=> **What is a Database ?**

A shared collection of logically related data designed to meet the information requirements of an organization.

A database is a structured electronic storage system for organizing and managing data efficiently.

Examples - MySQL, Oracle, MongoDB, MS Access, SQL Server, Informix, PostgreSQL,

=> **What is a Database Management System?**

A software system that enables users to define, create and maintain the database and which provides controlled access to the database.

=> **Actors on the DBMS**

- **Database Administrator**

  Coordinates all the activities of the database system.the database administrator has a good understanding of the enterprise's information resources and needs.

  Database administrator's duties include.

  - Schema definition
  - Granting user authority to access the database
  - Schema and physical organization modification
  - Storage structure and access method definition
  - Specifying integrity constraints
  - Acting as liaison with users
  - Monitoring performance and responding to changes in requirements

- **Database Users**

  Users are differentiated by the way they expect to interact with the system

  - **Application programmers –** interact with system through DML calls
  - **Sophisticated users –** form requests in a database query language
  - **Specialized users –** write specialized database applications that do not fit into the traditional data processing framework
  - **Naïve users –** invoke one of the permanent application programs that have been written previously

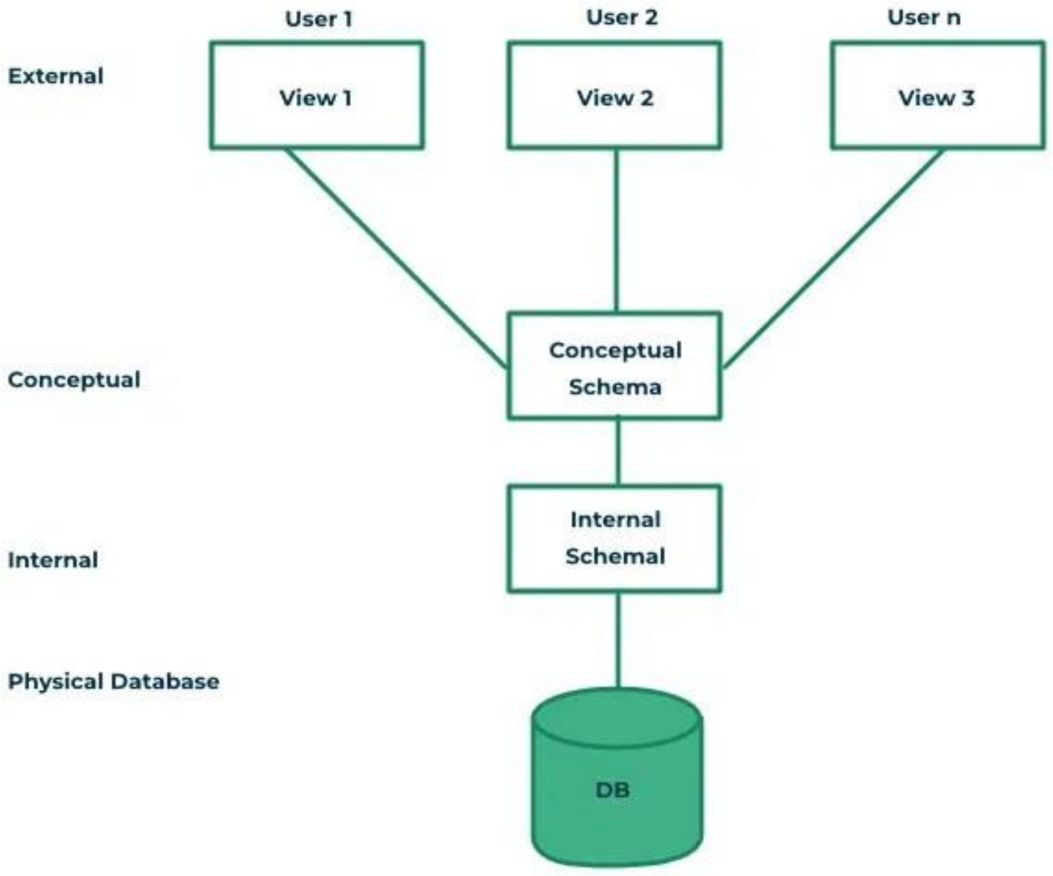
    Examples,
    people accessing database over the web, bank tellers, clerical staff

=> **Difference between Traditional File based System and DBMS**

Traditional File	DBMS
Redundant data can be present in a file system	In DBMS there is no redundant data.
Only one user can access data at a time.	Multiple users can access data at a time.
There is no data independence.	In DBMS data independence, mainly of two types: 1) Logical Data Independence. 2) Physical Data Independence.

File systems provide less security in comparison to DBMS.	DBMS has more security mechanisms as compared to file systems.
There is less data consistency in the file system.	There is more data consistency because of the process of normalization.
It doesn't provide an Inbuilt mechanism for backup and recovery of data.	It provides in house tools for backup and recovery of data.

=> **3-Tier Architecture in DBMS**



Defines DBMS schemas at three levels:

- **Internal schema**
- **Conceptual schema:** entities, data types, relationships, etc
- **External schemas** at the external level to describe the various user views.

Data Independence:

- **Logical Data Independence:** ability to change conceptual schema without affecting application programs

- **Physical Data Independence:** ability to change internal schema without affecting upper layers

=> **Data Model**

A Data Model in Database Management System (DBMS) is the concept of tools that are developed to summarize the description of the database.

Types of Relational Models

- Conceptual Data Model
- Representational Data Model
- Physical Data Model

Some Other Data Models

- Hierarchical Model

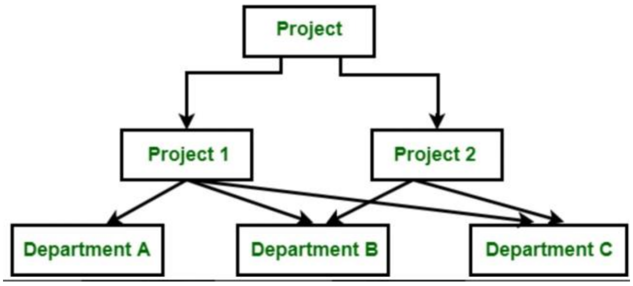
  Hierarchical Data Model: Hierarchical data model is the oldest type of the data model. It was developed by IBM in 1968. It organizes data in the tree-like structure.

  Hierarchical model consists of the following :

  - It contains nodes which are connected by branches.
  - The topmost node is called the root node.
  - Each node has exactly one parent.
  - One parent may have many child.
  - If there are multiple nodes appear at the top level, then these can be called as root segments.

- Network Model

  It is the advanced version of the hierarchical data model. To organize data it uses directed graphs. In this child can have more than one parent. It uses the concept of the two data structures Records and Sets.



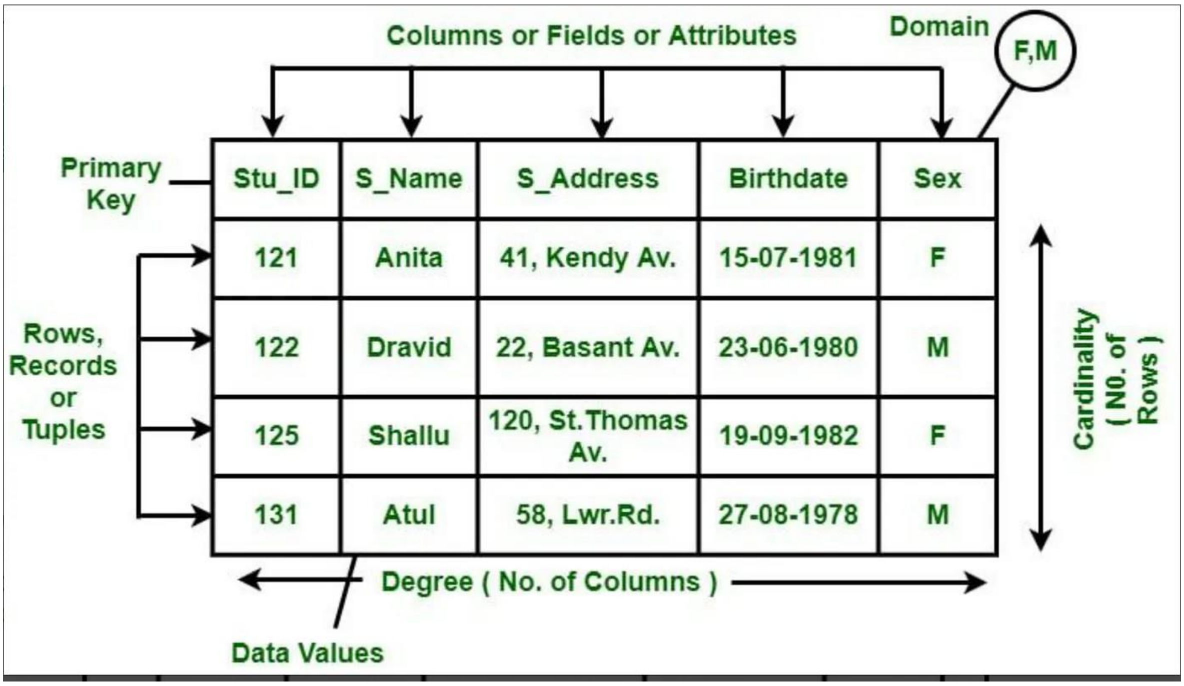
- Object-Oriented Data Model

Relational data model

The relational data model was developed by E.F. Codd in 1970. There are no physical links as they are in the hierarchical data model.

Following are the properties of the relational data model :

- Data is represented in the form of table only.
- It deals only with the data not with the physical structure.
- It provides information regarding metadata.
- At the intersection of row and column there will be only one value for the tuple.
- It provides a way to handle the queries with ease.



Example of a Relation

Relation Name → **STUDENT**; Attributes (arrows to each column); Tuples (arrows to each row)

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

In the STUDENT table, SSN is the key

The Schema (or description) of a Relation:
Denoted by R(A1, A2, ... An)
R is the name of the relation
The attributes of the relation are A1, A2, ..., An

Example:
CUSTOMER (<u>Cust-id</u>, Cust-name, Address, Phone#)

CUSTOMER is the relation name
Cust-id is the Key
Defined over the four attributes: Cust-id, Cust-name, Address, Phone#

A tuple is an ordered set of values (enclosed in angled brackets ‘< … >’)
Each value is derived from an appropriate domain.
A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:

< 632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000" >

This is called a 4-tuple as it has 4 values
A tuple (row) in the CUSTOMER relation.

The relation state is a subset of the Cartesian product of the domains of its attributes.

Definition Summary

Informal Terms	Formal Terms
Table	Relation
Column Header	Attribute
All possible Column Values	Domain
Row	Tuple
Table Definition	Schema of a Relation
Populated Table	State of the Relation

Relational Integrity Constraints

There are three main types of constraints in the relational model

- Key constraints
- Entity integrity constraints
- Referential integrity constraints

Categories of data models

1. Conceptual (high-level, semantic) data models

    Conceptual models include
    - Entity-relationship database model (ERDBM)

        Advantages
        - Exceptional conceptual simplicity
        - Visual representation
        - Effective communication tool
        - Integrated with the relational database model

        Disadvantages
        - No data manipulation language (no complete).
        - Loss of information content.
        - Limited constraint representation.
        - Limited relationship representation (internal relationship can not be depicted multiple relationships)

‘

- Object-oriented model (OODBM)

  Advantages
  - Adds semantic content
  - Visual presentation includes semantic content
  - Database integrity
  - Both structural and data independence

  Disadvantages
  - Lack of OODM
  - Complex navigational data access
  - Steep learning curve
  - High system overhead slows transactions

Relationships in Conceptual Models

- One-to-one (1:1)
- One-to-many (1:M)
- Many-to-many (M:N)

2. Physical (low-level, internal) data models

   Provide concepts that describe details of how data is stored in the computer.

3. Implementation (representational) data models

   Implementation models include
   - Hierarchical database model (HDBM)
   - Network database model (NDBM)
   - Relational database model (RDBM)
   - Object-oriented database model (ODBM)

1. Hierarchical database model (HDBM)

Logically represented by an upside down tree

- Each parent can have many children (segment linkage)
- Each child has only one parent

Advantages

- Conceptual simplicity
- Database security
- Data integrity
- Data independence

Disadvantages

- Complex implementation
- Lacks structural independence
- Difficult to manage and lack of standards
- Applications programming and use complexity (pointer based)

2. Network database model (NDBM)

Advantages

- Conceptual simplicity
- Data access flexibility
- Data independence
- Promotes database integrity
- Handles more relationship types (but all 1:M relationship)
- Conformance to standards

Disadvantages

- System complexity
- Lack of structural independence

3. Relational database model (RDBM)

Advantages

- Structural independence
- Improved conceptual simplicity
- Easier database design, implementation, management, and use
- Ad hoc query capability with SQL (4GL is added)
- Powerful database management system

Disadvantages

- Substantial hardware and system software overhead
- Poor design and implementation is made easy
- May promote “islands of information” problems

Features of relational DBMS.

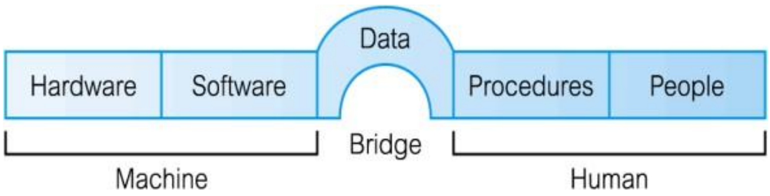
- 2-D tables (rows and columns representing records/tuples and fields/attributes)
- Dynamic links among tables
- Easy and flexible to design and use
- Data independence

=> **Database Systems**

A database system is more than just data or data in combination with DBMS software

A complete database system in an organization consists of four components. / Components of DBMS Environment

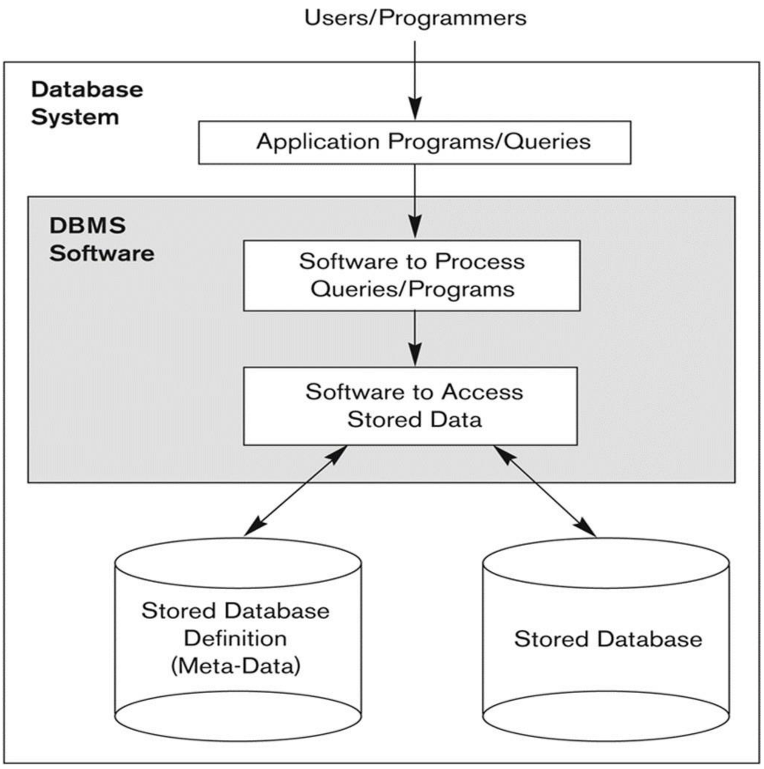
- Hardware
- Software
- Data
- Procedures (extra not lms)
- People -



Data Administrator (DA), Database Administrator (DBA), Database Designers (Logical and Physical), Application Programmers, End Users (naive and sophisticated)

A DBMS typically provides following services,

- A centralized data definition & data control facility known as a data dictionary/directory (DD/D) or catalog
- Data Security and integrity mechanisms
- Concurrent data access for multiple users
- User-oriented data query, manipulation, and reporting capabilities
- Programmer-oriented application system development capabilities



=> **Relational Model**

Relational Model - Properties

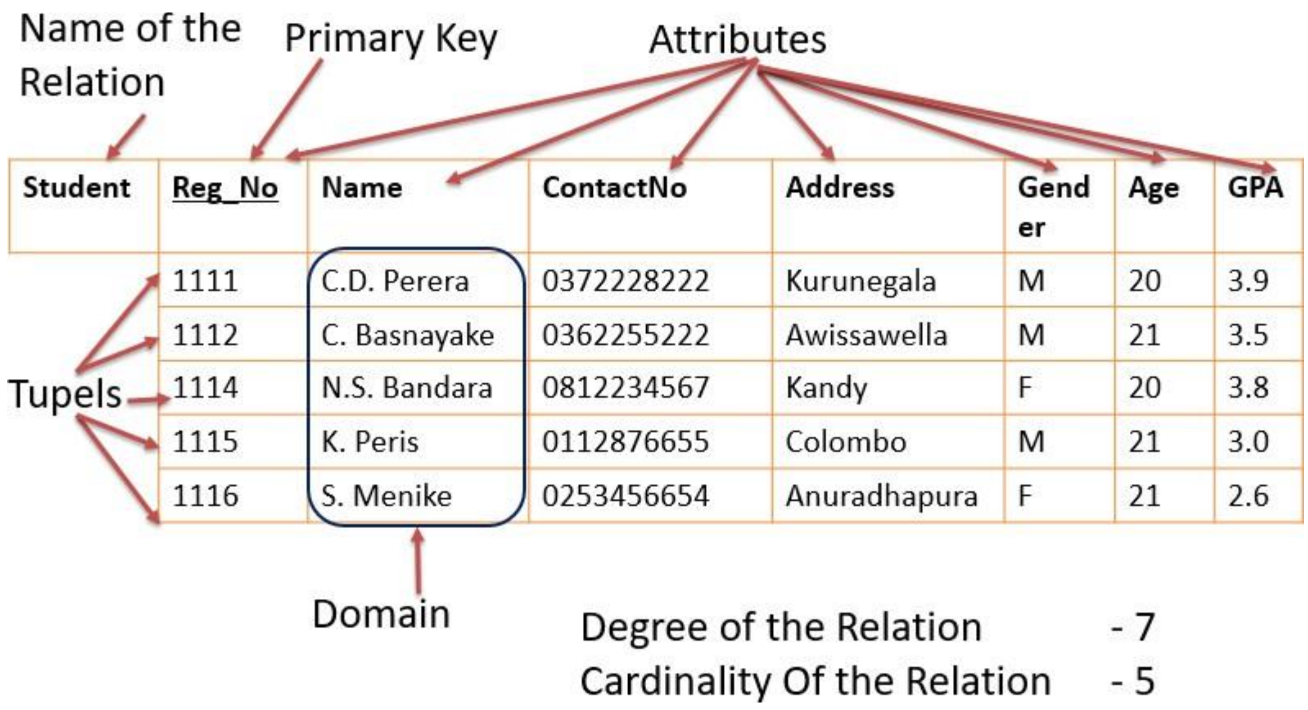
- Each relation (or table) in a database has a unique name.
- Each attribute (or column) within a table has a unique name.
- Each row is unique. no two rows in a relation are identical
- there can be no multi-valued attributes in a relation.
- An entry at the intersection of each row and column is atomic (or single-valued).

Attribute

- Each column in the relation is an **attribute** of the relation.
- The number of attributes of the relation is called the **DEGREE OF THE RELATION.**
- The set of possible values that an attribute may have is the **DOMAIN** of the attribute.

Tuple

- A **tuple** is a collection of values that makes up one row of relation.
- A tuple is the equivalent of a record.
- Number of tuples in a relation is its **CARDINALITY.**



Key

- **Primary Key -** Primary Key is an attribute that uniquely identifies each row in a relation
- **Foreign keys / (referential integrity) -**
  A foreign key is a set of columns in one table that serve as the primary key in another table
- **Composite Key / Concatenated Key -**
  Composite Key Is a primary key that consists of more than one attribute
- **Alternate Key -**
  Alternate Key is the candidate keys other than the chosen primary key are called alternate keys.
- **Candidate key -**
  Candidate key is a single key or a group of multiple keys that uniquely identify rows in a table
- **Super Key -**
  Super Key an attribute or a combination of attributes that is used to identify the records uniquely is known as Super Key.
- **Cardinality-** The cardinality of a table refers to the number of rows in the table
- **Degree -** The degree of a table refers to the number of columns
- **Query -** A query is a question about the data in your tables
- **Entity integrity -** Entity integrity constraint specifies that no primary key can be null

=> **Entity-Relationship (E-R) Model**

ERD is a data modeling technique used in software engineering to produce a conceptual data model of an information system.

Characteristics of E-R Model

- Semantic data model
- No physical DBMS
- Design tools and documentation for database structure
- Express the logical properties of an enterprise database

Components of E-R Model

1. Entity
2. Attribute
3. Key
4. Relationship
5. Structural constraints on relationship

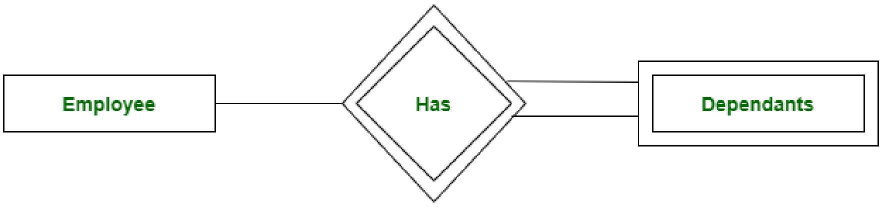
1. Entity

An object or concept Thing in real world with independent existence

"...anything (people, places, objects, events, etc.) about which we store information (e.g. supplier, machine tool, employee, utility pole, airline seat, etc.)."

Types

- Strong entity (parent, owner, dominant) - Strong Entity does not depend on other entity.
- Weak entity (child, dependent, or subordinate) - Weak entity is a depend on another entity



2. Attribute

attribute is Particular properties that describe entity

Types

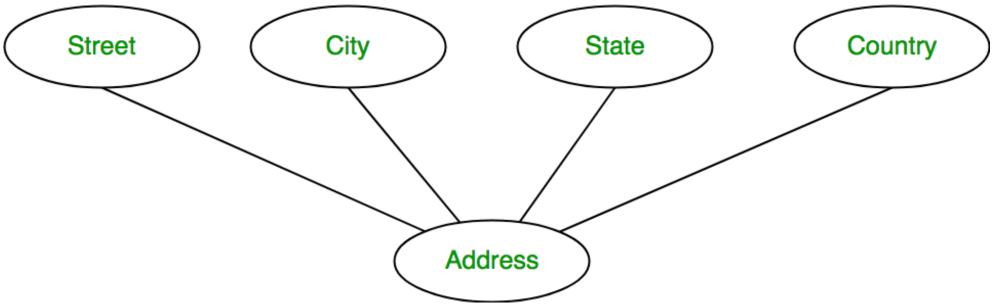
- Key attribute: Uniqueness property must hold for every entity set of the entity type
- Composite
- Multi-valued
- Derived
- Stored
- derived attributes

1. Key attribute

   The attribute which uniquely identifies each entity in the entity set is called the key attribute.

2. Composite Attributes

   Can be subdivided into smaller subparts



3. Multivalued Attribute

   An attribute consisting of more than one value for a given entity.

4. Derived Attributes

   An attribute that can be derived from other attributes of the entity type is known as a derived attribute.
   e.g.; Age (can be derived from DOB).

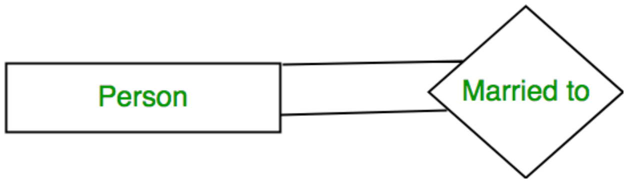
3. Degree of a relationship

Number of participating entities

Types
1. Unary (recursive relationship)
2. Binary
3. Ternary
4. Quaternary / N - ary

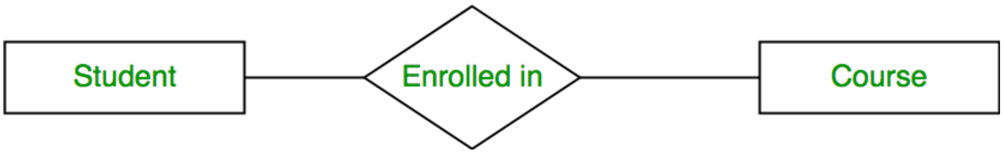
1. Unary (recursive relationship)

   When there is only ONE entity set participating in a relation, the relationship is called a unary relationship. For example, one person is married to only one person.



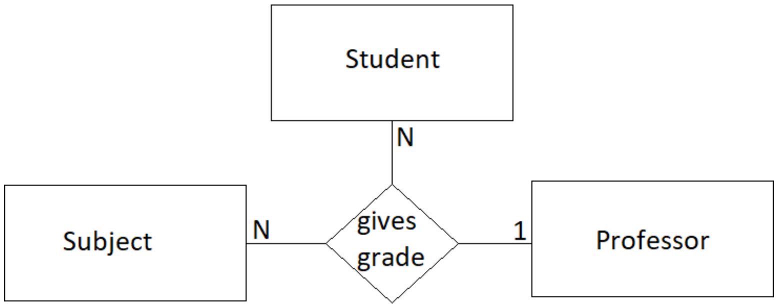
2. Binary relationship

   When there are TWO entities set participating in a relationship, the relationship is called a binary relationship. For example, a Student is enrolled in a Course.



3. Ternary relationship

   When there are three entities set participating in a relationship, the relationship is called a Ternary relationship.



4. Quaternary / N - ary relationship

   When there are n entities set participating in a relation, the relationship is called an n-ary relationship.

4. Structural constraints on relationships./ Cardinality

1. One-to-One

   In a particular hospital, the surgeon department has one head of department. They both serve one-to-one relationships

   Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one.

2. One-to-Many

In a particular hospital, the surgeon department has multiple doctors. They serve one-to-many relationships

3. Many-to-Many

In a particular company, multiple people work on multiple projects. They serve many-to-many relationships.

=> **Stages of the database system development life cycle**

1. Database planning
2. System definition
3. Requirements collection and analysis
4. Database design
5. DBMS selection (optional)
6. Application design
7. Prototyping (optional)
8. Implementation
9. Data conversion and loading
10. Testing
11. Operational maintenance

=> **EER Diagram (Enhanced Entity-Relationship diagrams)**

- **superclass (parent / Base) -** the class being inherited from
- **subclass (child / Derived) -** the class that inherits from another class

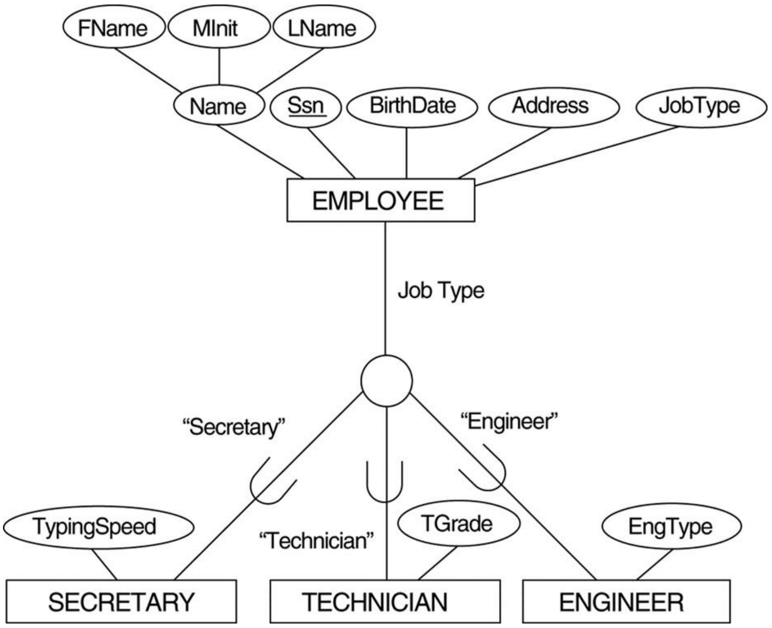
Attribute Inheritance

- **specialization:** maximizing the differences between members of an entity by identifying their distinguishing characteristics

- **generalization:** minimizing the differences between entities by identifying their common characteristics

EER - Specialization and Generalization Introduction || Enhanced Entity Relationship ...

1. **Example of specialization EER diagram**



mapping specialization the EER schema

(a) EMPLOYEE

SSN	FName	MInit	LName	BirthDate	Address	JobType

SECRETARY

SSN	TypingSpeed

TECHNICIAN

SSN	TGrade

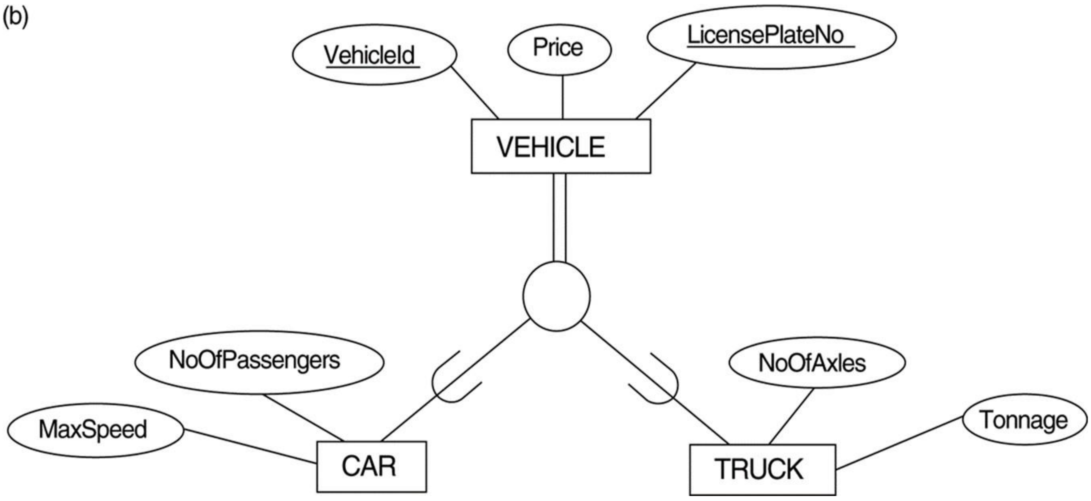
ENGINEER

SSN	EngType

2. **Example of Generalizing EER diagram**

CAR and TRUCK into the superclass VEHICLE.

(b)



mapping Generalizing the EER schema

(b) CAR

<u>VehicleId</u>	LicensePlateNo	Price	MaxSpeed	NoOfPassengers

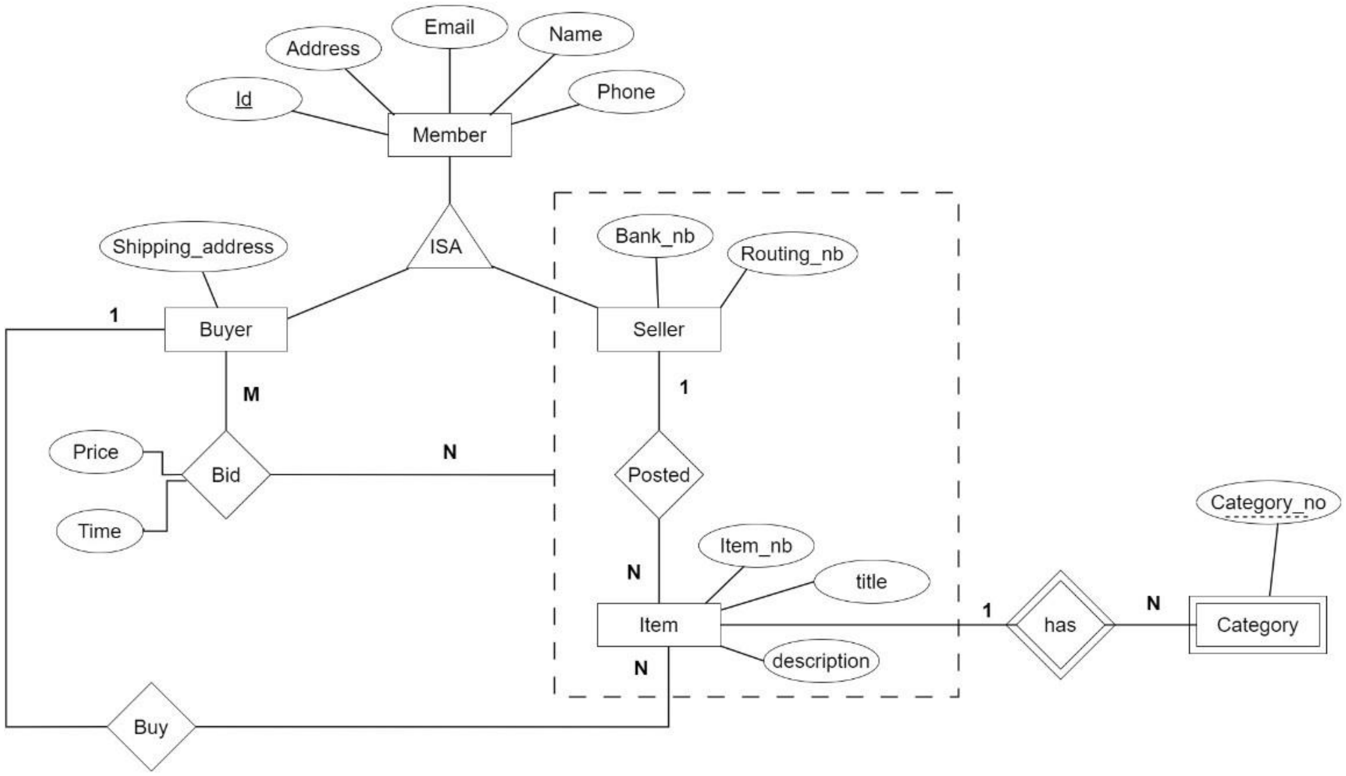
TRUCK

<u>VehicleId</u>	LicensePlateNo	Price	NoOfAxles	

Q1. Draw an EER diagram to cater the following requirements.

The members of an online auctioning site are identified by a unique member id and their names, email addresses, phone numbers and postal addresses should be maintained. A member may be a buyer or a seller. A buyer has a shipping address and a seller has a bank account number and routing number. Items are posted by a seller. When the seller posts the item a bid may be placed by buyers. A bidding price and the time of bid placement is recorded. The person with the highest bid price is declared as the winner and the buyer can then buy the item. Items are identified by a unique item number assigned by the system. Items are also described by an item title, an item description, a starting bid price and bidding increment. Item has a category with a category number. A category is identified by the category number and the item number together.

EER Diagram | Class Test | Y2S1 | DMS



=> **What is Normalization ?**

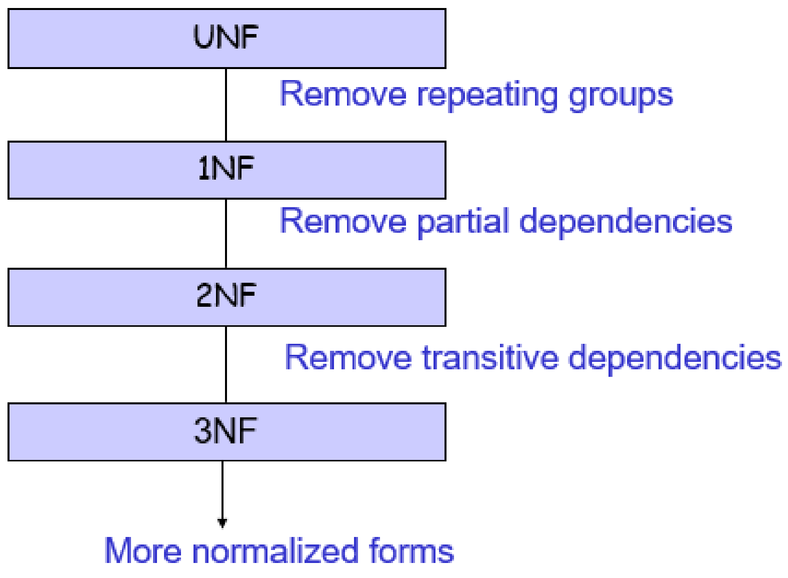
Normalization is the process of minimizing redundancy from a relation or set of relations.

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)

Data redundancy is undesirable because of the following anomalies

1. 'Insert' anomalies
2. 'Delete' anomalies
3. 'Update' anomalies

Normalization Flow



Advantages of Normalization

- Reduced data redundancy
- Simplified database design
- Easier database maintenance
- Improved data consistency
- Improved query performance

=> **SQL (Structured Query Language)**

SQL is a standard language for accessing and manipulating databases.

Numeric Types

- Integer (INT, INTEGER & SMALLINT)
- Real numbers (FLOAT, REAL, DOUBLE)
- Formatted numbers DECIMAL(i,j) or DEC(i,j) or NUMERIC(i,j)
- Character strings
  - Fixed length: CHAR(n) or CHARACTER(n)
  - Variable length: VARCHAR(n)
- Date and Time: DATE and TIME data types are supported in SQL2

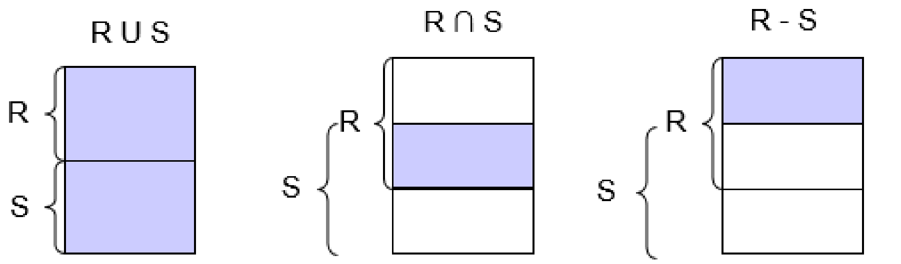
Three main categories of SQL

- DDL, Data Definition Language
- DCL, Data Control Language
- DML, Data Manipulation Language

SQL Aggregate Functions

- SUM
- AVG
- MIN
- MAX
- COUNT

Combining Results Tables



(a) Union (b) Intersection (c) Difference

UNION **INTERSECT** **EXCEPT**

=> **What is Database security?**

The mechanism that protects the database against intentional or accidental threats.

DB security is concerned with avoiding the following situation:

- Theft & fraud
- Loss of confidentiality (secrecy)
- Loss of privacy
- Loss of integrity
- Loss of availability

Countermeasures – Computer-Based Controls

1. **Authorization -**

   Authorization is the granting of a right or privilege that enables a subject to have legitimate access to a system or a system's object.

   Authorization is the process of giving someone the ability to access a resource.

2. **Authentication -**

   Authentication is a mechanism that determines whether a user is who he or she claims to be.

   Authentication is the process of validating the identity of a user or the integrity of a piece of data.

3. **Access control -**

The typical way to provide access controls for a database system is based on the granting and revoking of privileges.

Access Control is a Granting or denying approval to use specific resources

There are two types of Access controls

1. Discretionary Access Control (DAC)

   Most commercial DBMSs provide an approach called Discretionary Access Control (DAC),which manages privileges using SQL.

   The SQL standard supports DAC through the GRANT and REVOKE commands.

2. Mandatory Access Control (MAC)

   Some commercial DBMSs also provide an approach to access control called Mandatory Access Control (MAC), which is based on system-wide policies that cannot be changed by individual users.

   In this approach each database object is assigned a security class and each user is assigned a clearance for a security class, and rules are imposed on reading and writing of database objects by users.

   The SQL standard does not include support for MAC.

Approaches for securing a DBMS on the web

The security measures associated with DBMSs on the Web include

- proxy servers
- firewalls
- message digest algorithms and digital signatures
- digital certificates
- kerberos
- Secure Sockets Layer (SSL) and Secure HTTP (S-HTTP)
- Secure Electronic Transactions (SET) and Secure Transaction Technology (SST)
- Java security
- ActiveX security

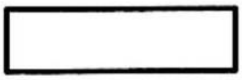
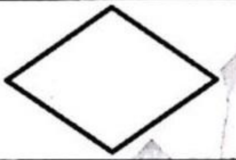
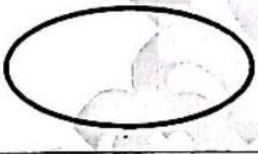
භූතාර්ථ සම්බන්ධතා අනුකෘතිය

Conceptual Design (ERD-Entity Relationship Diagram Modeling)

මෙම ආකෘතිය මුල් වරට Peter Chen විසින් 1976 දී හඳුන්වා දෙනලදී. පසුකාලීනව Charles Bachman විසින් සමහර මූලධර්ම වෙනස් කරන ලදී.

භූතාර්ථ සම්බන්ධතා අනුකෘතියක් තුලින් තවත් භූතාර්ථයක් අතර දත්ත ගලායන ආකාරය සහ වස්තූන් අතර සම්බන්ධතාවයක් ගොඩනගන ආකාරය මූලිකවම මෙම ආකෘතියේදී විස්තර කරයි.

භූතාර්ථ සම්බන්ධතා රූප සටහන් වල භාවිතා වන සංකේත **(ER Diagram Notation)**

භූතාර්ථ (Entity)	
සම්බන්ධතා (Relationship)	
උපලක්ෂණ (Attribute)	

භූතාර්ථ **(Entity)**

භූතාර්ථයක් යනු සැබෑ ලෝකයේ පවතින වස්තුවක් වන අතර සුවිශේෂී සර්වනාමයක් ලෙසද හඳුන්වයි. භූතාර්ථයක් ලෙස පුද්ගලයෙක්, ස්ථානයක්, හැකියාවක් හෝ සිද්ධියක් හැඳින්විය හැක.

උදා:- Student

භූතාර්ථ කුලක **(Entity Set)**

භූතාර්ථ කිහිපයක් එකිනෙකට භාවිතා කිරීම භූතාර්ථ කුලකයක් ලෙස හඳුන්වයි.

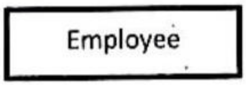
උදා:- 1. Set of all Person
2. Companies
3. Trees
4. Holidays

භූතාර්ථ වර්ග

Strong (Regular) Entity

එක් භූතාර්ථයක් මත නොයැපෙන භූතාර්ථ Strong Entity ලෙස හඳුන්වයි. මෙවා ස්වාධීන භූතාර්ථ ලෙසද හඳුන්වයි.

උදා:- 1. Employee

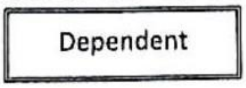


දුර්වල භූතාර්ථ

Weak Entity

දුර්වල භූතාර්ථයක් යනු වෙනත් භූතාර්ථයක් මත යැපෙන භූතාර්ථ වේ.

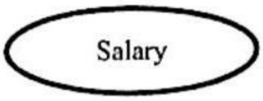
උදා:- 1. Dependent



උපලක්ෂණ **(Attributes)**

භූතාර්ථයක් විස්තර කිරීමට භාවිතා කරන තොරතුරු උපලක්ෂණ ලෙස හඳුන්වයි.

උදා:- Salary



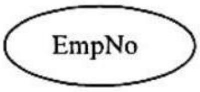
උපලක්ෂණ වර්ග

1. සරල උපලක්ෂණ (Simple Attributes)
2. සංයුක්ත උපලක්ෂණ (Composite Attributes)
3. බහු උපලක්ෂණ (Multi- Valued Attributes)
4. ව්‍යුත්පන්න උපලක්ෂණ (Derived Attributes)
5. ගබඩා උපලක්ෂණ (Stored Attributes)
6. විස්තරාත්මක උපලක්ෂණ
7. තනි අගය සහිත උපලක්ෂණ /පරමාණුක උපලක්ෂණ (Single Valued Attributes)

සරල උපලක්ෂණ (Simple Attributes)

උපලක්ෂණයක අඩංගු දත්ත නැවත කුඩා කොටස්වලට වෙන්කල නොහැකි නම් එවැනි උපලක්ෂණයක් සරල උපලක්ෂණයක් ලෙස හඳුන්වයි.

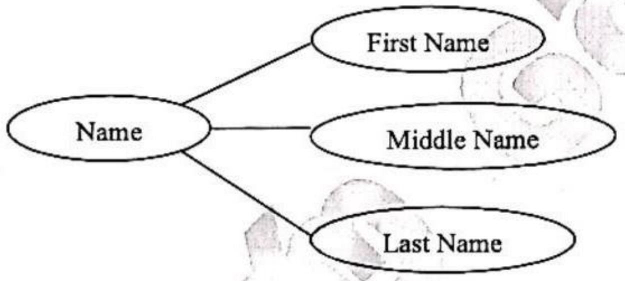
උදා:-



සංයුක්ත උපලක්ෂණ (Composite Attributes)

උපලක්ෂණයක් අංග කිහිපයක් එක්වීමෙන් සෑදී ඇත්නම් එවැනි උපලක්ෂණයක් සංයුක්ත උපලක්ෂණයක් ලෙස හඳුන්වයි

උදා:-



බහු උපලක්ෂණ (Multi- Valued Attributes)

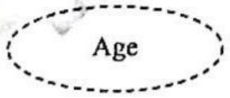
උපලක්ෂණයන් තුල අගයන් / දත්ත එකකට වැඩි ප්‍රමාණයක් ගබඩා කලයුතු වේ. එවැනි උපලක්ෂණයක් බහු අගයන් සහිත උපලක්ෂණයක් ලෙස හඳුන්වයි.

උදා:- 1. Employee Skills
2. Qualifications

ව්‍යුත්පන්න උපලක්ෂණ (Derived Attributes)

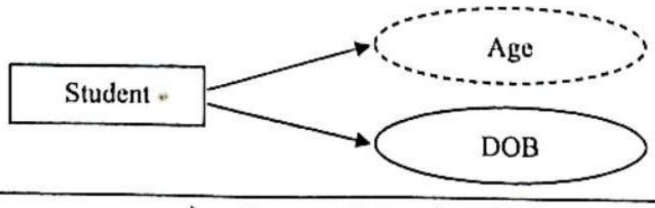
ගබඩා උපලක්ෂණයක් භාවිතාකර සාදා ගන්නා උපලැබියන ව්‍යුත්පන්න උපලක්ෂණයක් ලෙස හඳුන්වයි. වගු නිර්මාණයේදී ව්‍යුත්පන්න උපලක්ෂණ වගුව සඳහා භාවිතා නොකරයි.

උදා:-



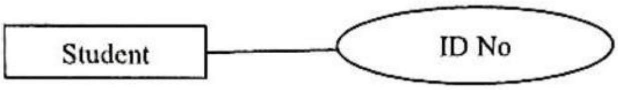
ගබඩා උපලක්ෂණ (Stored Attributes)

දත්ත පාදකයක වගුවක් තුල දත්ත ගබඩාකර තබා ගැනීමට භාවිතා කරන උපලක්ෂණය ගබඩා උපලක්ෂණයක් වේ. මෙම උපලක්ෂණයේ ඇති දත්තය භාවිතයෙන් ව්‍යුත්පන්න උපලක්ෂණයක් සාදා ගතහැක.



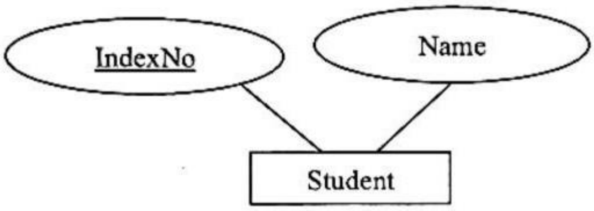
තනි අගයක් සහිත උපලක්ෂණ (Single Valued Attributes)

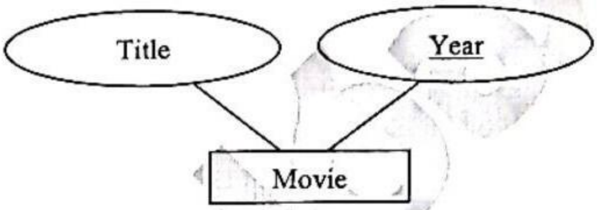
තනි අගයක් පමණක් ගබඩාකර තබා ගන්නා උපලක්ෂණයක් තනි අගයක් සහිත උපලක්ෂණයක් ලෙස හැඳින්වේ.



යතුරු උපලක්ෂණ / ප්‍රධාන උපලක්ෂණ (Key Attribute)

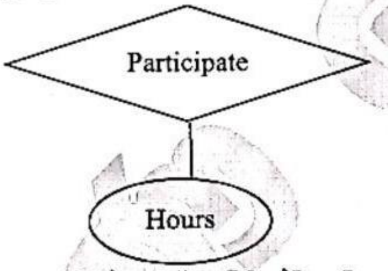
භූතාර්ථයක උපලැබියානයක් අනන්‍යව හඳුනාගැනීම සඳහා භාවිතා කරන උපලක්ෂණය යතුරු උපලක්ෂණය ලෙස හඳුන්වයි.



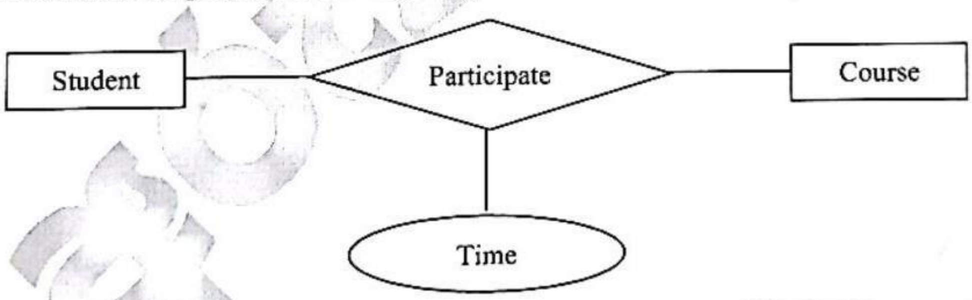


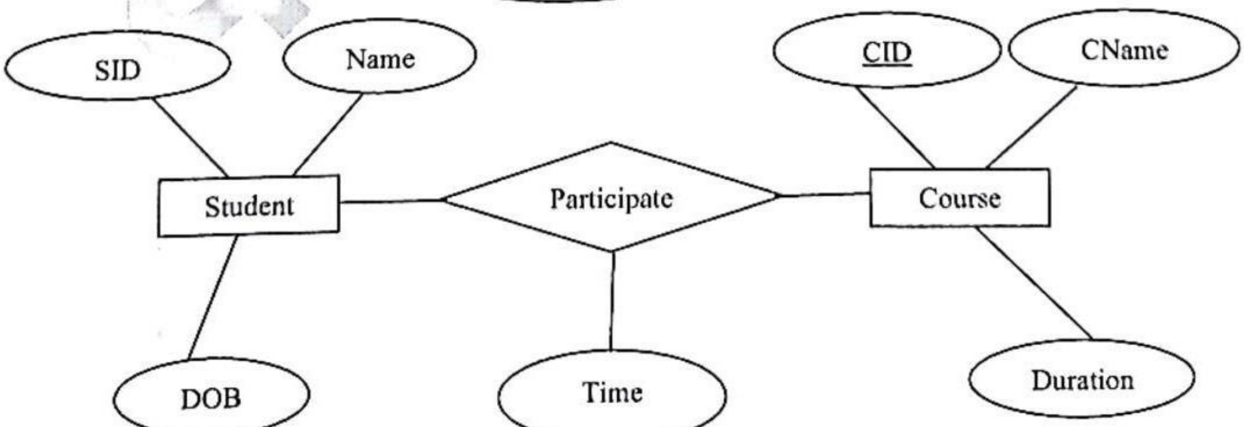
විස්තරාත්මක උපලක්ෂණ

සම්බන්ධතාව මත පැවරෙන උපලක්ෂණ විස්තරාත්මක උපලක්ෂණ ලෙස හඳුන්වයි.



උදා:- 1. ශිෂ්‍යයෙකු හා පාඨමාලාවක් අතර සම්බන්ධතාවයක් ගොඩනැගීමේදී පාඨමාලාව සඳහා ගතවන කාලය විස්තරාත්මක උපලක්ෂණයක් ලෙස හඳුන්වයි.





භූතාර්ථ හඳුන්වන (**Identifiers**)

නිදර්ශයක් අනන්‍යව හඳුනාගැනීම සඳහා භාවිතා කරන උපලක්ෂය හෝ උපලක්ෂණ කිහිපයක් භූතාර්ථයක් ලෙස හඳුන්වයි. ප්‍රධාන හඳුන්වන වර්ග 2කි.

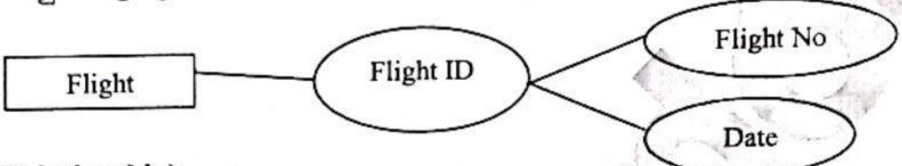
1. සරල හඳුන්වන
2. සංයුක්ත හඳුන්වන

සරල හඳුන්වන

සරල උපලක්ෂණයක් භාවිතා කරන හඳුන්වන සරල හඳුන්වන ලෙස හඳුන්වයි.

සංයුක්ත හඳුන්වන

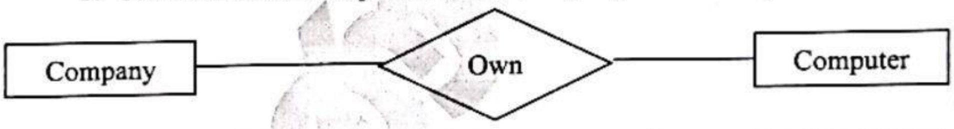
හඳුන්වනයක් සංයුක්ත උපලක්ෂණ භාවිතා කරයි නම් එය සංයුක්ත හඳුන්වනයක් වේ.



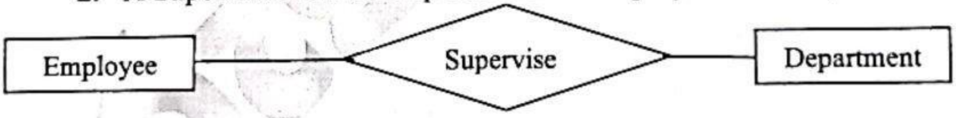
සම්බන්ධතා (**Relationship**)

- භූතාර්ථ 2ක් හෝ කිහිපයක් අතර ඇති කරන සබැඳියාව වේ.
- සම්බන්ධතාවය සර්වනාමයට අදාල ක්‍රියාවක් වේ.

1. An own relationship between a company and a computer



2. A Supervises relationship between an employee and a department



3. A performs relationship between an artist and a song

4. A proved relationship between a mathematication and a theorem

සම්බන්ධක කුලක (**Relationship Sets**)

භූතාර්ථ නිදර්ශන සම්බන්ධතාවය අනුව ඇති කරන ආකාරය ප්‍රධාන ආකාර 2ක් හැඳින්විය හැකිය.

1. මට්ටම (Degree)
2. ගණනීයතාව (Cardinality)

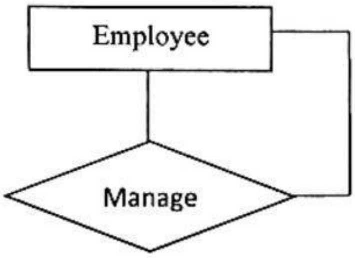
සම්බන්ධතා මට්ටම (Degree of Relationship)

සම්බන්ධතාවයකට අයත්වන භූතාර්ථ ප්‍රමාණය සම්බන්ධතා මට්ටම ලෙස හඳුන්වයි.

- ප්‍රධාන සම්බන්ධතා මට්ටම් 4කි.
  1. Unary (Degree 1) (Recursive)
  2. Binary (Degree 2)
  3. Ternary (Degree 3)
  4. N- ary (Degree above 3)

Unary Relationship

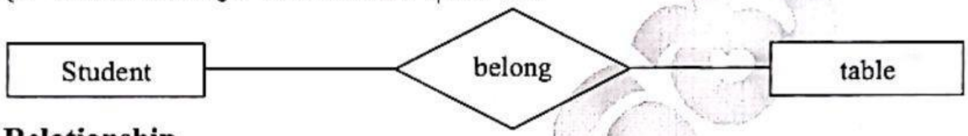
- සම්බන්ධතාවයකට එක් භූතාර්ථයක් පමණක් අයත් වේ නම් එය Unary වේ.
  උදා:- සේවකයෙක් සේවකයන් කිහිපදෙනෙක් පාලනය කරයි.



Binary Relationship

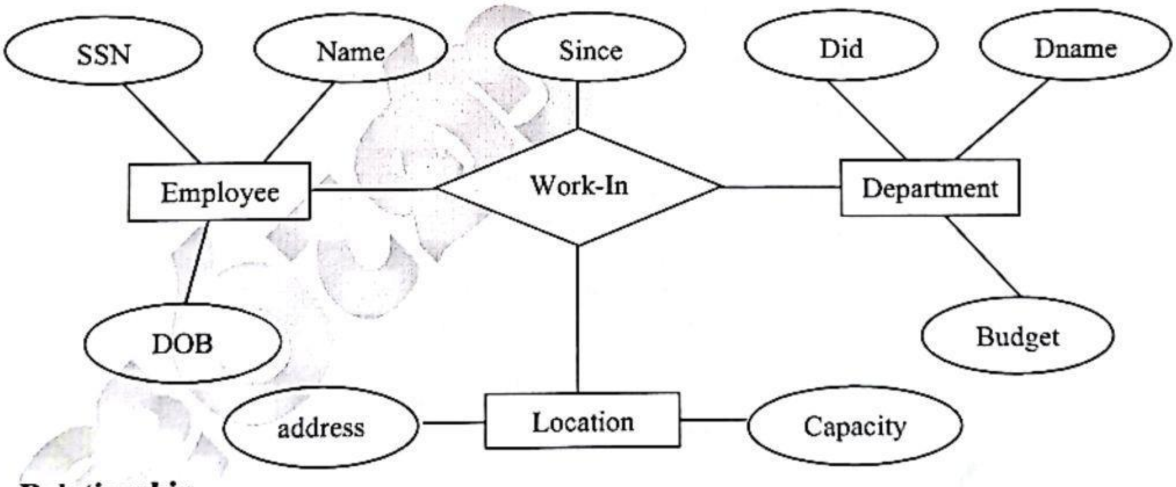
එක සම්බන්ධතාවයකට භූතාර්ථ 2ක් අයත් වේ නම් එවැනි සම්බන්ධතාවයක් ද්විමය සම්බන්ධතාවයක් ලෙස හඳුන්වයි.

උදා:- එක් ශිෂ්‍යයෙකුට එක් මේසයක් අයත් වේ.



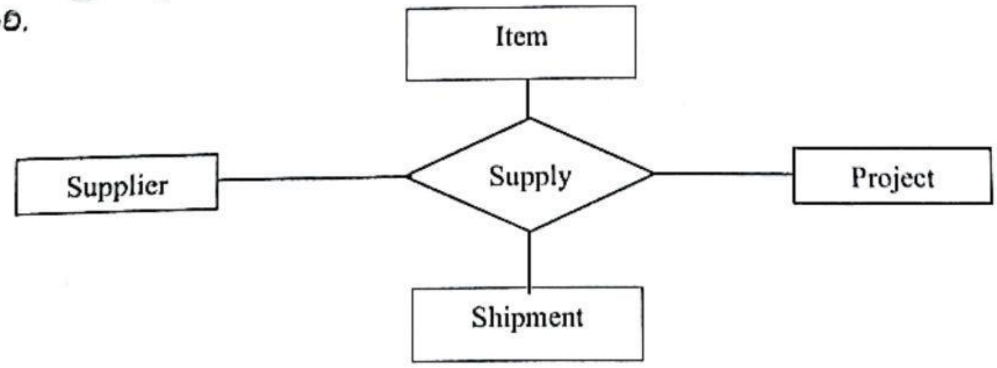
Ternary Relationship

සම්බන්ධතාවයකට භූතාර්ථ 3ක් සම්බන්ධ කලහැකි නම් එය තෘතික ලෙස හැඳින්වේ.



N-ary Relationship

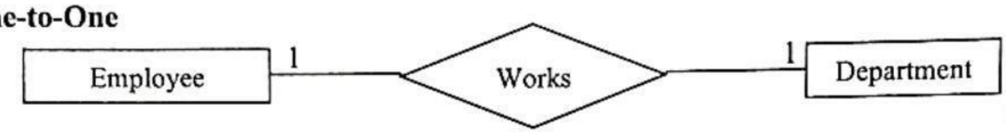
සම්බන්ධතාවයකට භූතාර්ථ 3කට වඩා වැඩි ප්‍රමාණයක් අයත් වේ නම් එය N-ary සම්බන්ධතාවයක් ලෙස හැඳින්වේ.



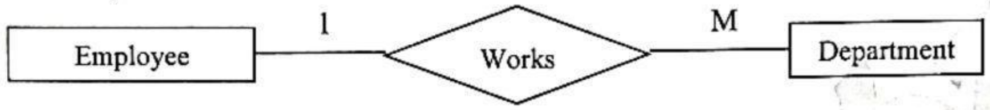
ගණනීයතාව **(Cardinality)**

- එක් භූතාර්ථයක් තවත් භූතාර්ථයක් සමඟ පැවතිය යුතු අනන්‍යතා සම්බන්ධතා ප්‍රමාණය ගණනීයතාවය ලෙස හඳුන්වයි.
- ගණනීයතා ප්‍රධාන ආකාර 3 කි.
    1. ඒක -ඒක One-to-One (1:1)
    2. ඒක - බහු One-to-Many (1:M)
    3. බහු - බහු Many-to-Many (M:N)

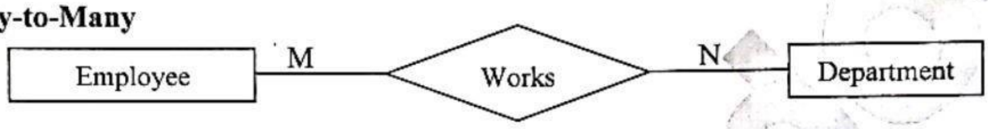
One-to-One



One-to-Many



Many-to-Many



වෛකල්පිත **(Optionally)**

සම්බන්ධතාවයකට අවම වශයෙන් ගොඩනැගිය හැකි සම්බන්ධතා වෛකල්පිත ලෙස හඳුන්වයි. (අත්‍යවශ්‍ය නොවන සම්බන්ධතා)

Optional

සහභාගිත්වය අවම හෝ ශූන්‍ය වේ නම් එය Optional ලෙස හඳුන්වයි.

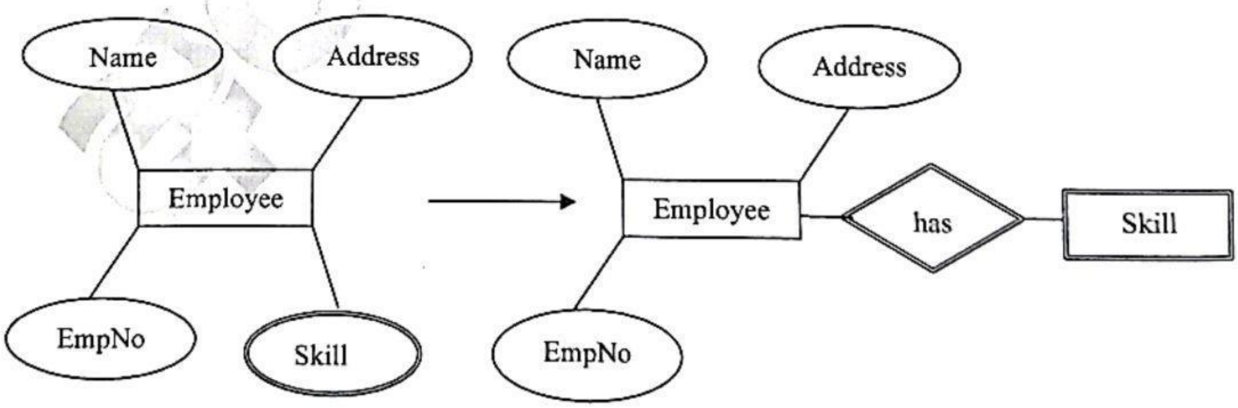
Mandatory

සහභාගිත්වය අවම එකක් හෝ කිහිපයක් වේ නම් එය Mandatory ලෙස හඳුන්වයි.

බහු වටිනාකම් සහිත උපලක්ෂණ ඉවත් කිරීම

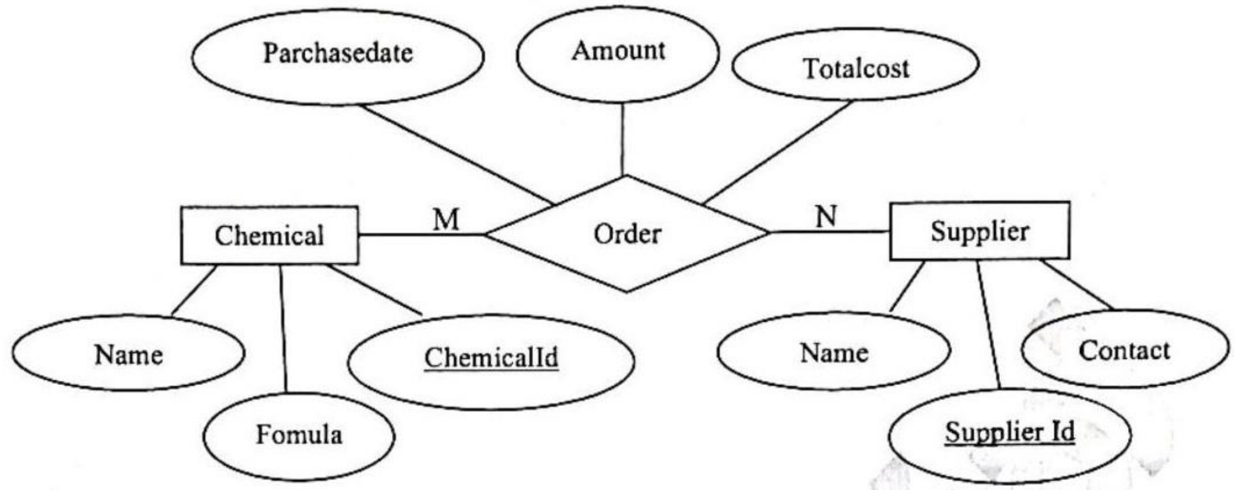
Remove Multivalue Attribute - Weak Entity

බහු උපලක්ෂණ සහිත භූතාර්ථයක් වගු බවට පත් කිරීමේදී එම උපලක්ෂණය සහිත දුර්වල භූතාර්ථයක් සාදයි.



බහු බහු සම්බන්ධතාවය ඉවත් කිරීම

Remove Many-to-Many Relationship



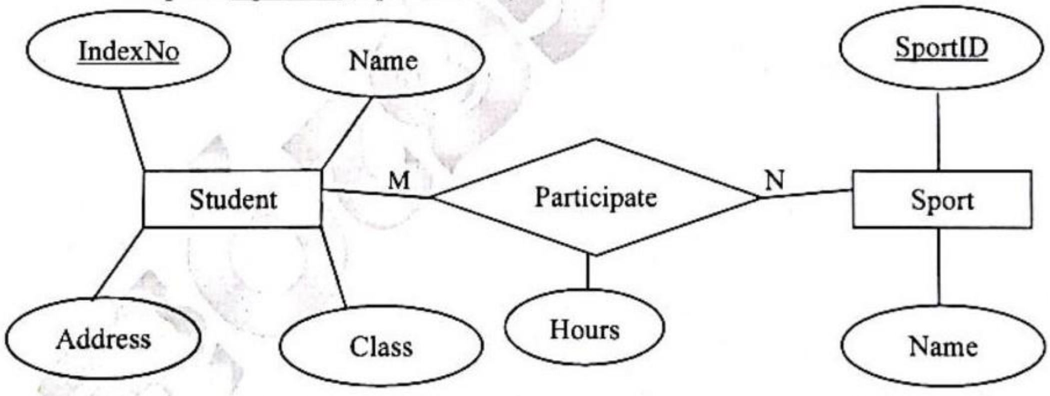
උදාහරණ 1

පාසලක සිසුහු වොලිබෝල්, ජවන හා පිටිය මළල ක්‍රීඩා, මේස පන්දු විවිධ ක්‍රීඩා සඳහා සහභාගී වෙති. ක්‍රීඩා සඳහා සහභාගී වන සිසු සිසුවියන් ඇතුල්වීමේ අංකය, ශිෂ්‍ය නාමය, නිවසේ ලිපිනය, පන්තිය හා සහභාගීවන ක්‍රීඩා ඇතුලත් නාම ලේඛනයක් පවත්වාගෙන යාමට විදුහල්පතිවරයාට අවසර ඇත. එක් සිසුවෙකුට එක් ක්‍රීඩාවකට වඩා සහභාගී වියහැකිය. නිශ්චිත ක්‍රීඩාවක් සඳහා එක් සිසුවෙකුට වඩා වැඩි සංඛ්‍යාවක් සිටිය හැක. එක් එක් ශිෂ්‍යයාට කලින් නියම කරන ලද කාල ප්‍රමාණයක් මත ක්‍රීඩා සඳහා සහභාගී වියහැකිය.

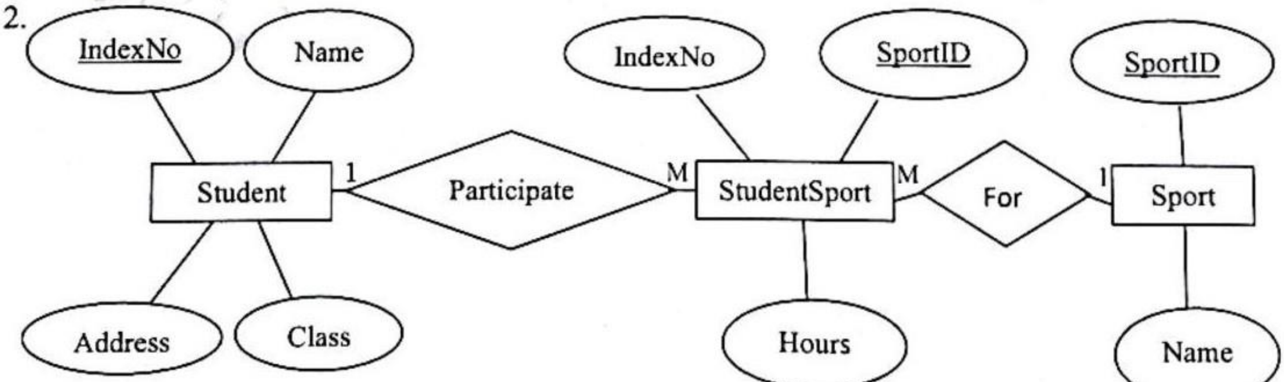
1. ඉහත සම්බන්ධතාවයට අදාල ER ආකෘතිය අඳින්න.
2. Many-to-Many අවස්ථාව හරකර නැවත ER ආකෘතිය අඳින්න.
3. භූතාර්ථවලට අදාල යතුරු උපලක්ෂණ ලියන්න.

Nouns: - Student (<u>Index No,</u> Name, Address, Class)
Sport (<u>Sport ID</u>, Sport Name)

1.



2.

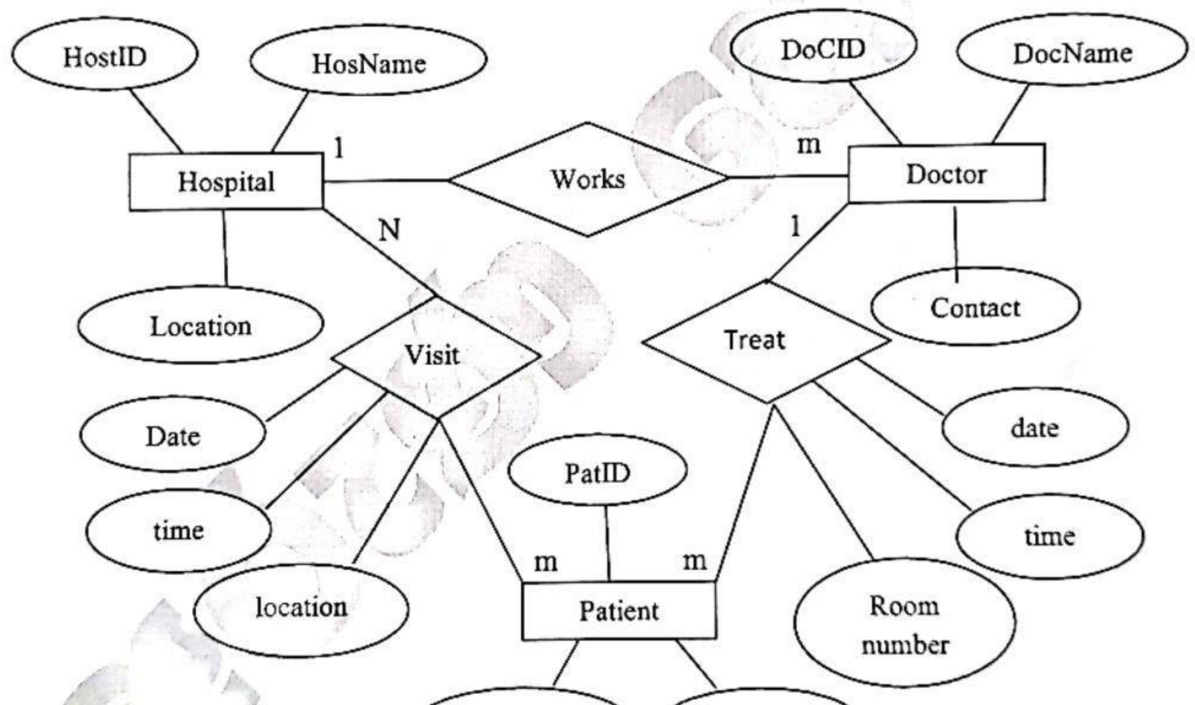


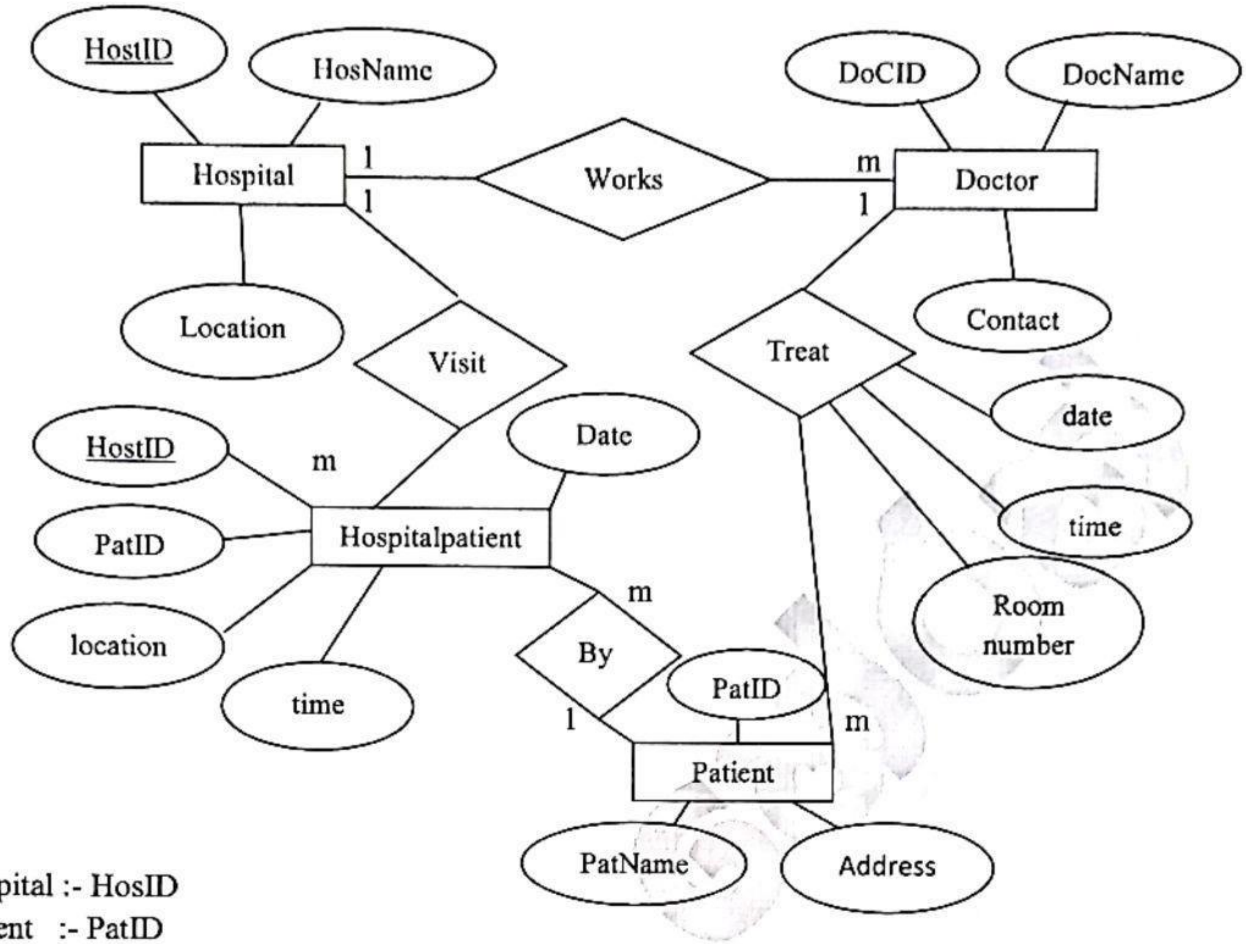
3.
Student:- Index No
Sport:- Sport ID
Studentsport:- Index No+Sport ID

උදාහරණ 2

සෞඛ්‍යය අමාත්‍යංශය විවිධ ආරෝග්‍යශාලා පාලනය කරයි. සෑම රෝගියෙකුටම විවිධ රෝගවලින් ප්‍රතිකාර ගතහැකිය. සෑම රෝහලකම එක් රෝගියෙකු සඳහා වෛද්‍යවරයෙක් අනුයුක්තව ඇත. සෑම රෝහලකම වෛද්‍යවරු කිහිපදෙනෙකු සිටී. නමුත් එක් වෛද්‍යවරයෙකු එක් රෝහලකට පමණක් සේවය කරයි.
එක් රෝහලක එක් වෛද්‍යවරයෙක් රෝගීන් කිහිපදෙනෙකුට ප්‍රතිකාර කරයි. රෝහලට එන රෝගීන්ට ප්‍රතිකාර ලබා ගැනීමට දිනයක් හා වේලාවක් කලින් ලබා දී ඇත.

Hospital (<u>HosID</u>, hosName, Location)
Patient (<u>PatID</u>, PatName, Address)
Doctor (<u>DocID</u>, DocName, Contact)

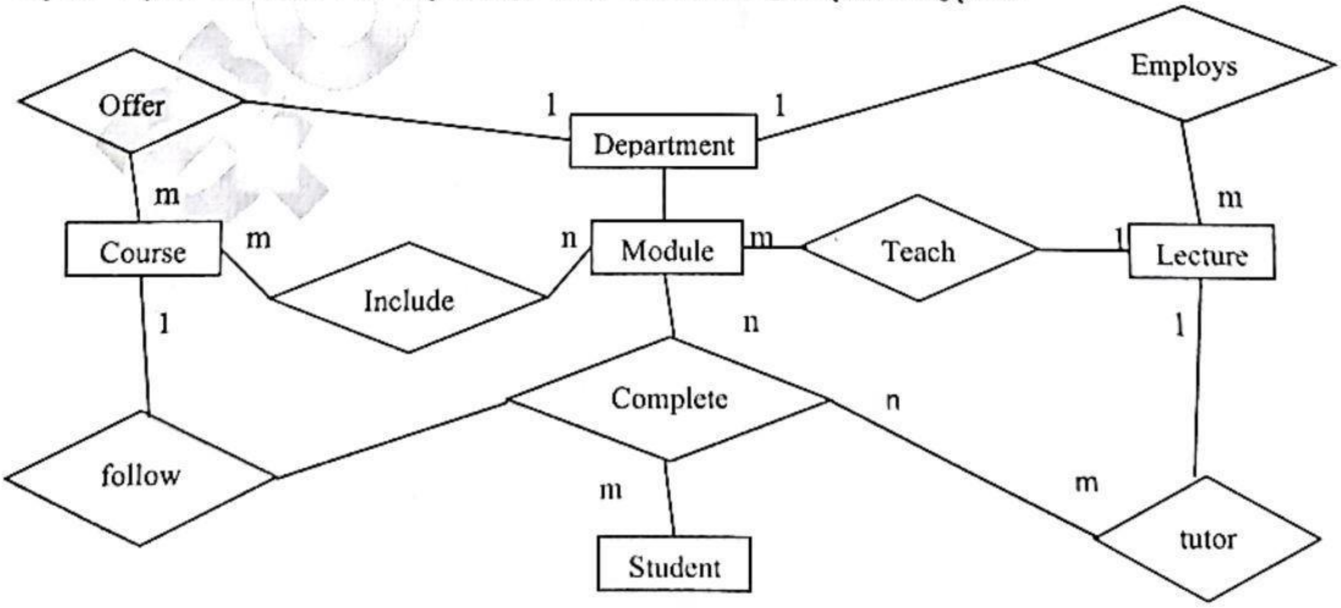




Hospital :- HosID
Patient :- PatID
Doctor :- DocID
Hospitalpatient:- HosID+ PatID

උදාහරණ 3

විශ්ව විද්‍යාලයක් තුල අධ්‍යයන අංශ කිහිපයක් ඇත. එම සෑම අංශයක්ම විවිධ පාඨමාලා පවත්වයි. සෑම පාඨමාලාවක්ම මොඩියුල කිහිපයකින් සමන්විත වේ. පාඨමාලාවක් සඳහා ලියාපදිංචි වු සෑම ශිෂ්‍යයෙකුම එම පාඨමාලාවට අදාල ඒකක සම්පූර්ණ කලයුතුය. අධ්‍යයන අංශවලට අනුයුක්ත දේශකයන් විසින් සෑම ඒකකයක් සඳහාම දේශන පවත්වයි. සෑම දේශයෙකුම ශිෂ්‍ය කණ්ඩායම් උපදේශකයෙකුද වේ.



ප්‍රමතකරණය (Normalization)

දත්ත සමුදායක ඇති දත්ත පිළිවෙළකට අනුව සැකසීම ප්‍රමතකරණය ලෙස සැලකේ. මෙහි දී වගු කිහිපයක් අතර සම්බන්ධතා නිර්මාණයේ දී රීති කිහිපයක් අනුගමනය කිරීමට යොමු කෙරේ. මෙමගින් දත්ත පුනර්යෝජනය (Data Redundancy) සහ දත්ත අක්‍රමිකතාව (Data Anomalies) අවම කර දත්ත සමුදායේ ඇතිවන අසංගතතාව ඉවත් කිරීමට හැකි වේ.

- **දත්ත පුනර්යෝජනය (Data Redundancy)**

  වගුවක් තුළ උපලැකියානක් (Record) ලෙස යෙදී ඇති දත්තයක් එම වගුව තුළ නැවත නැවත යෙදීම දත්ත පුනර්යෝජනය ලෙස හැඳින්වේ.

- **දත්ත අක්‍රමිකතාව (Data Anomalies)**

  දත්ත වගුවකට දත්ත ඇතුළත් කිරීමට යාමේ දී ඇති වන දෝෂ දත්ත අක්‍රමිකතාව ලෙස හැඳින්වේ. මේ යටතේ මූලික ප්‍රවේශ 3කි.

  1. දත්ත වගුව යාවත්කාලීන කිරීමට නොහැකි වීම. (Update Anomalies)
  2. දත්ත වගුවකින් දත්ත ඉවත් කිරීමට නොහැකි වීම. (Deletion Anomalies)
  3. වගුවකට දත්ත ඇතුළත් කිරීමේ දී වෙනත් දත්ත නිසා ඇතුලත් කිරීමට නොහැකි වීම. (Insertion Anomalies)

ප්‍රමතකරණය සඳහා යොදාගත හැකි උපක්‍රම ගණනාවකි. ඉන් කීපයකි.

1. පළමු ප්‍රමතකරණ විධිය (1st Normal form)
2. දෙවන ප්‍රමතකරණ විධිය (2st Normal form)
3. තෙවන ප්‍රමතකරණ විධිය (3st Normal form)

- **පළමුප්‍රමතකරණ විධිය (1st Normal form)**

  දත්ත වගුවක් නිර්මාණයේ දී,

  - එක හා සමාන උපලැකි (Fields) නැවත නැවත යෙදීම වැළැක්වීම.
  - එකිනෙක සම්බන්ධ උපලැකි (Fields) ඇතුළත් කර වෙනත් වගු නිර්මාණය කිරීම.
  - එකිනෙක සම්බන්ධ උපලැකි (Fields) ප්‍රාථමික යතුර මගින් හඳුනාගත හැකි බව තහවුරු කර ගැනීම.

  යන කරුණු පිළිබඳව මෙම ප්‍රමතකරණ විධියේ දී අවධානය යොමු කෙරේ.

- **දෙවන ප්‍රමතකරණ විධිය (2st Normal form)**

  දත්ත සමුදාය නිර්මාණයේ දී,

  - එකිනෙක සම්බන්ධ උපලැකි සහිත වගු වෙන් කර ගැනීමෙන් අනතුරුව එක ම උපලැකියක් නැවත තවත් වගුවක යෙදීම වැළැක්වීම,
  - නිර්මාණය කරන ලද වගු අතර ආගන්තුක යතුර භාවිතකර සම්බන්ධතාවයන් ගොඩ නැගීම,

  යන කරුණු පිළිබඳව මෙම ප්‍රමතකරණ විධියේ දී අවධානය යොමු කෙරේ.

- **තෙවන ප්‍රමතකරණ විධිය (3st Normal form)**

වගුවක් නිර්මාණයේ දී,

- එහි ඇති උපලැකි සියල්ලක්ම ප්‍රාථමික යතුර මත ක්‍රියාත්මක වන්නේදැයි සොයා බැලීම හා එසේ නොවන ඒවා ඇත්නම් ඒවා ඉවත් කිරීම,

යන කරුණු පිළිබඳව මෙම ප්‍රමතකරණ විධියේ දී අවධානය යොමු කෙරේ.

<u>වගුවක් ප්‍රමතකරණය කිරීම</u>

නිදර්ශනය : -

Student

STIndex	SName	ClassT	Grade	Coach1	Coach2	Coach3
710	Saman Silva	C20	11A	T125	T127	T130
515	NimalPerera	C25	12B	T127	T125	T130
712	SudathLekam	C20	11A	T130	T125	T127

එකම දත්තය ඇතුළත් කිරීම සඳහා වගුවක් තුළ උපලැකි කීපයක් තැබීමට ඉඩ නොදිය යුතුයි. ඒ අනුව සිසුන්ගේ තොරතුරු ඇතුළත් කරන වගුවක ක්‍රීඩා විෂයභාර ගුරුවරුන් කීප දෙනෙක් වෙත සිසුන් යොමු වීම පෙන්වීම සඳහා නිර්මාණය කර ඇති වගුවක් ඉහත දැක්වේ.

- **පළමුප්‍රමතකරණ විධිය (1st Normal form) - සමානනැවත නැවත යෙදීම වැළැක් වීම.**

මෙහි Coach1,Coach2,Coach3 වශයෙන් උපලැකි වෙන් වෙන් වශයෙන් නොයොදා ඒම දත්තය එක් උපලැකියක් වෙතට එක් කර ගත යුතුයි.

ඒ අනුව ඉහත වගුව පළමු ප්‍රමතකරණ විධියට සකස් කළ විට එහි ස්වරූපය පහත ආකාරය ගනී.

Student

STIndex	SName	ClassT	Grade	Coach
710	Saman Silva	C20	11A	T125
710	Saman Silva	C20	11A	T127
710	Saman Silva	C20	11A	T130
515	NimalPerera	C25	12B	T127
515	NimalPerera	C25	12B	T125

515	NimalPerera	C25	12B	T130
712	SudathLekam	C20	11A	T130
712	SudathLekam	C20	11A	T125
712	SudathLekam	C20	11A	T127

උපලැකි කීපයක තිබූ දත්ත මෙම වගුවේ දී එක් උපලැකියකට එක් කර ඇති අතර ඒ නිසා ම උපලැකියාන පුනර්යෝජනය (Redundancy) වී ඇත.

- **දෙවන ප්‍රමතකරණ විධිය (2st Normal form) - දත්තපුනර්යෝජනය වළක්වා ගැනීම.**

මෙසේ දත්ත පුනර්යෝජනය වැලක්වා ගැනීම සඳහා වගු දෙකක් සකස් කළ යුතුයි. එසේ වගු දෙකක් සකස් කිරීමෙන් ඉහත වගුවේ ඇති පුනර්යෝජනය නැති කර ගැනීමට හැකි වේ. පහත දැක්වෙනුයේ එසේ සකස් කර ගත් වගු දෙකයි.

Student_Details

STIndex	SName	ClassT	Grade
710	Saman Silva	C20	11A
515	NimalPerera	C25	12B
712	SudathLekam	C20	11A

Student_Teacher

STIndex	Coach
710	T125
710	T127
710	T130
515	T127
515	T125
515	T130
712	T130
712	T125
712	T127

එසේම මෙම වගු දෙක අතර ආගන්තුක යතුර නිර්මාණය කළ යුතු ආකාරය පිළිබඳවද මෙම අවස්ථාවේ දී අවධානය යොමු කළ යුතුයි.

- **තෙවන ප්‍රමතකරණ විධිය (3st Normal form) - ප්‍රාථමික යතුර මත රඳා නොපවතින උපලැකි වෙන් කිරීම.**

පහත වගුවේ ඇති සියලුම උපලැකි ප්‍රාථමික යතුර මත ක්‍රියාත්මක වන්නේ ද යන්න සොයා බැලිය යුතුයි. එසේ ම කිසියම් උපලැකි දෙකක් ප්‍රාථමික යතුරේ බලපෑමක් නොමැතිව හඳුනාගැනීමට හැකියාවක් ඇත්නම් ඒවා ඉවත් කර නව වගුවක් නිර්මාණය කළ යුතු යි.

Student_Details

STIndex	SName	ClassT	Grade
710	Saman Silva	C20	11A
515	NimalPerera	C25	12B
712	SudathLekam	C20	11A

ඉහත නිදර්ශනයේ Grade යන උපලැකිය මඟින් ClassT යන උපලැකිය හඳුනාගත හැකි අතර ඒ සඳහා ප්‍රාථමික යතුර බලපෑමක් සිදු නො වේ.

එහෙයින් මෙම තත්ත්වය මඟහරවා ගැනීම සඳහා නැවතත් මෙම වගුව වගු දෙකකට වෙන් කර ගැනීම කර ගත හැකියි.

Student_ClassT

STIndex	SName
710	Saman Silva
515	NimalPerera
712	SudathLekam

Student_Grade

ClassT	Grade
C20	11A
C25	12B
C20	11A

Numeric Types

Tiny Int :	127 සිට (-128) දක්වා
Small Int:	32,768 සිට (-32,767) දක්වා
Medium Int:	8,388,608 සිට (-8,388,608) දක්වා
Int :	2^{31} සිට ($-2^{31}-1$) දක්වා
Big Int:	2^{63} සිට ($-2^{63}-1$) දක්වා
Float :	දශමස්ථාන $1.1X10^{38}$ සිට ($-1.1X10^{38}$) දක්වා
Double:	දශමස්ථාන $1.7X10^{308}$ සිට ($-1.7X10^{308}$) දක්වා

String Types

CHAR:	නිශ්චිත අකුරු ප්‍රමාණයක් සහිතය
VARCHAR:	අකුරු ගණන නිශ්චිතව කිව නොහැක
BLOB:	බයිට 2^{16} ක දත්ත ඇතුලත්වේ

Constrains

NOT NULL:	ශුන්‍ය විය නොහැක
UNIQUE:	එකිනෙකට වෙනස් අනන්‍ය අගයන් විය යුතුය
PRIMARY KEY:	NOT NULL සහ UNIQUE වූ එකතුවකි
FOREIGN KEY:	
DEFAULT:	අගයක් ඇතුලත් නොකලවිට තිබිය යුතු අගය

Operators

=	Equal
<>	Not Equal
>	greater than
<	Less Than
>=	Less Than or Equal
<=	Greater than or Equal
LIKE	Search for a Pattern
BETWEEN	අගයන් දෙකක් අතර

Primary Key - ප්‍රාථමික යතුර

- උපලැකියානයක් වෙන්කර හඳුනාගැනීමට භාවිතා කරයි.
- අනන්‍ය වේ (Unique)
- ශුන්‍ය අගයක් (NULL) ලබාදිය නොහැක
- මුල් අගය නැවත වෙනස්කල නොහැක

Foreign Key - ආගන්තුක යතුර

- තවත් වගුවක ප්‍රාථමික යතුර සමඟ සම්බන්ධතාවක් ගොඩනැගීමට භාවිතා කරයි.
- ආගන්තුක යතුර අනන්‍ය විය යුතු නැත (ප්‍රාථමික යතුර අනන්‍ය විය යුතුය.)
- ප්‍රාථමික යතුරට වඩා වෙනස් නමක් තිබිය හැක.
- ආගන්තුක යතුරට NULL අගයක් තිබිය හැක.

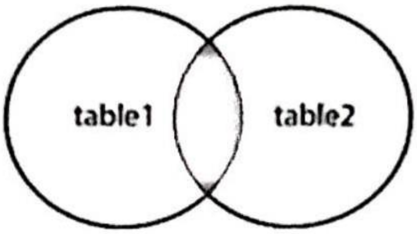
* **Data type** එක සමාන විය යුතුය

* ආගන්තුක යතුරක් තවත් වගුවක **Unique** උපලැකියට යොමුවක් විය යුතුය.

INNER JOIN

මෙහිදී වගු දෙකෙහි ම ගැලපෙන පොදු උපලැකි අගයන් පමණක් පෙන්වයි.

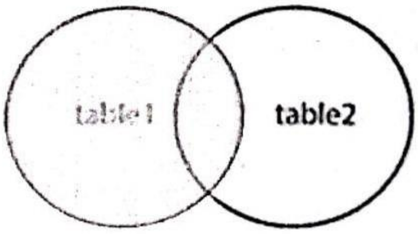
INNER JOIN



LEFT JOIN

මෙහිදී වම්පස වගුවේ සියලු උපලැකි හා දකුණුපස වගුවේ උපලැකි ඊට ගැලපෙන උපලැකි පමණක් පෙන්වයි. (දකුණුපස වගුවේ ගැලපෙන උපලැකි නැතිනම් NULL ලෙස පෙන්වයි.)

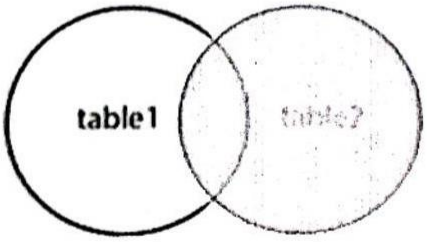
LEFT JOIN



RIGHT JOIN

මෙහිදී දකුණුපස වගුවේ සියලු උපලැකි හා වම්පස වගුවේ ඊට ගැලපෙන උපලැකි පමණක් පෙන්වයි. (වම්පස වගුවේ ගැලපෙන උපලැකි නැතිනම් NULL ලෙස පෙන්වයි.)

RIGHT JOIN



```
Microsoft Windows [Version 10.0.17763.168]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\KANISHKA VIRAJ>cd..

C:\Users>cd..

C:\>cd MAMP\bin\mysql\bin

C:\MAMP\bin\mysql\bin>mysql.exe -u root -p
Enter password: root

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.24 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

```
/* Creating a Database*/
CREATE DATABASE ict;

        mysql> create database ict;
        Query OK, 1 row affected (0.00 sec)

/* Deleting a Database*/
DROP DATABASE ict;

        mysql> drop database ict;
        Query OK, 0 rows affected (0.00 sec)
```

/*View all Existing Databases*/
SHOW DATABASES;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

/*Selecting a Database to Use*/
USE ict;

```
mysql> use ict;
Database changed
```

/*Show Currently Using Database*/
SELECT DATABASE();

```
mysql> select database();
+------------+
| database() |
+------------+
| ict        |
+------------+
1 row in set (0.00 sec)
```

/*Show Data Tables Inside Selected Database*/
SHOW TABLES;

```
mysql> show tables;
+---------------+
| Tables_in_ict |
+---------------+
| student       |
+---------------+
1 row in set (0.00 sec)
```

/*Creating a New Table*/

/* column_name data_type (size) CONSTRAINS */

```
CREATE TABLE student (
        reg_no smallint (5) PRIMARY KEY,
        name varchar (255) NOT NULL ,
        grade  tinyint (3) DEFAULT '11',
        city varchar (255),
        subject varchar (255),
        mark tinyint (5) NOT NULL
        );
```

/* View Description of a Table*/

DESC student;

```
mysql> desc student;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| reg_no  | smallint(5)  | NO   | PRI | NULL    |       |
| name    | varchar(255) | NO   |     | NULL    |       |
| grade   | tinyint(3)   | YES  |     | 11      |       |
| city    | varchar(255) | YES  |     | NULL    |       |
| subject | varchar(255) | YES  |     | NULL    |       |
| mark    | tinyint(5)   | NO   |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

/* Delete table from Database */

DROP TABLE student;

/* Adding a New Column to a Table */

ALTER TABLE student ADD COLUMN age tinyint (5) NOT NULL;

```
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| reg_no  | smallint(5)  | NO   | PRI | NULL    |       |
| name    | varchar(255) | NO   |     | NULL    |       |
| grade   | tinyint(3)   | YES  |     | 11      |       |
| city    | varchar(255) | YES  |     | NULL    |       |
| subject | varchar(255) | YES  |     | NULL    |       |
| mark    | tinyint(5)   | NO   |     | NULL    |       |
| age     | tinyint(5)   | NO   |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

```
/* Deleteing a Column From Table */
ALTER TABLE student DROP COLUMN age;

/* Method 1 : adding data to a table */
INSERT INTO student VALUES ('1000' , 'Sugath' , '13' , 'Ampara' , 'ICT' , '80');

/* View all the data in a table*/
SELECT * FROM student;
```

```
mysql> select * from student;
+--------+--------+-------+--------+---------+------+
| reg_no | name   | grade | city   | subject | mark |
+--------+--------+-------+--------+---------+------+
|   1000 | Sugath |    13 | Ampara | ICT     |   80 |
+--------+--------+-------+--------+---------+------+
1 row in set (0.00 sec)
```

```
INSERT INTO student VALUES ('1001','Kamal','12','Kandy','Sinhala','85');

/* Method 2 : adding data to a table */
INSERT INTO student (reg_no,name,grade,city,subject,mark) VALUES
                    ( '1002' , 'Supuni' , '13' , 'Kurunegala' , 'English' , '75' );

INSERT INTO student VALUES ('1003', 'Ganga', '12', 'Kandy', 'Physics', '88');
INSERT INTO student VALUES ('1004', 'Robinson', '12', 'Jaffna',  'Maths', '90');
INSERT INTO student VALUES ('1005', 'Kostha', '13', 'Hikkaduwa', 'Music', '60');
INSERT INTO student VALUES ('1006', 'Priyankara', '13', 'Galle', 'ICT', '90');
INSERT INTO student VALUES ('1007', 'Sujee', '12', 'Colombo', 'Maths', '75');


SELECT * FROM student;
```

```
mysql> SELECT * FROM student;
+--------+------------+-------+------------+---------+------+
| reg_no | name       | grade | city       | subject | mark |
+--------+------------+-------+------------+---------+------+
|   1000 | Sugath     |    13 | Ampara     | ICT     |   80 |
|   1001 | Kamal      |    12 | Kandy      | Sinhala |   85 |
|   1002 | Supuni     |    13 | Kurunegala | English |   75 |
|   1003 | Ganga      |    12 | Kandy      | Physics |   88 |
|   1004 | Robinson   |    12 | Jaffna     | Maths   |   90 |
|   1005 | Kostha     |    13 | Hikkaduwa  | Music   |   60 |
|   1006 | Priyankara |    13 | Galle      | ICT     |   90 |
|   1007 | Sujee      |    12 | Colombo    | Maths   |   75 |
+--------+------------+-------+------------+---------+------+
8 rows in set (0.00 sec)
```

SELECT reg_no, name FROM student;

```
mysql> SELECT reg_no, name FROM student;
+--------+------------+
| reg_no | name       |
+--------+------------+
|   1000 | Sugath     |
|   1001 | Kamal      |
|   1002 | Supuni     |
|   1003 | Ganga      |
|   1004 | Robinson   |
|   1005 | Kostha     |
|   1006 | Priyankara |
|   1007 | Sujee      |
+--------+------------+
8 rows in set (0.00 sec)
```

SELECT * FROM student WHERE reg_no = '1002';

```
mysql> SELECT * FROM student WHERE reg_no = '1002';
+--------+--------+-------+------------+---------+------+
| reg_no | name   | grade | city       | subject | mark |
+--------+--------+-------+------------+---------+------+
|   1002 | Supuni |    13 | Kurunegala | English |   75 |
+--------+--------+-------+------------+---------+------+
1 row in set (0.06 sec)
```

SELECT * FROM student WHERE mark='90';

```
mysql> SELECT * FROM student WHERE mark='90';
+--------+------------+-------+--------+---------+------+
| reg_no | name       | grade | city   | subject | mark |
+--------+------------+-------+--------+---------+------+
|   1004 | Robinson   |    12 | Jaffna | Maths   |   90 |
|   1006 | Priyankara |    13 | Galle  | ICT     |   90 |
+--------+------------+-------+--------+---------+------+
2 rows in set (0.03 sec)
```

SELECT name,subject FROM student WHERE mark = '90';

```
mysql> SELECT name,subject FROM student WHERE mark = '90';
+------------+---------+
| name       | subject |
+------------+---------+
| Robinson   | Maths   |
| Priyankara | ICT     |
+------------+---------+
2 rows in set (0.00 sec)
```

SELECT name,subject FROM student WHERE mark <> '90';

```
mysql> SELECT name,subject FROM student WHERE mark <> '90';
+--------+---------+
| name   | subject |
+--------+---------+
| Sugath | ICT     |
| Kamal  | Sinhala |
| Supuni | English |
| Ganga  | Physics |
| Kostha | Music   |
| Sujee  | Maths   |
+--------+---------+
6 rows in set (0.00 sec)
```

SELECT * FROM student WHERE mark >'60';

```
mysql> SELECT * FROM student WHERE mark >'60';
+--------+------------+-------+------------+---------+------+
| reg_no | name       | grade | city       | subject | mark |
+--------+------------+-------+------------+---------+------+
|   1000 | Sugath     |    13 | Ampara     | ICT     |   80 |
|   1001 | Kamal      |    12 | Kandy      | Sinhala |   85 |
|   1002 | Supuni     |    13 | Kurunegala | English |   75 |
|   1003 | Ganga      |    12 | Kandy      | Physics |   88 |
|   1004 | Robinson   |    12 | Jaffna     | Maths   |   90 |
|   1006 | Priyankara |    13 | Galle      | ICT     |   90 |
|   1007 | Sujee      |    12 | Colombo    | Maths   |   75 |
+--------+------------+-------+------------+---------+------+
7 rows in set (0.00 sec)
```

SELECT * FROM student WHERE name LIKE 'k%';

```
mysql> SELECT * FROM student WHERE name LIKE 'k%';
+--------+--------+-------+-----------+---------+------+
| reg_no | name   | grade | city      | subject | mark |
+--------+--------+-------+-----------+---------+------+
|   1001 | Kamal  |    12 | Kandy     | Sinhala |   85 |
|   1005 | Kostha |    13 | Hikkaduwa | Music   |   60 |
+--------+--------+-------+-----------+---------+------+
2 rows in set (0.07 sec)
```

SELECT * FROM student WHERE name LIKE '%a';

```
mysql> SELECT * FROM student WHERE name LIKE '%a';
+--------+------------+-------+-----------+---------+------+
| reg_no | name       | grade | city      | subject | mark |
+--------+------------+-------+-----------+---------+------+
|   1003 | Ganga      |    12 | Kandy     | Physics |   88 |
|   1005 | Kostha     |    13 | Hikkaduwa | Music   |   60 |
|   1006 | Priyankara |    13 | Galle     | ICT     |   90 |
+--------+------------+-------+-----------+---------+------+
3 rows in set (0.00 sec)
```

SELECT * FROM student WHERE mark > '80' AND subject = 'Maths';

```
mysql> SELECT * FROM student WHERE mark > '80' AND subject = 'Maths';
+--------+----------+-------+--------+---------+------+
| reg_no | name     | grade | city   | subject | mark |
+--------+----------+-------+--------+---------+------+
|   1004 | Robinson |    12 | Jaffna | Maths   |   90 |
+--------+----------+-------+--------+---------+------+
1 row in set (0.00 sec)
```

SELECT * FROM student WHERE mark > '80' OR subject = 'Maths' ;

```
mysql> SELECT * FROM student WHERE mark > '80' OR subject = 'Maths';
+--------+------------+-------+---------+---------+------+
| reg_no | name       | grade | city    | subject | mark |
+--------+------------+-------+---------+---------+------+
|   1001 | Kamal      |    12 | Kandy   | Sinhala |   85 |
|   1003 | Ganga      |    12 | Kandy   | Physics |   88 |
|   1004 | Robinson   |    12 | Jaffna  | Maths   |   90 |
|   1006 | Priyankara |    13 | Galle   | ICT     |   90 |
|   1007 | Sujee      |    12 | Colombo | Maths   |   75 |
+--------+------------+-------+---------+---------+------+
5 rows in set (0.00 sec)
```

SELECT * FROM student WHERE mark > '75' AND city = 'Kandy' AND subject='Sinhala';

```
mysql> SELECT * FROM student WHERE mark > '75' AND city = 'Kandy' AND subject='Sinhala';
+--------+-------+-------+-------+---------+------+
| reg_no | name  | grade | city  | subject | mark |
+--------+-------+-------+-------+---------+------+
|   1001 | Kamal |    12 | Kandy | Sinhala |   85 |
+--------+-------+-------+-------+---------+------+
1 row in set (0.00 sec)
```

SELECT * FROM student ORDER BY mark ASC;

```
mysql> SELECT * FROM student ORDER BY mark ASC;
+--------+------------+-------+------------+---------+------+
| reg_no | name       | grade | city       | subject | mark |
+--------+------------+-------+------------+---------+------+
|   1005 | Kostha     |    13 | Hikkaduwa  | Music   |   60 |
|   1002 | Supuni     |    13 | Kurunegala | English |   75 |
|   1007 | Sujee      |    12 | Colombo    | Maths   |   75 |
|   1000 | Sugath     |    13 | Ampara     | ICT     |   80 |
|   1001 | Kamal      |    12 | Kandy      | Sinhala |   85 |
|   1003 | Ganga      |    12 | Kandy      | Physics |   88 |
|   1004 | Robinson   |    12 | Jaffna     | Maths   |   90 |
|   1006 | Priyankara |    13 | Galle      | ICT     |   90 |
+--------+------------+-------+------------+---------+------+
8 rows in set (0.00 sec)
```

SELECT * FROM student ORDER BY name DESC;

```
mysql> SELECT * FROM student ORDER BY name DESC;
+--------+------------+-------+------------+---------+------+
| reg_no | name       | grade | city       | subject | mark |
+--------+------------+-------+------------+---------+------+
|   1002 | Supuni     |    13 | Kurunegala | English |   75 |
|   1007 | Sujee      |    12 | Colombo    | Maths   |   75 |
|   1000 | Sugath     |    13 | Ampara     | ICT     |   80 |
|   1004 | Robinson   |    12 | Jaffna     | Maths   |   90 |
|   1006 | Priyankara |    13 | Galle      | ICT     |   90 |
|   1005 | Kostha     |    13 | Hikkaduwa  | Music   |   60 |
|   1001 | Kamal      |    12 | Kandy      | Sinhala |   85 |
|   1003 | Ganga      |    12 | Kandy      | Physics |   88 |
+--------+------------+-------+------------+---------+------+
8 rows in set (0.00 sec)
```

UPDATE student SET mark=90 WHERE reg_no ='1000';

```
mysql> UPDATE student SET mark=90 WHERE reg_no ='1000';
Query OK, 1 row affected (0.10 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM student;
+--------+------------+-------+------------+---------+------+
| reg_no | name       | grade | city       | subject | mark |
+--------+------------+-------+------------+---------+------+
|   1000 | Sugath     |    13 | Ampara     | ICT     |   90 |
|   1001 | Kamal      |    12 | Kandy      | Sinhala |   85 |
|   1002 | Supuni     |    13 | Kurunegala | English |   75 |
|   1003 | Ganga      |    12 | Kandy      | Physics |   88 |
|   1004 | Robinson   |    12 | Jaffna     | Maths   |   90 |
|   1005 | Kostha     |    13 | Hikkaduwa  | Music   |   60 |
|   1006 | Priyankara |    13 | Galle      | ICT     |   90 |
|   1007 | Sujee      |    12 | Colombo    | Maths   |   75 |
+--------+------------+-------+------------+---------+------+
8 rows in set (0.00 sec)
```

UPDATE student SET city='Matara' WHERE reg_no='1006';

```
mysql> UPDATE student SET city='Matara' WHERE reg_no='1006';
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM student;
+--------+------------+-------+------------+---------+------+
| reg_no | name       | grade | city       | subject | mark |
+--------+------------+-------+------------+---------+------+
|   1000 | Sugath     |    13 | Ampara     | ICT     |   90 |
|   1001 | Kamal      |    12 | Kandy      | Sinhala |   85 |
|   1002 | Supuni     |    13 | Kurunegala | English |   75 |
|   1003 | Ganga      |    12 | Kandy      | Physics |   88 |
|   1004 | Robinson   |    12 | Jaffna     | Maths   |   90 |
|   1005 | Kostha     |    13 | Hikkaduwa  | Music   |   60 |
|   1006 | Priyankara |    13 | Matara     | ICT     |   90 |
|   1007 | Sujee      |    12 | Colombo    | Maths   |   75 |
+--------+------------+-------+------------+---------+------+
8 rows in set (0.00 sec)
```

DELETE FROM student;

```
mysql> DELETE FROM student;
Query OK, 7 rows affected (0.10 sec)

mysql> select * from student;
Empty set (0.00 sec)
```

DELETE FROM student WHERE reg_no='1000';

```
mysql> DELETE FROM student WHERE reg_no='1000';
Query OK, 1 row affected (0.15 sec)

mysql> select * from student;
+--------+------------+-------+------------+---------+------+
| reg_no | name       | grade | city       | subject | mark |
+--------+------------+-------+------------+---------+------+
|   1001 | Kamal      |    12 | Kandy      | Sinhala |   85 |
|   1002 | Supuni     |    13 | Kurunegala | English |   75 |
|   1003 | Ganga      |    12 | Kandy      | Physics |   88 |
|   1004 | Robinson   |    12 | Jaffna     | Maths   |   90 |
|   1005 | Kostha     |    13 | Hikkaduwa  | Music   |   60 |
|   1006 | Priyankara |    13 | Matara     | ICT     |   90 |
|   1007 | Sujee      |    12 | Colombo    | Maths   |   75 |
+--------+------------+-------+------------+---------+------+
7 rows in set (0.00 sec)
```

Student

reg_no	name	grade	city	subject	mark
1000	Sugath	13	Ampara	ICT	80
1001	Kamal	12	Kandy	Sinhala	85
1002	Supuni	13	Kurunegala	English	75
1003	Ganga	12	Kandy	Physics	88
1004	Robinson	12	Jaffna	Maths	90
1005	Kostha	13	Hikkaduwa	Music	60
1006	Priyankara	13	Galle	ICT	90
1007	Sujee	12	Colombo	Maths	75

Sport

S_index	Sport_name	Student_Reg_no	S_type
5000	Cricket	1004	Outdoor
5001	Netball	1002	Outdoor
5002	Chess	1004	Indoor
5003	Badminton	1007	Indoor
5004	Cricket	1001	Outdoor
5005	Badminton	1005	Indoor
5006	Cricket	1007	Outdoor

```
CREATE TABLE sport(
    s_index smallint (5),
    sport_name varchar (255),
    student_reg_no smallint (5),
    s_type varchar (10) DEFAULT 'Indoor',
    PRIMARY KEY (s_index),
    FOREIGN KEY (student_reg_no) REFERENCES student (reg_no)
);
```

```
mysql> CREATE TABLE sport(
    -> s_index smallint (5),
    -> sport_name varchar (255),
    -> student_reg_no smallint (5),
    -> s_type varchar (10) DEFAULT 'Indoor',
    -> PRIMARY KEY (s_index),
    -> FOREIGN KEY (student_reg_no) REFERENCES student (reg_no)
    -> );
Query OK, 0 rows affected (0.70 sec)
```

```
mysql> desc sport;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| s_index        | smallint(5)  | NO   | PRI | NULL    |       |
| sport_name     | varchar(255) | YES  |     | NULL    |       |
| student_reg_no | smallint(5)  | YES  | MUL | NULL    |       |
| s_type         | varchar(10)  | YES  |     | Indoor  |       |
+----------------+--------------+------+-----+---------+-------+
4 rows in set (0.04 sec)
```

```
INSERT INTO sport VALUES('5000','Cricket','1004','Outdoor');
INSERT INTO sport VALUES('5001','Netball','1002','Outdoor');
INSERT INTO sport VALUES('5002','Chess','1004','Indoor');
INSERT INTO sport VALUES('5003','Badminton','1007','Indoor');
INSERT INTO sport VALUES('5004','Cricket','1001','Outdoor');
INSERT INTO sport VALUES('5005','Badminton','1005','Indoor');
INSERT INTO sport VALUES('5006','Cricket','1007','Outdoor')
```

```
mysql> select * from sport;
+---------+------------+----------------+---------+
| s_index | sport_name | student_reg_no | s_type  |
+---------+------------+----------------+---------+
|    5000 | Cricket    |           1004 | Outdoor |
|    5001 | Netball    |           1002 | Outdoor |
|    5002 | Chess      |           1004 | Indoor  |
|    5003 | Badminton  |           1007 | Indoor  |
|    5004 | Cricket    |           1001 | Outdoor |
|    5005 | Badminton  |           1005 | Indoor  |
|    5006 | Cricket    |           1007 | Outdoor |
+---------+------------+----------------+---------+
7 rows in set (0.00 sec)
```

/* INNER JOIN */

SELECT student.name, sport.sport_name FROM student INNER JOIN sport ON student.reg_no=sport.student_reg_no;

```
mysql> SELECT student.name, sport.sport_name FROM student INNER JOIN sport ON student.reg_no=sport.student_reg_no;
+----------+------------+
| name     | sport_name |
+----------+------------+
| Robinson | Cricket    |
| Supuni   | Netball    |
| Robinson | Chess      |
| Sujee    | Badminton  |
| Kamal    | Cricket    |
| Kostha   | Badminton  |
| Sujee    | Cricket    |
+----------+------------+
7 rows in set (0.06 sec)
```

/* LEFT JOIN */

SELECT student.name, sport.sport_name FROM student LEFT JOIN sport ON student.reg_no=sport.student_reg_no;

```
mysql> SELECT student.name, sport.sport_name FROM student LEFT JOIN sport ON student.reg_no=sport.student_reg_no;
+------------+------------+
| name       | sport_name |
+------------+------------+
| Sugath     | NULL       |
| Kamal      | Cricket    |
| Supuni     | Netball    |
| Ganga      | NULL       |
| Robinson   | Cricket    |
| Robinson   | Chess      |
| Kostha     | Badminton  |
| Priyankara | NULL       |
| Sujee      | Badminton  |
| Sujee      | Cricket    |
+------------+------------+
10 rows in set (0.03 sec)
```

/* RIGHT JOIN */

SELECT student.name, sport.sport_name FROM student RIGHT JOIN sport ON student.reg_no=sport.student_reg_no;

```
mysql> SELECT student.name, sport.sport_name FROM student RIGHT JOIN sport ON student.reg_no=sport.student_reg_no;
+----------+------------+
| name     | sport_name |
+----------+------------+
| Robinson | Cricket    |
| Supuni   | Netball    |
| Robinson | Chess      |
| Sujee    | Badminton  |
| Kamal    | Cricket    |
| Kostha   | Badminton  |
| Sujee    | Cricket    |
+----------+------------+
7 rows in set (0.00 sec)
```