
103	A.J.Pawar	FY	C102	C++	60
104	A.D.Despande	SY	C202	Oracle	75

Personal

RollNo	Name	Class
101	P.M.Pathak	FY
102	S.M.Patil	SY
103	A.J.Pawar	FY
104	A.D.Despande	SY

SubjectInfo

RollNo	SubjectCode	SubjectName	Marks
101	C101	C	57
102	C201	DBMS	72
103	C102	C++	60
104	C202	Oracle	75

5.7 Normalized Forms: -

Why we need normalization: -

We normally use DBMS systems to efficiently access data, to manage data, to perform suitable operations on it. So if the data is in normal form it is very easy to access it as well as manage it for the system.

By doing normalization we can achieve: -

- Elimination of redundant data
- Ensure data dependencies means table only store related data.
- It will increase quality of database design.
- It also reduce amount of space that databases used. So it is very useful to use normalization.

Definition of Normalization: -

It is the process of decomposing a relation (table) based on functional dependency and primary key.

Types of Normalization: -

- Un-Normalized Form
- First Normal Form (1 NF)
- Second Normal Form (2 NF)
- Third Normal Form (3 NF)

5.7.1 Un-Normalized Form: -

What is Un-Normalized Form?

It is a relation that contains non-atomic values. A **non-atomic value means** it can be decomposed .To create UN normalize form we must list attribute of entity, identify repeating groups of attributes, their keys, identify main keys.

Example: Un-normalized Student Table:

RollNo	Name	Course_Code	CourseName	Fees
123	Ravi	C102	C	2500
		C103	C++	1200
		C104	OOPs	3200
124	Sumit	C102	C	2500
		C103	C++	1200
125	Trupta	C102	C	2500
		C103	C++	1200
		C104	OOPs	3200

Figure. 5.7.1: Un-normalized Student Table

For UnNormalize table we identify repeating groups in above tables that is CourseName, Course_Code and Fees. Key in this table is RollNo because all fields of table depend on RollNo.

5.7.2. First Normal Form (1NF): -

For 1NF it is must that each cell of the table must have single value i.e. atomic. Every row of table contains exactly one value for each attribute. No two rows in a table may have similar values.

For converting table in 1NF we must follow rules: -

1. Remove duplicate columns from the same table.
2. After removing duplicate columns, you have to make separate tables for each group of related data and recognize each row with a unique column (the primary key).

Steps to convert UNF into First Normalize Form: -

- First you have to identify groups of attributes that can act as key for UnNormalize table.
- In second step you have to identify repeating groups and separate repeating groups. (Repeating Groups are Fields that may be repeated several times for one entity)
- In last step you have to remove repeating group by entering proper data into empty columns of rows containing the repeating data.

Example: Consider Un-normalized Student Table in Figure. 5.7.1

Convert it to Normalized Student table as below:

RollNo	Name	Course_Code	CourseName	Fees
123	Ravi	C102	C	2500
123	Ravi	C103	C++	1200
123	Ravi	C104	OOPs	3200
124	Sumit	C102	C	2500
124	Sumit	C103	C++	1200
125	Trupta	C102	C	2500
125	Trupta	C103	C++	1200
125	Trupta	C104	OOPs	3200

From above table you will notice that, each row contain unique combination of values. It contains only atomic value. We cannot decompose values. So it is in 1NF.

5.7.3. Second Normal Form (2NF):-

After producing first normal form we found redundant data in tables so we use second normal form to normalize it again. Second normal form is use to reduce the amount of redundant data in a table. This is done by extracting it and placing that redundant data in new table(s) and creating relationships between those tables.

We can say that a relation R is said to be in 2NF if relation is in 1NF and every attribute is depend on Primary key.

Steps to convert 1NF into Second Normalize Form: -

- In first step you have to remove partial dependencies.
    - If primary key of relation is single attribute means that it has only one value, and then we did not require to convert it in 2NF.
    - But if primary key exists in relation is not single means primary key is composite attributes then we require to identify primary key and functional dependencies that exist in relation for 1NF.
        - If any partial dependency is depending upon primary key, then remove those dependencies by placing them in new relation along with copy of determinant.

Let's look at an example.

Example: -Consider Student table in Figure 5.7.1 in that table we reduce redundant data by extracting it from 1NF and placing that redundant data in new table so we create relationship between those table as shown below. We remove partial functional dependencies to produce 2NF.

In that table RollNo is primary key .All other attributes Name, Class dependent upon primary key, but attribute fees is upon CourseName which is not primary key of student table .So we remove partial functional dependencies to produce 2NF.

We can observe that Name attribute completely depend upon primary key RollNo whereas Fees is depend on Course_Code. Course_Code cannot individually determine unique record .We need RollNo along with Course_Code for determine Fees attribute.

We get two separate tables as shown below.

Table1: -RollNo, Name

Table2: -RollNo, Course_Code, CourseName, Fees

Following tables shows 2NF

RollNo	Name
123	Ravi
124	Sumit
125	Trupta

RollNo	Course_Code	CourseName	Fees
123	C102	C	2500
123	C103	C++	1200
123	C104	OOPs	3200
124	C102	C	2500
124	C103	C++	1200
125	C102	C	2500
125	C103	C++	1200
125	C104	OOPs	3200

5.7.4. Third Normal Form (3NF): -

In 2NF we found some columns that are not fully dependent upon primary keys exist in table so we need to convert relation into 3NF .In 3NF we first check that relation satisfies 1NF and 2NF, after that we remove columns that are not fully dependent upon the primary key. i.e. we remove transitive dependencies.

So we can say that a relation R is in 3NF, if it is in 2NF and it has no transitive dependency.

Before see an example remind what is transitive dependency?

If A->B, B->C, this statement implies that A->C.
Example: EmpId->Ename and Ename->EmpAddress
i.e. EmpId -> EmpAddress.

Steps to convert 2NF into Third Normalize Form: -

- First necessary condition is that relation must be in 2NF.
- In next step you have to identify transitive functional dependencies that exist in relation .If transitive functional dependency is exist then remove that dependencies by placing them in new relation along with copy of determinant.

Example: Determine the functional dependencies. Remove partial dependency and transitive dependencies in Figure 5.7.4.3 i.e. convert it into 3NF.

Student = (RollNo, Name, Course_Code, Course_Name, Fees)

RollNo	Name	Course_Code	CourseName	Fees
123	Ravi	C102	C	2500
123	Ravi	C103	C++	1200
123	Ravi	C104	OOPs	3200
124	Sumit	C102	C	2500
124	Sumit	C103	C++	1200
125	Trupta	C102	C	2500
125	Trupta	C103	C++	1200
125	Trupta	C104	OOPs	3200

Figure. 5.7.4.3 Student table

Solution: -Remove transitive dependencies.

In above example Name is depend upon Primary key RollNo and other attribute Fee is depend upon CourseName, which is depend upon Course_Code.

RollNo -> Name

Transitive dependencies: - Course_Code->CourseName->Fees

First check is their partial dependents or transitive dependencies exist in given relation? If transitive dependencies exist then remove it by divide the relation into new relations.

Personal (RollNo, Name, Course_Code)

Course (Course_Code, CourseName, Fees)

Implementing the Split up would mean:-

Personal

RollNo	Name	Course_Code
123	Ravi	C102
123	Ravi	C103
123	Ravi	C104
124	Sumit	C102
124	Sumit	C103
125	Trupta	C102
125	Trupta	C103
125	Trupta	C104

Primary Key

Course

Course_Code	CourseName	Fees
C102	C	2500
C103	C++	1200
C104	OOPs	3200

Foreign key is an attribute that appears as a non-key attribute in one relation and as a primary key in one

5.8. De-normalization:-

De-normalization is the process of converting normalized form into unnormalized form. It is a technique to move from higher to lower normal forms of database.

We used denormalization for following purposes: -

- To optimize the performance of a database.
- To speed up database access
- Sometime users want to access data in different ways.
- In database many critical queries and reports exist which depends upon data from more than one table.
- Sometime users want to apply many calculations to one or many columns for expected result.

Advantages of Denormalization: -

Denormalization gives Performance benefits by: -

- Minimizing the need for joins.
- Precomputing aggregate values, that is, computing them at data modification time, rather than at select time
- Reducing the number of tables, in some cases

Disadvantages of Denormalization:-

- It makes updating process slow.
- It increase the size of tables.
- It makes coding more complex.
- It is mostly application specific .So if application changes, it need to re-evaluate.

Summary: -

In this chapter we learnt what is decomposition also Lossy and loss less decomposition. We focused on normalization, different forms of normalization. Along with that we learnt de-normalization, and their types.

- Decomposition:-

  If **R** is a relation scheme then $\{\mathbf{R_1, R_2, ..., R_n}\}$ is a decomposition
  if $\mathbf{R = R_1 \; U \; R_2 \; U \; ... \; U \; R_n}$

- Lossy Decomposition: -

  If **R** is a relation scheme and $\mathbf{R' = R_1 \; U \; R_2 \; U \; ... \; U \; R_n}$
  If we get $\mathbf{R \neq R'}$ then is called as Lossy decomposition.

- Loss less Decomposition: -

  If **R** is a relation scheme, if $\mathbf{R' = R_1 \; U \; R_2 \; U.. \; U \; R_n}$ if we get **R=R'** then is called as Loss less decomposition

- Functional Dependency**: -** one entity dependents on another entity

- Full Functional Dependency:-If relation R and Functional Dependency X->Y,Y is fully functionally dependent on X and there should not be any Z->Y, Where Z is a proper subset of X.

- Partial Functional Dependency: - Dependency in which on one or more non-key attributes are functionally dependent on part of the primary key

- Transitive Functional Dependency: - A Functional dependency between two (or more) non-key attributes in relation.

- Normalization: - It is the process of decomposing a relation (table) based on functional dependency and primary key.

- 1NF: - A Relation is said to be in 1NF if each cell of the table must have single value.

- 2NF: - if it is in 1NF, and each and every attribute is depends on Primary key.

- 3NF-if it is in 2NF, it has no transitive dependency.

- Denormalization: - it is the process of attempting to optimize the performance of a database by adding redundant data or by grouping data.

Multiple Choice Questions: -

1. Goals for relational Database Design is __________________

a) Avoid redundant data
b) Ensure that relationships among attributes are represented
c) Facilitate the checking of updates for violation of database integrity constraints.
d) All of above

2. If **R** is a relation scheme, if **R' = R_1 U R_2 U.. U R_n** if we get **R=R'** then is called as ________decomposition.

a) Loss less b)Lossy C
b) Both a and b d)None of above

3. Functional dependencies (FD) used to express __________ between attributes.

a) Constraints b)Values
b) Both a and b d)Properties

4. _________ is a relationship between or among attribute.

a) Functional dependency
b) Value dependency
c) Both a and b
d) None of Above

5. A Functional dependency between two (or more) non-key attributes in relation is known as ______________.

a) Bi-directional Functional Dependency
b) Transitive Functional Dependency
c) Full Functional Dependency
d) Partial Functional Dependency

6. A _________ is called as tuple generating dependencies.

a) Multivalued Functional Dependency
b) Transitive Functional Dependency
c) Full Functional Dependency
d) Partial Functional Dependency

7. The process of decomposing a relation (table) based on functional dependency and primary key is known as ________.

a)Decomposition b)Lossy Decomposition
c)Normalization d)Denormalization

8. A Relation is said to be in ________ if each cell of the table must have single value.

a)UnNormalize Form b)Second Normal Form
c) First Normal Form d) None of Above

9. The process of attempting to optimize the performance of a database by adding redundant data or by grouping data is known as __________.

a)Normalization b)Denormalization c)Decomposition d)None of Above

10) Denormalization gives Performance benefits by: -

a) Minimizing the need for joins.
b) Precomputing aggregate values, that is, computing them at data modification time, rather than at select time
c) Reducing the number of tables, in some cases
d) All of above

****

Chapter 6

Transactions

Topics to Learn

6.0 Introduction
6.1 Transaction Concept
6.2 Properties of Transactions
6.3 Transaction Terminology
6.4 Transaction States
6.5 Concurrent Execution of Transactions
6.6 Operations on a Transaction
6.7 Concurrency Control
6.8 Schedules
6.9 Recoverability

6.0 Introduction

In our day-to-day life we perform several activities like purchasing of groceries, vegetables, paying bills, etc. Each activity consists of many steps. For example, while paying an electricity bill, we have to go to the bank, withdraw money from bank, then go to the electricity office, stand in queue and finally pay the bill and again go home. Paying electricity bill is one activity but it contains many steps to perform it. Similarly Transaction is a single activity, which consists of many individual steps to perform the work. That's why transaction is always considered as an atomic unit or logical unit.

In this chapter we will first see the transaction concept and how a particular transaction is carried out. Then we will see the properties of transaction, which every transaction should satisfy. Then we will study the transaction terminology in which we will go through the basic concepts that will be used through out the chapter. A transaction before its completion travels through many states. These states will be explained in transaction states.

In any computing system, there are many programs executing in the system at the same time. This concept and the problems related to concurrency will be described in concurrent execution of transactions. There are various operations, which can be performed on the database; we will then see the operations permissible on the database. After operations, we will see the concept of schedules. A schedule is the order in which transactions are executed in the system.

Serializibility is the mechanism used to control the concurrent execution of transactions. We will study the concept of Serializibility and also its types. The last topic covered in this chapter is Recoverability. Recoverability is removing the effects of incomplete transactions. Then we will study the various Recoverable Schedules.

So let's start with the Transaction Concept.

6.1 Transaction Concept

The execution of a program that includes database access operations is called the transaction. A transaction is a unit of program execution that accesses and updates various data items. Whenever we read from and/or write (update) a database, we create a transaction. If the database operations in a transaction do not modify the data in the database, but only reads the data, then the transaction is called as the read-only transaction. A transaction is a single unit of database operations that are either completed entirely or not performed at all. The transaction should be completed or aborted, no intermediate states are allowed. The transaction consists of simple SQL statements such as select, insert, delete, etc. The more common transactions are formed by usually two or more database requests.

E.g..: Consider a simple example of calculating percentage of six subjects:

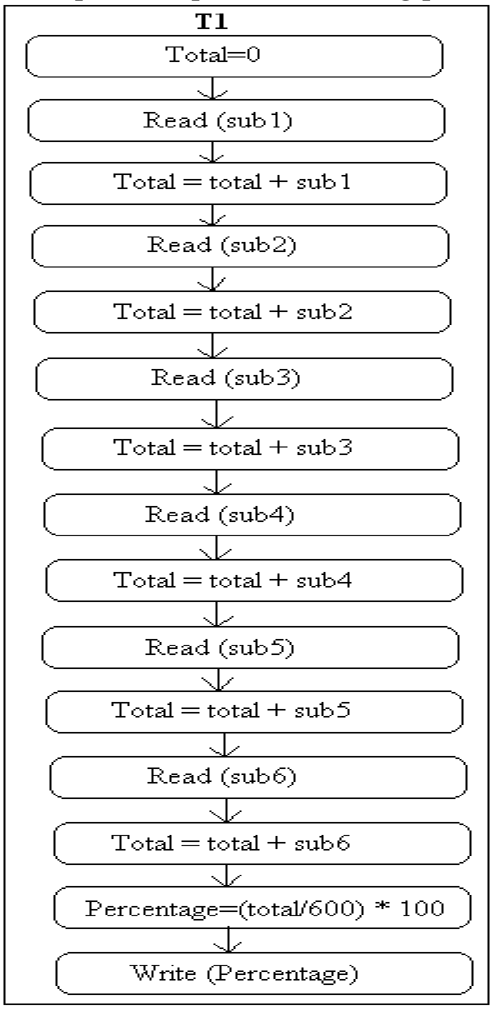


Figure 6.0: Simple transaction.

In this example, we first read the value of each subject and then add it in the value of total. After the total is calculated, we calculate the percentage and value of percentage is written back to the database.

Database Management System controls and executes all the transactions so as to ensure that the database is always in a consistent state. The steps in which the transaction is executed are:

i) The data item sub1 is read from permanent storage to primary memory and added in the variable total, then sub2 is read and added in total and so on.
ii) Finally percentage is calculated.
iii) The updated value of percentage is written back from primary memory to the permanent storage.

When the transaction is executing, the DBMS must ensure that the data item used should not be accessed by any other transaction.

6.2 Properties of Transaction:

The atomic transaction has several properties. These are called as the ACID properties.

1) **Atomicity**: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.

2) Consistency Preservation: A correct execution of a transaction must take the database from one consistent state to another.

3) **Isolation**: A transaction should not make its updates visible to other transactions until it is committed. This property is useful in multi-user database environment because several different users can access and update the database at the same time.

4) **Durability or Permanency**: Once a transaction changes the database and the changes are committed, these changes must never be lost because of the software or hardware failures.

6.2.1 Consistency and Isolation

Users are responsible for ensuring transaction consistency, i.e. the user should make sure that the transaction would leave the database in a consistent state after its completion.

For example, consider the transaction to transfer money from one account to another. The consistency criterion is that funds transfer between the bank account should not change the total amount of money in the accounts. The user will debit one account leaving the database in an inconsistent state, but he will credit the second account to make the database consistent. So the database is in consistent state after the entire transaction is completed. If a faulty transfer takes place where the transaction credits the second account with half of the debited amount from the first account, then the DBMS cannot be expected to detect the inconsistencies due to such errors in the user's program's logic.

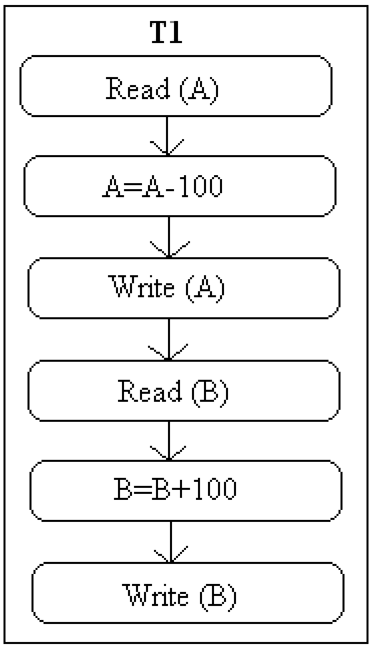


Figure 6.1: A consistent transaction.

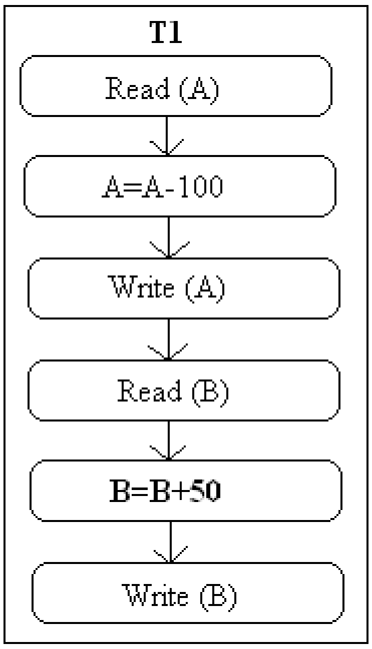


Figure 6.2: An inconsistent transaction.

The isolation property is ensured by guaranteeing that even though actions of several transactions are interleaved, the net effect is identical to executing all transactions one after the other in some serial order. If two transactions T1 and T2 are executed concurrently the net effect is guaranteed to be equivalent to executing T1 followed by T2 or T2 followed by T1. This is enforced by the concurrency control method.

6.2.2 Atomicity and Durability

Transactions can be incomplete for three kinds of reason:

1) Firstly it can be aborted, or terminated unsuccessfully, by the DBMS because of some internal reason, during execution. If the transaction is aborted by DBMS, then it will restart automatically and execute as a new transaction.
2) Secondly, the system may crash while one or more transactions are in progress.
   E.g. power failure.
3) Thirdly, a transaction may encounter an unexpected error and decide to terminate itself.
4)

DBMS maintains a record called log of all write operations performed on the database. To ensure the atomicity property, DBMS has to undo the actions of incomplete transactions. A transaction that terminates or aborts may leave the database in an inconsistent state. Thus DBMS has to remove the updates of such incomplete transactions by undoing the actions of those particular transactions.

The log is also useful in ensuring durability. If the system crashes before the changes made by a completed transaction is written to disk, the log is used to recognize and restore the changes when the system restarts. The DBMS component, which ensures atomicity and durability, is called the recovery manager.

6.3 Transaction Terminology

1) **Commit**: A transaction that completes the execution successfully is called as a committed transaction. The committed transaction should always take the database from one consistent state to another. The changes made by the committed transaction should be permanently stored in the database even if there is any system failure.
2) **Abort**: If there are no failures then all the transactions complete successfully. But transaction may not always complete its execution successfully then the transaction is called as aborted.
3) **Roll back**: If we want to obey the atomicity property then all the changes made by the aborted transactions must be undone. When we undo the changes of a transaction we say that the transaction has been rolled back.
4) **Restart**: If a transaction is aborted because of hardware or software failure, a transaction restarts as a new transaction
5) **Kill**: A transaction is killed, if there is some internal logic problem, or input problem or output problem.
6) **Throughput**: It is the average number of transactions completed in a given amount of time.
7) **Average response time**: It is the average time taken by a transaction to complete after it has been submitted.

6.4 Transaction States

The transaction may be in one of the following states while executing:

1) **Active**: This is the initial state. The transaction is in this state while it is executing.
2) **Partially Committed**: The entire transaction has been executed, but not yet committed.
3) **Failed**: The transaction cannot execute normally.
4) **Aborted**: The transaction is rolled back and the database is stored to its prior state.
5) **Committed**: After a successful completion.

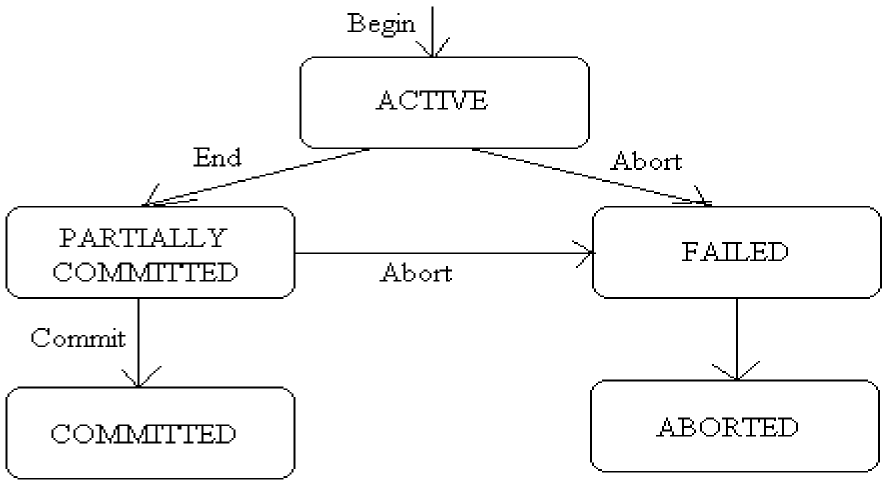


Figure 6.3: Transaction States.

A transaction starts its execution in active state. When the transaction completes, it enters the partially committed state. Here, the transaction has finished the execution, but some recovery protocols need to ensure that a system failure will not result in an inability to record the changes of the transactions permanently. Once this check is successful, the transaction is said to have reached the commit point and enters the committed state. Once a transaction is committed then it has finished the execution successfully and all its changes must be recorded permanently in the database.

The database system writes enough information to disk. So, that even in case of failure the updates of a transaction can be re-created when the system restarts. When this information is written then the transaction enters the committed state.

A transaction enters a failed state when the DBMS finds that the transaction is not executing normally. Such a transaction is rolled back. Then, it enters the aborted state. After a transaction aborts, it may restart or get killed.

Note: We should be careful while writing the changes to the database. Most systems allow such writes only after the transaction has entered the committed state.

6.5 Concurrent Execution of Transactions

If only one user can use the system then it is called as single-user system. E.g. Personal computers at home. If many users can use the system at the same time, then it is called as multi-user system. E.g..: For example, reservation system, bank system, stock exchange, etc. The database system can be classified as the number of users who can use the system concurrently, i.e. at the same time.

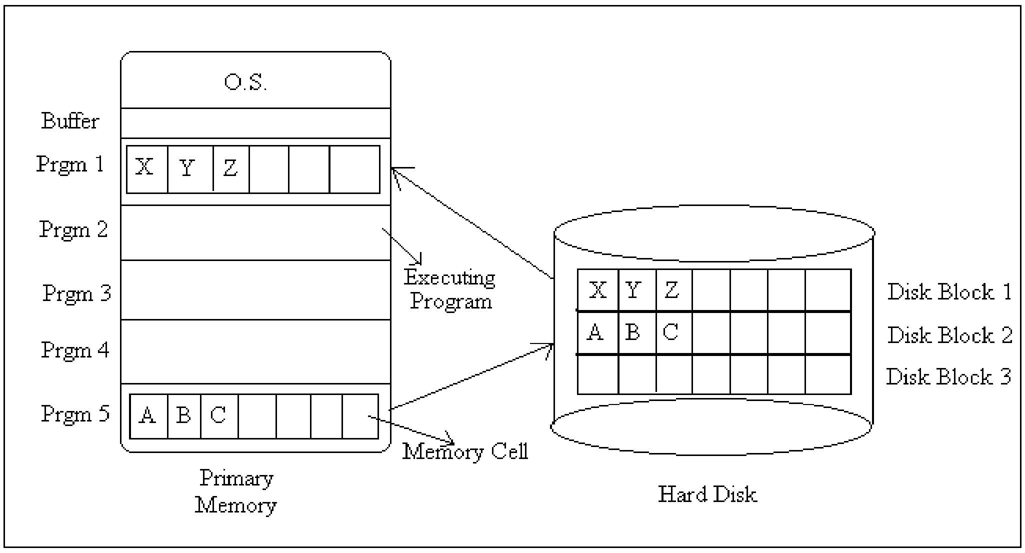


Figure 6.4: Multiprogramming System.

The Figure 6.4 represents the working of Multiprogramming system. There is primary memory and hard disk shown in the Figure. In multiprogramming system, there are various programs kept in memory at the same time and one program is in execution. Primary memory consists of operating system and programs. In primary memory, there is buffer. Buffer is small part of memory, which is used for temporarily storing a disk block. In this Figure, there are five programs Prgm 1 to Prgm 5 in memory and Prgm 2 is currently executing. The hard disk is also shown. On the hard disk, programs are stored in disk blocks. Each disk block consists of memory cells, which are used to store data items.

Multi-user systems are based on the concept of multiprogramming, which allows the computer to process multiple programs at the same time. In multiprogramming operating systems, it executes some commands from one program, then suspend that program and execute some commands from the next program and so on. A program is resumed again at the point it was suspended when it gets the CPU turn. Hence, many programs are interleaved.

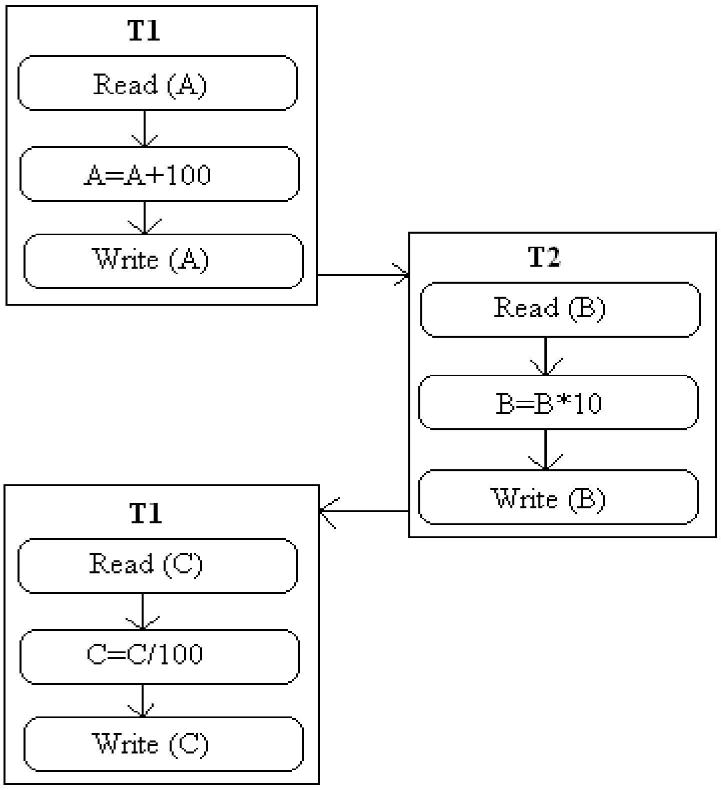


Figure 6.5: Interleaving of transactions.

Ensuring isolation property while performing such concurrent execution is difficult, but is required for performance reasons.

1) **Improved Throughput & Resource utilization**: While one transaction is waiting for a page to be read in from disk, the CPU can process another transaction. This is because I/O activity can be done in parallel with CPU activity in a computer. Overlapping I/O and CPU activity reduces the amount of time disks and processors are idle, and increases system throughput. The resource utilization also increases with decrease in idle time.

2) **Reduced Waiting Time**: Interleaved execution of a short transaction with a long transaction usually allows the short transaction to complete quickly. In serial execution, a short transaction could get stuck behind a long transaction leading to unpredictable delays in average response time.

6.6 Operations on a Transaction

We will deal with the transaction at the level of data item. The database access operations are:

1) Read-item (X): Reads the database item named X into a program variable. The Read-item (X) includes the following steps:
   - Find the address of disk block that contains item X.
   - Copy the contents of disk block into a buffer in main memory.
   - Copy the item X from the buffer to the program variable named X.

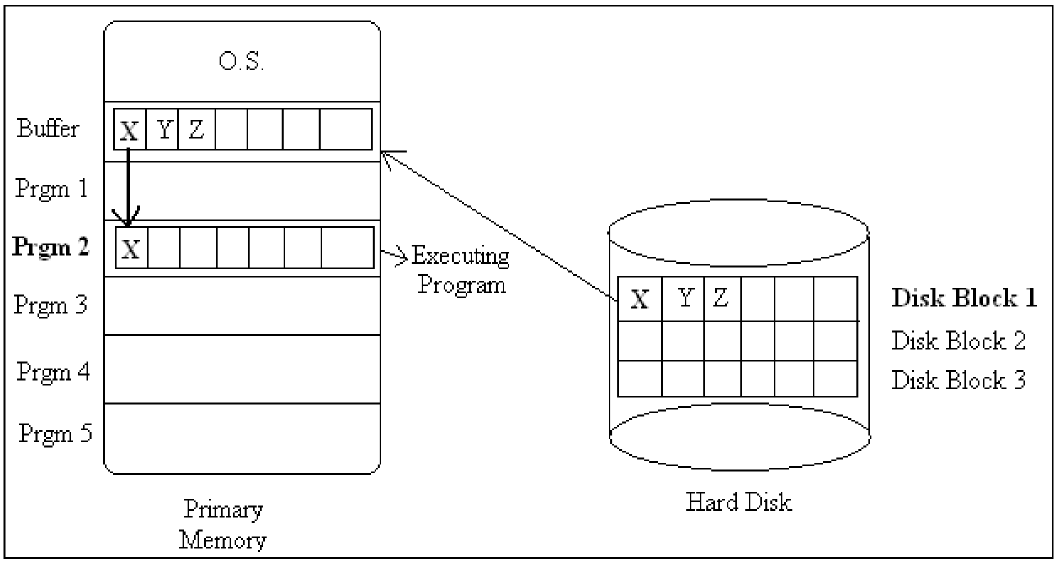


Figure 6.6: Read-item (X) operation.

2) Write-item (X): Writes the value of a program variable X into the database item named X. Write-item (X) includes the following steps:

- Find the address of the disk block that contains item X.
- Copy the contents of disk block into a buffer in main memory.
- Copy item X from program variable named X into its correct location in the buffer.
- Store the updated block from the buffer back to disk.

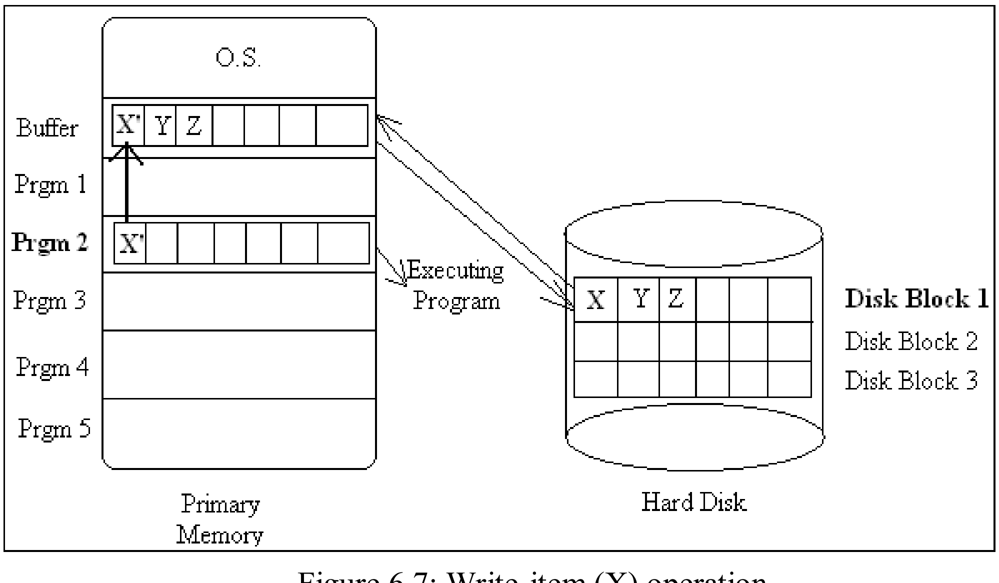


Figure 6.7: Write-item (X) operation.

Note: Here we assume that the program variable and the database item name are same.

6.7 Concurrency Control

In multi-user systems, programs are executed concurrently i.e. at the same time. Concurrency control is a mechanism used to ensure that the programs do not interfere with other executing programs. The major task performed by concurrency control protocols is to manage the concurrent operations on the database by executing programs. As we have seen, Database is an application that manages data and allows fast storage and retrieval of that data. Multiple users can concurrently share the data stored in the database. The database can remain consistent, if all the programs are only reading the data in the database. The problem arises when some programs are reading the data in the database and some program is updating the same data in the database, leaving the database in an inconsistent state.

Several problems may occur if concurrent executions are not controlled. We will
now study some of these problems. Consider a simple example of transfer of money between two accounts. Suppose there is a joint account (account operated by two persons) of Mr. Sharma & Mrs. Sharma. Mr. Sharma wants to make a payment of Rs. 1000/- to Mr. Malhotra. So, he performs a transaction T1 to transfer money to Mr. Malhotra's account. At the same time, Mrs. Sharma also performs a transaction T2 to deposit Rs. 2000/- in their joint account. We refer to Mr. Sharma's account as "S" and Mr. Malhotra's account as "M". The problems, which can arise, are as follows:

6.7.1 Multiple Update Problem

Multiple Update problem occurs when two transactions access the same database item and are interleaved in a way, which results in incorrect final value of the database item.

Suppose that T1 and T2 have started approximately at the same time and the operations are interleaved as shown in Figure 6.8. Also assume that the value of S is 10000 before transactions T1 and T2 started their execution.

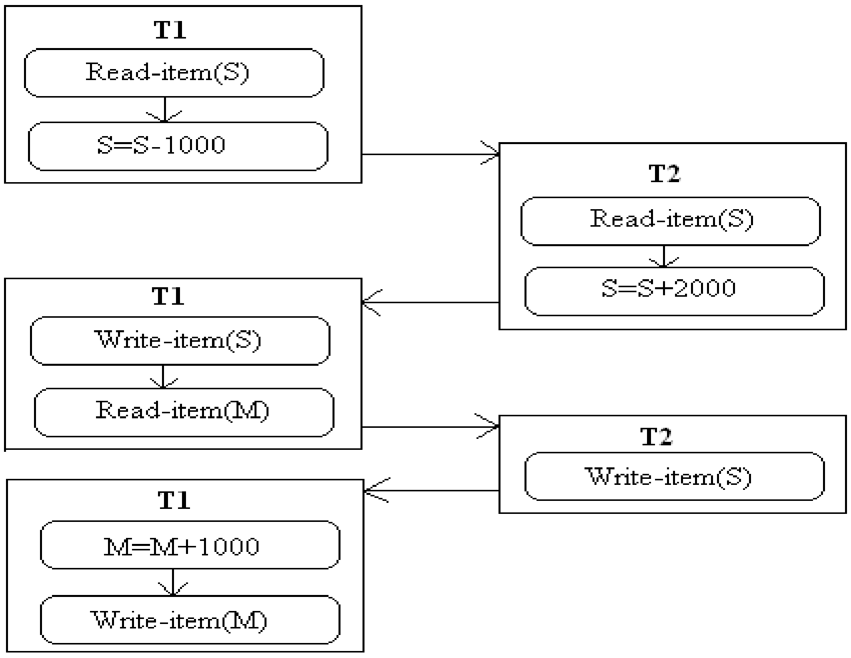


Figure 6.8: Multiple update problem.

As shown above, transaction T1 first reads the value of data item S. It then subtracts Rs. 1000/- from S. Now the value of S is Rs. 9000/-. After subtraction, transaction T1 is interleaved and transaction T2 starts its execution. Transaction T2 also reads the value of data item S that was read by T1. Transaction T2 adds Rs. 2000/- to the value of S, which makes the value of S equal to Rs. 12000/-. Transaction T2 is suspended and T1 resumes its execution. T1 writes the updated value i.e. 9000 to the database. It then reads the value of M and interleaves again. After this T2 also performs write operation. It writes the value of S to the database i.e. 12000, which is incorrect. The value of S updated by T1 is lost, leaving the database in an inconsistent state. The final value of S is Rs. 12000/-, which is incorrect. The actual value of S after transactions T1 and T2 should be Rs. 11000/-

6.7.2 Uncommitted Dependency (Dirty Read) Problem

The Uncommitted Dependency problem occurs when one transaction updates a database item and writes it into the database and after some time it fails due to some reason. This updated value is accessed by another transaction before it is changed back to its original value. The updated value read by the transaction is called as dirty read because it has been modified by the transaction that has not completed or committed.

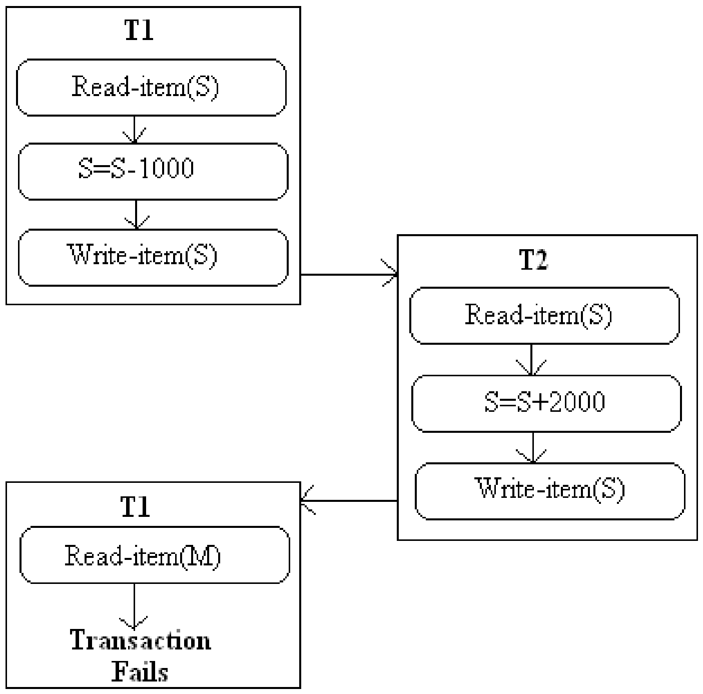


Figure 6.9: Uncommitted Dependency problem.

Here, transaction T1 changes the value of S and fails. So the system must discard this change. Before it can be done, the transaction T2 reads the temporary value of S, which is incorrect. The value of item S that is read by T2 is called as dirty read.

6.7.3 Incorrect Analysis Problem

If the transaction is calculating an aggregate function on a number of records while the other transaction is updating some of these records then the aggregate functions may take some values before updating and other after updating. For example, let's add a new account of Mrs. Sharma, which she operates individually. And she performs a transaction to calculate the balance of her account and the joint account. We refer to this new account as "X".

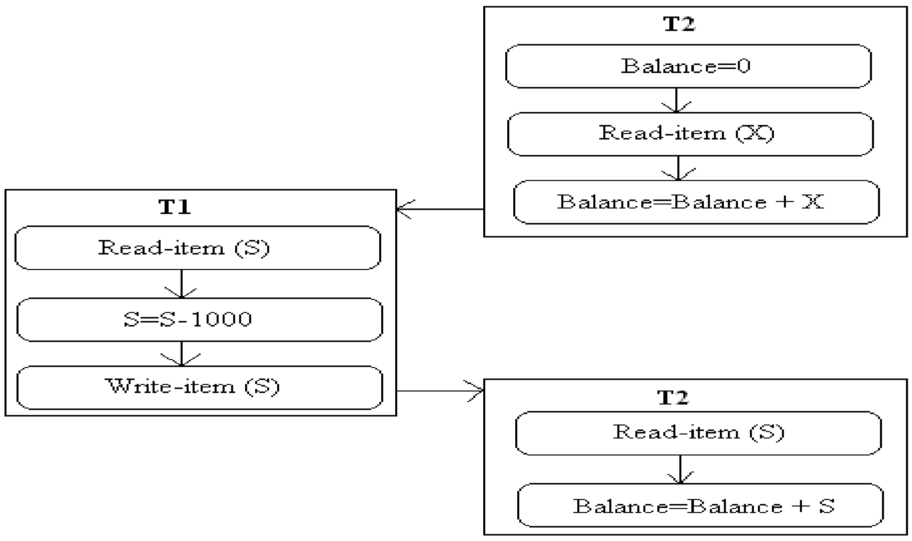


Figure 6.10: Incorrect Analysis Problem.

Here, transaction T2 has read the value of X before updating and the value of S after updating. So, the final value of balance is incorrect.

Another problem that may occur is the **unrepeatable read**. Here the transaction T1 reads an item twice, and the item is change by another transaction T2 between the two reads. Here, T1 will receive different values for its two reads for the same item even though T1 has not modified the item in between.

6.8 Schedules

Instructions in the transaction are executed in a particular sequence to accomplish the task. Schedules are used to represent the sequence in which the instructions are executed. A schedule is a timely ordered sequence of the instructions in the transactions. A schedule S of n transactions T1, T2, …Tn is an ordering of the operations of the transactions subject to the constraints that, for each transaction Ti that is in S, the operations of Ti in S must appear in the same order in which they occur in Ti.

The instructions from other transaction Tj can be interleaved with the operation of Ti in S. A schedule for a set of transactions must consist of all the instructions of those transactions and must maintain the order in which they appear in individual transaction.

Consider the simple banking system, which has number of accounts and a set of transactions, which access and update those accounts. Consider two transactions T1 and T2, which transfer funds from one account to another. Transaction T1 transfer Rs.500 from account A to account B, it is defined as,

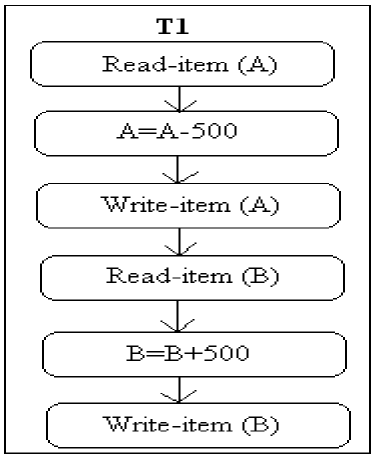


Figure 6.11: Transaction T1.

Transaction T2 transfers 25 percent of the balance from account A to account B. It is defined as,

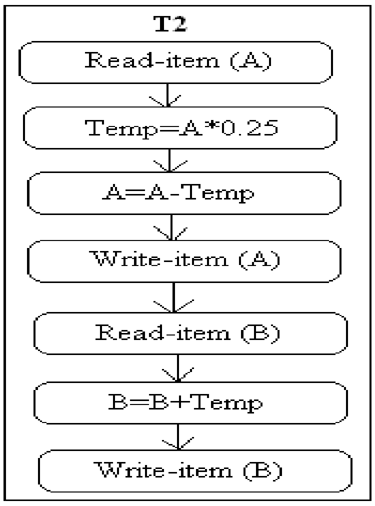


Figure 6.12: Transaction T2.

The schedules are serial schedules if it consists of a sequence of instructions from various transactions, where the instructions belonging to one single transaction appear together in that schedule for a set of n transactions there exist n! different valid serial schedules. Suppose, the values of A and B is Rs.1000 and Rs.2000 resp. The final values after schedule S1 are Rs.375 and Rs.2625.

When the system executes several transactions concurrently then schedule need not be a serial. If two transactions are running concurrently the system may execute one transaction for some time and other transaction for some time and then again switch back to first transaction. With concurrent execution CPU time is shared among all the transaction.

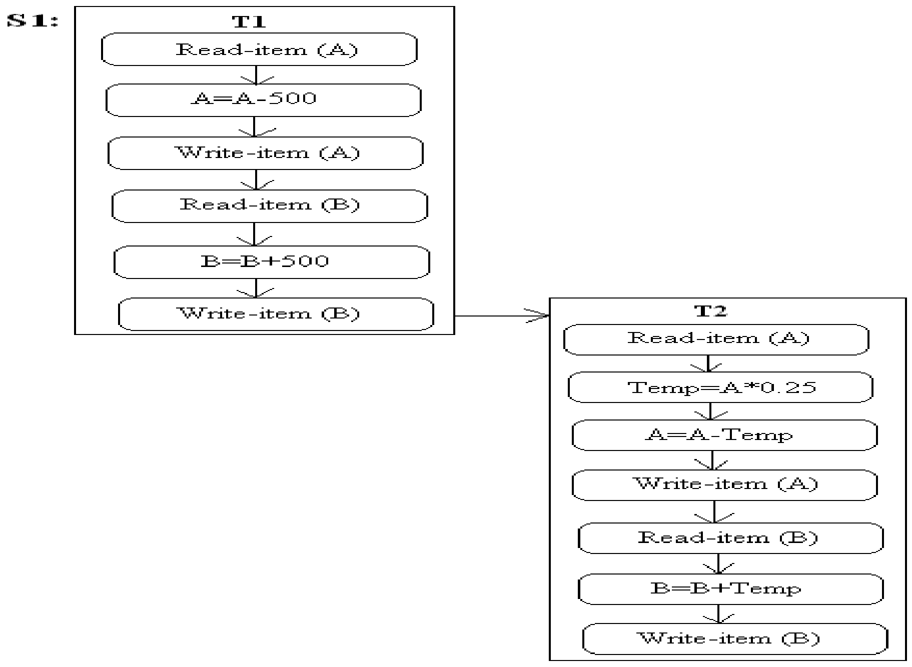


Figure 6.13: A serial schedule T1 followed by T2.

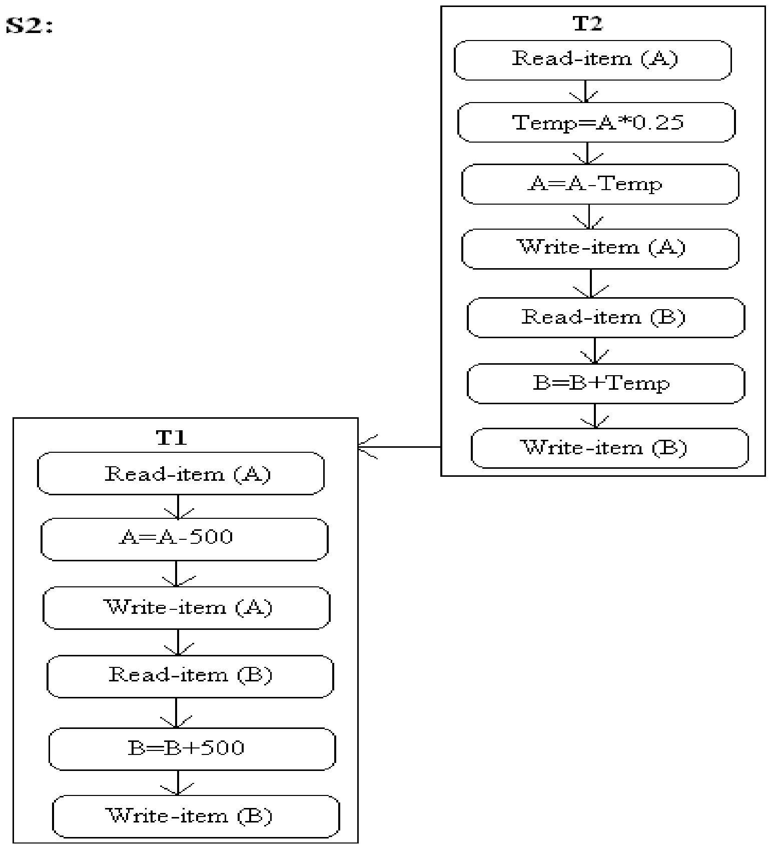


Figure 6.14: A serial schedule T2 followed by T1.

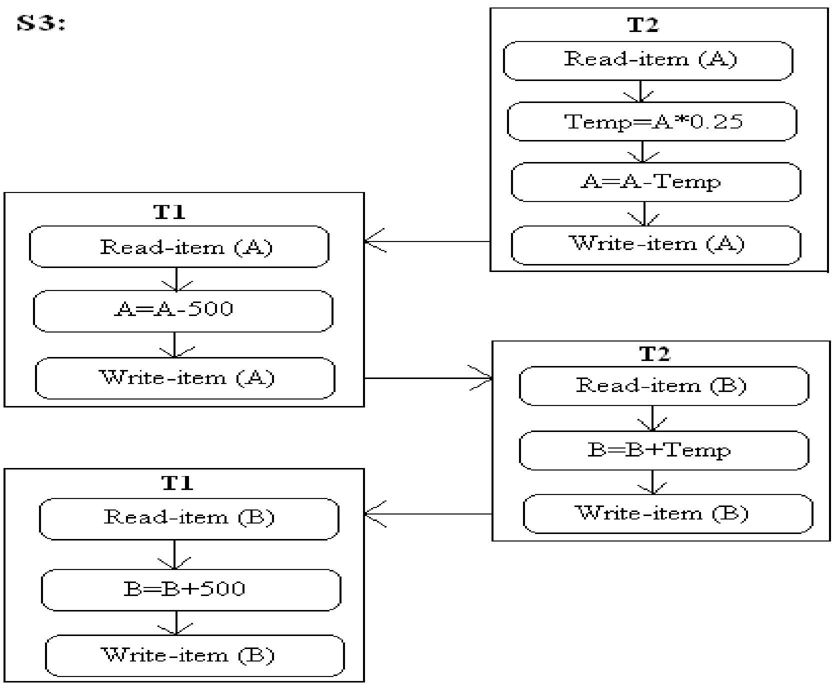


Figure 6.15: A concurrent schedule.

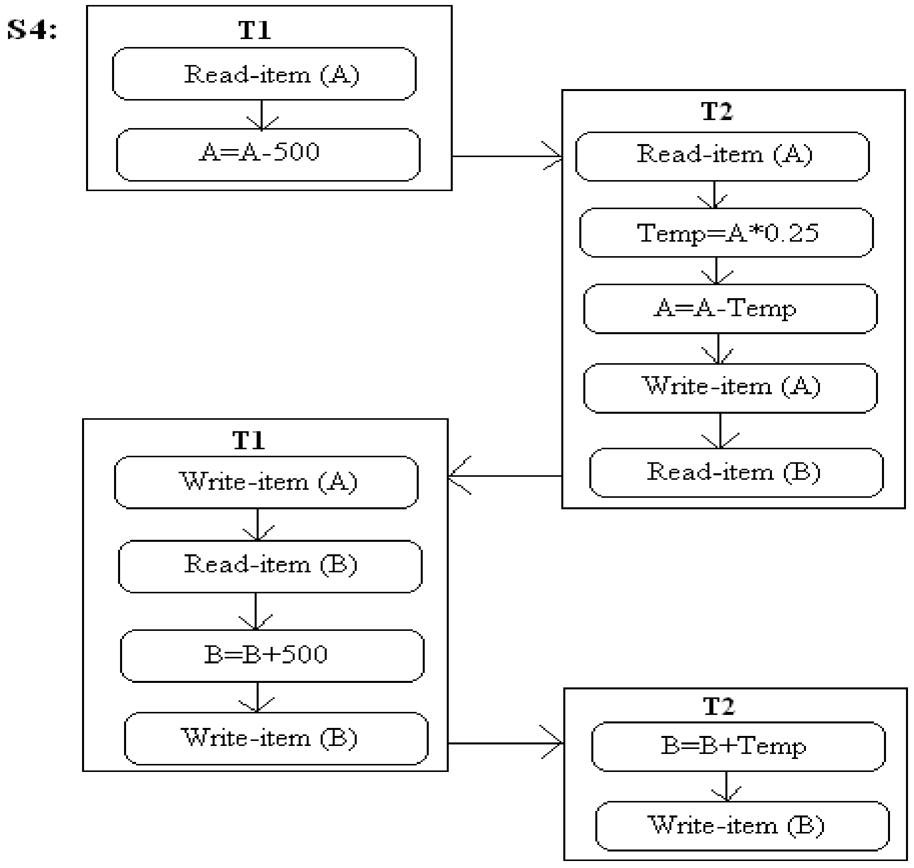


Figure 6.16: A concurrent schedule.

Several execution sequences are possible. The schedule S3 will produce the same result as schedule S1. But all concurrent executions may not result in a correct state. Consider the schedule S4. The final values of A and B are incorrect.

The final state is an inconsistent state. After executing the schedule S4, the values of A and B are Rs.500 and Rs.2125 resp.

It is the job of Concurrency control mechanism to ensure that schedules that are executed concurrently should always result as if they were executed in a serial manner. This is done to ensure the consistency of the database.

6.9 Recoverability

If a transaction fails we have to undo the effect of the transaction and bring the database to the consistent state. So when a transaction aborts, all the transactions dependent on the aborted transaction should also be aborted and rolled back. To achieve this, there are certain restrictions imposed on the type of schedules permitted in the system. To bring the database back to a consistent state, there is a limit on the acceptable schedules. We will now see the types of schedules:

6.9.1 Recoverable Schedules

A Recoverable Schedule is a schedule in which, if a transaction Tj reads a data item previously written by a transaction Ti, then the commit operation of Ti should appear before the commit operation of Tj. Suppose if transaction Ti fails for some reason, we need to undo the updates of this transaction. In the concurrent execution it is also necessary to ensure that any transaction Tj which was dependent on Ti is also aborted.

A recoverable schedule is shown in the following Figure:

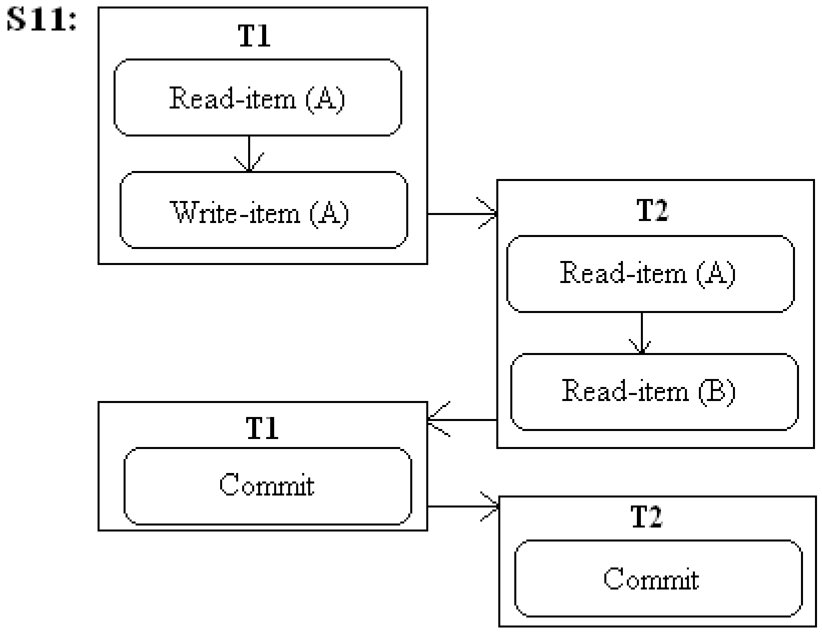


Figure 6.24: A recoverable schedule

Consider the example of two transactions T1 and T2. T2 has just one instruction read (A) and it commits before T1 does. Suppose T1 fails before it commits. Since T2 has read the value of data item X written by T1 we should abort T2 also. But T2 has already committed and cannot be aborted. So it is impossible to recover correctly from the failure of T1. So it is a non-recoverable schedule.

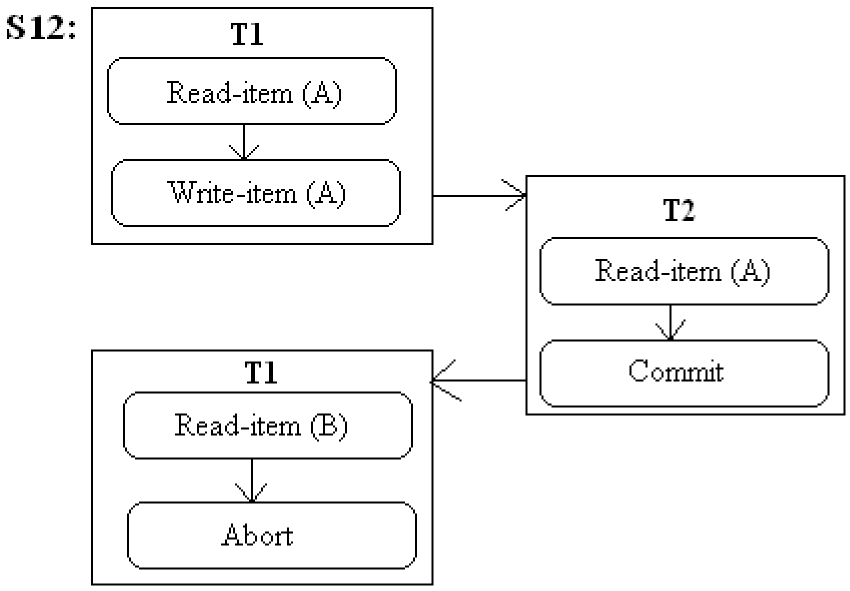


Figure 6.25: A non-recoverable schedule

Note: If a transaction Ti aborts, Tj would read an inconsistent database. Hence the database must ensure that schedules are recoverable.

6.9.2 Cascading Rollback

A single transaction failure leads to a series of transaction rollbacks. This is called as Cascading Rollback. Consider the following schedule where none of the transactions has yet committed (so the schedule is recoverable)

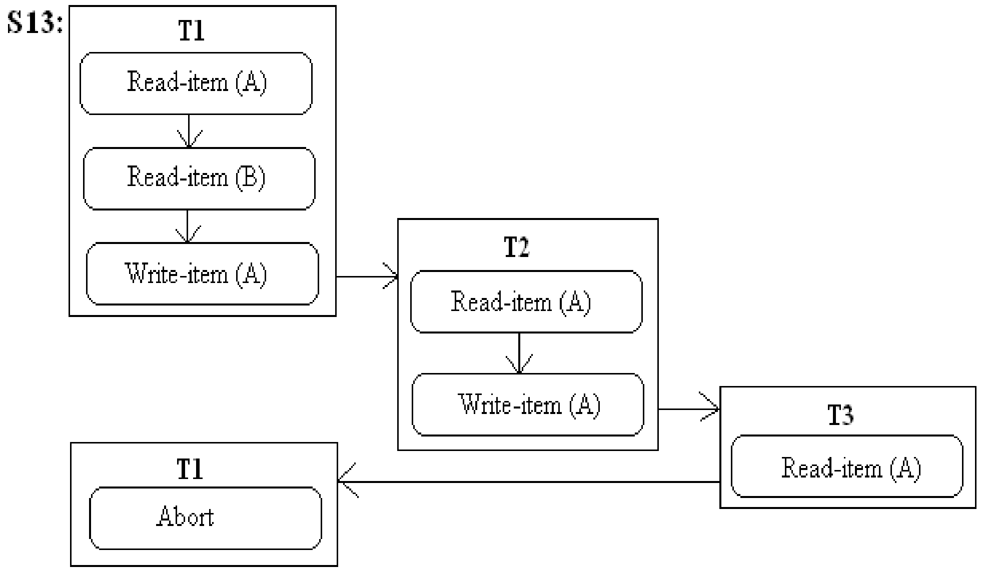


Figure 6.26: Cascading Rollback

If T1 aborts due to some reason, then the transactions T2 and T3 must also be rolled back because they have read the value of A which is updated by T1.
Note: Cascading rollbacks can lead to a large amount of work.

6.9.3 Cascadeless Schedules

For each pair of transactions Ti and Tj such that Tj reads a data item previously written by Ti, then the commit operation of Ti should appear before the read operation of Tj. This schedule is called as Cascadeless Schedule. In this type of schedule cascading rollbacks cannot occur.

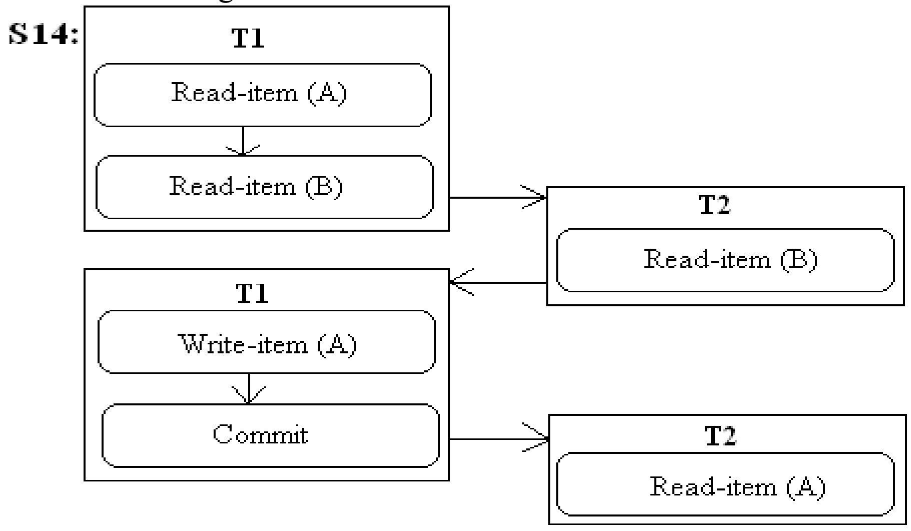


Figure 6.27: Cascadeless Schedule

Note: Every Cascadeless schedule is also recoverable.

6.9.4 Strict Schedules

Strict schedules as the name suggests is a more restrictive schedule than the other types of schedules. In this schedule, transactions cannot read or write a data item 'X' until the last transaction that wrote the data item 'X' has committed or aborted. Strict schedules simplify the recovery process. In a strict schedule, the process of undoing a write operation of an aborted transaction is easy because one simply has to restore the 'before-image' of the data item.

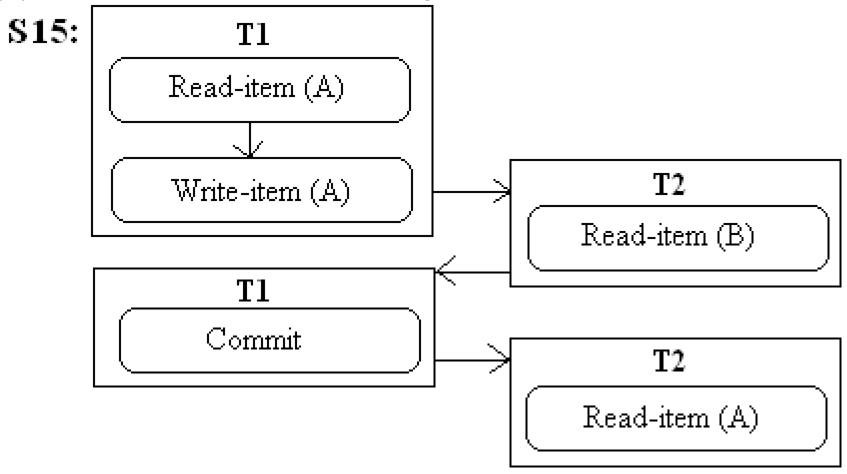


Figure 6.28: Strict Schedule S15

Summary

- A transaction is a unit of program execution that accesses and updates various data items.

- A transaction is a single unit of database operations that are either completed entirely or not performed at all. The transaction should be completed or aborted, no intermediate states are allowed.

- **ACID Properties**
    - **Atomicity**: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
    - **Consistency Preservation**: A correct execution of a transaction must take the database from one consistent state to another.
    - **Isolation**: A transaction should not make its updates visible to other transactions until it is committed.
    - **Durability**: Once a transaction changes the database and the changes are committed, these changes must never be lost.

- **Committed Transaction**: A transaction that completes the execution successfully is called as a committed transaction**.**

- **Aborted**: A transaction may not always complete its execution successfully then the transaction is called as aborted.

- **Roll Back:** When we undo the changes of a transaction we say that the transaction has been rolled back.

- **Transaction States**
    - **Active**: This is the initial state. The transaction is in this state while it is executing.
    - **Partially Committed**: The entire transaction has been executed, but not yet committed.
    - **Failed**: The transaction cannot execute normally.
    - **Aborted**: The transaction is rolled back and the database is stored to its prior state.
    - **Committed**: After a successful completion.

- **Operations on Transaction**
    - **Read-item (X)**: Reads the database item named X into a program variable.
    - **Write-item (X)**: Writes the value of a program variable X into the database item named X.

- **Concurrency control**: Concurrency control is a mechanism used to ensure that the programs do not interfere with other executing programs. The major task performed by concurrency control protocols is to manage the concurrent operations on the database by executing programs

- **Schedules**: Schedules are used to represent the sequence in which the instructions are executed in the system.

- **Recoverability**: If a transaction fails we have to undo the effect of the transaction and bring the database to the consistent state.

- **Recoverable Schedule**: A Recoverable Schedule is a schedule in which, if a transaction Tj reads a data item previously written by a transaction Ti, then the commit operation of Ti should appear before the commit operation of Tj.

- **Cascading Rollback**: A single transaction failure leads to a series of transaction rollbacks. This is called as Cascading Rollback.

- **Cascadeless Schedule**: For each pair of transactions Ti and Tj such that Tj reads a data item previously written by Ti, then the commit operation of Ti should appear before the read operation of Tj. This schedule is called as Cascadeless Schedule.

- **Strict Schedules** :In this schedule, transactions cannot read or write a data item 'X' until the last transaction that wrote the data item 'X' has committed or aborted.

Multiple Choice Questions

1) Which of the following is a transaction state?
   a) Active b) Commit c) Aborted d)All of the above

2) Which of the following is a transaction state when the normal execution of the transaction cannot proceed?
   a) Failed b) Active c) Aborted d) Committed

3) Which of the following is a problem resulting from concurrent execution of transaction?
   a) Incorrect analysis b) Multiple update
   c) Uncommitted dependency d)All of these

4) Which of the following is not one of the ACID properties?
   a)Atomicity b)Completion c)Isolation d)Durability

5) In which of the following state, the transaction is executing?
   a) Failed b) Active c) Committed d) Aborted

6) Removing the effects of transaction is called as
   a) Abort b) Commit c) Rollback d) Undo

7) In which schedule, the transaction is brought back to a consistent state after failure?
   a) Serial Schedule b) Recoverable Schedule
   c) Strict Schedule d) None

8) In which type of schedule, transactions are interleaved and then again resumed?
   a)Serial Schedule b)Non-Serial Schedule
   c) Concurrent Schedule d) None

****