

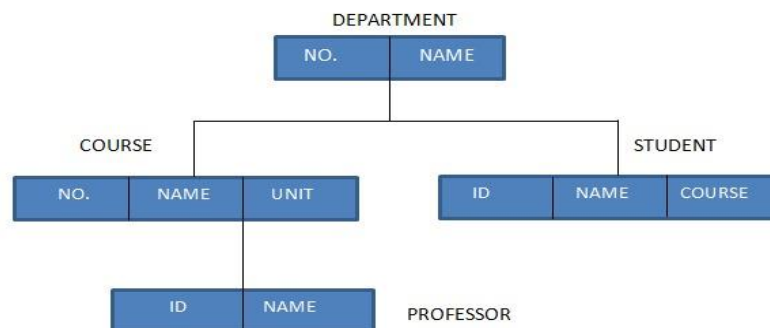
CHAPTER: 2 RELATIONAL DATA MODEL & SECURITY & INTEGRITY SPECIFICATION

2.1 Data Models

- A data model is a collection of concepts that can be used to describe the structure of a database.
- A data model is underlying structure of the database.
- This is the way which describes the design of database at physical, logical & view level.
- **Types of Data Models**
 - Hierarchical Model
 - Network Model
 - Relational Model
 - E-R model

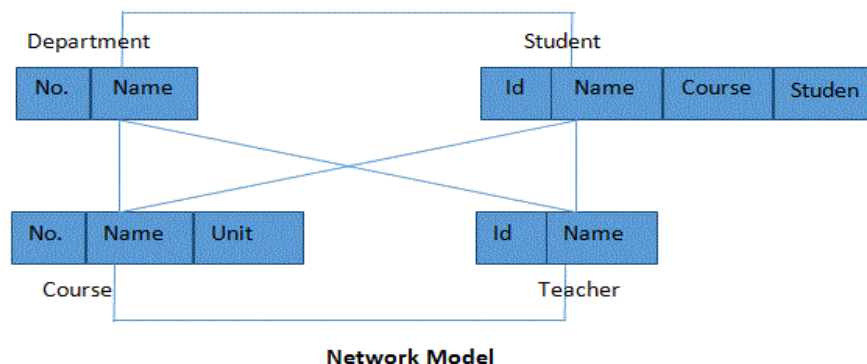
1. Hierarchical Model

- It is a tree like structure with one to many relationships.
- The structure is based on the rule that one parent can have many children but children are allowed only one parent.



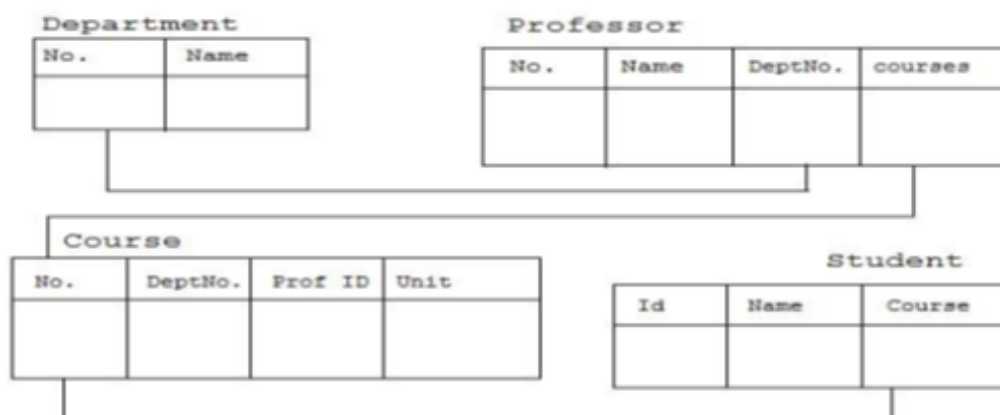
2. Network Model

- In the network model, entities are organized in a graph, in which some entities can be accessed through several path.



3. Relational Model

- In this model, data is organized in two-dimension tables called relations. The tables or relation are related to each other.



Q. Compare Network and Hierarchical Model.

S.N.	Hierarchical Model	Network Model
1.	It is based on tree like structure with one root.	It is based on records and links.
2.	Supports one to many relationships.	Supports many to many relationships
3.	Less popular	More popular than Hierarchical
4.	The main application of hierarchical data model is in the mainframe database system	Network model is upgraded version of the hierarchical model so used in the networks.
5.	It does not uses client server architecture.	It uses client server architecture.
6.	Uses pointers to relate data	Uses links to relate data

Q. Compare Network and Relational model.

S.N.	Network Model	Relational Model
1.	It is based on the tree like structure.	This is based model i.e. it is collection of rows & columns.
2.	Supports Many to Many & One to Many relationships.	Supports Many to Many & One to One relationships.
3.	It is not much popular	Much Popular
4.	Invented by Charles Bachman	Invented by E. F. Codd

Q. Compare Hierarchical and Relational model.

S.N.	Hierarchical Model	Relational Model
1.	It is based on tree like structure with one root.	This is based model i.e. it is collection of rows & columns.
2.	Supports one to many relationships.	Supports Many to Many & One to One relationships.
3.	Less popular	Much Popular
4.	The main application of hierarchical data model is in the mainframe database system	There are many application of the relational model which are unlimited.

2.2 Introduction to Relational Model

- Relational Model represents the database as a collection of tables.
- A table is a database object that stores data in form of rows and columns.
- Each row in the table represents collection of related data value.
- **Tuple:** In relational model, a row is called as tuple.
- **Attribute:** A column header is called as an attribute.
- **Degree:** The degree of relation is number of attributes of the table.
- **Domain:** All permissible values of attributes is called as a domain. Or it is the set of values of the same data types.
- **Cardinality:** Number of rows in the table is called as cardinality.
- **E.g. Create table Student_details (RollNo number(3),Name varchar2(15));**

Student_details:

Roll No.	Name
11	Nita
12	Rakesh
13	Ramesh

- In the above example Student_details is the name of Relation.
- There are two attributes RollNo and Name so Degree is 2.
- In the relation there are three tuples (rows) so Cardinality is 3.

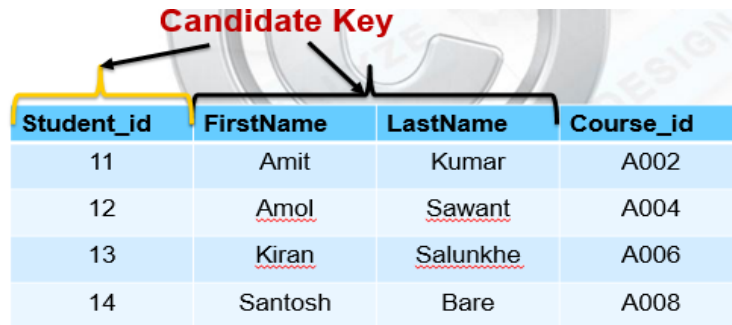
Database Keys

- Keys are very important part of Relational database.
 - They are used to establish and identify relation between tables.
 - They also ensure that each record within a table can be uniquely identified by combination of one or more fields within a table.
1. **Super Key:** - Super Key is defined as a set of attributes within a table that uniquely identifies each record within a table. Super Key is a superset of Candidate key.

2. Candidate Key:-

- It is a subset of super key.
- It is single field or the combination of fields that uniquely identifies each record in the table.

Candidate Key

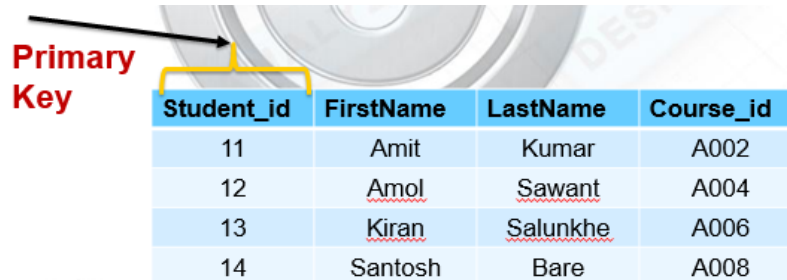


Student_id	FirstName	LastName	Course_id
11	Amit	Kumar	A002
12	<u>Amol</u>	<u>Sawant</u>	A004
13	<u>Kiran</u>	<u>Salunkhe</u>	A006
14	Santosh	Bare	A008

3. Primary Key:-

- It is a key that uniquely identify each record in a table.
- It cannot accept null, duplicate values.
- Most tables should have a primary key, and each table can have only ONE primary key.

Primary Key



Student_id	FirstName	LastName	Course_id
11	Amit	Kumar	A002
12	<u>Amol</u>	<u>Sawant</u>	A004
13	<u>Kiran</u>	<u>Salunkhe</u>	A006
14	Santosh	Bare	A008

- The following SQL creates a PRIMARY KEY on the "P_Id" column when the "Persons" table is created:
- **SQL> CREATE TABLE Persons(Student_id number(2) NOT NULL, FirstName varchar2(25), LastName varchar2(25) NOT NULL, PRIMARY KEY (Student_id));**

4. Foreign Key:-

- FOREIGN KEY in one table points to a PRIMARY KEY in another table.
- It can accept multiple null, duplicate values.
- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.
- SQL FOREIGN KEY Constraint on CREATE TABLE
- The following SQL creates a FOREIGN KEY on the "P_Id" column when the "Orders" table is created:
SQL> CREATE TABLE Orders (O_Id number(2) NOT NULL, OrderNo number(5) NOT NULL, P_Id number(2), PRIMARY KEY (O_Id), FOREIGN KEY (P_Id) REFERENCES Persons(P_Id));

- Look at the following two tables: The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	2
4	24562	1


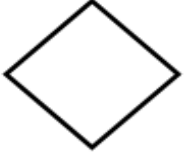




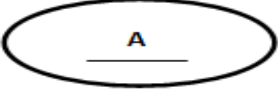
Note that the "P_Id" column in the "Orders" table points to the "P_Id" column in the "Persons" table.

The "P_Id" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

The "P_Id" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

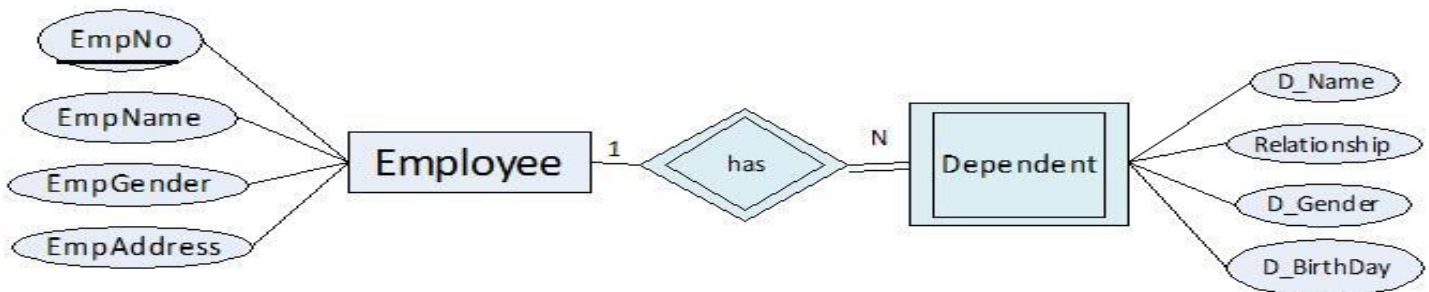
2.3 E-R Model

- Entity Relationship(ER) Diagram is a visual representation of data that describes how data is related to each other.
- The ER model a high level data model that is useful in developing a conceptual design for a database.
- Creation of an ER diagram helps the designers to understand & to specify the desired components of the database & the relationships among those components.
- Entity:** An entity is a thing or object in the real world with an independent existence. An entity may be an object with a physical existence.
- Attribute:** Describing properties of an entity is called attributes. For example, a student entity may have name, class, and age as attributes.
- Symbols and Notations of E-R Model:-**

	Rectangle Represent entities sets
	Diamonds represent relationship sets
	Lines link attribute to entity set and entity sets to relationship sets
	Represent attributes(Ellipses)
	Represent multi value attribute (Double Ellipses)
	Represent derived attribute (Dashed Ellipses)
	Indicates primary key attributes (Underline)

Components of E-R Model

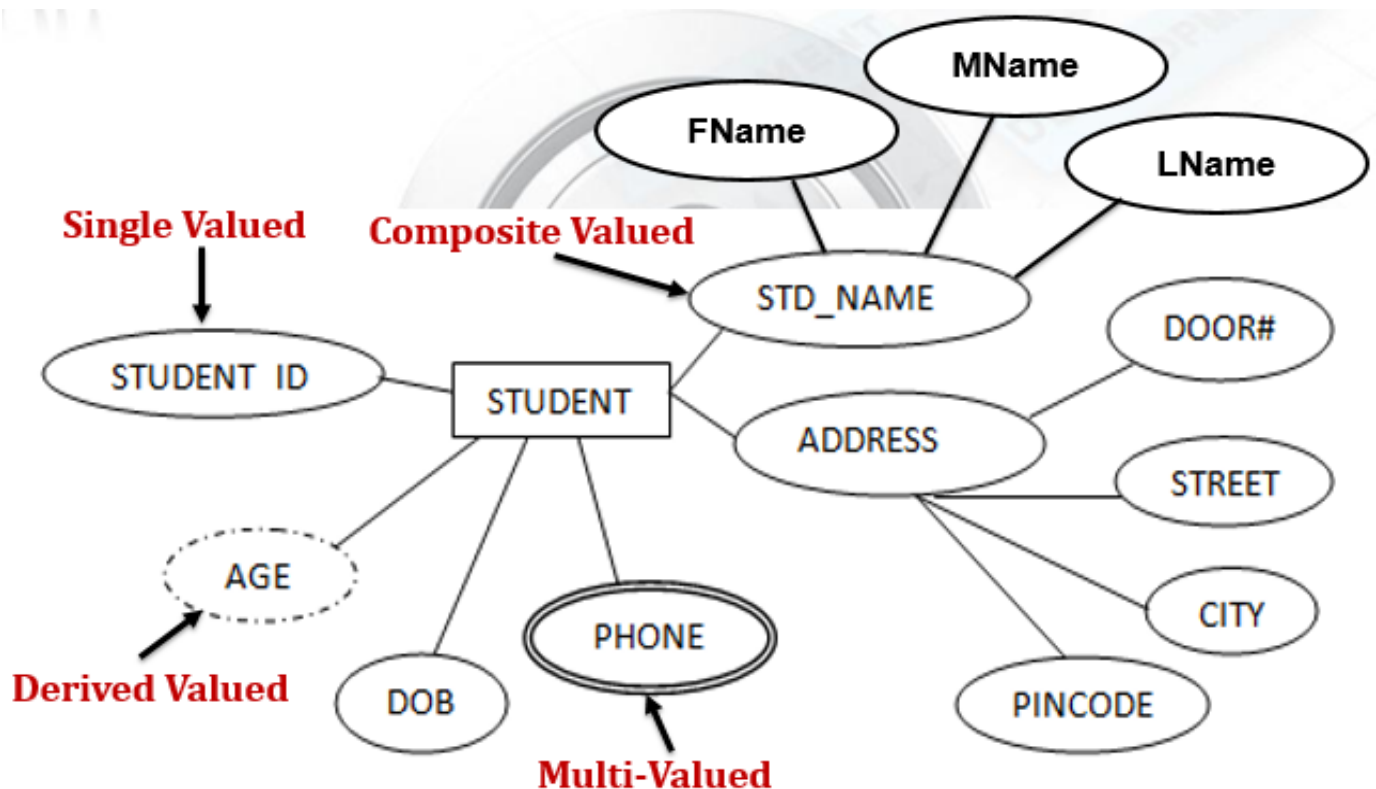
- **Entities: -** An entity is any object in the system that we want to model & store information about database. Groups of the same type of objects are called entity types or entity sets.
- **Entity Set: -** An entity set is a set of entities of the same type that share the same properties or attributes.
- **Strong Entity Set:** An entity set that have sufficient attributes to form a primary key is called as strong entity set.
- **Weak Entity Set:** An entity set that does not have sufficient attribute to form a primary key is called as Weak Entity Set.



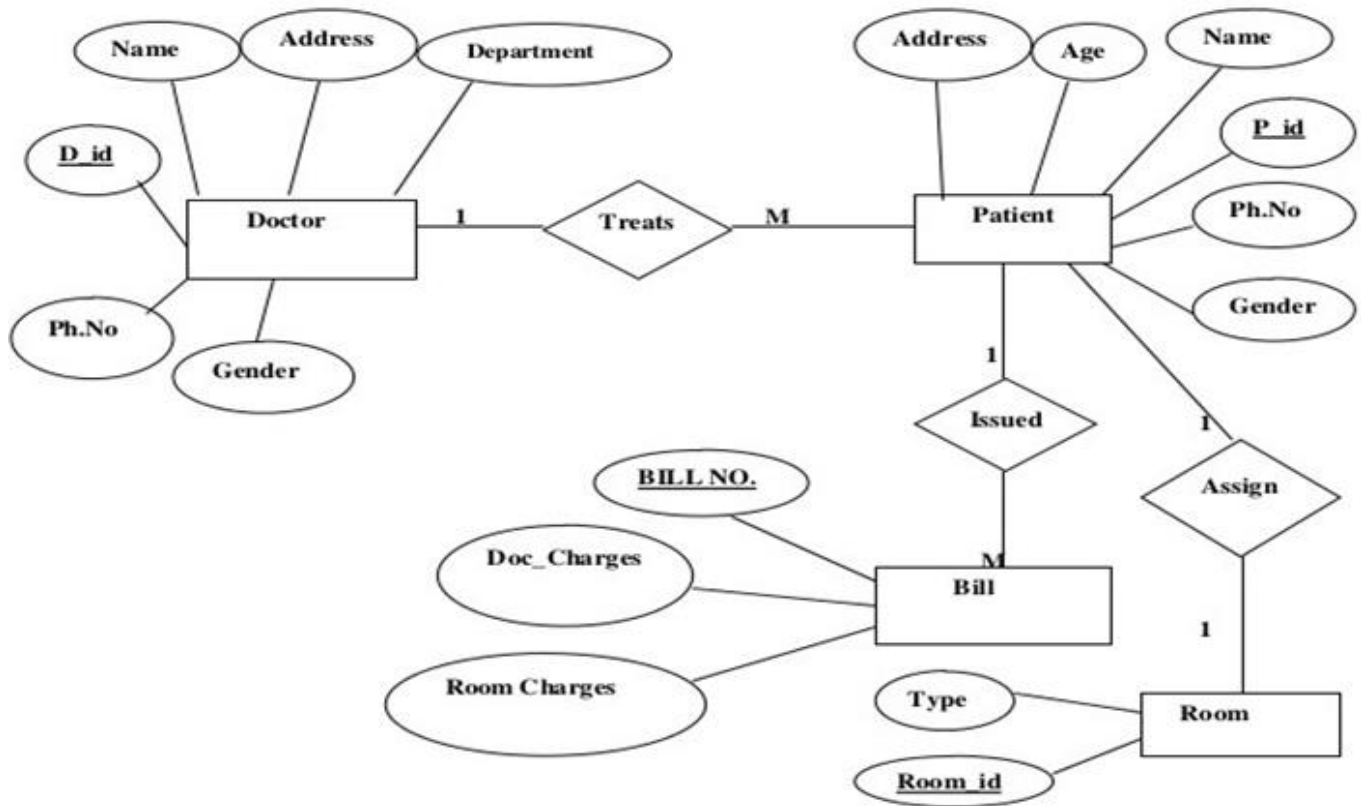
- In the above diagram Employee is a Strong Entity and Dependent is Weak Entity as it depends on Employee.

Types of Attributes

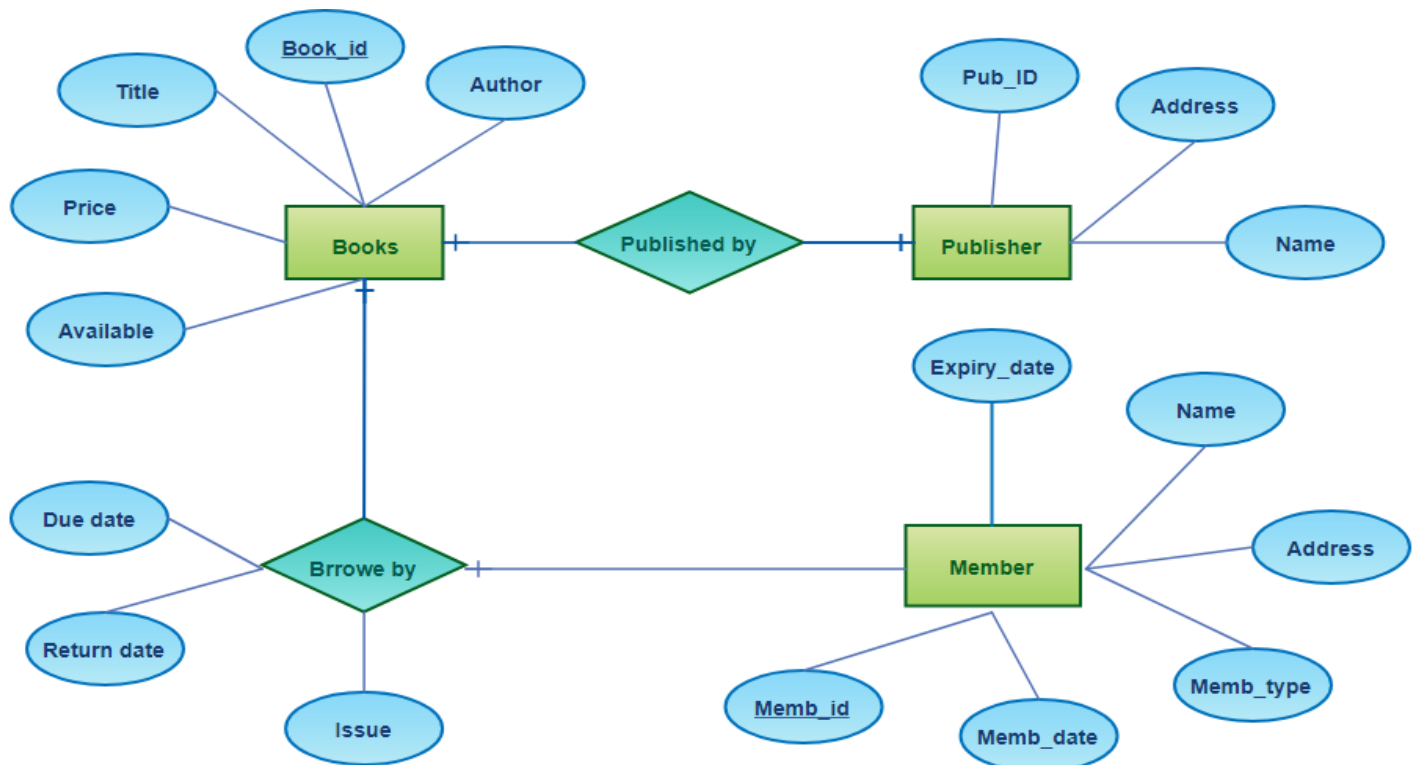
- **Single valued Attributes:** An attribute, that has a single value for a particular entity is known as single valued attributes. For example, age of a employee entity.
- **Multi valued Attributes:** An attributes that may have multiple values for the same entity is known as multi valued attributes. For example colors of a car entity.
- **Composite Attribute:** Attribute can be subdivided into two or more other Attribute. For Example, Name can be divided into First name, Middle name and Last name.
- **NULL Attribute:** It is used when an entity does not have a value for an attribute. i.e. missing or not known.
- **Derived Attribute:** Attributes derived from other stored attribute. For example age from Date of Birth and Today's date.



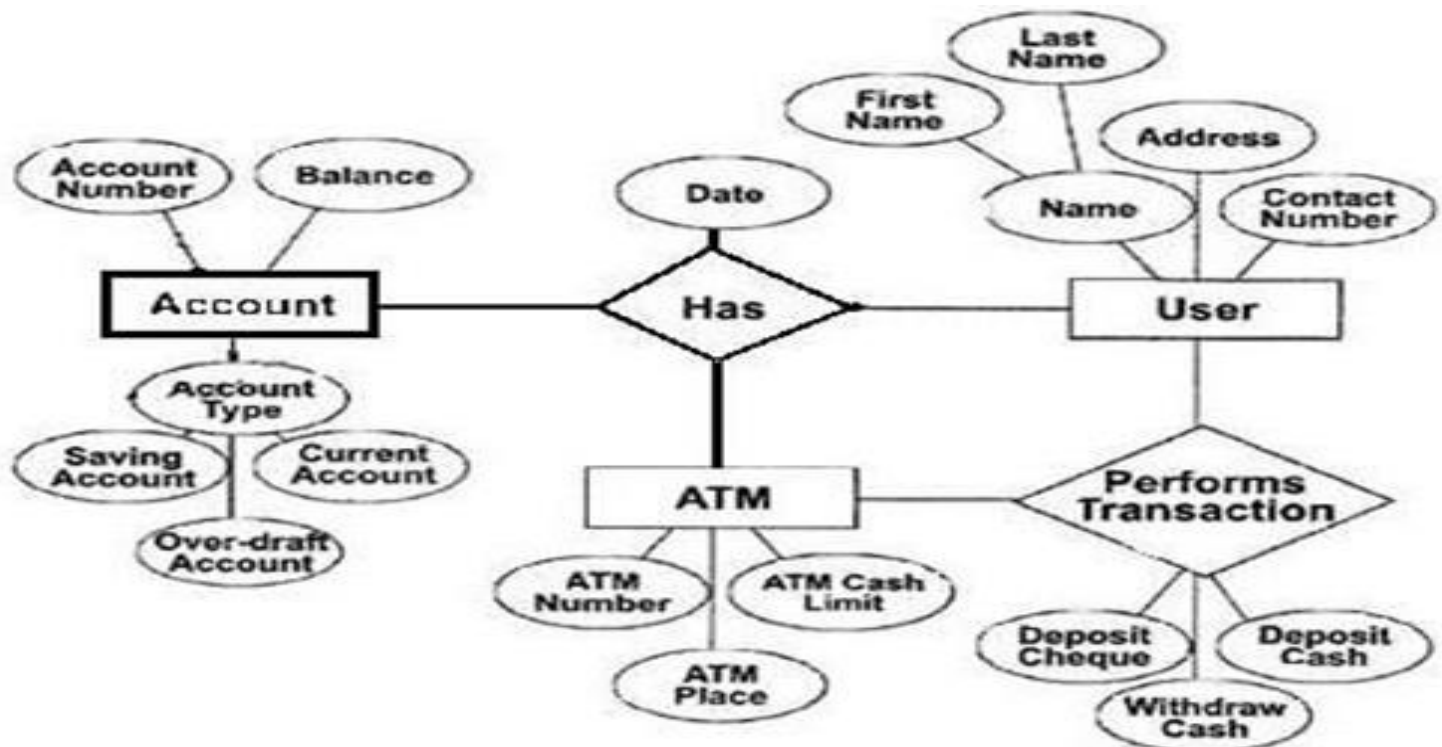
Q. Draw an E-R diagram of hospital management system. (Correct E-R Diagram - 4 marks)



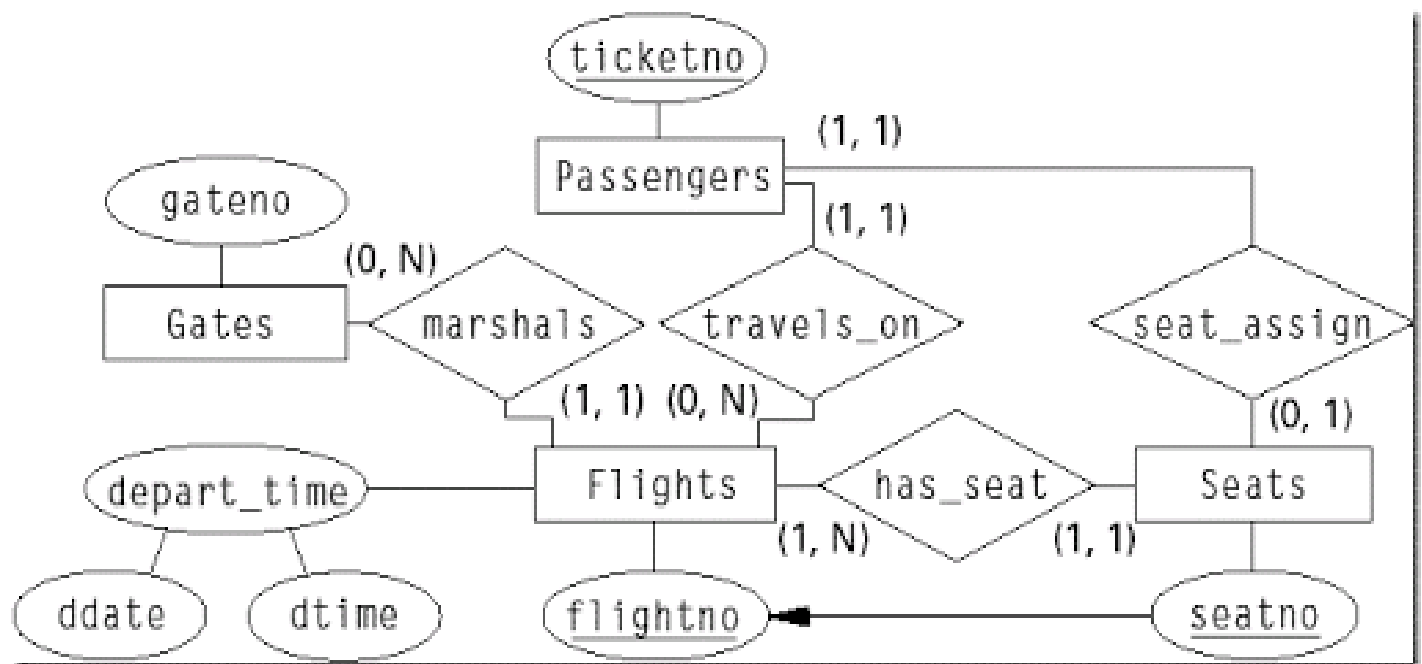
Q. Draw an E-R diagram of Library management system



Q. Draw an E-R diagram of Banking System

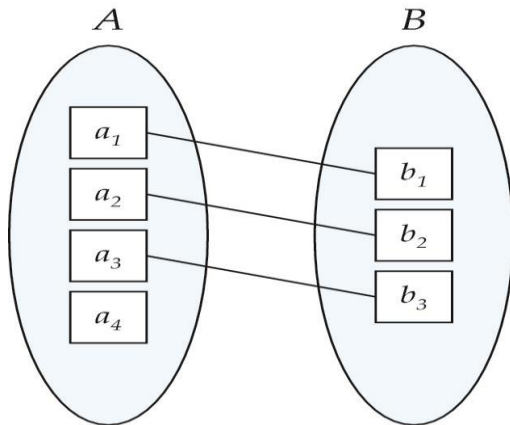


Q. Draw an E-R diagram of Airline Reservation System



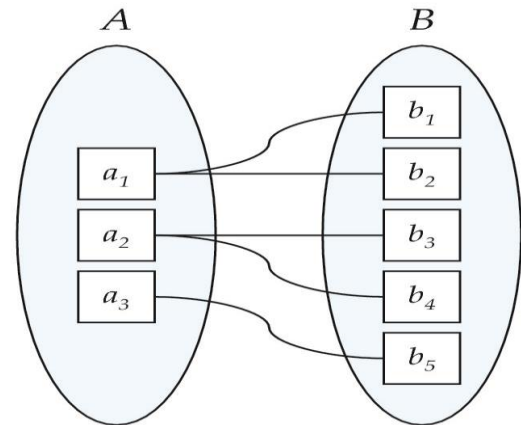
Mapping Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.
- 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



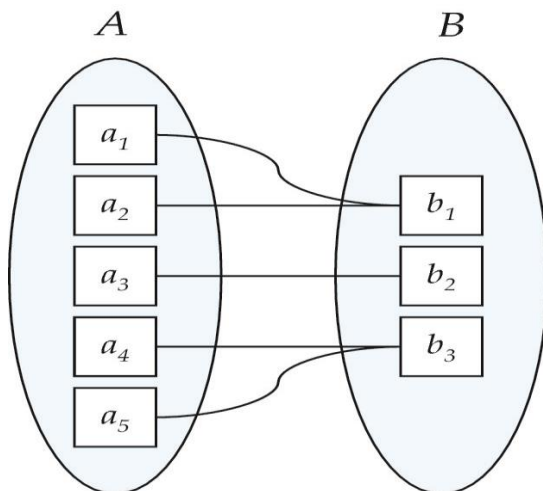
(a)

ONE-to ONE: - One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



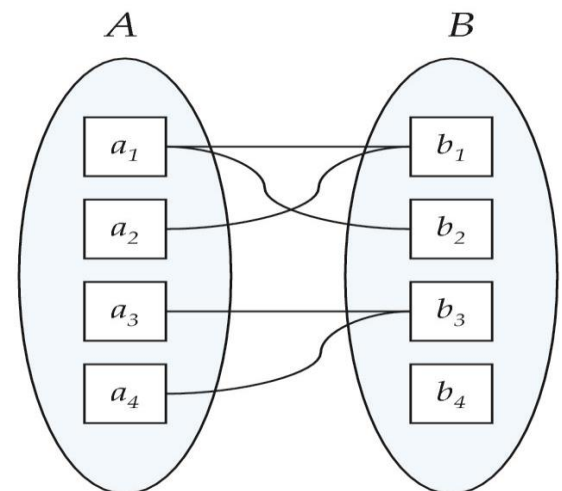
(b)

ONE-to-MANY:- One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.



(a)

MANY-to-ONE:- More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.



(b)

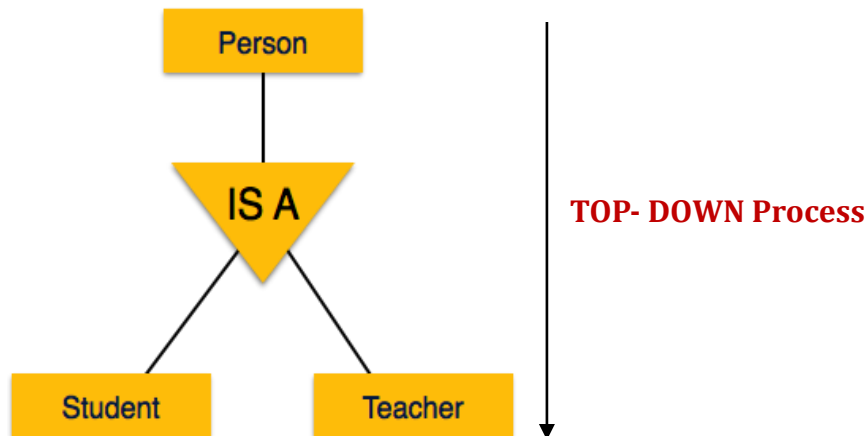
MANY-to-MANY One entity from A can be associated with more than one entity from B and vice versa.

2.4 Enhanced Entity Relationship Diagram:-

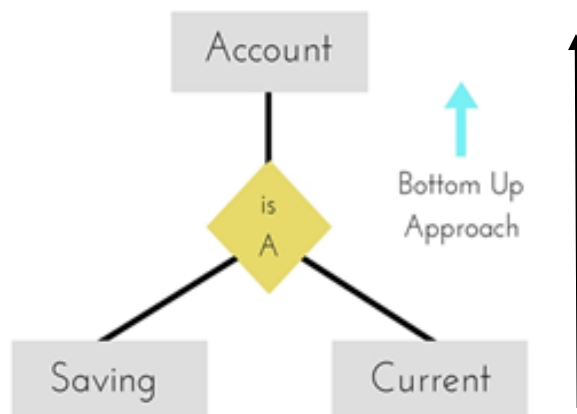
Q. Describe enhanced E-R model with the help of example.

Ans:-

- The enhanced entity-relationship model is a high level or conceptual data model incorporating extensions to the original Entity – Relationship model (E-R) model, used in design of database.
- EER model includes all modeling concepts of the ER model. In addition, EER includes: Subclasses and superclasses, specialization and generalization.
- **Specialization**
 - Specialization is opposite to Generalization.
 - The process of subgrouping within sets is called specialization.
 - It is a top-down approach in which one higher level entity can be broken down into two lower level entity.
 - In specialization, some higher level entities may not have lower-level entity sets at all.



- **Generalization**
 - Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity.
 - A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set.



2.5 Relational Algebra

- Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output.
- It uses operators to perform queries. An operator can be either unary or binary.
- The fundamental operations of relational algebra are as follows –

1. Select : σ (Sigma)	6. Intersection: \cap (Cap)
2. Project : π (Pi)	7. Difference: - (Minus)
3. Cartesian product: \times (Times)	8. Rename: ρ (rho)
4. JOIN: $ \times $ (Bow-tie)	
5. Union : \cup (Cup)	

1. The SELECT Operation Or Select : σ (Sigma) :

- The SELECT operation is used to choose a *subset* of the tuples from a relation that satisfies a selection condition.
- Unary Relational Operations
- The SELECT operation is denoted by σ <selection condition> (R) i.e. Relation name. Where the symbol σ (sigma) is used to denote the SELECT operator.
- The SQL Query is, **SQL> SELECT * FROM EMP WHERE DeptNo=20;**
- The equivalent relational algebra is, σ DeptNo=20(EMP).
- We use relational operators $\{=, <, \leq, >, \geq, \neq\}$ & logical operator AND (\wedge), OR (\vee), NOT (\neg).
- Conditions can be combined together using AND (\wedge), OR (\vee).**
- Consider following table**

EMP Table

EmpNo	LastName	Job	HirDate	Sal	DeptNo.
7369	Sharma	Developer	17-Dec-07	80000	20
7499	Mante	Tester	17-Dec-14	60000	30
7521	Johnson	DBA	17-Dec-15	50000	30

DEPT Table

DeptNo	DeptName	Location
10	Store	Mumbai
20	Research	Pune
30	Testing	Bangalore
40	Marketing	Pune

- Example: 1** Find empno of employee working in department 20 & LastName as 'Sharma'.
The SQL Query is, **SQL> SELECT EmpNo FROM EMP WHERE DeptNo=20 and LastName='Sharma';**
- Equivalent RA Query is, σ DeptNo=20 \wedge LastName='Sharma' (EMP)
- Example: 2** Find employees having salary greater than 20000.
- The SQL Query is, **SQL> SELECT * FROM EMP WHERE SAL>20000;**
- Equivalent RA Query is σ Sal>20000(EMP)

2. The PROJECT Operation:

- The PROJECT operation, selects certain columns from the table and discards the other columns.
- The SELECT operation chooses some of the rows from the table while discarding other rows.
- The general form of the PROJECT operation is $\pi \langle \text{attribute list} \rangle (R)$

Where π (π) is the symbol used to represent the **PROJECT** operation, and $\langle \text{attribute list} \rangle$ is the desired sublist of attributes from the attributes of relation R .

- **For Example 1:-** To list each employee's first and last name and salary, we can use the PROJECT operation as follows:

$\pi \text{ Lname, Fname, Salary}(\text{EMPLOYEE})$

3. SELECT and PROJECT: - It can be combined together.

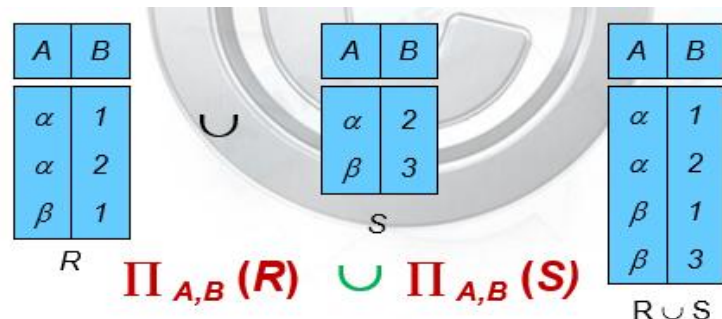
For Example:- Find LastName of employees working in deptno. 20.

Ans: - $\pi \text{ LastName } (\sigma \text{ deptno}=20(\text{EMPLOYEE}))$

4. Set Operations:

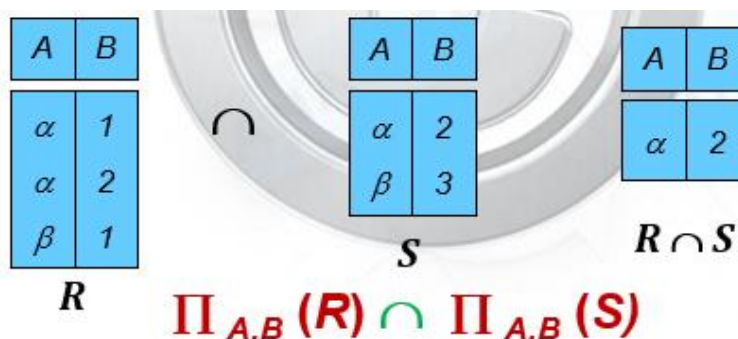
1. Union:-

- The result of this operation, denoted by $R \cup S$, is a relation that includes all tuples that are either in R or in S or in both R and S .
- Duplicate tuples are eliminated.
- Union Operation – Example



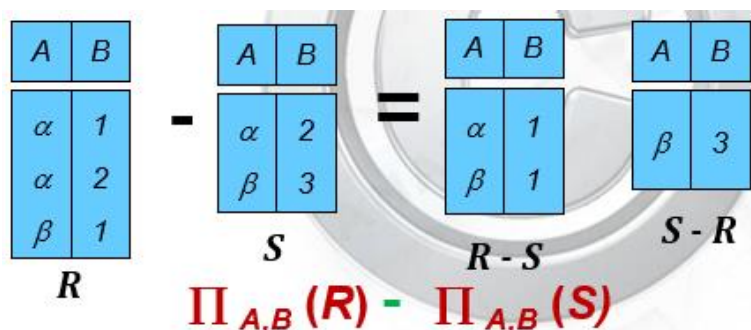
2. Intersection:-

- The result of this operation, denoted by $R \cap S$, is a relation that includes all tuples that are in both R and S .
- Intersection Operation – Example



3. SET DIFFERENCE (or MINUS):

- The result of this operation, denoted by $R - S$, is a relation that includes all tuples that are in R but not in S .
- Set Difference Operation – Example

**5. CARTESIAN PRODUCT (CROSS PRODUCT) Operation:-**

- The Cartesian product is also an operator which works on two sets.
- It is sometimes called the Cross Product or Cross Join.

$R(A_1, A_2, \dots, A_n) \times S(B_1, B_2, \dots, B_m)$ is a relation Q

■ Relations R, S :

A	B
α	1
β	2

R

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

S

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

$R \times S$

6. The JOIN Operation:

- It is a binary operation & a combination of certain Selections & a Cartesian product into one operation.
- The JOIN operation, denoted by \bowtie , is used to combine *related tuples* from two relations into single “longer” tuples.
- The join operator allows the combination of two relations to form a single new relation.
- Types JOIN Operation**

1. INNER Join: This is a simple JOIN in which the result is based on matched data as per the condition specified in the query.

- Inner Join Syntax :**

SQL> SELECT * FROM table_name1 INNER JOIN table_name2 on
table_name1.column_name = table_name2.column_name;

- Inner Join Example :**

SQL> SELECT * from emp inner join dept on emp.id = dept.id;

- The Natural Join Operation

Syntax:- $R \bowtie_{\text{(Join Condition)}} S$

Relations R:

Col A	Col B
A	1
B	2
D	3
F	4
E	5

R JOIN R ColA = S.SColA S			
A	1	A	1
D	3	D	3
E	5	E	4

$\Pi_{\text{colA, colB, SColA, SColB}} (\sigma_{R.colA=S.SColA} (R \bowtie S))$

Relations S:

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

R JOIN R ColB = S.SColB S			
A	1	A	1
B	2	C	2
D	3	D	3
F	4	E	4

$\Pi_{\text{colA, colB, SColA, SColB}} (\sigma_{R.colB=S.SColB} (R \bowtie S))$

2. The OUTER JOIN Operation: (Outer Join is based on both matched and unmatched data.)

- There are 3 forms of outer join, depending on which data is to be kept.

1. LEFT OUTER JOIN: - Keep data from the left hand table. Notation $\bowtie \sqsubset$

SQL> SELECT * FROM EMP LEFT OUTER JOIN dept ON (emp.id=dept.id);

R Left Outer Join R ColA = S.SColA S			
A	1	A	1
D	3	D	3
E	5	E	4
B	2	---	---
F	4	---	---

2. RIGHT OUTER JOIN: - Keep data from the right hand table. Notation $\bowtie \sqsupset$

SQL> SELECT * FROM EMP RIGHT OUTER JOIN dept on (emp.id=dept.id);

R Right Outer Join R ColA = S.SColA S			
A	1	A	1
D	3	D	3
E	5	E	4
---	---	C	2

3. **FULL OUTER JOIN:** - Keep data from the both table. Notation \bowtie

SQL> SELECT empname, sal from EMP FULL OUTER JOIN dept on emp.id = dept.id;

R Full Outer Join R CoIA = S.SCoIA S			
A	1	A	1
D	3	D	3
E	5	E	4
B	2	---	---
F	4	---	---
---	---	C	2

7. Rename Operator

- It is used to rename.
- It is denoted as ρ (rho).
- The rename operator returns an existing relation under a new name. $\rho A (B)$ is the relation B with its name changed to A.

Q.1. Consider the following Relational algebra schema

STUDENT (RNO, Name, DOB, Percentage, DNO)

DEPARTMENT (DNO, DNAME, HEAD)

Write relational algebra expressions:

- Find students name and course from Computer Dept.
- Get the students name who has percentage greater than 70.

Ans:- 1. $\Pi \text{ Name, DNAME } (\sigma \text{ dname} = \text{'Computer'} (\text{STUDENT} \bowtie \text{DEPARTMENT}))$

2. $\Pi \text{ Name } (\sigma \text{ Percentage} > 70 (\text{STUDENT}))$

Q. 2. Consider the structure as

Product_Master = {prod_id, prod_name, rate}

Purchase_details = {prod_id, quantity, dept_no, purchase_date}

Write a relational algebra expression for the following:

- Get product_id, product_name and quantity for all purchased product.
- Get the products with rates between 100 and 4500.

Ans:- i) $\Pi \text{ prod_id, prod_name, quantity } (\sigma \text{ Product_Master.prod_id} = \text{Purchase_details.prod_id} (\text{Product_Master} \bowtie \text{Purchase_details}))$

ii) $\Pi \text{ prod_name, rates } (\sigma \text{ rates} > 100 \wedge \text{rates} < 4500 (\text{Product_Master}))$

Q. 3. Consider student schema (studid, studname, studaddr, studcity, studper) Write relational algebra expression of the following:

- Find the name of the student those who scored first class.
- Find studid, studaddr from the student database.

- Ans:- 1. $\Pi_{studname} (\sigma_{studper \geq 60} (student))$
2. $\Pi_{studid, studaddr} (student)$

2.6 Relational Calculus

- Relational calculus is a non-procedural query language. It uses mathematical predicate calculus instead of algebra.
- It informs the system what to do with the relation, but does not inform how to perform it.
- There are two types of relational calculus –
 1. Tuple Relational Calculus (TRC)
 2. Domain Relational Calculus (DRC).

1. Tuple Relational Calculus

- It is a non-procedural calculus. It describes information without giving a specific procedure for obtaining that information.
- A query in tuple calculus is expressed as $\{t \mid P(t)\}$ or $\{t \mid \text{condition}(t)\}$ i.e. the set of all tuples (t) such that predicate (P[condition]) is true for 't'.
- We use $t[a]$ to denote the value of tuple on attribute 'a' & we use ' $t \in R$ ' to denote that tuple 't' is in relation 'R'.
- There are different symbols with specific meaning which can be used to write tuple calculus expression;-
 1. \in belong to
 2. \exists there exists
 3. \forall for all
 4. \neg not
 5. \Rightarrow implies
 6. \wedge and
 7. \vee or
- **Example:** - Find records of employees where salary is more than 20000.
Ans :- $\{t \mid t \in \text{employee} (t[\text{salary}] > 20000)\}$

2. Domain Relational Calculus

- In contrast to tuple relational calculus, domain relational calculus uses list of attribute to be selected from the relation based on the condition.
- It is same as TRC, but differs by selecting the attributes rather than selecting whole tuples.
- It is denoted as below: $\{ \langle a_1, a_2, a_3, \dots, a_n \rangle \mid P(a_1, a_2, a_3, \dots, a_n) \}$ Where $a_1, a_2, a_3, \dots, a_n$ are attributes of the relation and P is the condition.
- **For example,**
 1. Select EMP_ID and EMP_NAME of employees who work for department 10.

Ans: - $\{ \langle \text{EMP_ID}, \text{EMP_NAME} \rangle \mid \langle \text{EMP_ID}, \text{EMP_NAME} \rangle \in \text{EMPLOYEE} \wedge \text{DEPT_ID} = 10 \}$

2. Display the information of employee for salary greater than 60000.

Ans: - $\{ \langle E, N, C, S \rangle \mid \langle E, N, C, S \rangle \in \text{EMPLOYEE} \wedge S > 60000 \}$.

2.7 Database design

- Database design is the process of producing a detailed [data model](#) of [database](#).
- The term database design can be used to describe many different parts of the design of an overall [database system](#).
- Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data.
- The process of doing database design generally consists of a number of steps which will be carried out by the database designer.
 1. Determine the data to be stored in the database.
 2. Determine the relationships between the different data elements.
 3. Superimpose a logical structure upon the data on the basis of these relationships.

2.7.1 Functional Dependency

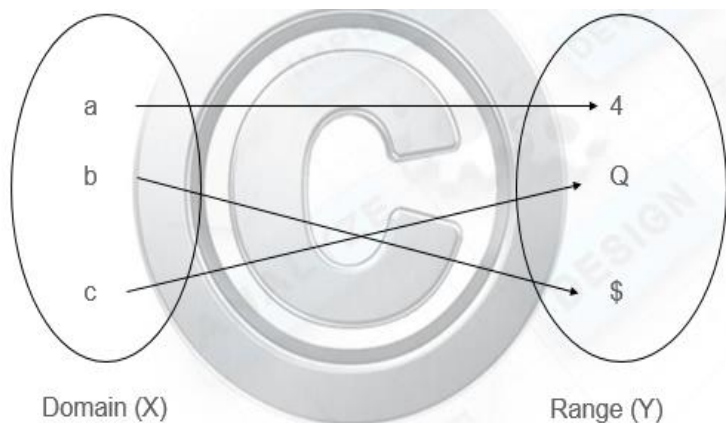
- A functional dependency occurs when one attribute in a relation uniquely determine another attribute.

OR

- Let 'R' be a relation and let X and Y be any arbitrary attributes of R, then it can be said that Y is functionally dependent on X if and only if, each X value is associated with precisely one Y value. And it can be shown as $X \rightarrow Y$.

e.g. $\text{emp_id} \rightarrow \text{ename}$ (meaning ename functionally dependent on emp_id)

Emp_id	Ename
101	Sachin
102	John
103	clark
104	taylor



In the sense that for every ename there Exists a unique emp_id.

Recall that if X uniquely determines Y, then Y is functionally dependent on X.

2.7.2 Multivalued Dependency

Definition of MVD

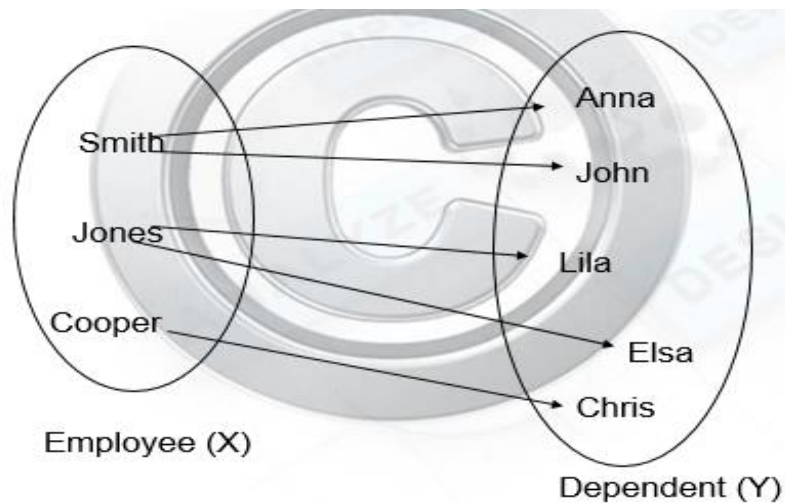
- A multivalued dependency is a full constraint between two sets of attributes in a relation.
- In contrast to the functional independency, the multivalued dependency requires that certain tuples be present in a relation.
- Therefore, a multivalued dependency is also referred as a tuple-generating dependency. The multivalued dependency also plays a role in 4NF normalization.
- **Multivalued dependencies** occur when the presence of one or more rows in a table implies the presence of one or more other rows in that same table.

OR

- **A multivalued dependency (MVD):**

In a relational table R with columns A, B and C then

$R.A \twoheadrightarrow R.B$ (column A multi determines column B) is true if and only if the set of B –values matching a given pair of A-values and C- values in R depends only on A-value and is independent of C-value.



2.8 Integrity Constraints

- **Data integrity:** - it refers to the correctness & completeness of the data in a database.
- **Constraints** enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the data integrity during an update/delete/insert into a table.
- **Integrity Constraints:** - It is a mechanism used to prevent invalid data entry into the table. It means that you are enforcing some conditions on the attributes.
- **Types of Integrity Constraints**
 1. Domain Integrity Constraints
 2. Entity Integrity Constraints
 3. Referential Integrity Constraints

1. Domain Integrity Constraints

1). **Not Null:** By default all columns in tables allows null values. When a NOT NULL constraint is enforced on column or set of columns it will not allow null values.

- **Syntax for NOT NULL**

SQL> Create Table Table_name (column_name Data_type, Column_name Data_type **Not Null**);

- **Example:**

SQL> Create Table Student (Roll_no Number (5), Name Varchar2 (20) **Not Null**);

- **After table creation not NULL can be added as:**

SQL> Alter table EMP modify empname varchar2 (20) not Null;

2). **CHECK:** The constraint defines a condition that each row must satisfy. A single column can have multiple check condition.

- **Syntax:-**

SQL> Create Table Table_name (Column_name1 Data_type, Column_name2 Data_type **Constraint Constraint_name Check <Condition>**);

- **Example:**

SQL> Create Table EMP (Id Number (5), Name Varchar2 (10), Sal Number (10) **Constraint Chk_sal Check (Sal>15000)**);

- **After table creation Check constraint can be added as:**

SQL> Alter table EMP add constraint chk_emp check (salary>15000);

2. Entity Integrity Constraints:

3). **Primary Key constraint:** It is use to avoid redundant / duplicate value entry within the row of specified column in table. It restricts null values too.

- **Syntax:**

Create Table Table_name (Column_name1 Data_type, Column_name2 Data_type **Constraint Constraint_name Primary Key**);

- **Example:**

SQL> Create Table Emp (Id Number (5) constraint Id_pk Primary Key, Name Varchar2 (10), Sal Number (10));

- **After table creation primary key can be added as:**

SQL> Alter table EMP add constraint pk_emp Primary Key (empid);

4). **Unique Constraint:** The UNIQUE constraint uniquely identifies each record in a database table. The UNIQUE and PRIMARY KEY constraints both provide a guarantee for uniqueness for a column or set of columns.

- **Syntax:**

Create Table Table_name (Column_name1 Data_type, Column_name2 Data_type **Constraint Constraint_name Unique**);

- **Example:**

Create Table Persons (p_id Number Constraint P_uk Unique, Firstname Varchar2 (20), City Varchar2 (20));

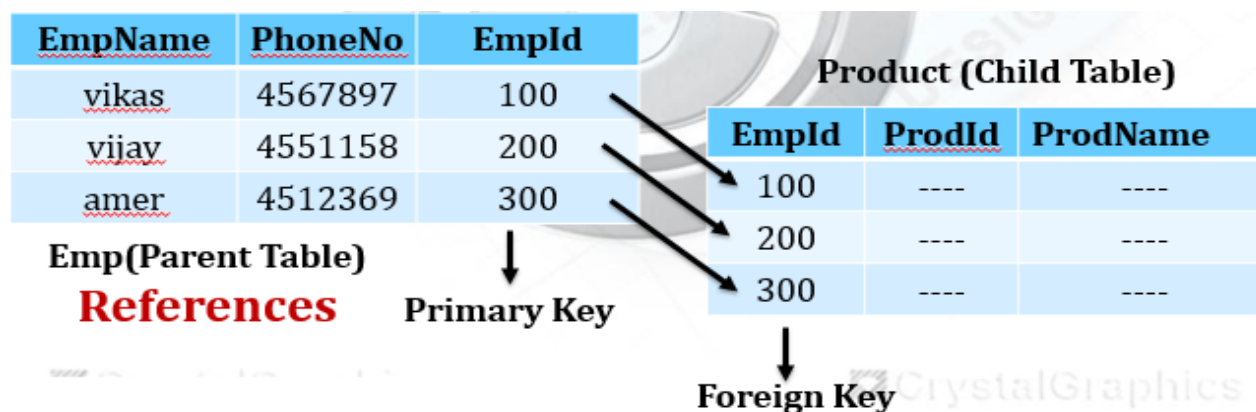
- **After table creation unique can be added as:**

SQL> Alter table EMP add constraint ph_uk unique (phoneno);

3. Referential Integrity Constraint:

5). Referential Integrity Constraint:

- It used to establish a parent child relationships between two tables.
- A value of foreign key is derived from the primary key.
- Primary key is defined in a parent table & foreign key is defined in child table.



- **Syntax:**

Create Table Table_name (Column_name1 Data_type (size) References Parent_table_name (Primary Key attribute), Column_name2 Data_type);

- **Example:**

Create Table Product (EmpId Number (5) References Emp (EmpId), ProdName varchar2 (10));

- **After table creation the foreign key can be added as:**

SQL> Alter table product add constraint fk_prod foreign key (EmpId) references Emp (EmpId);

2.9 What do you mean by database security?

- Database security refers to the collective measures used to protect and secure a database or database management software from illegal use and malicious threats and attacks.
- Database security covers and enforces security on all aspects and components of databases like Data stored in database, Database server, DBMS.

2.9.1 Database Security Requirements

- 1. Authentication:** System verifies a user's identity.
- 2. Authorization:** Which database operations that user may perform (like read, update, drop etc.) and which data objects that user may access.
- 3. Secure Storage of Sensitive Data:** Once confidential data has been entered, its integrity and privacy must be protected on the databases and servers wherein it resides.
- 4. Integrity:** Data integrity means that data is protected from deletion and corruption.
- 5. Availability:** A secure system makes data available to authorized users, without delay.
- 6. Confidentiality:** A secure system ensures the confidentiality of data. Confidentiality is the protection of personal information. This means that it allows individuals to see only the data they are supposed to see.

2.10 Normalization

- Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.
- It is the process of efficiently organizing data in a database.
- There are two goals of the normalization process:
 1. Eliminating redundant data.
 2. Ensuring data dependencies.
- Database normalization is a database schema design technique, by which an existing schema is modified to minimize redundancy and dependency of data.

2.10.1 Database Normalization

- The main goal of Database Normalization is to restructure the logical data model of a database to:
- Eliminate redundancy
- Organize data efficiently
- Reduce the potential for data anomalies.

2.10.2 Levels of Normalization

- Levels of normalization based on the amount of redundancy in the database.
- Various levels of normalization are:
 - First Normal Form (1NF)
 - Second Normal Form (2NF)
 - Third Normal Form (3NF)
 - Boyce-Codd Normal Form (BCNF)
 - Fourth Normal Form (4NF)
 - Fifth Normal Form (5NF)
 - Domain Key Normal Form (DKNF)

1. First Normal Form (1NF)

- A database table is said to be in 1NF if it contains no repeating fields/columns. The process of converting the UNF (Un-normalized table) table into 1NF.
- The requirements to satisfy the 1st NF:
 - a. Each table has a primary key: minimal set of attributes which can uniquely identify a record
 - b. The values in each column of a table are atomic (No multi-value attributes allowed).
 - c. There are no repeating groups: two columns do not store similar information in the same table.
- As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.
- **Example:** Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

Emp_id	Emp_name	Emp_address	Emp_mobile
101	Herschel	New Delhi	8912312390
102	John	Kanpur	8812121212 9900012222
103	Ron	Chennai	7778881212
104	Smith	Bangalore	9990000123 8123450987

Two employees (John & Smith) are having two mobile numbers so the company stored them in the same field as you can see in the table above.

- This table is **not in 1NF** as the rule says “each attribute of a table must have **atomic (single) values**”, the Emp_mobile values for employees John & smith violates that rule.
- To make the table complies with 1NF we should have the data like this:

Emp_id	Emp_name	Emp_address	Emp_mobile
101	Herschel	New Delhi	8912312390
102	John	Kanpur	8812121212
102	John	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Smith	Bangalore	9990000123
104	Smith	Bangalore	8123450987

2. Second Normal Form (2NF)

- A table is said to be in 2NF if both the following conditions hold:
 - Table is in 1NF (First normal form)
 - Prime attribute – an attribute, which is a part of the prime-key, is known as a prime attribute.
 - Non-prime attribute – an attribute, which is not a part of the prime-key, is said to be a non-prime attribute.
- If we follow second normal form, then every non-prime attribute should be fully **functionally** dependent on prime key attribute.
- Example:** Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this:

Teacher_id	Subject	Teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate Keys:
{Teacher_id, Subject}
Non-prime attribute:
Teacher_age

The table is in **1 NF** because each attribute has atomic values. However, **it is not in 2NF because non-prime attribute Teacher_age is dependent on Teacher_id alone which is a proper subset of candidate key.** This violates the rule for 2NF.

- To make the table complies with 2NF we can break it in two tables like this:

Teacher_details Table:

Teacher_id	Teacher_age
111	38
222	38
333	40

Teacher_subject table:

Teacher_id	Subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

3. Third Normal Form (3NF)

- A table design is said to be in 3NF if both the following conditions:
 - Table must be in 2NF
 - [Transitive functional dependency](#) of non-prime attribute on any super key should be removed.
- An attribute that is not part of any [candidate key](#) is known as non-prime attribute.
- In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency $X \rightarrow Y$ at least one of the following conditions hold:
 - X is a [super key](#) of table
 - Y is a prime attribute of table
 - An attribute that is a part of one of the candidate keys is known as prime attribute.
- Example:** Suppose a company wants to store the complete address of each employee, they create a table named Employee_details that looks like this:

Emp_id	Emp_name	Emp_zip	Emp_state	Emp_city	Emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UP	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

Here, Emp_state, Emp_city & Emp_district dependent on Emp_zip. And, Emp_zip is dependent on Emp_id that makes non-prime attributes (Emp_state, Emp_city & Emp_district) transitively dependent on super key (Emp_id). This violates the rule of 3NF.

- To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

Emp_id	Emp_name	Emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

Employee Table

Emp_zip	Emp_state	Emp_city	Emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

Employee_zip Table

4. Boyce Codd Normal Form(BCNF)

- Definition: A relation R is in Boyce-Codd normal form (BCNF) if and only if every determinant is a candidate key.
- The process of converting the table into BCNF is as follows:
 1. Remove the non-trivial ($x \rightarrow y$, y is not subset of x) functional dependency.
 2. Make separate table for determinants.
- **Example:**

Consider the relation **SUPPLIER (SNO, SNAME, PH_NO, CITY)** having **SNO** and **SNAME** unique. In this there are 2 determinants **SNO, SNAME** as **PH_NO** and **CITY** dependence upon them and both are candidate keys. So this is in BCNF.

Q. State properties of Boyce Codd Normal Form.

1. BCNF: A relation R is in Boyce - Codd normal form if and only if every determinant is a candidate key.
2. In BCNF non-trivial functional dependency is preserved for super key.
3. A table can be in 3NF but not in BCNF.
4. 3NF does not deal satisfactorily with the case of a relation with overlapping candidates keys, in such case BCNF can be used.
5. The violation of BCNF means that the table is subject to anomalies.

5. Fourth Normal Form (4NF)

- A database table is said to be in 4NF if it is BCNF & primary key has one to one relationship to all non-keys fields. **OR**
- We can also said a table to be in 4NF if is in BCNF & contains no multi-valued dependencies.
- The process of converting the table into 4NF is as follows:
 1. Remove the multivalued dependency
 2. Make separate table for multivalued fields.
- **4NF of below table is as follows:**

Emp_Name	Skills	Language
Mohan	C#	Hindi
Mohan	ASP.net	Hindi
Mohan	SQL Server	Hindi
Mohan	C#	English
Mohan	ASP.net	English
Mohan	SQL Server	English

Emp_Name	Skills
Mohan	C#
Mohan	ASP.net
Mohan	SQL Server

Fourth Normal Form (4NF)

Emp_Name	Language
Mohan	Hindi
Mohan	English

6. Fifth Normal Form (5NF)

- A database table is said to be in 5NF if it is 4NF & contains no redundant values. OR
- We can also said table to be in 5NF if it is 4NF & contains no join dependencies.
- The process of converting the table into 5NF is as follows:
 1. Remove the join dependency
 2. Break the database table into smaller & smaller tables to remove all data redundancy.
- **5NF of below table is as follows:**

Company	Product	Supplier
Godrej	Soap	Mr. Amit
Godrej	Shampoo	Mr. Pavan
Godrej	Shampoo	Mr. Amit
H. Lever	Soap	Mr. Amit
H. Lever	Shampoo	Mr. Pavan
H. Lever	Soap	Mr. Sachin

Company	Product
Godrej	Soap
Godrej	Shampoo
H. Lever	Soap
H. Lever	Shampoo

Company	Supplier
Godrej	Mr. Amit
Godrej	Mr. Pavan
H. Lever	Mr. Amit
H. Lever	Mr. Pavan
H. Lever	Mr. Sachin

Product	Supplier
Soap	Mr. Amit
Shampoo	Mr. Pavan
Shampoo	Mr. Amit
Soap	Mr. Sachin

**Fifth Normal Form
(5NF)**

IMPORTANT QUESTIONS:-**SUMMER-2016 (38 Marks)**

1. List various data models. (*Listing of any 2 models - 1 mark each*)
Ans: Relational Database Model, Hierarchical Model, Network Model, E-R Model
2. Define attribute and entity. (*Definition of Attribute - 1 mark; Entity - 1 mark*)
3. Define normalization. (*Definition - 2 marks*)
4. List and explain the types of integrity constraints in detail. (*Listing - 1 mark; Any Two Constraints explanation - 1 ½ marks each*)
5. State properties of Boyce Codd Normal Form. (*Any two properties - 2 marks each*)
6. Describe Relational model with example. (*Explanation - 2 marks; example - 2 marks*)
7. Consider the following Relational algebra schema
STUDENT (RNO, Name, DOB, Percentage, DNO) DEPARTMENT (DNO, DNAME, HEAD)
Write relational algebra expressions:
i) Find students name and course from Computer Dept.
ii) Get the students name who has percentage greater than 70
8. Consider the structure as
Product_Master = {prod_id, prod_name, rate}
Purchase_details = {prod_id, quantity, dept_no, purchase_date}
Write a relational algebra expression for the following:
i) Get product_id, product_name and quantity for all purchased product
ii) Get the products with rates between 100 and 4500.
9. Draw an E-R diagram of hospital management system. (*Correct E-R Diagram - 4 marks*)
10. Compare network and hierarchical model. (*Any 4 points - 1 mark each*)
11. Explain 3NF with example. (*Explanation - 2 marks; example - 2 marks*)

WINTER-2016 (42 Marks)

1. List any two data model. (*Any 2 models: 1 mark each*)
2. What is multi-valued dependency?
3. List different relational algebraic operators any four.
4. Explain strong entity and weak entity set. (*Strong Entity Set: 2 marks, Weak Entity set :2marks*)
5. Explain functional dependencies and 2 NF with example. (*Functional dependency: 2 marks, 2NF: 2marks*)
6. Explain Database security with its requirements. (*Database security: 2marks, Requirements: 2 marks*)
7. Explain entity integrity constraints with syntax and example. (Primary key constraint:2marks, Unique key constraint:2marks)
8. Explain tuple relational calculus with example. (Explanation: 2 marks, any one Example: 2marks)
9. Consider student schema(studid, studname, studaddr, studcity, studper) Write relational algebra expression of the following:
i) Find the name of the student those who scored first class.
ii) Find studid, studaddr from the student database.
10. Explain BCNF with example. (Explanation :2 marks, Example: 2 marks)
11. Explain Domain integrity constraint with syntax and example.
12. Explain the term specialization and generalization with suitable example.

WINTER-2015 (30 Marks)

1. What is domain and entity?(Definition of Domain – 1 Mark; Definition of Entity – 1 Mark)
2. What is meant by database normalization?(Database Normalization – 2 Marks)
3. What do you mean by database security?
4. Explain any four integrity constraints.(Any 4 Integrity constraint – 1 Mark each)
5. Describe enhanced E-R model with the help of example.(Description – 2 Marks, Diagram - 2 Marks)
6. Describe functional dependencies with example. (Explanation of functional dependency – 2 Marks, Example – 2 Marks)
7. Define the term.
 - (i) Candidate key
 - (ii) Primary key (Candidate key definition – 2 Marks, Primary key Definition – 2 Marks)
8. Explain tuple relational calculus with example.(Description - 3 Marks, example - 1 Mark)
9. Explain not null constraints by suitable example. (Explanation -2 Marks, Example -2 Mark)