

- Data - Fact that can be recorded or stored
 - e.g. Person Name, Age, Gender and Weight etc.
- Database - Collection of logically related data
 - e.g. Books Database in Library, Student Database in University etc.
- Management - Manipulation, Searching and Security of data
 - e.g. Viewing result in GTU website, Searching exam papers in GTU website etc.
- System - Programs or tools used to manage database
 - e.g. SQL Server Studio Express, Oracle etc.
- DBMS - A Database Management System is a software for creating and managing databases.
- Database Management System (DBMS) is a software designed to define, manipulate, retrieve and manage data in a database.
 - e.g. MS SQL Server, Oracle, My SQL, SQLite, MongoDB etc.

Advantages of DBMS (Summary)

- Reduce data redundancy (duplication)
 - Avoids unnecessary duplication of data by storing data centrally.
- Remove data inconsistency
 - By eliminating redundancy, data inconsistency can be removed.
- Data isolation
 - A user can easily retrieve proper data as per his/her requirement.
- Guaranteed atomicity
 - Either transaction executes 0% or 100%.
- Allow implementing integrity constraints
 - Business rules can be implemented such as do not allow to store amount less than Rs. 0 in balance.
- Sharing of data among multiple users
 - More than one users can access same data at the same time.
- Restricting unauthorized access to data
 - A user can only access data which is authorized to him/her.
- Providing backup and recovery services
 - Can take a regular auto or manual backup and use it to restore the database if it corrupts.

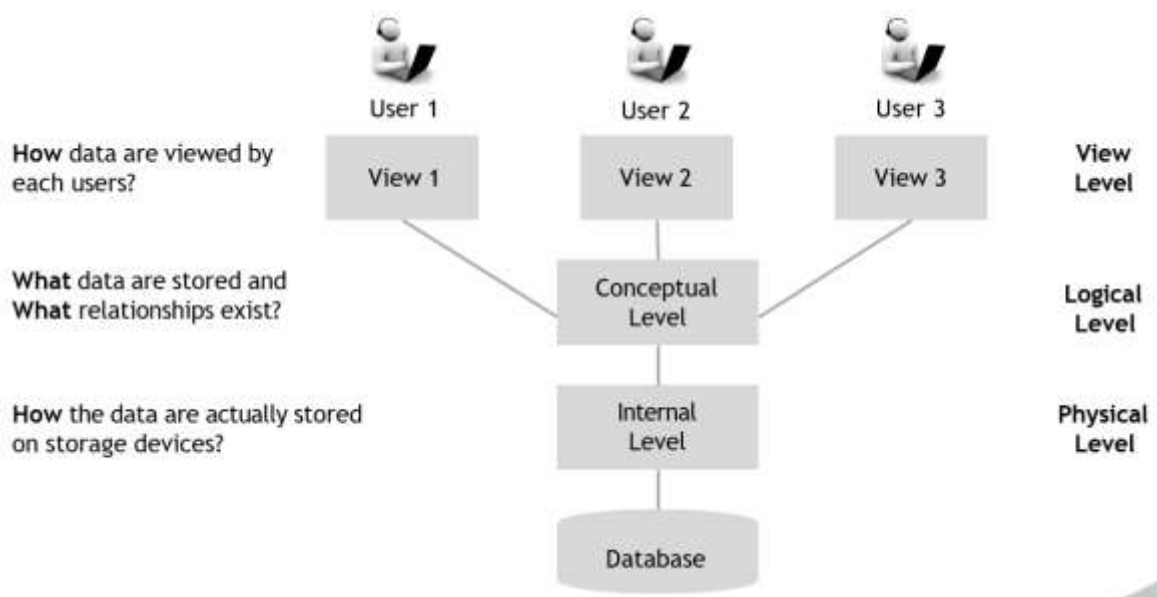
Basic terms

- Data
 - Data is raw, unorganized facts that need to be processed.
- Information
 - When data is processed, organized, structured or presented in a given context so as to make it useful, it is called information.
- Metadata
 - Metadata is data about data.
- Data dictionary
 - A data dictionary is an information repository which contains metadata.
- Data warehouse
 - A data warehouse is an information repository which stores data.
- Field
 - A field is a character or group of characters that have a specific meaning.
- Record / Tuple
 - A record is a collection of logically related fields.

- Instance
 - The data which is stored in the database at a particular moment of time is called an instance of the database.
- Schema
 - The overall design of a database is called schema.
 - The basic structure of how the data will be stored in the database is called schema.

3 Levels ANSI SPARC Database System

- Internal level (Physical level)
 - It describes how a data is stored on the storage device.
 - Deals with physical storage of data.
 - Structure of records on disk - files, pages, blocks and indexes and ordering of records
 - Internal view is described by the internal schema.
- Conceptual level (Logical level)
 - What data are stored and what relationships exist among those data?
 - It hides low level complexities of physical storage.
 - For Example, STUDENT database may contain STUDENT and COURSE tables which will be visible to users but users are unaware about their storage.
 - Database administrator works at this level to determine what data to keep in the database.
- External level (View level)
 - It describes only part of the entire database that an end user concern or how data are viewed by each user.
 - Different user needs different views of the database, so there can be many views in a view level abstraction of the database. Used by end users and application programmers.
 - End users need to access only part of the database rather than the entire database.



Data Abstraction in DBMS

- Database systems are made-up of complex data structures.
- To ease the user interaction with database, the developers hide internal irrelevant details from users.
- This process of hiding irrelevant details from user is called data abstraction.

Types of Data Independence

- Physical Data Independence
 - Physical Data Independence is the ability to modify the physical schema without requiring any change in logical (conceptual) schema and application programs.
 - Modifications at the internal levels are occasionally necessary to improve performance.
 - Possible modifications at internal levels are changes in file structures, compression techniques, hashing algorithms, storage devices, etc.
- Logical Data Independence
 - Logical data independence is the ability to modify the conceptual schema without requiring any change in application programs.
 - Modification at the logical levels is necessary whenever the logical structure of the database is changed.
 - Application programs are heavily dependent on logical structures of the data they access. So any change in logical structure also requires programs to change.

Types of Database Users

- Naive Users (End Users)
 - Unsophisticated users who have zero knowledge of database system
 - End user interacts to database via sophisticated software or tools
 - e.g. Clerk in bank
- Application Programmers
 - Programmers who write software using tools such as Java, .Net, PHP etc...
 - e.g. Software developers
- Sophisticated Users
 - Interact with database system without using an application program
 - Use query tools like SQL
 - e.g. Analyst
- Specialized Users (DBA)
 - User write specialized database applications program
 - Use administration tools
 - e.g. Database Administrator

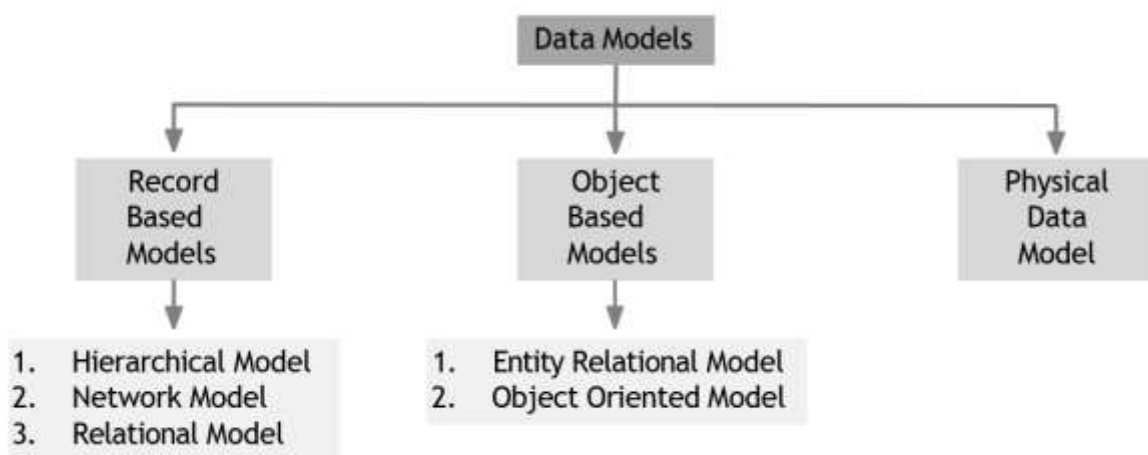
Role of DBA (Database Administrator)

- Schema Definition
 - DBA defines the logical schema of the database.
- Storage Structure and Access Method Definition
 - DBA decides how the data is to be represented in the database & how to access it.
- Defining Security and Integrity Constraints

- DBA decides on various security and integrity constraints.
- Granting of Authorization for Data Access
 - DBA determines which user needs access to which part of the database.
- Liaison with Users
 - DBA provide necessary data to the user.
- Assisting Application Programmer
 - DBA provides assistance to application programmers to develop application programs.
- Monitoring Performance
 - DBA ensures that better performance is maintained by making a change in the physical or logical schema if required.
- Backup and Recovery
 - DBA backing up the database on some storage devices such as DVD, CD or magnetic tape or remote servers and recover the system in case of failures, such as flood or virus attack from this backup.

Data Models

Data models define how the logical structure of a database is modeled.



Refer Unit1 PPT.

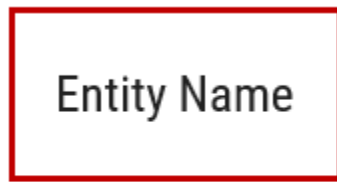
- ▶ What is Database Design?
 - ↳ Database Design is a collection of processes that facilitate the **designing, development, implementation** and **maintenance** of enterprise database management systems.
- ▶ What is E-R diagram?
 - ↳ E-R diagram: (Entity-Relationship diagram)
 - ↳ It is **graphical (pictorial) representation** of database.
 - ↳ It uses different types of symbols to represent different objects of database.

Entity

- ▶ An entity is a **person**, a **place** or an **object**.
- ▶ An entity is represented by a **rectangle** which contains the name of an entity.

Entity Set

- ▶ It is a **set (group)** of entities of same type.



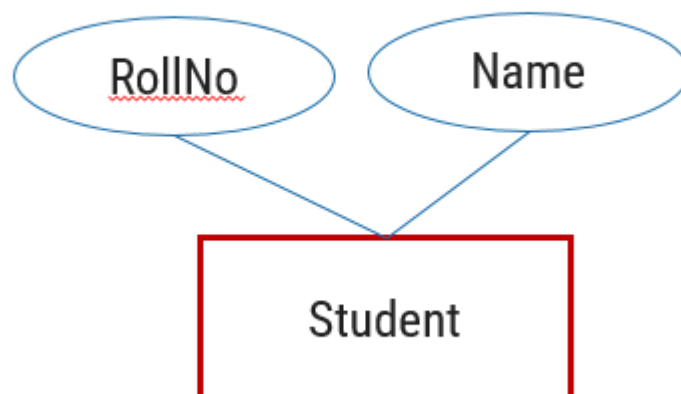
Symbol

Attributes

- ▶ Attribute is **properties** or details about an entity.
- ▶ An attribute is represented by an **oval** containing name of an attribute.

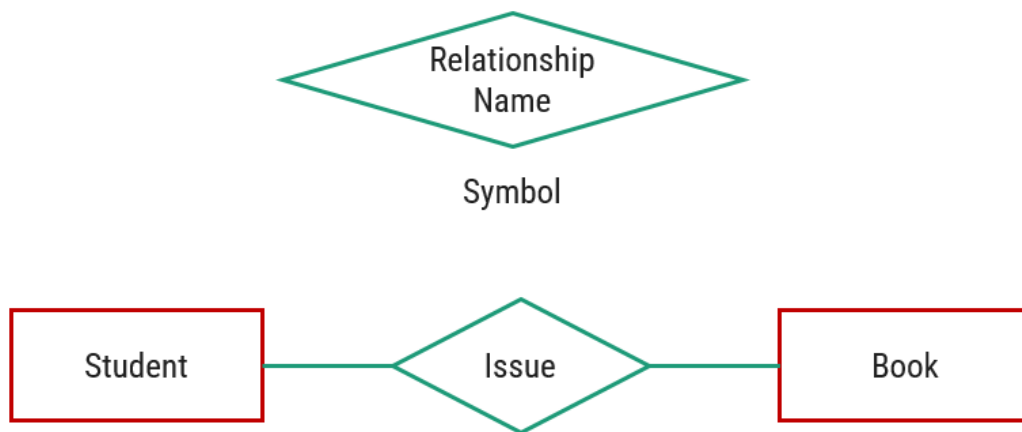


Symbol



Relationship

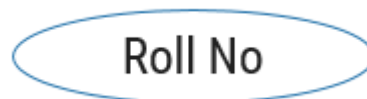
- ▶ Relationship is an **association** (connection) between several entities.
- ▶ It should be placed between two entities and a line connecting it to an entity.
- ▶ A relationship is represented by a **diamond** containing relationship's name.



Types of Attributes

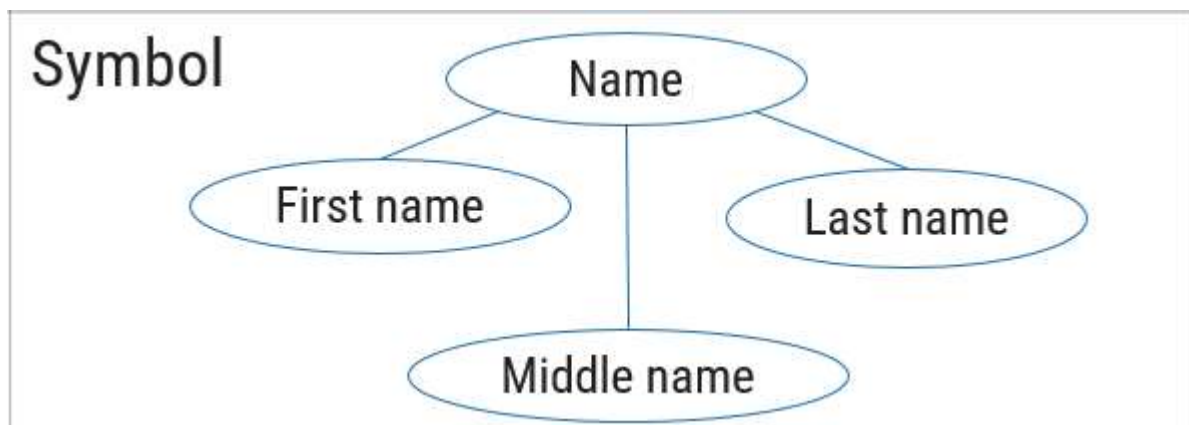
1) Simple Attribute

- Cannot be divided into subparts
- E.g. RollNo, CPI



2) Composite Attribute

- Can be divided into subparts
- E.g. Name (first name, middle name, last name)
- Address (street, road, city)



3) Single-valued Attribute

- Has single value
- E.g. RollNo, CPI

Roll No

4) Multi-valued Attribute

- Has multiple (more than one) value
- E.g. PhoneNo (person may have multiple phone nos)
- EmailID (person may have multiple emails)

Phone No

5) Stored Attribute

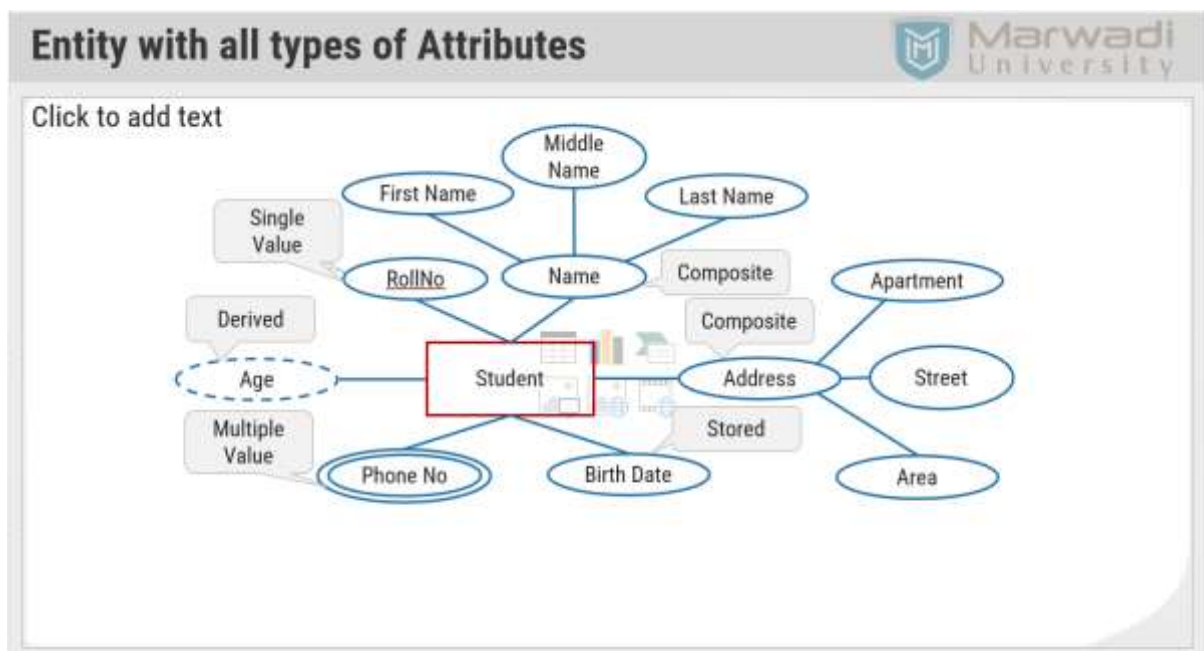
- It's value is stored manually in database
- E.g. Birthdate

Birthdate

6) Derived Attribute

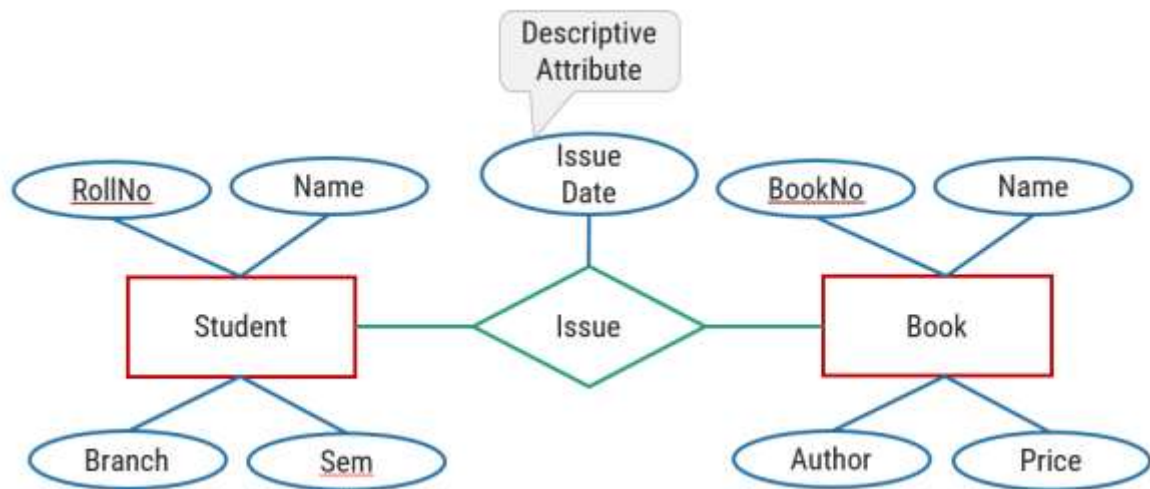
- It's value is derived or calculated from other attributes
- E.g. Age (can be calculated using current date and birthdate)

Age



Descriptive Attribute

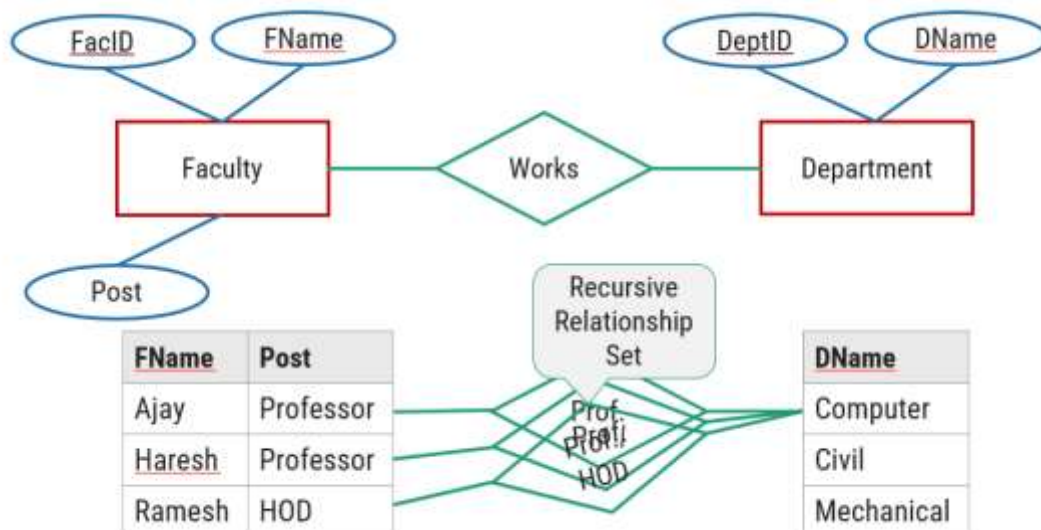
- **Attributes of the relationship** is called descriptive attribute.



Role

- Roles are indicated by labeling the lines that connect diamonds (relationship) to rectangles (entity).
- The labels "Coordinator" and "Head" are called roles; it specify the function that an entity plays in a relationship.

Recursive Relationship Set



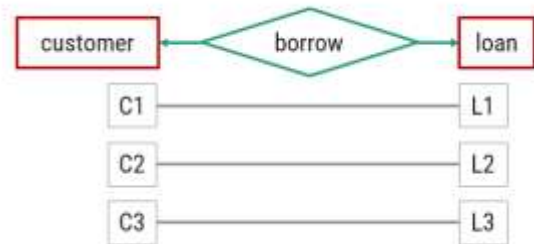
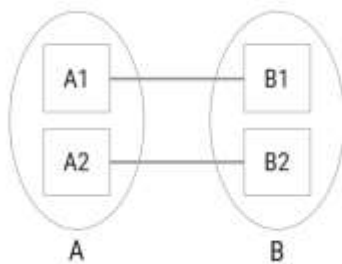
Mapping Cardinality (Cardinality Constraints)

- It represents the **number of entities of another entity set** which are **connected to an entity** using a relationship set.
- It is most **useful in describing binary relationship sets**.

- For a binary relationship set the mapping cardinality must be one of the following types:
 - ↳ One to One
 - ↳ One to Many
 - ↳ Many to One
 - ↳ Many to Many

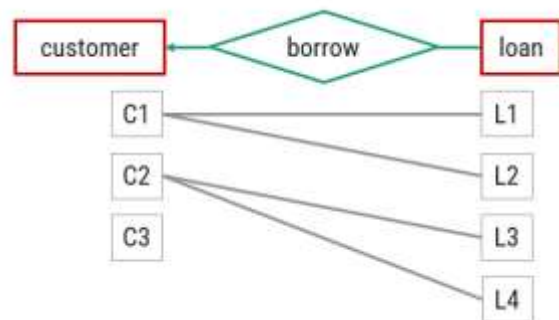
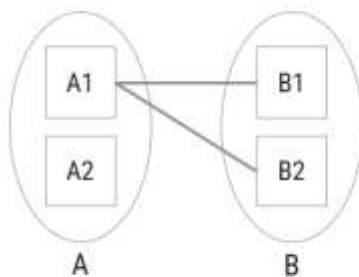
One-to-One relationship (1 – 1)

- An entity in **A** is associated with **only one entity in B** and an entity in **B** is associated with **only one entity in A**.



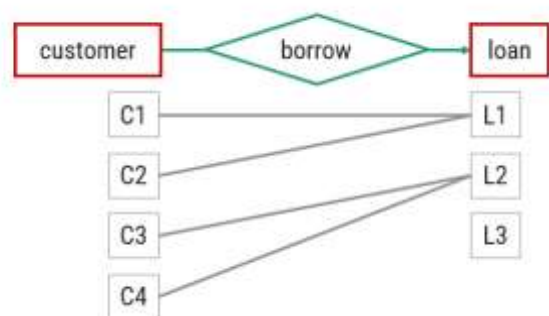
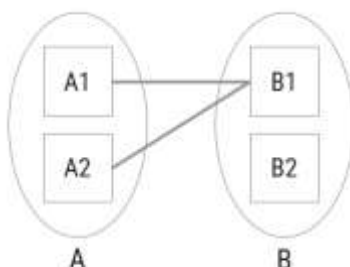
One-to-Many relationship (1 – N)

- An entity in **A** is associated with **more than one entities in B** and an entity in **B** is associated with **only one entity in A**.



Many-to-One relationship (N – 1)

- An entity in **A** is associated with **only one entity in B** and an entity in **B** is associated with **more than one entities in A**.



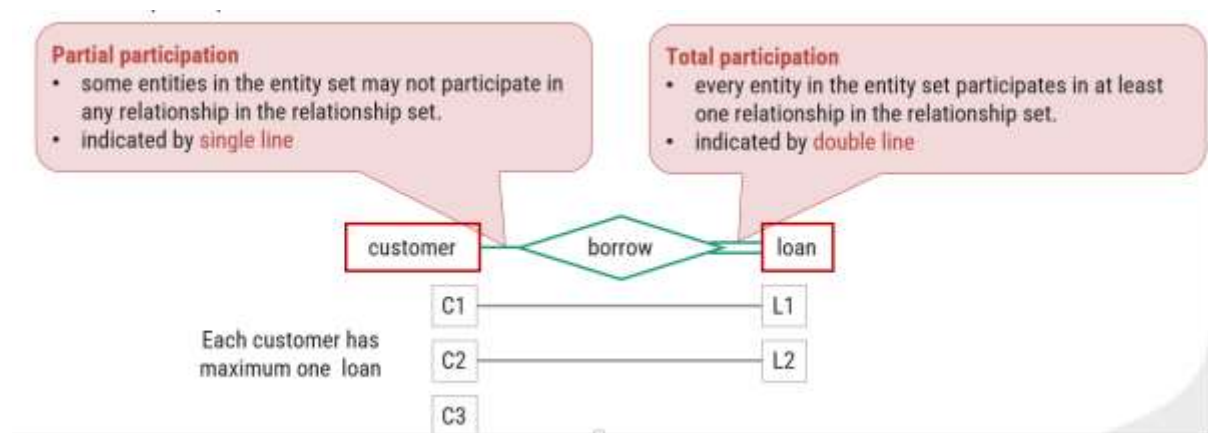
Many-to-Many relationship (N – N)

- ▶ An entity in **A** is associated with more than one entities in **B** and an entity in **B** is associated with more than one entities in **A**.



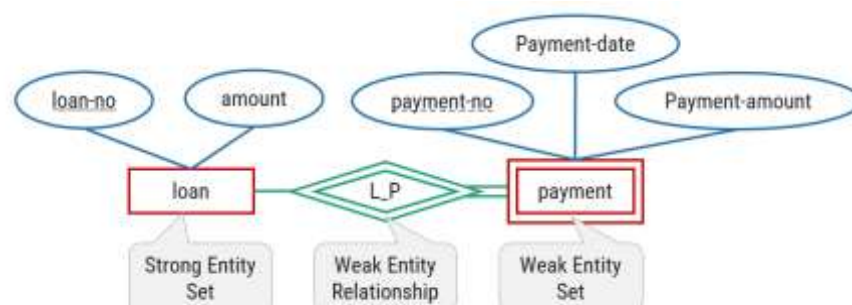
Participation Constraints

- ▶ It specifies the **participation of an entity set** in a relationship set.
- ▶ There are two types participation constraints
 - ↳ Total participation
 - ↳ Partial participation



Weak Entity Set

- ▶ An entity set that does not have a primary key is called weak entity set.



- ▶ The **existence of a weak entity set** depends on the **existence of a strong entity set**.
- ▶ The **discriminator (partial key)** of a weak entity set is the set of **attributes that distinguishes all the entities** of a weak entity set.
- ▶ The **primary key** of a weak entity set is created by **combining the primary key of the strong entity set** on which the weak entity set is existence dependent and the **weak entity set's discriminator**.
- ▶ We underline the discriminator attribute of a weak entity set with a **dashed line**.
- ▶ Payment entity has payment-no which is discriminator.
- ▶ Loan entity has loan-no as primary key.
- ▶ So primary key for payment is **(loan-no, payment-no)**.

Superclass v/s Subclass

Super Class

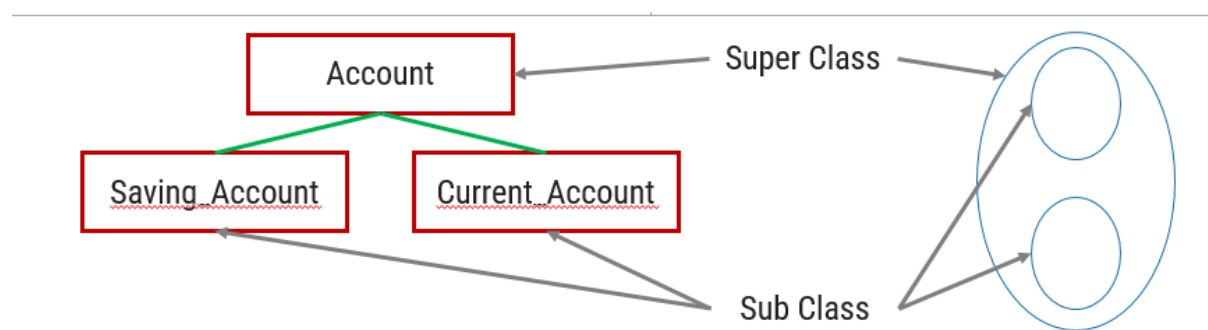
A superclass is an entity from which **another entities can be derived**.

E.g, an entity account has two subsets saving_account and current_account So an **account is superclass**.

Sub Class

A subclass is an entity that is **derived from another entity**.

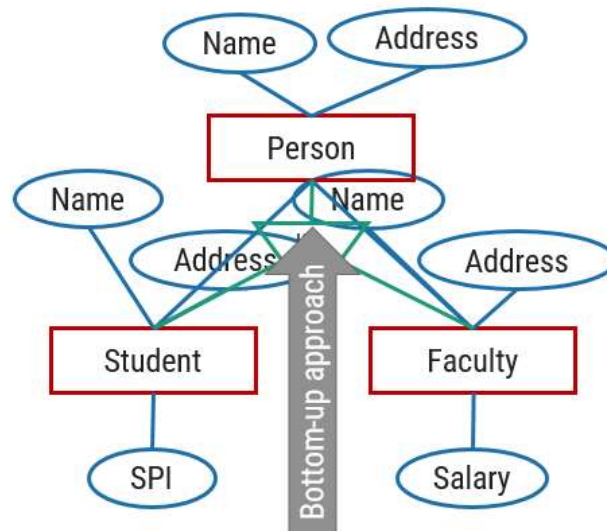
E.g, saving_account and current_account entities are derived from entity account. So **saving_account and current_account are subclass**.



Generalization v/s Specialization

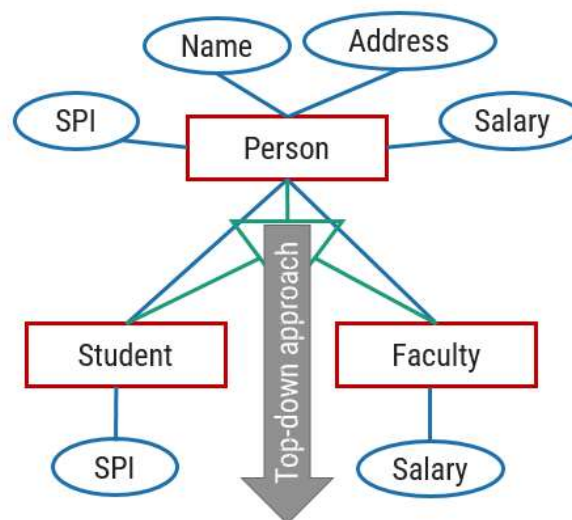
Generalization

- It **extracts the common features** of **multiple entities** to **form a new entity**.
- The process of **creation of group from various entities** is called generalization.
- It is **Bottom-up** approach.
- The process of taking the **union of two or more lower level entity** sets to produce a higher level entity set.
- It starts from the number of entity sets and creates high level entity set using some common features.



Specialization

- It **splits an entity to form multiple new entities** that **inherit some feature of the splitting entity**.
- The process of **creation of sub-groups within an entity** is called specialization.
- It is **Top-down** approach.
- The process of taking a **sub set of higher level entity set** to form a lower level entity set.
- It starts from a single entity set and creates different low level entity sets using some different features.

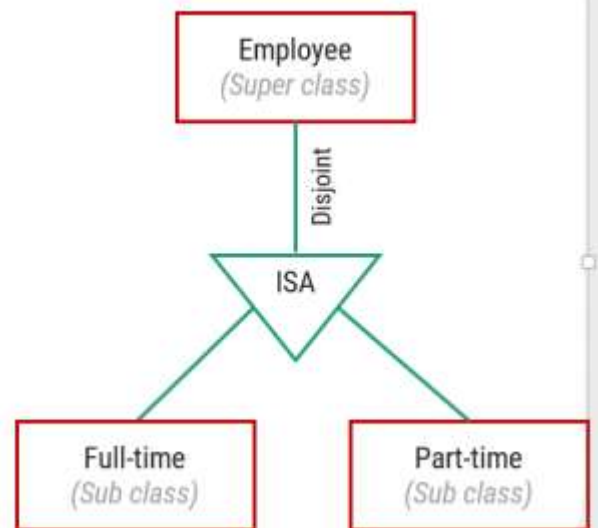
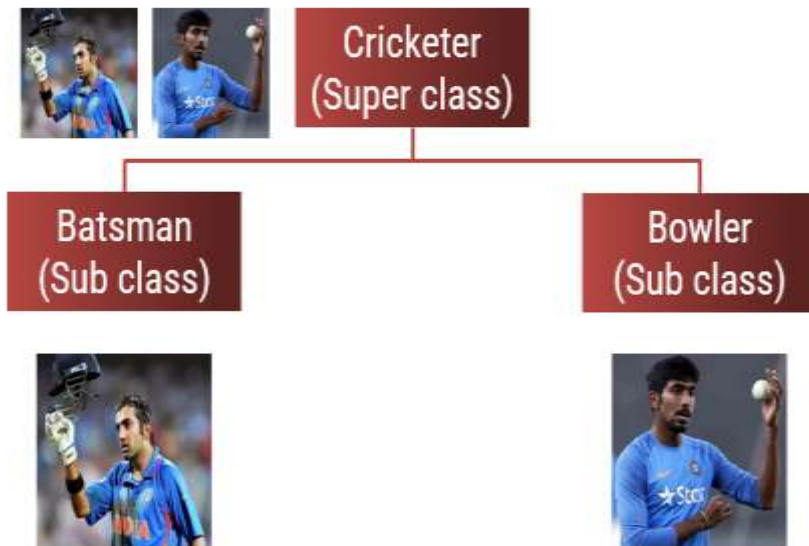


Disjoint Constraint

- ▶ It describes **relationship between members of the superclass and subclass** and indicates whether member of a superclass can be a member of one, or more than one subclass.
- ▶ Types of disjoint constraints
 - Disjoint Constraint
 - Non-disjoint (Overlapping) Constraint

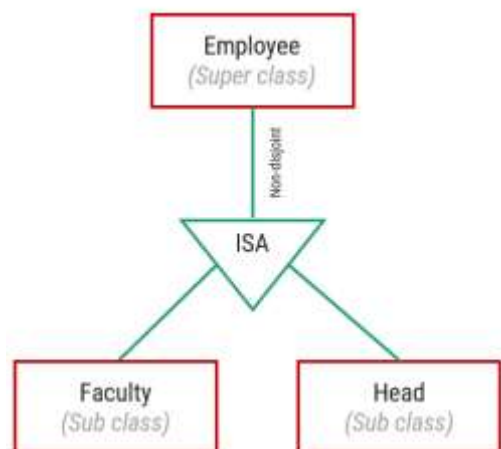
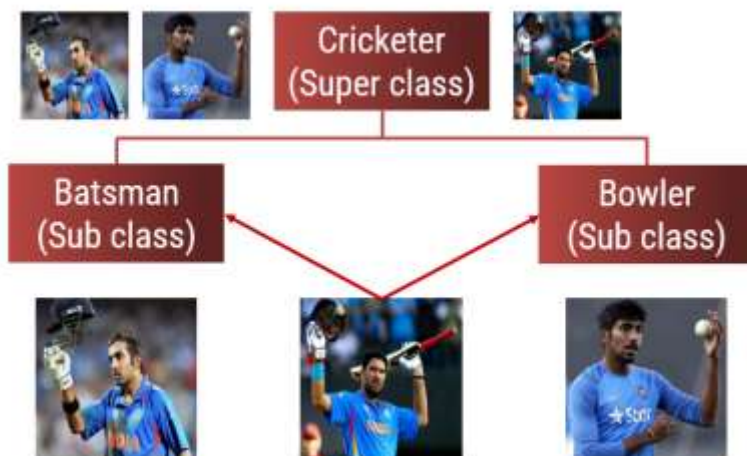
Disjoint Constraint

- ▶ It specifies that the **entity of a super class can belong to only one lower-level entity set** (sub class).
- ▶ Specified by 'd' or by writing **disjoint** near to the ISA triangle.
- ▶ All the players are associated with only one sub class either (Batsman or Bowler).



Non-disjoint (Overlapping) Constraint

- ▶ It specifies that an **entity of a super class can belong to more than one lower-level entity set** (sub class).
- ▶ Specified by 'o' or by writing **overlapping** near to the ISA triangle.



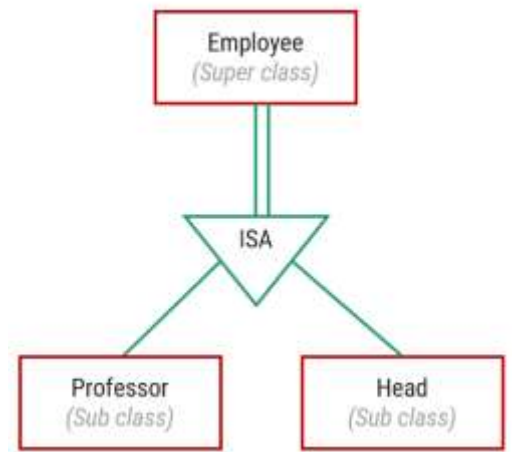
- ▶ One player (Yuvraj singh) is associated with more than one sub class.

Participation (Completeness) Constraint

- ▶ It determines **whether every member of super class must participate as a member of subclass or not.**
- ▶ Types of participation (Completeness) Constraint
 - ↳ Total (Mandatory) participation
 - ↳ Partial (Optional) participation
- ▶ All the players are associated with minimum one sub class either (Batsman or Bowler).

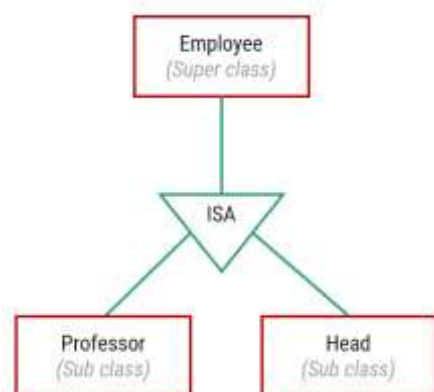
Total (Mandatory) Participation

- ▶ Total participation specifies that **every entity in the superclass must be a member of some subclass** in the specialization.
- ▶ Specified by a **double line** in E-R diagram.



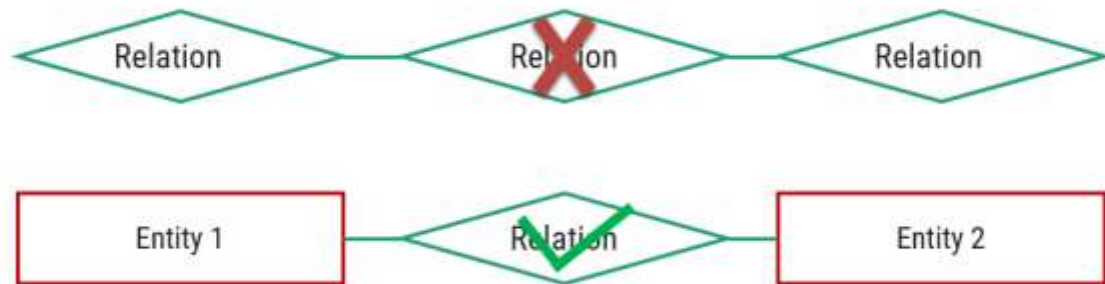
Partial (Optional) Participation

- ▶ Partial participation specifies that **every entity in the super class does not belong to any of the subclass** of specialization.
- ▶ Specified by a **single line** in E-R diagram.

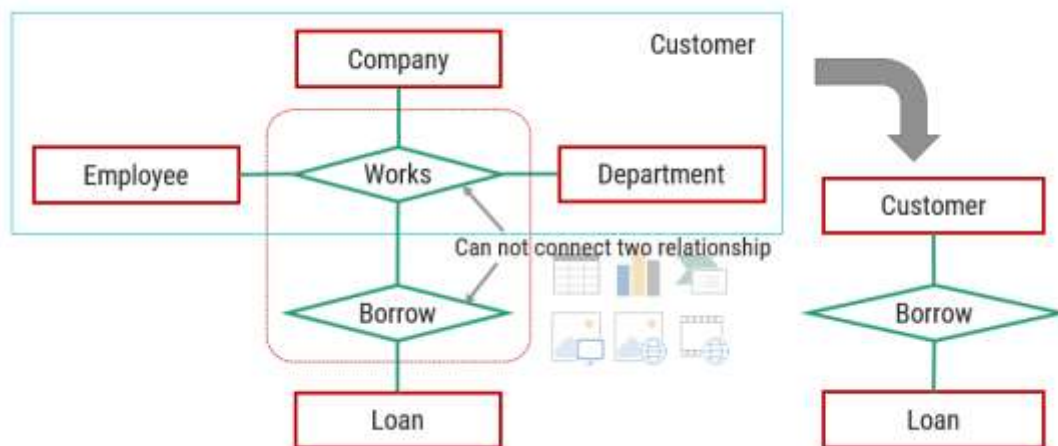


Limitation of E-R diagram

- ▶ In E-R model we **cannot express relationships between two relationships**.



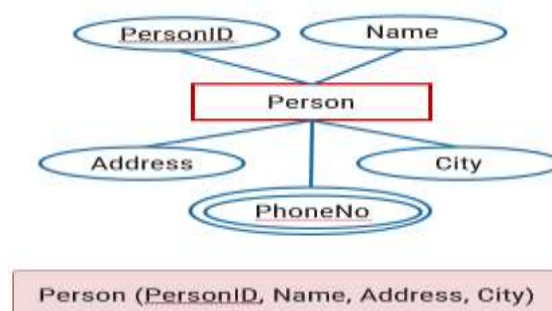
Process of creating an entity by combining various components of E-R diagram is called aggregation.



Reduce the E-R diagram to database schema

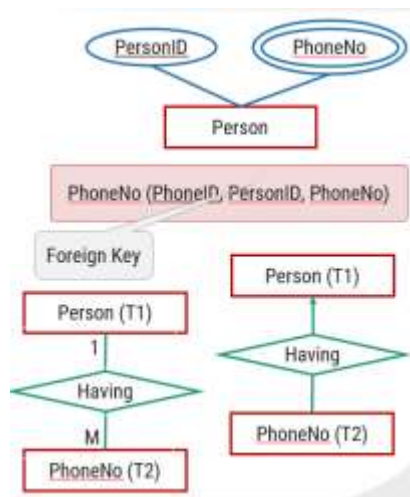
Step 1: Reduce **Entities** and **Simple Attributes**:

- An **entity** of an ER diagram is **turned into a table**.
- Each **attribute** (except multi-valued attribute) **turns into a column** (attribute) in the table.
- **Table name** can be same as **entity name**.
- **Key attribute** of the entity is the **primary key** of the table which is usually underlined.
- It is highly recommended that every table should start with its primary key attribute conventionally named as TablenameID.



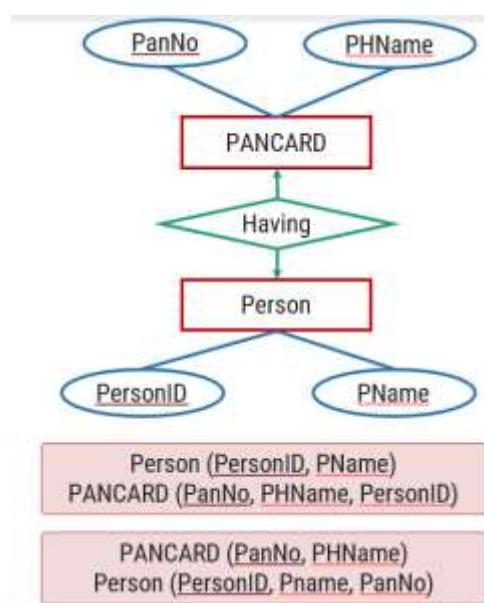
Step 2: Reduce **Multi-valued Attributes**:

- ▶ **Multi-value attribute** is turned into a **new table**.
- ▶ Add the **primary key** column into **multi-value attribute's table**.
- ▶ Add the **primary key column** of the **parent entity's table** as a **foreign key** within the **new (multi-value attribute's) table**.
- ▶ Then make a **1:N relationship** between the Person table and PhoneNo table.



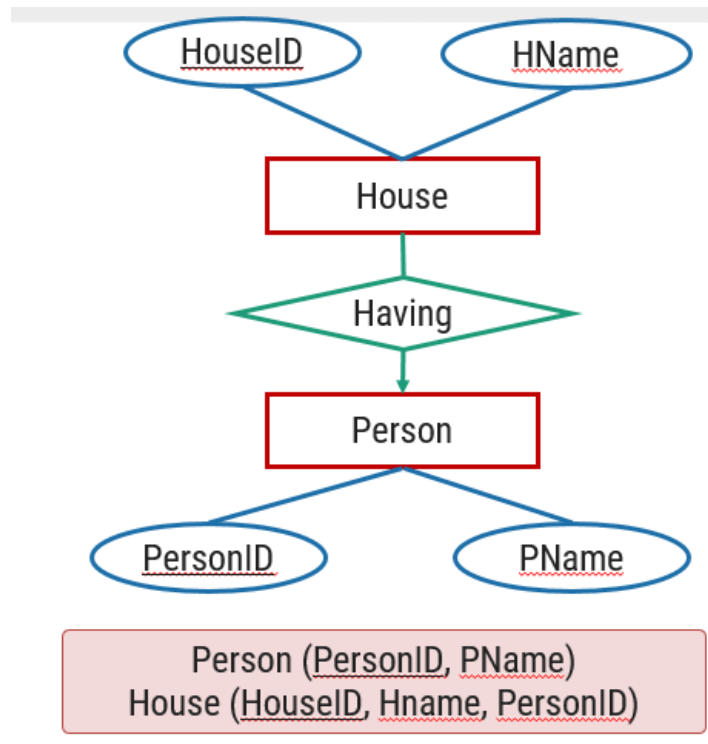
Step 3: Reduce **1:1 Mapping Cardinality**:

- ▶ Convert **both entities** in to **table** with proper attribute.
 - ▶ Place the **primary key** of any **one table** in to the **another table** as a **foreign key**.
 - ▶ Place the primary key of the PANCARD table PanNo in the table Persons as Foreign key.
- OR
- ▶ Place the primary key of the Person table PersonID in the table PANCARD as Foreign key.



Step 4: Reduce **1:N Mapping Cardinality**:

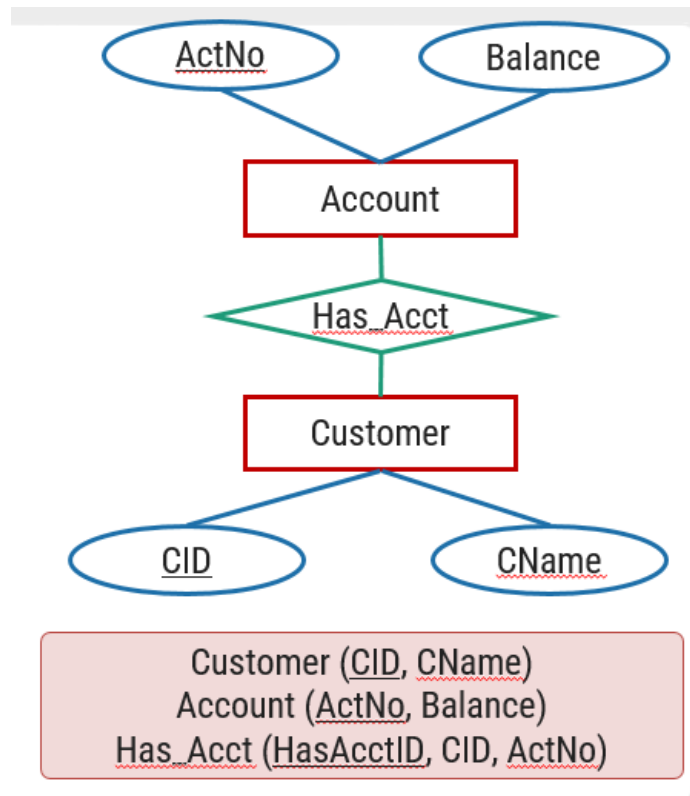
- ▶ Convert **both entities** in to **table** with proper attribute.
- ▶ Place the **primary key of table having 1 mapping** in to the another **table having many cardinality as a Foreign key**.
- ▶ Place the primary key of the Person table PersonID in the table House as Foreign key.



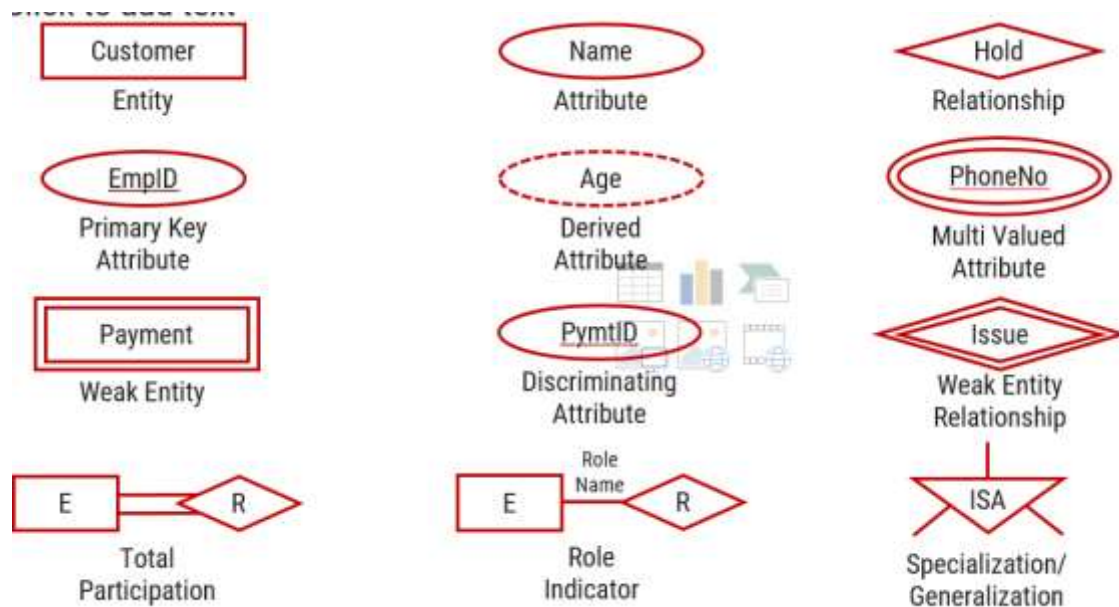
Step 5: Reduce **N:N Mapping Cardinality**:

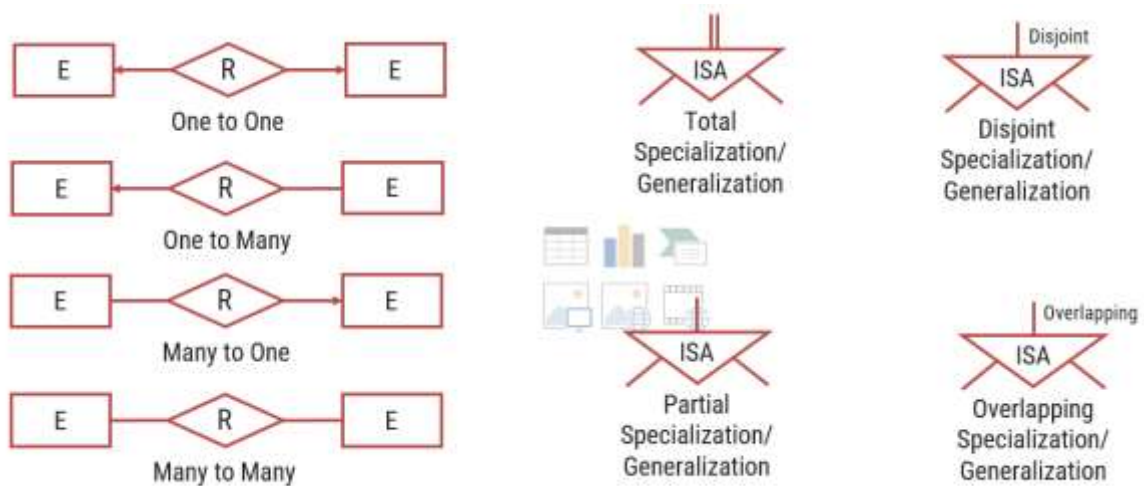
- ▶ Convert both **entities** in to **table** with proper attribute.
- ▶ Create a **separate table for relationship**.
- ▶ Place the **primary key of both entities table** into the **relationship's table as foreign key**.
- ▶ Place the primary key of the Customer table CID and Account table Ano in the table Has_Acct as Foreign key.

Summery of Symbols used in E-R diagram



Summary of Symbols used in E-R diagram





Integrity Constraints

- ▶ Integrity constraints are a **set of rules**. It is used to **maintain the quality** of information.
- ▶ Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- ▶ Thus, integrity constraint is used to **guard against accidental damage** to the database.
- ▶ Various Integrity Constraints are:
 - ↳ Check
 - ↳ Not null
 - ↳ Unique
 - ↳ Primary key
 - ↳ Foreign key
- ▶ Check
 - ↳ This constraint defines a business rule on a column. All the rows in that column must satisfy this rule.
 - ↳ Limits the data values of variables to a **specific set, range, or list of values**.
 - ↳ The constraint can be applied for a single column or a group of columns.
 - ↳ E.g. value of SPI should be between 0 to 10.
- ▶ Not null
 - ↳ This constraint ensures all rows in the table contain a definite value for the column which is specified as not null. Which means a **null value** is not allowed.
 - ↳ E.g. name column should have some value.
- ▶ Unique
 - ↳ This constraint ensures that a column or a group of columns in each row have a **distinct (unique)** value.
 - ↳ A column(s) can have a null value but the values cannot be duplicated.
 - ↳ E.g. enrollmentno column should have unique value.
- ▶ Primary key
 - ↳ This constraint defines a column or combination of columns which uniquely identifies each row in the table.
 - ↳ Primary key = **Unique key + Not null**
 - ↳ E.g. enrollmentno column should have unique value as well as can't be null.

- ▶ Foreign key (referential integrity constraint)
 - ↳ A referential integrity constraint (foreign key) is specified between two tables.
 - ↳ In the referential integrity constraints, if a foreign key column in table 1 refers to the primary key column of table 2, then every value of the foreign key column in table 1 must be null or be available in primary key column of table 2.



Table (Relation): A database object that holds a collection of data for a specific topic. Table consist of rows and columns.

Column (Attribute): The vertical component of a table. A column has a name and a particular data type; e.g. varchar, decimal, integer, datetime etc.

Record (Tuple): The horizontal component of a table, consisting of a sequence of values, one for each column of the table. It is also known as row.

A database consists of a collection of tables (relations), each having a unique name.

Domain is a set of **all possible unique values** for a specific column. Domain of Branch attribute is (CE, CI, ME, EE)

Key

- ▶ Super Key
 - ↳ A super key is a set of one or more **attributes whose values uniquely identifies each record** within a relation (table).
- ▶ Candidate Key
 - ↳ A candidate key is a **subset of a super key**.
 - ↳ A candidate key is a single attribute or the least combination of attributes that uniquely identifies each record in the table.
 - ↳ A candidate key is a **super key for which no proper subset is a super key**.
 - ↳ **Every candidate key is a super key but every super key is not a candidate key**.
- ▶ Primary Key
 - ↳ A primary key is a **candidate key that is chosen by database designer** to identify tuples uniquely in a relation (table).
 - ↳ A primary key **may have one or more attributes**.
 - ↳ There is **only one primary key** in the relation (table).
 - ↳ A primary key **attribute value cannot be NULL**.
 - ↳ Generally, the **value of a primary key attribute does not change**.
- ▶ Alternate Key
 - ↳ An alternate key is a **candidate key that is not chosen by database designer** to identify tuples uniquely in a relation.

- ▶ Foreign Key
 - ↳ A foreign key is **used to link two relations** (tables).
 - ↳ A foreign key is an **attribute** or collection of attributes in one table that **refers to the primary key in another table**.
 - ↳ A table containing the foreign key is called the child table, and the table containing the primary key is called the parent table.

Relational Algebra Operations

Operator	Description
Selection	Display particular rows/records/tuples from a relation
Projection	Display particular columns from a relation
Cross Product	Multiply each tuples of both relations
Joins	Combine data or records from two or more tables <ol style="list-style-type: none"> 1. Natural Join / Inner Join 2. Outer Join <ol style="list-style-type: none"> 1. Left Outer Join 2. Right Outer Join 3. Full Outer Join
Set Operators	Combine the results of two queries into a single result. <ol style="list-style-type: none"> 1. Union 2. Intersection 3. Minus / Set-difference
Division	Divides one relation by another
Rename	Rename a column or a table

Selection Operator

- ▶ Symbol: σ (Sigma)
- ▶ Notation: $\sigma_{condition}$ (Relation)
- ▶ Operation: **Selects tuples** from a relation that **satisfy a given condition**.
- ▶ Operators: =, <>, <, >, <=, >=, \wedge (AND), \vee (OR)

Projection Operator

- ▶ Symbol: π (Pi)
- ▶ Notation: $\pi_{attribute\ set}$ (Relation)
- ▶ Operation: **Selects specified attributes** of a relation.
- ▶ It **removes duplicate tuples** (records) from the result.

Cartesian Product / Cross Product

- ▶ Symbol: \times (Cross)
- ▶ Notation: *Relation-1 (R1) \times Relation-2 (R2)* **OR** *Algebra-1 \times Algebra-2*
- ▶ Operation: It will **multiply each tuples** of Relation-1 to each tuples of Relation-2.
 - ↳ Attributes of Resultant Relation = Attributes of R1 + Attributes of R2
 - ↳ Tuples of Resultant Relation = Tuples of R1 * Tuples of R2
- ▶ If both relations have some attribute with the same name, it can be distinguished by combining **relation-name.attribute-name**.

Example Perform Cross Product between Student and Result.

Answer (Student) X (Result)

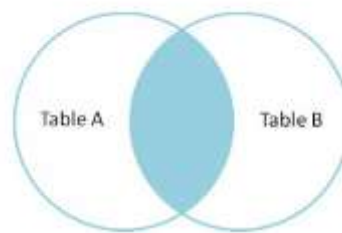
Student			Result	
RNo	Name	Branch	RNo	SPI
101	Raju	CE	101	8
102	Mitesh	ME	102	9

If both relations have some attribute with the same name, it can be distinguished by combining **relation-name.attribute-name**.

Output				
Student.RNo	Name	Branch	Result.RNo	SPI
101	Raju	CE	101	8
101	Raju	CE	102	9
102	Mitesh	ME	101	8
102	Mitesh	ME	102	9

Natural Join / Inner Join

- ▶ Symbol: \bowtie
- ▶ Notation: $Relation-1 (R1) \bowtie Relation-2 (R2)$ OR $Algebra-1 \bowtie Algebra-2$
- ▶ Operation: Natural join will **retrieve consistent data** from multiple relations.
 - ↳ It **combines records** from different relations that **satisfy a given condition**.
- ▶ To perform a Natural Join there must be **one common attribute (column)** between two relations.



Example Perform Natural Join between Student and Result.

Answer (Student) \bowtie (Result)

Student		
RNo	Name	Branch
101	Raju	CE
102	Mitesh	ME

Result	
RNo	SPI
101	8
103	9

Output			
RNo	Name	Branch	SPI
101	Raju	CE	8

Steps performed in Natural Join

Step:1 Perform Cross Product				
Student.RNo	Name	Branch	Result.RNo	SPI
101	Raju	CE	101	8
101	Raju	CE	103	9
102	Mitesh	ME	101	8
102	Mitesh	ME	103	9

Step:2 Removes inconsistent tuples				
Student.RNo	Name	Branch	Result.RNo	SPI
101	Raju	CE	101	8

Step:3 Removes an attribute from duplicate			
RNo	Name	Branch	SPI
101	Raju	CE	8

Outer Join


- ▶ In natural join some records are missing, if we want that missing records then we have to use outer join.

Three types of Outer Join

Sr.	Outer Join	Symbol
1	Left Outer Join	$\bowtie\leftarrow$
2	Right Outer Join	$\rightarrow\bowtie$
3	Full Outer Join	$\bowtie\leftarrow\rightarrow$

- ▶ To perform a Outer Join there must be **one common attribute (column)** between two relations.

Left Outer Join



▶ Symbol: $\bowtie\leftarrow$


▶ Notation: $Relation-1 (R1) \bowtie\leftarrow Relation-2 (R2)$ OR $Algebra-1 \bowtie\leftarrow Algebra-2$

▶ Operation:

- Display **all the tuples of the left relation** even through there is no matching tuple in the right relation.
- For such kind of **tuples having no matching**, the attributes of right relation will be **padded with NULL** in resultant relation.

Example Perform Left Outer Join between Student and Result.


Student			Result	
RollNo	Name	Branch	RollNo	SPI
101	Raj	CE	101	8
102	Meet	ME	103	9



Output			
RollNo	Name	Branch	SPI
101	Raj	CE	8
102	Meet	ME	NULL

Exercise What is the output of $(Result) \bowtie\leftarrow (Student)$.

Right Outer Join



▶ Symbol: $\rightarrow\bowtie$


▶ Notation: $Relation-1 (R1) \rightarrow\bowtie Relation-2 (R2)$ OR $Algebra-1 \rightarrow\bowtie Algebra-2$

▶ Operation:

- Display **all the tuples of right relation** even through there is no matching tuple in the left relation.
- For such kind of **tuples having no matching**, the attributes of left relation will be **padded with NULL** in resultant relation.

Example Perform Right Outer Join between Student and Result.

Student			Result	
RollNo	Name	Branch	RollNo	SPI
101	Raj	CE	101	8
102	Meet	ME	103	9



Output			
RollNo	Name	Branch	SPI
101	Raj	CE	8
103	NULL	NULL	9

Exercise What is the output of $(Result) \rightarrow\bowtie (Student)$.

Full Outer Join

► Symbol: \bowtie

► Notation: *Relation-1 (R1) \bowtie Relation-2 (R2)* OR *Algebra-1 \bowtie Algebra-2*

► Operation:

- Display **all the tuples of both of the relations**. It also pads null values whenever required. (Left outer join + Right outer join)
- For such kind of **tuples having no matching**, it will be **padded with NULL** in resultant relation.

Example Perform Full Outer Join between Student and Result.

Answer (Student) \bowtie (Result)

Student			Result	
RollNo	Name	Branch	RollNo	SPI
101	Raj	CE	101	8
102	Meet	ME	103	9



Output			
RollNo	Name	Branch	SPI
101	Raj	CE	8
102	Meet	ME	NULL
103	NULL	NULL	9

Exercise What is the output of (Result) \bowtie (Student).

Set Operators

- Set operators **combine the results of two or more queries** into a single result.

Three types of Set Operators

Sr.	Set Operator	Symbol
1	Union	\cup
2	Intersect / Intersection	\cap
3	Minus / Set difference	$-$

Conditions

- Set operators will take two or more queries as input, which must be union-compatible.
- Both queries should have **same (equal) number of columns**
- Corresponding **attributes should have the same data type or domain**

Conditions to perform Set Operators

Conditions-1: Both queries should have same (equal) number of columns.

Student	Faculty
RNo	Fid
101	101
102	102
103	103

Conditions-2: Corresponding attributes should have the same data type.

Student	Faculty
RNo	Fid
101	101
102	102
103	103

Union Operator

- ▶ Symbol: \cup
- ▶ Notation: $Relation-1 (R1) \cup Relation-2 (R2)$ OR $Algebra-1 \cup Algebra-2$
- ▶ Operation:
 - It displays all the tuples/records belonging to the first relation (left relation) or the second relation (right relation) or both.
 - It also **eliminates duplicate tuples** (tuples present in both relations appear once).

Example Perform Union between Customer and Employee.

Customer
Name
Raju
Suresh
Meet

Employee
Name
Meet
Suresh
Manoj

Answer $(Customer) \cup (Employee)$

Output
Name
Manoj
Meet
Raju
Suresh

Exercise Is there any difference in the output if we swap the tables in Union operator. $(Employee) \cup (Customer)$.

Intersect/ Intersection Operator

- ▶ Symbol: \cap
- ▶ Notation: $Relation-1 (R1) \cap Relation-2 (R2)$ OR $Algebra-1 \cap Algebra-2$
- ▶ Operation:
 - It displays all the tuples/records belonging to both relations. OR
 - It displays all the tuples/records which are common from both relations.

Example Perform Intersection between Customer and Employee.

Customer
Name
Raju
Suresh
Meet

Employee
Name
Meet
Suresh
Manoj

Answer $(Customer) \cap (Employee)$

Output
Name
Meet
Suresh

Exercise Is there any difference in the output if we swap the tables in Intersection. $(Employee) \cap (Customer)$.

Minus/ Set difference Operator

- ▶ Symbol: $-$
- ▶ Notation: $Relation-1 (R1) - Relation-2 (R2)$ OR $Algebra-1 - Algebra-2$
- ▶ Operation:
 - It displays all the tuples/records belonging to the first relation (left relation) but not in the second relation (right relation).

Example Perform Set difference between Customer and Employee.

Customer
Name
Raju
Suresh
Meet

Employee
Name
Meet
Suresh
Manoj

Answer $(Customer) - (Employee)$

Output
Name
Raju

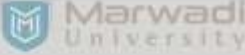
Exercise Is there any difference in the output if we swap the tables in Set difference. $(Employee) - (Customer)$.

Division Operator

- ▶ Symbol: \div (Division)
- ▶ Notation: $Relation1 (R1) \div Relation2 (R2)$ **OR** $Algebra1 \div Algebra2$
- ▶ Condition:
 - ↪ Attributes of relation2/algebra2 must be a proper subset of attributes of relation1/algebra1.
- ▶ Operation:
 - ↪ The output of the division operator will have attributes =
All attributes of relation1 – All attributes of relation2
 - ↪ The output of the division operator will have tuples =
Tuples in relation1, which are associated with the all tuples of relation2.

Division Operator Example

Example add text Division operation between Student and Subject.



Student

Name	Subject
Raj	DBMS
Raj	DS
Meet	DS
Meet	DF
Rohit	DBMS
Rohit	DS
Rohit	DF
Suresh	DBMS
Suresh	DF
Suresh	DS

Subject

Subject
DBMS
DS
DF

Answer (Student) \div (Subject)

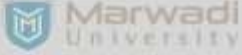
Output

Name
Rohit
Suresh

Find Out Name of all Students who enroll for all subjects.

Division Operator Example

▶ Click to add text



A

Sno	PNo
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S5	P4

B1

PNo
P2

B2

PNo
P2
P4

B3

PNo
P1
P2
P4

B4

PNo
P2
P5

Algebra (A) \div (B1)

Output

SNo
S1
S2
S3
S4

Algebra (A) \div (B2)

Output

SNo
S1
S4

Algebra (A) \div (B3)

Output

SNo
S1

Algebra (A) \div (B4)

Output

SNo

Rename Operator

- ▶ Symbol: ρ (Rho)
- ▶ Notation: $\rho_A (X_1, X_2, \dots, X_n)$ (Relation)
- ▶ Operation:
 - ↳ The rename operation is used to **rename the output relation**.
 - ↳ The result of rename operator are also relations with new name.
 - ↳ The **original relation name can not be changed** when we perform rename operation on any relation.
- ▶ How to use:
 - ↳ $\rho_x(E)$
Returns a relation E under a new name X.
 - ↳ $\rho_{A_1, A_2, \dots, A_n}(E)$
Returns a relation E with the attributes renamed to A1, A2, ..., An.
 - ↳ $\rho_{x(A_1, A_2, \dots, A_n)}(E)$
Returns a relation E under a new name X with the attributes renamed to A1, A2, ..., An.

Rename Operator Example

Example Rename table

Student		
RNo	Name	CPI
101	Raj	8
102	Meet	9
103	Jay	7

Algebra $\rho_{Person}(Student)$

Person		
RNo	Name	CPI
101	Raj	8
102	Meet	9
103	Jay	7

Example Rename attributes

Student		
Rno	Name	CPI
101	Raj	8
102	Meet	9
103	Jay	7

Algebra $\rho_{(RollNo, StudentName, SPI)}(Student)$

Student		
RollNo	StudentName	SPI
101	Raj	8
102	Meet	9
103	Jay	7

Rename Operator Example

Example Rename table and attributes both

Student		
Rno	Name	CPI
101	Raj	8
102	Meet	9
103	Jay	7

Algebra $\rho_{Person(RollNo, StudentName)}(\pi_{RNo, Name}(Student))$

Person	
RollNo	StudentName
101	Raj
102	Meet
103	Jay

Example Rename particular attributes

Student		
Rno	Name	CPI
101	Raj	8
102	Meet	9
103	Jay	7


Algebra $\rho_{StudentName / Name}(Student)$

Student		
Rno	StudentName	CPI
101	Raj	8
102	Meet	9
103	Jay	7

Aggregate Functions

- ▶ Symbol: g or G
- ▶ Notation: $g_{\text{function-name(column), function-name(column), ..., function-name(column)}}(\text{Relation})$
- ▶ Operation:
 - ↳ It takes a more than one value as input and returns a single value as output (result).
- ▶ Aggregate functions are:
 - ↳ Sum (It returns the sum (addition) of the values of a column.)
 - ↳ Max (It returns the maximum value for a column.)
 - ↳ Min (It returns the minimum value for a column.)
 - ↳ Avg (It returns the average of the values for a column.)
 - ↳ Count (It returns total number of values in a given column.)

Aggregate Functions Example



Rno	Name	Branch	Semester	CPI
101	Ramesh	CE	3	9
102	Mahesh	EC	3	8
103	Suresh	ME	4	7
104	Amit	EE	4	8
105	Anita	CE	4	8
106	Reeta	ME	3	7
107	Rohit	EE	4	9
108	Chetan	CE	3	8
109	Rakesh	CE	4	9

Example Find out sum of CPI of all students.

Answer $g_{\text{sum(CPI)}}(\text{Student})$

Output

sum
73

Example Find out maximum & minimum CPI.

Answer $g_{\text{max(CPI), min(CPI)}}(\text{Student})$

Output

max	min
9	7

Example Count the number of students.

Answer $g_{\text{count(Rno)}}(\text{Student})$

Output

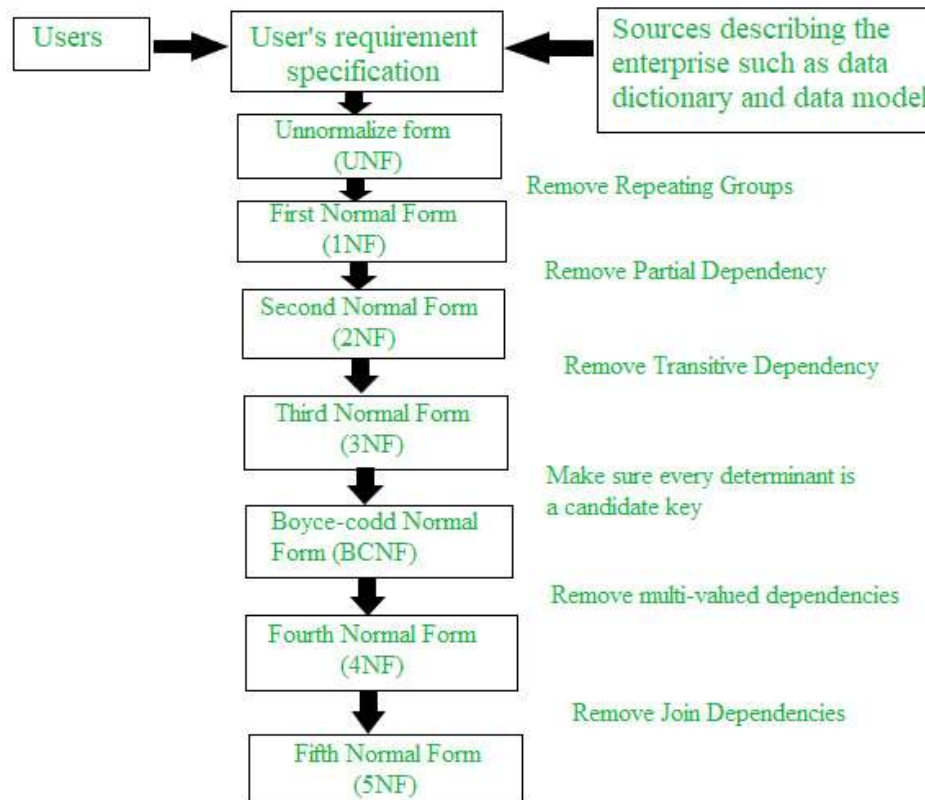
count
9

Example Find out average of CPI of all students.

Answer $g_{\text{avg(CPI)}}(\text{Student})$

Output

avg
8.11



Armstrong's axioms OR Inference rules

- Armstrong's axioms are a set of rules used to infer (derive) all the functional dependencies on a relational database.

Reflexivity

- If B is a subset of A
- then $A \rightarrow B$

Augmentation

- If $A \rightarrow B$
- then $AC \rightarrow BC$

Self-determination

- If $A \rightarrow A$

Transitivity

- If $A \rightarrow B$ and $B \rightarrow C$
- then $A \rightarrow C$

Pseudo Transitivity

- If $A \rightarrow B$ and $BD \rightarrow C$
- then $AD \rightarrow C$

Decomposition

- If $A \rightarrow BC$
- then $A \rightarrow B$ & $A \rightarrow C$

Union

- If $A \rightarrow B$ and $A \rightarrow C$
- then $A \rightarrow BC$

Composition

- If $A \rightarrow B$ and $C \rightarrow D$
- then $AC \rightarrow BD$