Unit-1

Database Systems: Introducing the database and DBMS, Files and File Systems, Problems with File System and advantages of Database Management systems.
Data Models: The importance of Data models, Data Model Basic Building Blocks, Business Rules, The evaluation of Data Models, Degree of Data Abstraction.

Unit-II

The Relational Database Model: A logical view of Data, Keys, Integrity Rules, Relational Set Operators, The Data Dictionary and the system catalog, Relationships within the Relational Database, Data Redundancy revisited, Indexes, Codd's relational database rules.
Entity Relationship Model: The ER Model, Developing ER Diagram,

Unit-III

Normalization of database tables: Database Tables and Normalization, The need for Normalization, The Normal forms and High level Normal Forms, denormalization.

Unit-IV

Introduction to SQL: Data Definition Commands, Data Manipulation Commands, Select queries, Advanced Data Definition Commands, Advanced Select queries, Virtual Tables, Joining Database Tables.

Unit-V

Advanced SQL: Relational Set Operators, SQL Join Operators, Subqueries and correlated queries, SQL Functions, Oracle Sequences, and Procedural SQL.

UNIT -1

DATABASE SYSTEMS

INTRODUCING THE DATABASE AND DBMS:

Data:

Data is defined as a collection of raw facts or data items placed in a file.

Example: Student marks details (rno,name,marks), employee details(empno, name, salary) .

Information:

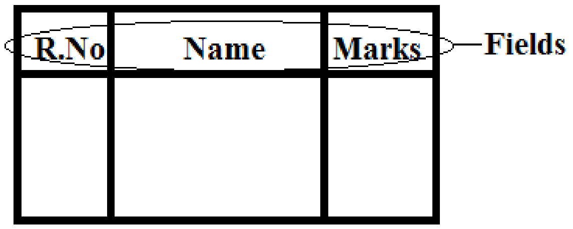
Information is the result of processing raw data to reveal its meaning. Processed data is called information or result of data processing.

Example: students who get 70 marks in computers.

Field:

A character or group of characters that has a specific meaning. A field is used to define and store data. It is also called attribute.

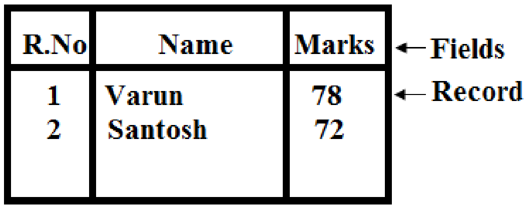
Example:



Record:

A record can be defined as collection of fields or attributes.

Example:

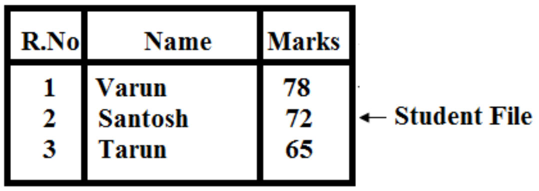


R.No	Name	Marks
1	Varun	78
2	Santosh	72

Fields
Record

File:

A File can be defined as collection of records.


← Student File

R.No	Name	Marks
1	Varun	78
2	Santosh	72
3	Tarun	65

Database:

a. A Database can be defined as collection of inter related files.
b. It is a shared-integrated computer structure that stores both end users data and meta data.
c. In database the data is independent of programs. In database data sharing refers multi users and programs processing done simultaneously.
   Example: Student marks system, Employee salary System

End Users:

The users who should process daily operations of an organization using database application program.

Meta Data:

Meta Data is nothing but Data about Data. It helps fastest data accessing mechanism.

It provides a description of data characteristics and set of relationships that link the data found within the database (self describing data called meta data)

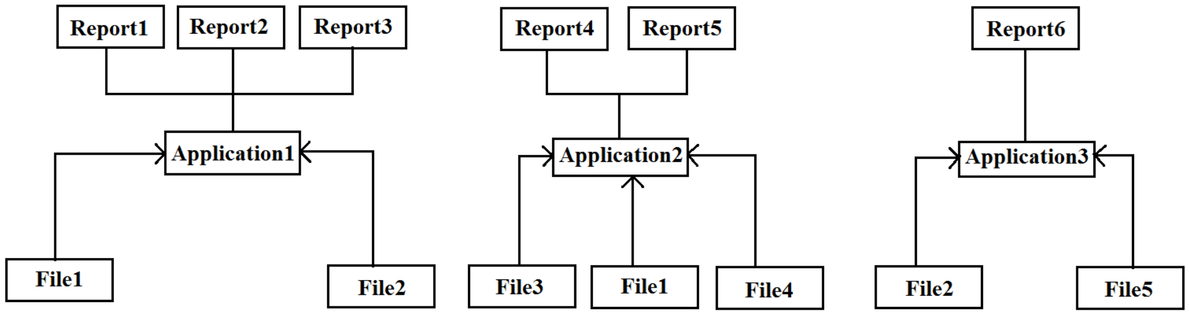
Example: Index of the text book.

DBMS:

a. DBMS is a collection of programs that are used to manage the database structure and controls the access of information stored in the Database.
b. A DBMS is a collection of database, database utilities (DBMS software) operated by end users or application programmers and finally administered by DBA.
c. A DBMS works as an interface between end-users and database.
d. DBMS is a software package that can be used to define, construct, manipulating the database using efficient query language.

FILES AND FILE SYSTEMS:

- ✓ In the conventional file processing system each and every subsystem maintains same files.
- ✓ These will be duplication (redundancy) of data between various subsystems of the information system.
- ✓ The concept of the conventional file processing system is shown below, which consists of the 3 applications namely App1, App2, App3.



- ✓ The different inputs and outputs of these applications are summarized below. Some files are duplicated in different subsystems in conventional file processing System.

Name of the Application	Input	Output
Application-1	File1 and File2	Report1, Report2, Report3
Application-2	File1 , File3 and File2	Report4, Report5
Application-3	File 2 and File 5	Report6

PROBLEMS WITH FILE MANAGEMENT SYSTEM:

- ❖ In File Processing System, there is the possibility to place the common files on different sub systems.
- ❖ It causes the problem of data redundancy.
- ❖ The following are the drawbacks of File management System.

1. Uncontrolled redundancy of data
2. Inconsistency of data
3. Inflexibility of data processing
4. Poor enforcement standards
5. Low programming productivity
6. Limited data sharing
7. Low data security
8. Excessive program maintenance
9. Doesn't support adhoc queries
10. Concurrent problem

1. Uncontrolled Redundancy of Data:
    a. In conventional file processing system each and every department has its individual data without coordination.
    b. The word redundancy refers to repetition (duplication) of data.
    c. In File processing system same data will be available in more than one subsystem.
    d. This will result in increased disk-space, increased time of entry and inconsistency of data.

2. Inconsistency of data:
    a. The uncontrolled redundancy of data will permit same data available in many different subsystems across the organization, so the inconsistency of data increased.
    b. While performing basic data processing operations like deletion, updating, retrieve etc this inconsistency will give misleading results.

3. Inflexibility of data processing
    a. Conventional File Processing System follows top-down approach.
    b. Entities and attributes/fields must be decided while designing the system in the beginning.
    c. As a result system may not be flexible to modify in future.

4. Poor Enforcement of Standards
    a. In CFPS, every subsystem will follow its own standards while finalizing the name of the field, field width, type etc.
    b. If any user made any modifications on the data standards of a file it effect the other users who works on the same file.
    c. There will be serious error due to mismatch of fields while integrating the files.
    d. It refers to poor enforcement of data standard.
5. Low Programming Productivity
    a. Programming productivity is a measure of time taken to develop an application.
    b. Whenever the program maintenance is high and takes more time to implement then automatically reduced the productivity.
    c. Inflexibility and Poor enforcement of standards causes low programming productivity in file management system.
6. Limited Data Sharing
    a. In File Management System various departments of the system managing the files without coordination.
    b. Each and every user has the right to access all files but he can able to access limited files only.
    c. File Management system is not centralized so it has limited data sharing.
7. Low Data security
    a. In Conventional File Processing System each subsystem will perform its own tasks individually.

   b. It is not fully secured because there is no private rules and passwords defined on each subsystem.
8. Excessive Program Maintenance
   a. In File Processing System data is not centralized.
   b. Every subsystem contains different files. It is not possible to modify the system software because we have to perform modifications to all the system individually.
   c. Because of this the maintenance of program in high.
9. Doesn’t Support Adhoc Queries
   a. Adhoc query refers the queries which were designed by using a strong syntactical rules of query language.
   b. In File Management System there is no Adhoc queries to access the records.
10. Concurrent Access Problem
   a. File Management System doesn’t support multi-user environment.
   b. It doesn’t support multiple users to access the records and manipulations at the same time.

ADVANTAGES OF DATABASE MANAGEMENT SYSTEM:

➔ DBMS receives all the application requests and translate them into Database understandable format to fulfill the user request.

➔ The following are the benefits/advantages of DBMS.

1. Reduced Data Redundancy
2. Consistency of Data
3. Flexibility of Data Processing
4. Better enforcement of standards
5. High Programming Productivity
6. Enhanced Data sharing
7. Improved Data Security
8. Reduced Program maintenance
9. Supports Adhoc queries
10. Supports Concurrent access

1. Reduced Data Redundancy
   a. The data of database is maintained at centralized location.
   b. As a result of centralization, repetition of data can be avoided.
   c. In DBMS we have key constraints which reduce the redundancy of data.
2. Consistency of Data
   a. In DBMS we can use key constraints to eliminate data redundancy.
   b. Hence there is no redundancy in DBMS, so the data is not mixed.
   c. The data is very consist for both end users and database server.
3. Flexibility of Data Processing
   a. Database System follows bottom-up approach.
   b. If any error occur at any level of the project it could not affect the other level.

   c. Each level has individual error detection and correction principles.
   d. So data is flexible for data processing.
4. Better Enforcement of Standards
   a. Database system follows bottom-up approach.
   b. Therefore various departments of the organization will follow a common notation while designing the attributes.
   c. It is very easy to integrate different files and generate required reports whenever required.
5. High Programming Productivity
   a. Since the database system free from redundancy of data, inconsistency of data and limited data sharing we can easily write programs on database in less time.
   b. So to implement the projects productivity is increased.
6. Enhanced Data Sharing
   a. In DBMS data will be stored in a centralized location.
   b. It provides an environment for end users to access and share more data easily.
   c. The end users have the rights to share and access any file at any time. There is no limitation of data access and sharing in DBMS.
7. Improved Data Security
   a. DBMS provides high security for files by using usernames and strong passwords.
   b. All files are under the control of DBA (Database Administrator).
   c. DBA should provide different access permissions to different types of users.
   d. Unauthorized persons cannot access data.
8. Reduced program maintenance
   a. DBMS follows both graphical and virtual representation of data.
   b. It needs a less number of programs for maintaining the files in DBMS.
   c. Different applications are developed under the coordination of the DBA.
9. Support Adhoc Queries
   a. DBMS follows strong syntactical query language.
   b. So it is easy for data accessing, retrieving and data modifications.
   c. So DBMS follows Adhoc query language rules.
10. Supports Concurrent Access
   a. Concurrent access means multiple users can access their information at the same time.
   b. DBMS supports parallel accessing as well as concurrent access.

DATA MODELS

THE IMPORTANCE OF DATA MODELS

❖ A Data Model is a relatively simple representation, usually graphical, of more complex real-world data structures.

- ❖ A Data Model represents data structures and their characteristics, relations, constraints, transformations, and other constructs with the purpose of supporting a specific problem domain.
- ❖ Data Models can facilitate interaction among the designer, the applications programmer and the end user.
- ❖ A Data Model can be defined as the blueprint of the database structure.

The Data Model includes the following:

1. Characteristics of Data Structure
2. Data Standards
3. The Relationship defined in database
4. Data base constraints
5. The Data type assigned to the fields
6. The Data transformation in between the fields

IMPORTANCE OF DATA MODEL:

- ↪ The Data Model maintains the communication in between database designers, application developers and end users.
- ↪ The Data Model gives a clear view of database structure included in Database design.
- ↪ The Data Model enforces the characteristics of data structure which were involved in database design.
- ↪ The Data Model gives specifications, relationships which were involved in database for application programmers.

TYPES OF DATA MODELS:

1. Hierarchical Data Model (HDM)
2. Network Data Model (NDM)
3. Relational Data Model (RDM)
4. Object Oriented Relational Data Model (ORDM)
5. Entity Relational Data Model (ERDM)
   The classification of data model done based on the relationships (associations).

DATA MODEL BASIC BUILDING BLOCKS:

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

→Entity:- It is a type of object in the real world.

Ex: a person, place, thing

Ex: customer table, employee table etc.

→Attribute:- An Attribute is a characteristic of an entity.

Ex: customer entity having attributes such as customer_name, phone, address etc.

→Constraints:- A constraint is a restriction placed on the data. Constraints are important, they help to ensure data integrities.

Ex: student marks between 0 to 75

Ex: Each class must have 60 students only.

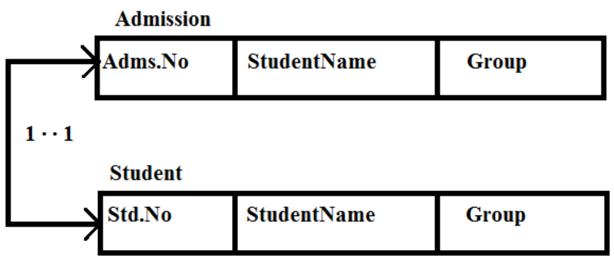
CLASSIFICATION OF RELATIONSHIPS:

The relationships or associations or mappings are classified as below.

a. One-to-One association (1:1 or 1..1 or < ------- >)
b. One-to-Many association (1:M or 1..M or <------>>)
c. Many-to-One association (M:1 or M..1 or << ------- >)
d. Many-to-Many association (M:M or M..M or << ------ >>)
e. One-to-One conditional association (1-0-1 or 1..0..1)

ONE-TO-ONE ASSOCIATION

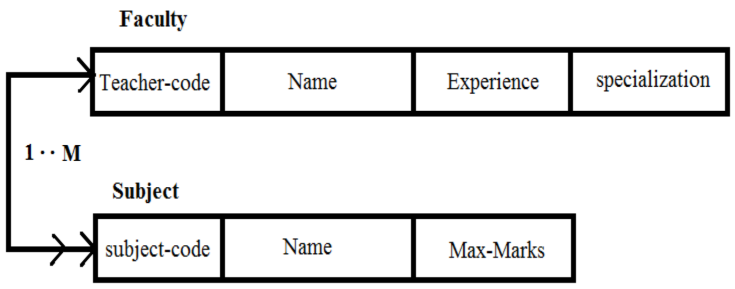
If a record of one file is associated with only one record of another file, such an association is said to be one-to-one association.



In the above example the association between Admission and Student relations is one-to-one association. That is a single admission number has only a single student number.

ONE-TO-MANY ASSOCIATION

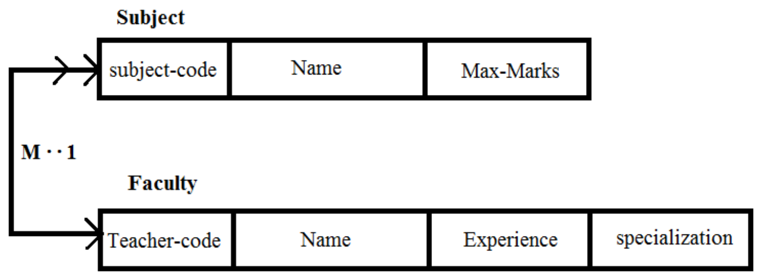
If a record of one file is associated with more than one record in another file, such an association is said to be one-to-many association.



In the above example the association between Faculty and Subject relations is one-to-many association. That is a single faculty can teach one or more subjects.

MANY-TO-ONE ASSOCIATION

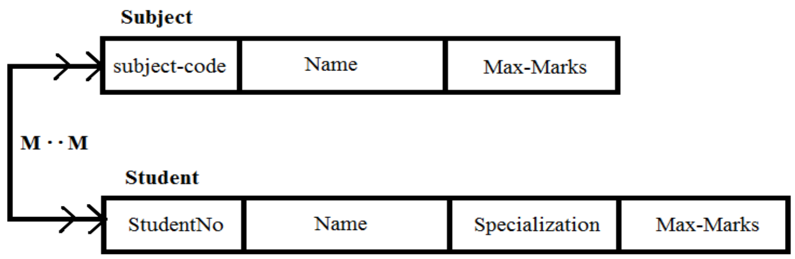
If multiple records of one file are associated with only one record in another file, such an association is said to be many-to-one association.



In the above example the association between Subject and Faculty relations is many-to-one association. That is multiple subjects may taught by a single faculty member.

MANY-TO-MANY ASSOCIATION

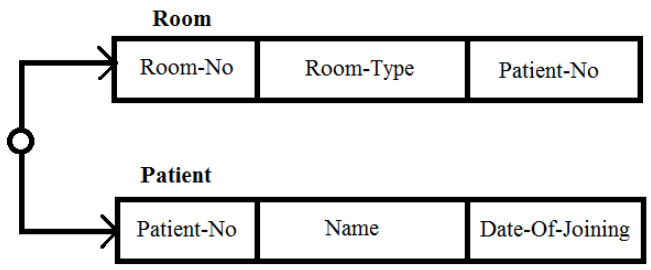
If each and every record of one file is associated with one or more records of another file, such an association is said to be many-to-many association.



In the above example the association between Subject and Student relations is many-to-many association. That is multiple subjects read by multiple students and multiple students may read multiple subjects.

ONE-TO-ONE CONDITIONAL ASSOCIATION

If a particular record of a file will be associated to only one record of another file for only a certain period based on some condition, such an association is said to be one-to-one conditional association.



In the above example the association between Room and Patient relations is one-to-one conditional association. That is a record of the Room file will have association with one record of the Patient file as long as the patient stays in that room, otherwise that record of the Room file will not have an association with the Patient file.

BUSINESS RULES:

- A business rule is a brief, precise and unambiguous description of a policy procedure of an organization.
- Business rules are used to define entities, attributes, relationships and constraints.
- Effective business rules must be easy to understand to ensure every person in the organization shares a common interpretation.
- The process of identifying and documenting business rules is essential to database design for the following reasons.
  1. They can be a communication tool between users and designers.
  2. They allow the designer to understand the nature role and scope of the data.
  3. They allow the designers to develop relationship participation rules and constraints to create an accurate data model.

EVOLUTION OF DATA MODELS (or) CLASSIFICATION DATA MODELS (or) TYPES OF DATA MODELS:

Based on the relationships the data models can be classified as below.

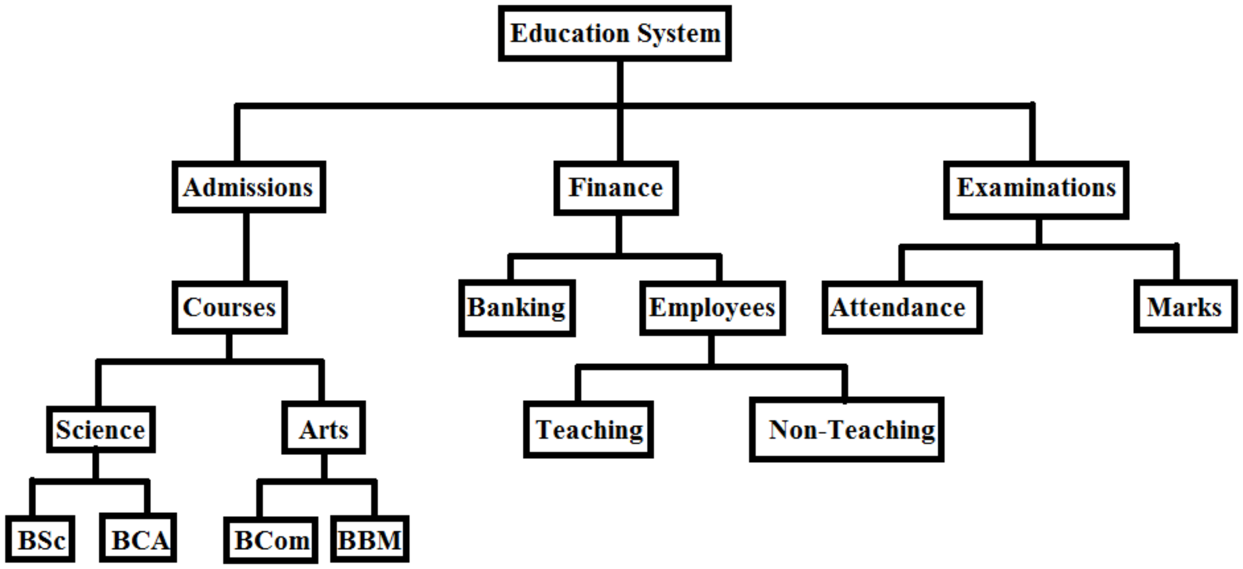
1. Hierarchical Data Model
2. Network Data Model
3. Relational Data Model
4. Object-Oriented Data Model
5. Entity Relational Data Model

HIERARCHICAL DATA MODEL

a. In this data model the structure of database is in the form of inverse tree.
b. This data model supports only one-to-one and one-to-many associations.
c. In this data model the data is placed in nodes. Nodes are basically two types.
   1. Parent / Super Node
   2. Child / Sub Node
d. The parent node / child node placed at one position represents levels.
e. Different number of levels placed in hierarchical data model are named as segments.

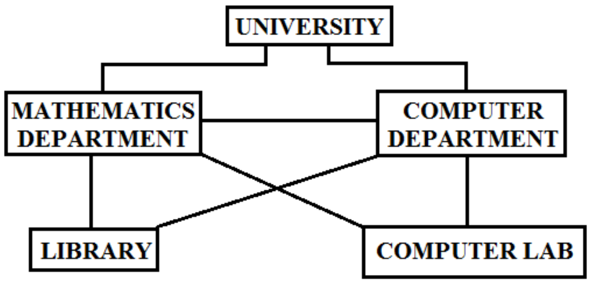
<u>Disadvantages (or) Limitations</u>

a. This model could not support many-to-many and many-to-one associations.
b. It is very difficult to maintain very complex relationship between application programs and end users.
c. This data model could not support Data Independency that is if any modification is done at any level it may effect another level.
d. The first implemented hierarchical database application program is “SYBASE”.



NETWORK DATAMODEL:

- The network data model was developed to overcome the limited scope of hierarchical model.
- Network data model can be used to design complex database structure.
- In network data model, multiple parent-child relationships are used.
- The network data model uses a network structure, which is a data structure consisting of nodes and branches.
- In network data model the database files can be divided into two parts.
  1. Owner part
  2. Member part
- A relationship defined between any two files is named as SET.
- The first implemented network database application program is IBM mainframe computers.
- Network data model includes DDL (Data Definition Language) and DML (Data Manipulation Language) in DBMS.
- The main difference of network model with hierarchical model is that a network model permits a child to have more than one parent, where as hierarchical model not allows a child to have multiple parents.
- In network data model data access is more flexible.



ADVANTAGES:

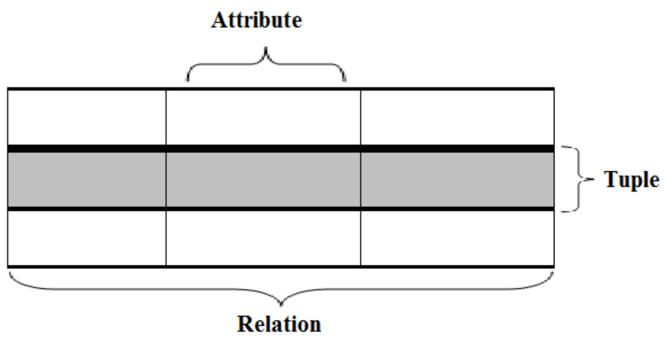
- Network data model is simple and easy to implement.
- Network data model can handle one-to-one and many-to-many associations.
- There is always a link between the parent and child segments in this data model.

DISADVANTAGES:

- Since all the records use pointers, database structure becomes more complex.
- Insertion, updating and deletion operations more complex in this model.
- In this data model structural independence is not present.

RELATIONAL DATAMODEL:

- It is one of the most popular data model implemented by E.F.Codd.
- Relational data model supports all kinds of associations.
- The structure of the database in relational data model is in the form of table.
- A table can be defined as collection of rows and columns.
- The rows are called records or tuple and columns are called fields or attributes.
- Relational data model could not support data redundancy. So two tables or two fields could not have the same name.
- There is no restriction for creating tables.
- Every relational database table having at least one unique field called “key field”.
- In relational data model the data is fully independent that is the modifications applied on one data item/value/table/field could not affect other data item/value/field/table.
- It supports all kinds of database constraints.
- The first implemented relational database application program is “ORACLE”. It is the combination of both SQL and PL/SQL.



ADVANTAGES:

- Relational data model is simpler than hierarchical and network data models.
- Changes in the structure do not affect the data access.
- Relational data model achieves both data independence and structural independence.
- Ad hoc query capability is based on SQL.

DISADVANTAGES:

- Relational database system need more powerful hardware computers and data storage devices.
- Setting up and maintaining the relational database is more expensive. To set up a relational database, you need to purchase special software.
- Ease of design can lead to bad design.

DEGREE OF DATA ABSTRACTION:

The characteristic of DBMS which specifies program independency and data independency is known as Data Abstraction.

- ↪ The end users can access, retrieve and manipulate any data at any time independently. But the data can be processed based on currently processed Query only.
- ↪ The degree of data abstraction specifies the frame work of Data model.
- ↪ The database can also be viewed from different levels of abstraction to reveal different levels of details.
- ↪ From a bottom-up manner, we may find that there are three levels of abstraction in the database.

EXTERNAL LEVEL:

1. In this database view, maximum details about the database will be hidden from the user.
2. Only the restricted portion of the database is available to end users, because an end user does not need to know everything about the structure of the entire database.
3. This model purely related to end user only.
4. It implements highest level of data abstraction.
5. E-R Diagrams will be used to represent the external views.

CONCEPTUAL LEVEL:

1. This view will provide some more details about the database to the user like structure or schema details of the database.
2. Conceptual level describes what type of data is stored in the database and what relationships exist among those data.
3. In this level a user know the information about the attributes of each table, the common attributes in the different tables.
4. It implements middle level of data abstraction.

INTERNAL LEVEL:

1. This level is concerned with the physical storage of the data.
2. It also describes what data is stored in tables.
3. It implements lowest level of data abstraction.
4. Both internal level and physical level is considered as a single level, but the difference is physical level is managed by operating system under the direction of DBMS, while the internal level is managed by DBMS.

UNIT-II

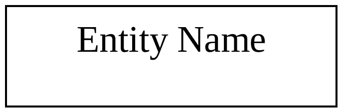
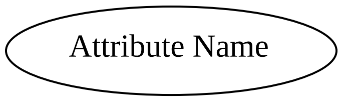
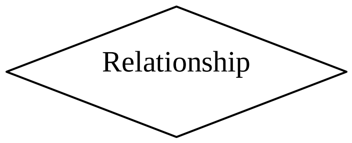
ENTITY RELATIONSHIP MODEL

THE E-R MODEL:

1. Entity – Relationship model is a high level conceptual model developed by Peter Chen in 1976 to facilitate database design.
2. The E-R Model is the generalization of the earlier available models like hierarchical and the network model.
3. Entity-Relationship model consists of set of objects called entities and relationships among the entity sets.
4. A basic component of the E-R model is the Entity-Relationship diagram which is used to visually represent data objects and the relationships between them.
5. The E-R Diagram used for representing E-R model can be easily converted into relations (tables) in relational model.
6. It is a top-down approach to database design.
7. It is very simple and easy to understand by various types of users and designers.
8. It provides an easily understood pictorial map for the database design.

BUILDING BLOCKS:

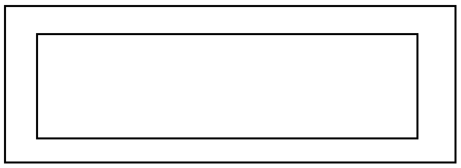
1. The E-R model can be carried out with the help of pictorial representation of entities, attributes and relationships.
2. The basic building blocks of Entity-Relationship diagram are Entity, Attribute and Relationship.
3. The following are the representations.

Entity	Data objects about which information can be collected and stored. Entities are represented by Rectangles.	
Attribute	Characteristics or properties of an entity. Attributes are represented by ellipses and is directly connected to entity.	
Relationship	An association among entities. Relationships are represented by diamond shaped box and name of the relationship is written inside it.	

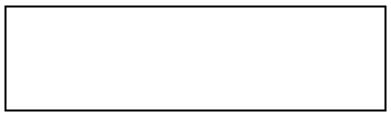
TYPES OF ENTITIES:

Entities are classified into two types.

1. WEAK ENTITY:
   If the existence of an entity is depends upon another entity such an entity is said to be weak entity. In other words an entity that does not has an attribute that act as a primary key is called as weak entity. It is represented by using double rectangle.



2. STRONG ENTITY:
   If the existence of an entity does not depend up on another entities such an entity is said to be strong entity. In other words an entity that has an attribute that act as a primary key is called as strong entity. It is represented by using single rectangle.



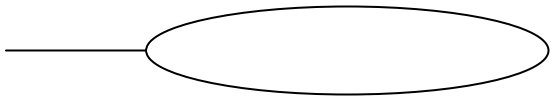
TYPES OF ATTRIBUTES:

The following are the various types of attributes.

1. Simple attribute
2. Composite attribute
3. Multi valued attribute
4. Derived attribute

SIMPLE ATTRIBUTE:

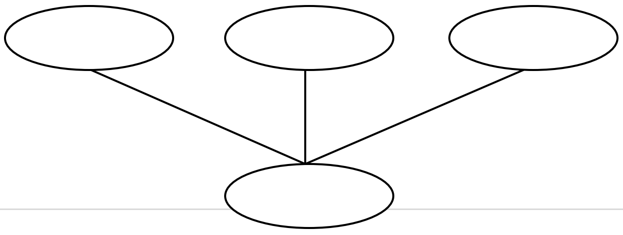
An attribute that consist of a single atomic value is called simple attribute and it cannot be sub divided.



For example age of an employee or student.

COMPOSITE ATTRIBUTE:

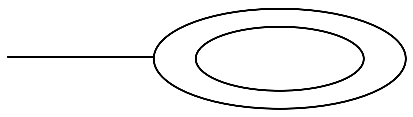
An attribute that can be divided into sub parts or components is called composite attribute.



For example the field name in a STUDENT or EMPLOYEE entity can be divided into First-Name, Middle-Name and Last-Name.

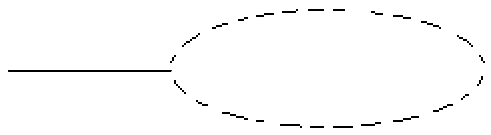
MULTI VALUED ATTRIBUTE:

An attribute that consists of multiple values is called as multi valued attribute. For example a STUDENT may have multiple phone numbers or email IDs.



DERIVED ATTRIBUTE:

An attribute that's value is derived form a stored attribute is called as derived attribute. For example the attribute age, its value is derived from the stored attribute Date_of_Birth.



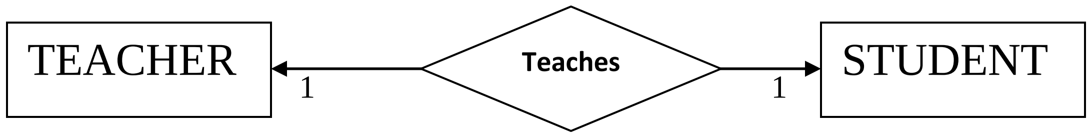
DEGREE PF REALTIONSHIP:

The degree of a relationship is the number of entities associated with the relationship. The degree of relationship is also called as **"cardinality".**

ONE-TO-ONE RELATIONSHIP:

Only one entity of the first set is related to only one entity of the second set, such a relationship is called one-to-one relationship.

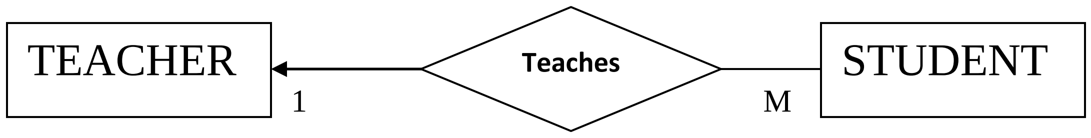
For example a teacher teaches student. Only one teacher is teaching to only one student.



ONE-TO-MANY RELATIONSHIP:

Only one entity of the first set is related to multiple entities of the second set, such a relationship is called one-to-many relationship.

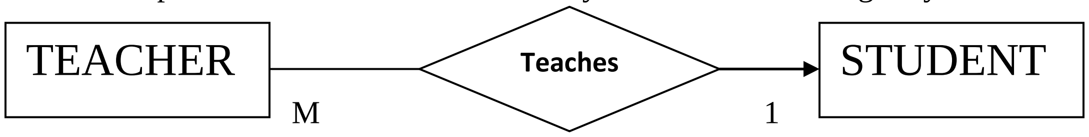
For example a teacher teaches students. Only teacher is teaching multiple students.



MANY-TO-ONE RELATIONSHIP:

Multiple entities of the first set are related to only one entity in the second set, such a relationship is called many-to-one relationship.

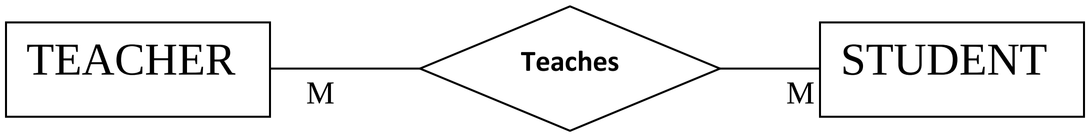
For example teachers teach a student. Many teachers are teaching only to one student.



MANY-TO-MANY RELATIONSHIP:

Multiple entities of the first set are related to multiple entities of the second set, such a relationship is called many-to-many relationship.

For example teachers teach students. Many teachers teaching many students.



DEVELOPING E-R DIAGRAMS:

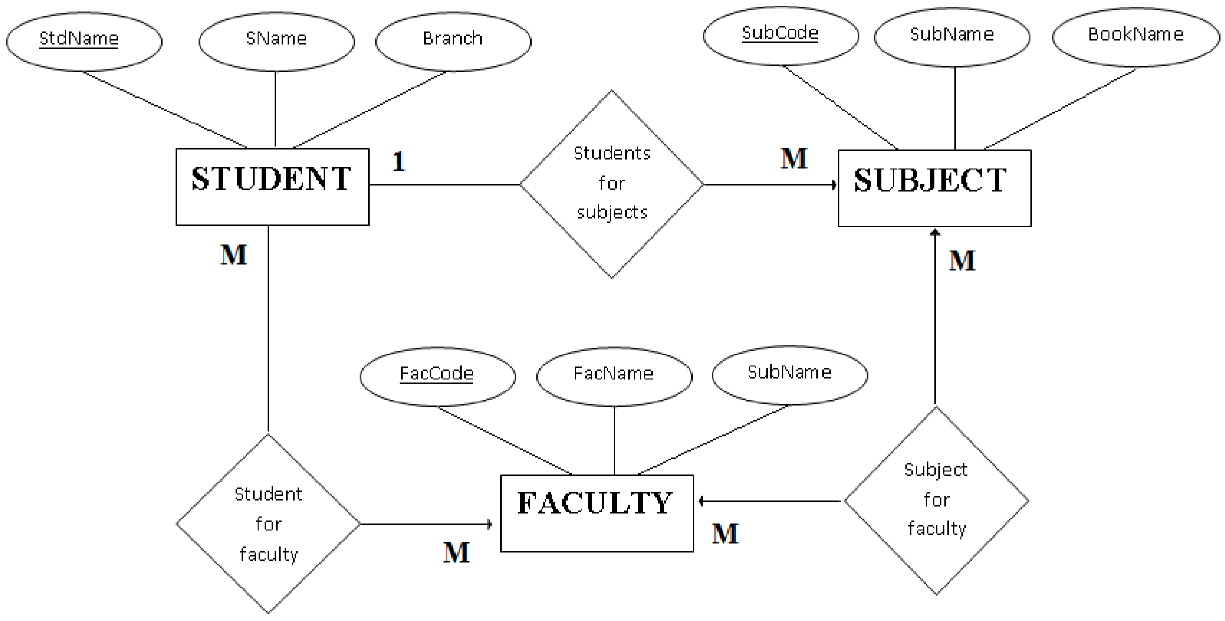
1. The name of the entity must be in upper case characters.
2. The relationship must and should have an identification (name of relation).
3. The relationship identification may be either in active voice or passive voice.
4. The relationship in the diagram either may be in single or double direction and this direction represents relationship type.

Let us consider 3 entities with fields as follows:

Entity – 1 : STUDENT (StdNum , SName , Branch)

Entity – 2 : SUBJECT (SubCode, SubName, BookName)

Entity – 3 : FACULTY (FacCode, FacName, SubName)



THE RELATIONAL DATABASE MODEL

A LOGICAL VIEW OF DATA:

1. A table is a two-dimensional structure composed of rows and columns.
2. Each row or tuple represents a single entity.
3. Each column represents an attribute and each column has a distinct name.
4. Each intersection of row and column represents a single data value.
5. All values in a column must be the same data format.
6. Each column has a specific range of values called as domain.
7. Each table must have an attribute or group of attributes that uniquely identifies each row.

KEYS:

- ❖ A KEY consists of one or more attributes that determine other attributes.
- ❖ Keys are also used to establish relationships among tables and to ensure the integrity of the data.
- ❖ Keys are important because they are used to ensure that each row in a table is identifiable.

SUPER KEY:-

A group of attributes which collectively used to identify a record uniquely from a table is called as super key.

PRIMARY KEY:-

An attribute which is used to identify a record uniquely from a table is called as primary key and it doesn’t allow duplicate values and null values.

🡇Primary Key **STUDENT Table**

Stdno	Admsno	Sname	Spec
1	101	Sailu	BCA
2	102	Mounika	BCA
3	103	Ramya	BCA

CANDIDATE KEY:-

Each individual key in super key which uniquely identifies a record uniquely is called as candidate key.

UNIQUE KEY:

An attribute that uniquely identifies a record from a table is called as unique key and it accepts null values.

ALTERNATE KEY:-

Keys other than candidate key are called as alternate key also known as secondary key used to retrieve data values from a table.

🡻Primary Key **🡻Secondary Key**

Stdno	Admsno	Sname	Spec
1	101	Sailu	BCA
2	102	Mounika	BCA
3	103	Ramya	BCA

FOREIGN KEY:-

An attribute or combination of attributes in one table whose values must either match the primary key in another table.

🡻Primary Key **STUDENT TABLE**

Stdno	Sname	Spec
1	Sailu	BCA
2	Mounika	BCA
3	Ramya	BCA

ADMISSION TABLE

🡻Primary Key **🡻 Foreign key**

Admsno	Stdno	Sname	Spec
101	1	Sailu	BCA
102	2	Mounika	BCA
103	3	Ramya	BCA

INTEGRITY RULES:

The responsibility of a Database administrator is to maintain the integrity of the database and prevent it from corruption.

An integrity constraint is a condition that is specified on a database schema, and restricts the data that can be stored in the database.

Integrity constraints are divided into three main categories:

1. Entity integrity
2. Domain integrity
3. Referential integrity

DOMAIN INTEGRITY

1. Domain integrity rules are associated with maintaining the correctness of attribute values.
2. It deals with the entries of the data values.
3. We can achieve domain integrity by using data type, NOT NULL and CHECK constraints.
4. NOT NULL constraint is used to verify the data value of field is not null.
5. CHECK constraint is used to verify the range and data types of data values.

ENTITY INTEGRITY

1. Entity integrity rule specifies that each entity should be identified uniquely.
2. Entity integrity is a constraint on primary key value and unique key value.
3. PRIMARY KEY doesn't have repeated data and null values and used to identify an entity uniquely.
4. UNIQUE KEY doesn't have repeated data but accepts null values and is used to identify an entity uniquely.

REFERENTIAL INTEGRITY:

1. Referential integrity is achieved through FOREIGN KEY CONSTRAINT.
2. A Foreign Key is an attribute within one relation that matches the primary key in another relation.
3. In relational database sometimes it requires to ensure that a value that appears in one relation field is also available for some other field in another relation. This is known as referential integrity and is expressed in terms of foreign key.

DATABSE CONSTRAINS

- ✓ Database integrity means the completeness, correctness and consistency of data.
- ✓ The database must be complete, accurate, valid, consistent can be achieved using integrity constraints.
- ✓ One of the most important responsibility of a DBA is to maintain the integrity of database and prevent it from becoming corrupted.
  - ✓ **Types of Integrity Constraints**
    1. **Domain Integrity (CHECK, NOT NULL)**
    2. **Entity Integrity (PRIMARY KEY, UNIQUE)**
    3. **Referential Integrity (FOREIGN KEY)**

1) Domain Integrity:-

- ✓ It checks the validity of entries for a given column.There are 2 domain integrityconstraints .
  a) **CHECK** b)**NOT NULL**

(a) CHECK:-

- ✓ Using CHECK constraint checks the range of the record values in a table.
- ✓ Using CHECK constraint checks the validity of entries for a given column.

Syntax:- SQL\> CREATE TABLE tablename (field1 datatype(length), field2 datatype(length), ….. CHECK condition);

Example:- SQL\> CREATE TABLE student (stdno number(10) PRIMARY KEY,fee number(10) CHECK (fee BETWEEN 15000 AND 25000));

(b) NOT NULL:-

- ✓ NOT NULL constraint could not accept empty values in a column.
- ✓ If any fields assigned as NOT NULL in a table that field must & should be mandatory.

Syntax:- SQL\> CREATE TABLE tablename (field1 datatype(length) NOT NULL,

field2datatype(length), …..);

Example:- SQL\> CREATE TABLE student(stdno number(10) PRIMARY KEY, sname varchar2(20) NOT NULL,spec varchar2(10));

2) Entity Integrity:-

- ✓ Entity integrity rules ensure that it should be easy to identify each entity in database.
  - ✓ There are 2 types of Entity integrity
    - a) **PRIMARY KEY**
    - b)**UNIQUE**

(a) PRIMARY KEY:-

- ✓ A key field that identifies all the records of a table uniquely is known as primary key, it does not accept null values& duplicate values.
- ✓ If any filed in a database table assigned as primary key constraints the record values of that field never accept repeated values and null values at the time of inserting.

```
Syntax:-  SQL\>  CREATE TABLE  tablename  (field1  datatype(length) PRIMARY  KEY,

field2datatype(length), …..);

Example:-   SQL\>  CREATE TABLE  student  (stdno  number(10) PRIMARY KEY,

                          Sname  varchar2(20)  NOT  NULL,   spec  varchar2(10));
```

(b) UNIQUE KEY:-

- ✓ The UNIQUE constraint uniquely identifies each record in a database table.
- ✓ A UNIQUE constraint acceptNULL values & unique values but not accept duplicate values.

```
Syntax:-  SQL\>  CREATE TABLE  tablename  (field1  datatype(length) UNIQUE,

field2datatype(length), …..);

Example:-   SQL\>  CREATE TABLE  student (stdno  number(10)   UNIQUE, sname  varchar2(20)
NOT  NULL,spec  varchar2(10));

SQL\> Table Created.

SQL\>desc   student;
```

3) Referential Integrity:-

- ✓ The referential integrity constraint by using foreign key we can define the integration between two database tables.
- ✓ Whenever a field can be assigned as **foreign key** in one table that field references the **primary key** in another table.
- ✓ The table which contain the **foreign key** is named as **detailed(child)** table. The table which contain the **references** field is named as **master table(parent).**

a) **FOREIGN KEY (REFERENCES) :-**

Syntax:- SQL\> CREATE TABLE parent_tablename (field1 datatype(length) REFERENCES child_tablename, field2 datatype(length), …..);

Example:- SQL\> CREATE TABLE student (stdno number(10) PRIMARY KEY, admsno number(10) REFERENCES admission,sname varchar2(20) NOT NULL,spec varchar2(10));

4) DEFAULT constraint:- The DEFAULT constraint is used to insert a default value into a column. The default value will be added to all new records, if no other value is specified.

Example:- SQL\> Create Table Student(stdno number(5) not null,

Spec varchar2(10) DEFAULT ' BCA');

Examples of PRIMARY KEY, FOREIGN KEY, CHECK Constraints:-

1) **Parent Table (primary key):-**
```
SQL\>  CREATE TABLE  student(rno  NUMBER(5)  PRIMARY  KEY,
           sname VARCHAR(20)  NOT NULL,  spec   VARCHAR2(10)   NOT NULL);
```

2) **Child Table (foreign key):-**
```
SQL\>  CREATE TABLE  marks(hno  NUMBER(10)  PRIMARY  KEY,
 rnoNUMBER(5)  REFERENCES  student,   java  NUMBER(5) NOT NULL,
dbms  NUMBER(5)  NOT NULL,  maths  NUMBER(5)  NOT NULL
CHECK(maths BETWEEN  0  AND 75) );
```

3) **Display records of both tables at a time:-**
```
SQL\>SELECT  student.rno,  sname, spec,  marks.hno, marks.dbms,  marks.java,
          marks.maths   FROM  student,  marks  WHERE student.rno = marks.rno;
```

MARKS table

hno	Rno	Java	Dbms	Maths
9001	1	60	72	72
9002	2	55	45	63
9003	3	65	70	58
9004	4	70	69	45

STUDENT table

Rno	Sname	Spec
1	Sailu	BSC
2	Murthy	BCA
3	Bharath	BCOM
47	Indu	BBM

Relational SET OPERATORS (Relational Algebra)

- ✓ In Oracle database SQL supports 8 types of relational algebra and relational SET operators.
- ✓ SET operators are used to combine information of similar types from one or more than one table.
- ✓ SQL joins tends to combine columns from different tables, whereas SQL SET operators combine rows from different queries.
  - o Before apply set operators on 2 or more tables, these tables must satisfies "union compatibility" relation.
  - o **Union compatibility:-** Two or more tables must and should have same field names, same data type and approximate field length.

✓ **Types of SET operators:-**

1) **UNION**
2) **INTERSECT**
3) **MINUS (difference)**
4) **DIVIDE**
5) **PRODUC T(Cartesian product)**
6) **PROJECT**
7) **SELECT**
8) **JOIN**

1) UNION:-

- ☞ The UNION operators joins the outputs of two or more queries(SELECT statement) into a single set of rows and columns having distinct(unique) records.
- ☞ The UNON operator combines the output of two selected queries without include duplicate records.

```
Syntax:-        SELECT  column  list   FROM   table1
                UNION
                SELECT  column  list   FROM   table2  [Order by clause];
```

STUDENT table

Stdno	Sname	Spec	Fee
1	Sailu	BSC	12000
2	Mounika	BCOM	10000
3	Ramya	BCA	20000
4	Indu	BA	8000

ADMISSION table

Admsno	Spec	Fee
1	BSC	12000
2	BCOM	10000
3	BCA	20000
4	BBM	15000

☞ Apply UNION operator on both tables.

```
Example:- SELECT  spec, fee   FROM   STUDENT
          UNION
          SELECT  spec, free  FROM  ADMISSION;
```

Output :-

Spec	Fee
BSC	12000
BCOM	10000
BCA	20000
BA	8000
BBM	15000

2) INTERSECT :-

☞ The INTERSECT operator returns the rows that are common between two sets of rows. It returns the records that appear in the both the tables.

```
Syntax:- SELECT  column  list   FROM   table1
         INTERSECT
         SELECT  column  list   FROM   table2;
```

☞ Apply INTERSECT operator on both tables.

```
Example:-SELECT  spec, fee   FROM   STUDENT
         INTERSECT
         SELECT  spec, free  FROM  ADMISSION;
```

Output :-

Spec	Fee
BSC	12000
BCOM	10000
BCA	20000

3) MINUS :-

☞ The MINUS operator returns the rows unique to first query.

☞ It combines records from 2 queries and returns the first table records which were not placed in the 2nd table.

```
Syntax:-SELECT  column  list   FROM   table1
         MINUS
        SELECT  column  list   FROM   table2;
```

☞ Apply MINUS operator on both tables.

```
Example - 1:-SELECT  spec, fee   FROM   STUDENT
             MINUS
             SELECT  spec, free  FROM  ADMISSION;
```

Output :-

Spec	Fee
BA	8000

```
Example - 2:-SELECT  spec, fee   FROM   ADMISSION
             MINUS
             SELECT  spec, free  FROM  STUDENT;
```

Output :-

Spec	Fee
BBM	15000

4) DIVIDE :-

- ☞ This operator specifies table1 is DIVIDE by table2 to produce table3 whenever both tabl1 & table2 using a common field.
- ☞ The DIVIDE operation uses one single column table (i.e. columna) as the divisor and 2 column table(i.e. column a & b) as the dividend.
- ☞ The table must have a common column (i.e.column a)

Table **A**

Sname	**Avg**
Xyz	70
abc	60
pqr	50
xyz	50
Pqr	90

DIVIDE

Table **B**

Sname
Xyz
Pqr

- The output of a Table A divided by Table B is.
- The only value associated with both 'xyz' & 'pqr' is 50 only.

Output :-

Avg
50

5) PRODUCT (or) Cartesian Product :-

- ☞ This operator returns all the possible pairs of records from the two tables..
- ☞ If table1 having 3 records and table2 having 3 records then the product of e tables will return 3 x 3 = 9 records.

STUDENT

Stdno	**Sname**
1	Murthy
2	Sailu
3	Bharat

PRODUCT

SUBJECTS

Spec	**Subj**
BSC	English
BSC	Maths
BSC	Computers

Output:-

Stdno	**Sname**	**Spec**	**Sunj**
1	Murthy	BSC	**English**
1	Murthy	BSC	**Maths**
1	Murthy	BSC	**Computers**
2	**Sailu**	BSC	English
2	**Sailu**	BSC	Maths
2	**Sailu**	BSC	Computers
3	Bharat	BSC	**English**
3	Bharat	BSC	**Maths**
3	Bharat	BSC	**Computers**

Example:- SQL\> SELECT stdno, sname, spec, subj FROM STUDENT, SUBJECT;

6) PROJECT :-

- ☞ This operator returns the vertical subset of relational database table.
- ☞ it returns the selected field from the database table.(it returns single or multiple fields)

STUDENT - Table

Stdno	Sname	Fee
1	Sailu	12000
2	Mounika	10000
3	Ramya	20000
4	Indu	8000

PROJECT fee yields →

Fee
12000
10000
20000
8000

7) SELECT :-

☞ The SELECT statement is used to query or retrieve data from a table in the database.

☞ A query may retrieve information from specified columns or all columns in a table.

Syntax :-

```
SELECT  columns_list   FROM   table_name [ WHEREcaluse]   [ GROUP BY
clause ] [ HAVING  clause ]  [ ORDER BY  clause ] ;
```

Display ALL fields Example :-

```
        SQL\>SELECT  *  FROM   STUDENT;
```

Display some specified fields Example :-

```
        SQL\>SELECT  Stdno, Sname FROM   STUDENT;
```

Display some specified fields based on Condition:-

```
        SQL\>SELECT  Stdno, Sname FROM   STUDENT    WHERE    RNO=3;
```

8) JOIN :-

☞ SQL joins are used to combine columns from different tables.

☞ The connection between tables is established through the WHERE clause, called SQL Join condition.

Syntax:-

```
        SELECT  table1.column1,  table2.column2,……., tableN.columnN       FROM
        table1, table2 …..   WHERE   table1.column1   = table2.column2;
```

☞ **Types of JOINS:-**

a) **NATURAL Join**

b) **EQUI Join**

c) **THETA Join**

$\left.\begin{array}{l}\text{b) EQUI Join}\\ \text{c) THETA Join}\end{array}\right\}$ Inner Joins

d) **OUTER JOIN**

   i) **LEFT OUTER Join**

   ii) **RIGHT OUTER Join**

NATURAL Join:- (New Style)

- ✓ It merges the records from both tables and returns the records having common values in common fields and eliminates duplicate records.
- ✓ **Syntax:- SELECT column_list FROM table1 NATUREAL JOIN table2;**
- ✓ **Example:- select *from subject natural join marks;**

SUBJECT table

Subj code	Subjname	Spec
m101	**maths**	**bsc**
a101	**accounts**	**bcom**
e101	english	ba
c101	computers	bca

MARKS table

Hno	Subjcode	Subjname
101	**m101**	**maths**
103	**a101**	**accounts**
102	c102	commerce
104	e102	economics

Output:-

SUBJ CODE	SUBJ NAME	SPEC	HNO
m101	**Maths**	**bsc**	**101**
a101	**accounts**	**bcom**	**103**

EQUI Join:- (Old Style)

- ✓ In Equi Join uses the equality(=) comparison operator for join table.
- ✓ Returns only the rows of both tables that based on join condition in the WHERE clause.

Syntax:-SELECT column_list FROM table1 WHERE table1.column = table2.column;

- ✓ **Example:-**
  SELECT * from subject, marks WHEREsubject.subjcode=marks.subjcode;

Output:-

SUBJCODE	SUBJNAME	SPEC	HNO	SUBJCODE	SUBJNAME
m101	**maths**	**bsc**	**101**	**m101**	**maths**
c101	**computers**	**bca**	**102**	**c101**	**commerce**
a101	**accounts**	**bcom**	**103**	**a101**	**accounts**

THETAJoin:-

- ✓ The condition specifies inner join uses other comparison operators but not use "="
- ✓ It uses operators LIKE, AND, OR etc..then that of join operator is known as "theta join"

OUTER JOIN:-It merges the records from two tables and returns the records having common values in common fields but also include unmatched values based on condition.

i) **LEFT OUTER Join:-**The left outer join returns not only the rows matching the join condition, but also rows in the leftside table with unmatched values in the right side table.

SUBJECT table **MARKS table**

Subjcode	Subjname	Spec
m101	**maths**	**bsc**
a101	**accounts**	**bcom**
e101	english	ba
c101	computers	bca

Hno	Subjcode	Subjname
101	**m101**	**maths**
103	**a101**	**accounts**
102	c102	commerce
104	e102	economics

- ✓ **Syntax:- SELECT column_list FROM table1 LEFT JOINtable2 ON condition;**
- ✓ **Example:- SELECT * from subject LEFT JOIN marks ONsubject.subjcode=marks.subjcode;**

Output:-

SUBJCODE	SUBJNAME	SPEC	HNO	SUBJCODE	SUBJNAME
m101	maths	Bsc	101	m101	maths
a101	accounts	bcom	103	a101	accounts
e101	english	Ba	-	-	-
c101	computers	Bca	-	-	-

ii) RIGHT OUTER Join:- The right outer join returns not only the rows matching the join condition, but also rows in the right side table with unmatched values in the left side table..**Example:-**

SELECT * from subject RIGHT JOIN marks ONsubject.subjcode=marks.subjcode;

SUBJCODE	SUBJNAME	SPEC	HNO	SUBJCODE	SUBJNAME
m101	maths	bsc	101	m101	maths
a101	accounts	bcom	103	a101	accounts
-	-	-	104	e102	economics
-	-	-	102	c102	commerce

THE DATA DICTIONARY AND SYSTEM CATALOG:

1. Data dictionary provides a detailed description of all tables in the database created by user and designer.
2. Data dictionary contains all of the attributes and characteristics for each table in the system.
3. Data dictionary contains metadata that is data about data.
4. The data dictionary sometimes called as “the database designer’s database”.
5. Like data dictionary, system catalog contains metadata.
6. The system catalog can be described as a detailed system data dictionary that describes the following
   a. Data about table names
   b. Table’s creator and creation date
   c. Number of columns in each table
   d. Data type corresponding to each column
   e. Index file names
   f. Index creators
   g. Access privileges

DATA REDUNDANCY:

1. Data redundancy leads to data anomalies, which can destroy the effectiveness of the database.
2. Relational database makes it possible to control data redundancy by using foreign keys.
3. The use of foreign keys control data redundancy, but they do not totally eliminate the problem because the foreign key values can be repeated many times.
4. The proper use of foreign keys minimizes data redundancies and data anomalies will be destroyed.

INDEXES:

1. An index is an orderly arrangement used to logically access rows in a table.
2. An index is used to locate a needed item quickly.
3. Indexes in the relational database environment works like indexes in book.
4. An index is composed of an index key and a set of pointers.
5. An index is an ordered arrangement of keys and pointers.
6. Each key points to the location of the data identified by the key.
7. An index can be used to retrieve data more efficiently, but indexes can also be used by a DBMS to retrieve data ordered by a specific attribute or attributes.
8. A table may have many indexes, but each index is associated with only table.
9. The index key can have multiple attributes.

CODD'S RELATIONAL DATABASE RULES:

1. In 1985, Dr. E.F.Codd published a list of 12 rules to define a relational database system.
2. An RDBMS product has to satisfy at least 6 of the 12 rules to be accepted as a full-fledged RDBMS.
3. There is no RDBMS package available that satisfies all the 12 rules.
4. The following are the Codd's rules.

a. **INFORMATION RULE:** All information in a relational database must be logically represented as column values in rows within tables.

b. **GUARANTEED ACCESS:** Every value in a table is guaranteed to be accessible through a combination of table name, primary key and column name.

c. **SYSTEMATIC TREATMENT OF NULL VALUES:** Null values must be represented and treated in a systematic way, independent of data type.

d. **DYNAMIC ONLINE CATALOG BASED ON THE RELATIONAL MODEL:** The meta data must be stored and managed as ordinary data, that is in tables within database. Such data must be available to authorized users using the standard database relational language.

e. **COMPREHENSIVE DATA SUBLANGUAGE:** The relational database must support one well defined, declarative language for data definition, view definition, data manipulation, integrity constraints, authorization and transaction management.

f. **VIEW UPDATING:** Any view that is theoretically updatable must be updatable through system.

g. **HIGH LEVEL INSERT, UPDATE AND DELETE:** The database must support set-level insert, update and delete.

h. **PHYSICAL DATA INDEPENDENCE:** Application programs and ad hoc facilities are logically unaffected when physical access methods or storage structure are changed.

i. **LOGICAL DATA INDEPENDENCE:** Application programs and ad hoc facilities are logically unaffected when changes are made to the table structure that is changing order of columns and inserting new columns.

j. **INTEGRITY INDEPENDENCE:** All relational integrity constraints must be definable in the relational language and stored in the system catalog, not at the application level.

k. **DISTRIBUTION INDEPENDENCE:** The end users and application programs are unaware and unaffected by the data location.

l. **NO-SUBVERSIONS:** If the system supports low-level access to the data, users must not be allowed to bypass the integrity rules of the database.

UNIT-III

NORMALIZATION OF DATABASE TABLES

1. The table is the basic building block of database design.
2. It is possible to create poor table structures even in good database design.
3. Normalization is the analysis of functional dependency between attributes/data items of user view. Normalization reduces a complex user view into small and stable sub relations.
4. Normalization is a process of evaluating and correcting table structure to minimize data redundancy, thereby reducing data anomalies.
5. Normalization works through a series of stages called normal forms.
6. You will occasionally need to denormalize some portions of a database design to meet performance requirements.
7. Denormalization produces a lower level normal form, that is a 3NF will be converted into a 2NF through denormalization.

There are different levels of normalization as listed below.

1. Unnormalized Form (UNF)
2. First Normal Form (1NF)
3. Second Normal Form (2NF)
4. Third Normal Form (3NF)
5. Boyce-Codd Normal Form (BCNF)
6. Fourth Normal Form (4NF)
7. Fifth Normal Form (5NF)

Unnormalized Form:

A relation which satisfies all the properties of a relation, such relation is considered as unnormalized form.

First Normal Form (1NF):

A relations is said to be in the first normal form if it is already in unnormal form and it has no repeating group.

Second Normal Form (2NF):

A relation is said to be in the second normal form if it is already in the first normal form and it has no partial dependency.

Third Normal Form (3NF):

A relation is said to be in the third normal form if it is already in the second normal form and it has no transitive dependency.

Boyce-Codd Normal Form (BCNF):

A relation is said to be in Boyce-Codd Normal Form if it is already in third normal form and every determinant is a candidate key. It is a stronger version of the third normal form.

Fourth Normal Form (4NF):

A relation is said to be in the Fourth Normal Form if it is already in Boyce-Codd Normal Form and it has no multi-valued dependency.

Fifth Normal Form (5NF):

A relation is said to be in the fifth normal form if it is already in Fourth Normal Forma and it has no Join Dependency.

Partial Dependency:

If a relation has more than one key field then all the non key fields must be depend on all the key fields but some non key fields depends only on one key field. Such dependency is called partial dependency.

Transitive Dependency:

All the non key fields in a relation must be depends on primary key or key field, but in some relations some set of non key fields depend on another non key field. Such dependency is called transitive dependency.

Determinant:

A determinant is any field i.e., simple or composite on which some other field is fully functionally dependent.

Multi-valued Dependency:

Consider three fields X,Y,Z in a relation. If for each value of X, there is a well defined set of values Y and a well-defined set of values of Z and the set of values of Y is independent of the set of values of Z, then multi-valued dependency exists.

Join Dependency:

A relation which has a join dependency cannot be decomposed by projection into other relations without any difficulty and undesirable results. The occurrence of this type of dependency is very rare.

For most of the practical applications, it is sufficient to normalize each and every relation up to 3NF.

UNNORMAL FORM:

Consider the following INVOICE relation to explain the normalization.

INVOICE(CUS-ID,CUS-NAME,CUS-ADDR(ISBN,TITLE,AUTHOR-NAME,AUTHOR-COUNTRY,QTY,UNIT-PRICE)). → Relation-1

The relation INVOICE has satisfy all the properties of a relation so it is in Un Normal Form(UNF).

FIRST NORMAL FORM:

Relation-1 is in UNF. The fields in the inner most set of parentheses put together is known as a repeating group. This will result in redundancy of data for first three fields. This redundancy leads to inconsistency.

So INVOICE relation is divided into two sub relations as follows.

CUSTOMER(CUS-ID,CUS-NAME,CUS-ADDR) → Relation-2

CUSTOMER-BOOK(CUS-ID,ISBN,TITLE,AUTHOR-NAME,AUTHOR-COUNTRY,QTY, UNIT-PRICE) → Relation-3.

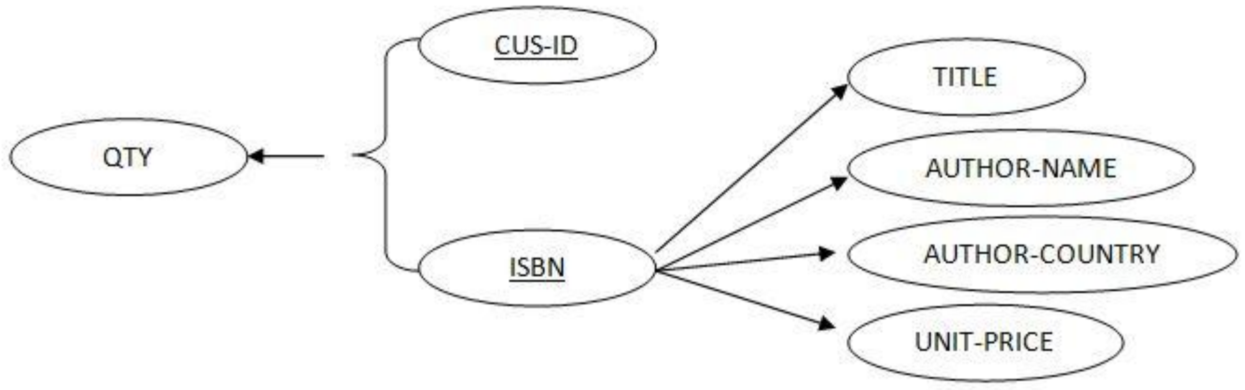
Now each of the above Relations 2 and 3 is in 1NF.

In Relaation-3 CUS-ID and ISBN combination works as a key field.

SECOND NORMAL FORM:

In Relation-2 CUSTOMER there is only one key field and hence there is no possibility of partial dependency in CUSTOMER relation. Hence it is considered to be in 2NF without any modifications.

In Relation-3 CUSTOMER-BOOK there are two key fields CUS-ID and ISBN and in this relation partial dependency existed. The dependency diagram is shown below.



QTY depends on CUS-ID and ISBN, but the remaining non key fields TITLE, AUTHOR-NAME, AUTHOR-COUNTRY and UNIT-PRICE depend only on ISBN. This is called partial dependency.

The existence of partial dependency leads to insertion anomaly, deletion anomaly and update anomaly.

INSERTION-ANOMALY:

If we want to insert a new book details it is not possible. It is possible only at least one customer purchases that book. This is called insertion anomaly.

UPDATE-ANOMALY:

If we want to change any of the non key fields it will result into inconsistency because the same data available in more than one record. This is called update anomaly.

DELETION-ANOMALY:

If a book has been purchased by only one customer and if you delete that record then the book details will also be lost. This is called deletion anomaly.

We can remove partial dependency in Relation-3 by dividing it into two sub relations as shown below.

SALES(<u>CUS-ID</u>,<u>ISBN</u>,QTY) → Relation - 4

BOOK-AUTHOR(ISBN,TITLE,AUTHOR-NAME,AUTHOR-COUNTRY,

UNIT-PRICE) → Relation – 5

Now the Relations 2,4,5 are in 2NF.

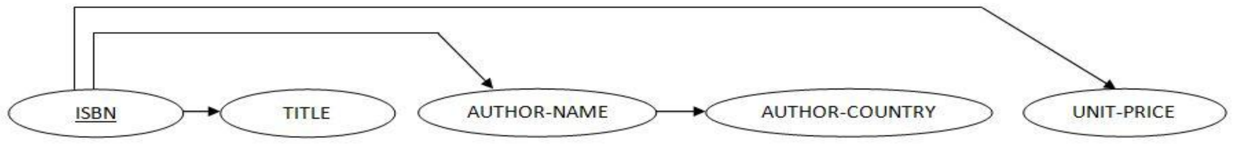
THIRD NORMAL FORM:

In Relation – 4 there is only one secondary key and there is no possibility of transitive dependency. Hence Relation-4 SALES is said to be in 3NF.

Similarly, in Relation – 2 there is no transitive dependency and it is said to be in 3NF.

In Relation – 5 AUTHOR-COUNTRY is a non key field and that is depend up on another non key field AUTHOR-NAME. This is called transitive dependency.

The transitive dependency in Relation – 5 leads to the following anomalies.



INSERTION-ANOMALY:

It will be difficult to include author's details in Relation – 5 because there should be at least one published book to insert the details of the resident author. This is called insertion anomaly.

UPDATE-ANOMALY:

There will be redundancy, inconsistency and excessive search time to locate the record for a given author while updating records since author's details are duplicated. This is called update anomaly.

DELETION-ANOMALY:

If an author wrote only one book, if that book is deleted then author's details will be lost. This is called deletion anomaly.

To overcome the above anomalies Relation – 5 is divided into two sub relations as shown below.

BOOK(ISBN,TITLE,UNIT-PRICE,AUTHOR-NAME) → Relation – 6

AUTHOR(AUTHOR-NAME,CUTHOR COUNTRY) → Relation – 7

In Relation – 6 AUTHOR-NAME is underlined with dotted line to represent foreign key.

Hence the Relations 2,4,6,7 are said to be in 3NF.

CUSTOMER(CUS-ID,CUS-NAME,CUS-ADDR) → Relation-2

SALES(CUS-ID,ISBN,QTY) → Relation - 4

BOOK(ISBN,TITLE,UNIT-PRICE,AUTHOR-NAME) → Relation – 6

AUTHOR(AUTHOR-NAME,CUTHOR COUNTRY) → Relation – 7