Database Management System :-

Data :-

→ Data is the collection of information, facts, figures, that is stored in or used by computer.

Database :-

→ The collection of structure (or) organised data usually refer to the database, content information releavant to interprice

Database management system :-

→ To manage the database by a tool is known as database management system.

→ Database management system is a software, which is responssible for performing all the types of operation such as insert, retrive, update, dilate with database.

→ Database management System is a intermediate between developer and database.

Application of DBMS :-

→ Railways
→ Airline
→ Tours and Travels.
→ Educational institute.
→ Finance.
→ Banking etc.

Goals of DBMS :-

→ It is providing a way to store and retrive database information, i.e both convinient and efficient (user friendly)

→ Ensuring the safety of information.

10/7/25

Disadvantages of file system
or
Advantages of data base

→ Conquercing and inconsistency. (duplicate) (if change on update in one place not update in somewhere else)

→ Difficulty in data accessing (user should know the exact location of the data).

→ Data isolation. (Combining of different files data into a single file for specific task) Ex- Excel, Updated Excel, CSV file

→ Integrity problems (increasing risk of data errors)
Ex- You have 4000, you withdraw 5000.

→ Atomicity Problem
(all the transaction must be Atomic means it means it most happen entirely or not at all. and it is difficult to ensure atomicity in a conventional file processing system).

→ Concurrent access. (Same file multiple updations by users)
Ex- 1 shint 2 user accessing, father and mother sending 1000 to you.

→ Security :- Unauthorized access and modification happen is file system.

File system	DBMS
→ In file system duplication of data and inconsistency happens in high amount.	→ In DBMS duplication of data and inconsistency is low.
→ Accessing of data is difficult in file system.	→ Accessing of data is compenatively easy in DBMS.
→ file system does not provide Data isolation facilities.	→ DBMS provide data isolation facilities.
→ Increase of risk of data errors in file system.	→ Decrease risk of data errors in DBMS.
→ File system not provide support to complicated transaction.	→ DBMS provide support and make easy to the complicated transaction.
→ File system does not offer concurrent access facilities.	→ DBMS offers concurrent access facilities.
→ File system offers lesser security.	→ DBMS offers high security.
→ Filesystem does not provide backup and recovery of data if it is lost.	→ DBMS provide backup and recovery of data if it is lost.

Components of DBMS:-

→ Hardware - hard disk, system, printer, CPU, monitor, keyboard
→ software - operating system.
→ data - include user data, metadata, application data.
→ users.
→ Procedure:-
① log in to DBMS.
② Use a particular DBMS facility.
③ Start and stop the DBMS
④ Make a backup copy of data base.
⑤ Handle the hardware and software failures.

Users

→ User can access the data on demand of by using the application and interface provided by DBMS.

① Naive Users :- The users who were donot know on aware of data base but use them.
Ex- kids, old age

② Application Programer:- Those who are designed the Programs for database software.

③ Sophisticated User. - Those who are accessing the database by writing the query.

④ Specialized user - Those who are design the data base.

⑤ Database Adminstrator (DBA) - Those who can grant and revoke the permission
Ex: group admin.

→ DBA can be a single person on group of members.
→ DBA is responssible for everything related to database i.e the DBA is responsible for authonising access to the data base. by grant and revoke permission to the users for coordinating and monitoring its use, managing backups and repairing damages due to hardware and software failures.
→ In case of small organisation the role of DBA is Performed by a single person and in case of large

Organisation a group of DBA's who shares responsibilities.

→ Example of popular DBMS.

MYSQL, Oracle, Microsoft SQL Service.

View of data :-

→ A database system is the collection of interrelated data and set of programme, that can allow users to access and modify these data.

→ A major purpose of database is to provide users with an abstract view of data, i.e the system hides certain details of how the data stored and maintained.

Data Abstraction :-

→ To hide the complexity from users through several levels of abstraction to simplify user interactions with the system.

→ The process of hiding irrelavant details from user is known as data abstraction.

→ Data abstraction can be divided to three levels

1. Physical level / Internal schema. (where is the data).
2. Logical level / Conceptual schema. (Programme, concept, logic)
3. View level / External schema. (overview)

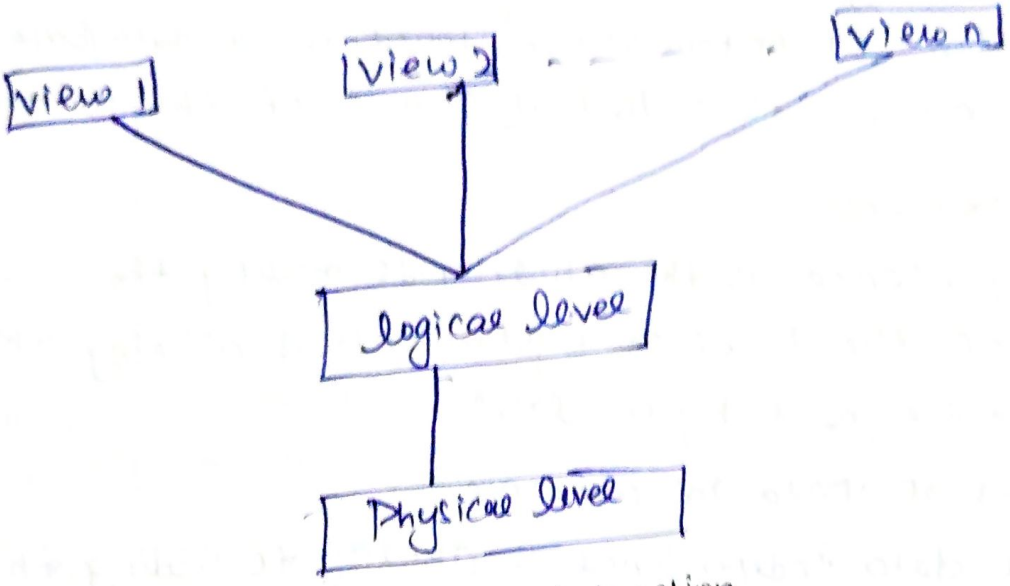
NOTE

The overall design of our database is called schema.

① Physical level / Internal Schema :-

→ This is the lowest level of data abstraction, It
→ It describes how data is actually stored in database.
→ You can get the complex data structure details at this level.
→ Ex- let's say we are storing customer information in a customer table. At physical level these records can be described as blocks of storage (bytes, gigabytes, terabytes etc). in memory. These details are often hidden from the Programmers

Data abstraction:



Three levels of data abstraction.

② Logical level / Conceptual schema:-

→ This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.

→ At the logical level these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented.

→ The programmers generally work at this level because they are aware of such thing about database system.

③ View level / External schema:-

→ Highest level of data abstraction.

→ This level describes the user interaction with database system.

→ At view level, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

Database schema and instance :-

→ The overall design of the database is called schema.

→ The collection of information that stored in a database at a particular moment is called instance of the database.

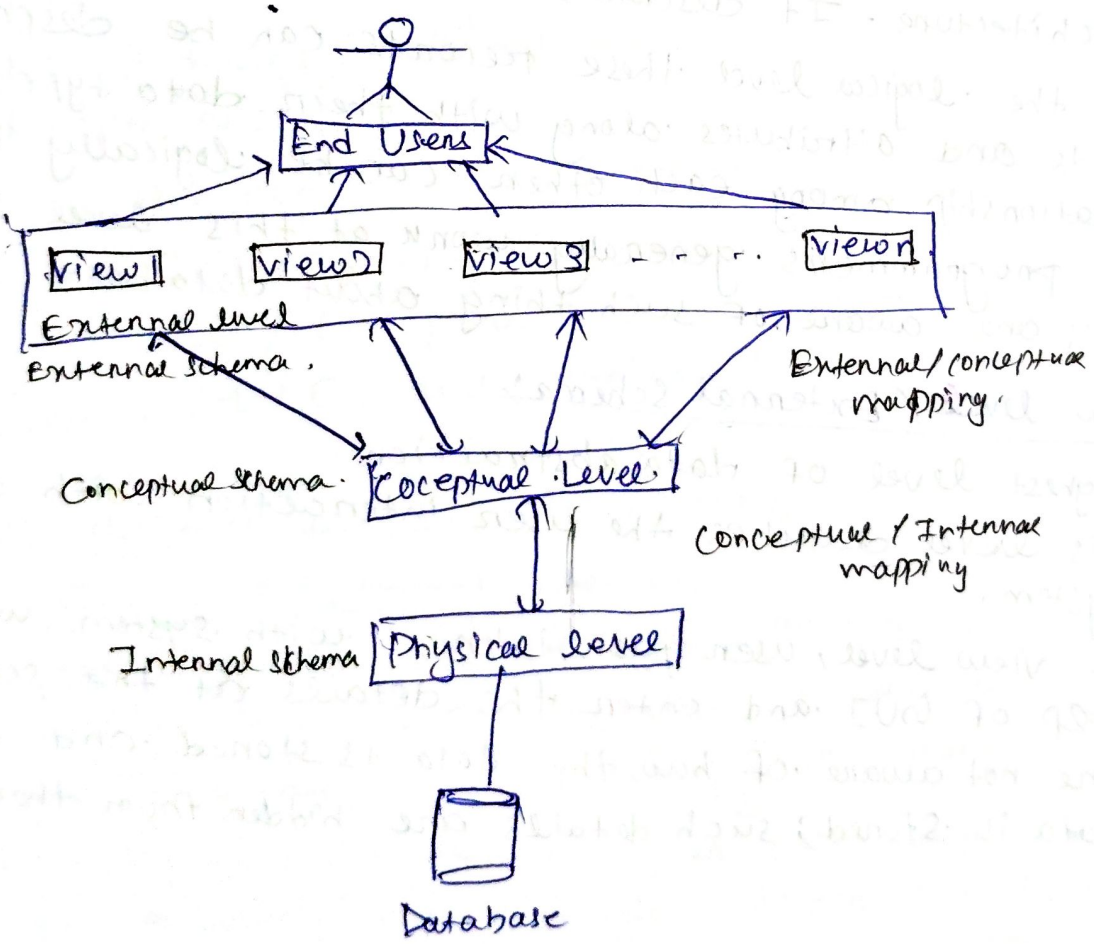
Data independence :-

→ Data independence is the ability to modify the schema at one level of the database system without altering the schema at the next higher level.

→ Two types of Data Independence,

i) Logical data independence :- ability to modify the logical schema without changing the external view and application.

ii) Physical data independence :- ability to modify the physical schema without changing the logical schema.



Three level DBMS Architecture.

Mapping:-

→ Process of transforming request and result between three level is called mapping.

→ There are 2 types of mapping.
  i) Conceptual / internal mapping.
  ii) External / conceptual mapping.

i) conceptual / internal mapping:-

→ The conceptual / internal mapping defines the correspondence between the conceptual view and the Stone database.

→ It specifies how conceptual record and fields are represented at the internal level.

→ It relates conceptual schema with internal schema.

ii) External / conceptual mapping:-

→ The external / conceptual mapping defines the correspondence between a particular external view and conceptual view.

→ It relates each external schema with conceptual schema.

DBMS Architecture:-

Centralized:-

→ It is the database stone, located as always maintained at single location only.

Decentralized:-

→ It is the database does not have central owner on single authority.

→ The design of a DBMS depends on its architecture.

→ The DBMS architecture depends upon how users are connected to the database to get their request done.

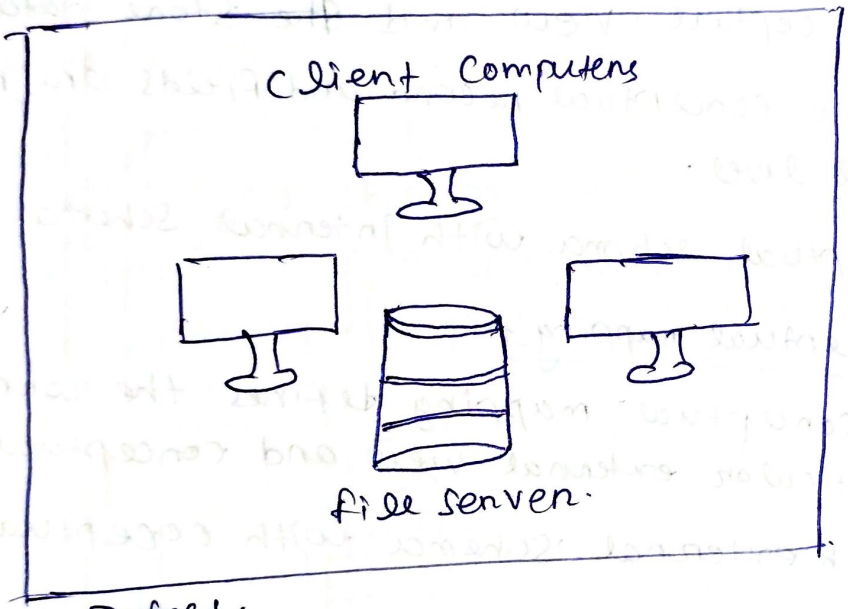
→ Database Architecture can be seen as single tier or multi-tier.

→ These tiers are classified as follows that is.

① 1-tier Architecture
② 2-tier Architecture
③ 3-tier Architecture

① 1- tier Architecture :-

→ Basically, a one-tier architecture keeps all of the elements of an application, including the interface, middle ware and back-end data in one place.



→ Ex- Own Project.

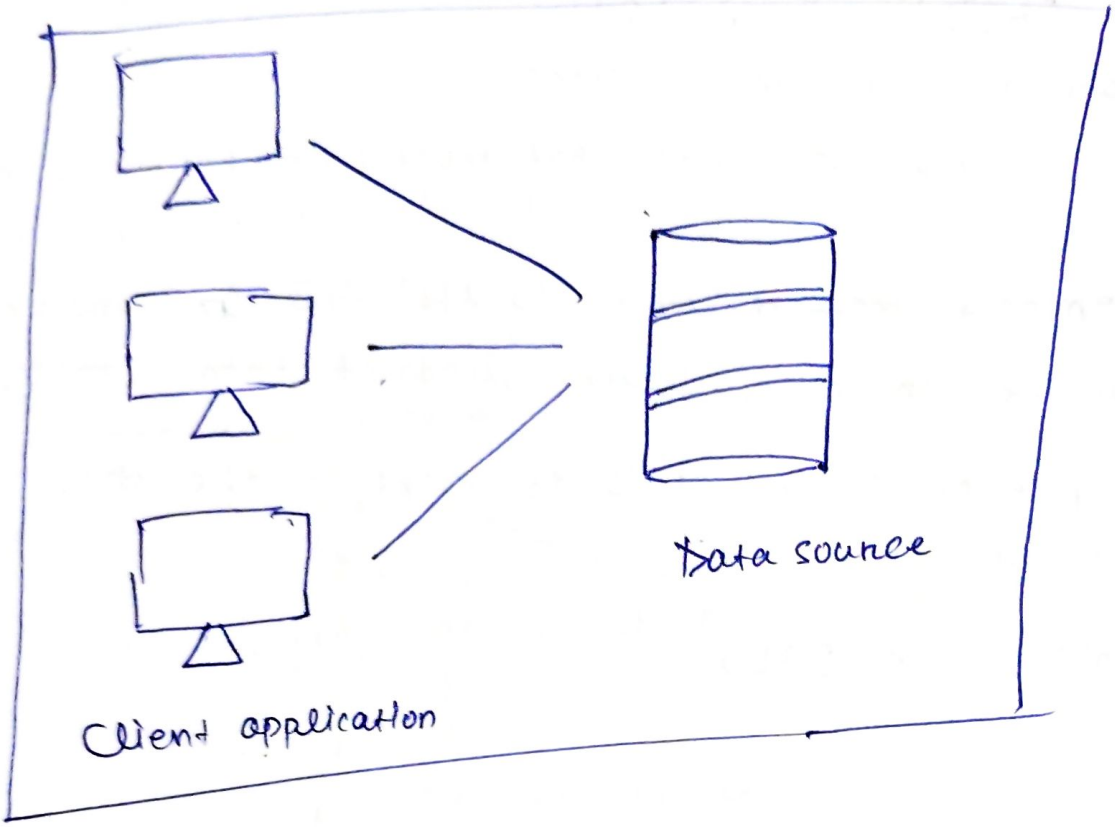
② 2- Tier Architecture :-

→ Two-Teen Architecture is basically same as the basic - client-server architecture.

→ In 2-tier architecture application on the client end can directly communicate with database and server.

→ Ex- Railways ticket reservation, the clerk work as client, and use the database directly.

12/09/2025



3. 3-Tier Architecture :-

→ A 3-Tier architecture separates its tier from each other based on complexity of the user and how they use the data present in the database.

→ It is the most widely used architecture.

i) Data layer :- At this layer the database resides along with its query processing language.

ii) Application layer - At this layer the application server and the programmer that access the database.

iii) Client layer :- End users operate on this layer and they know nothing about any existance of database.

→ Ex- : ERP Section, mobile app of IRCTC.

Constraints :- (Relational model) :-

→ Constraints are nothing but conditions.

→ Every relation has some conditions that must hold for it to be a valid relation.

→ Constraints enforce limit to the data or type of data that can be inserted, deleted, updated from a table.

→ The whole purpose of constraints is to maintain the data integrity into a table.

→ Constraints available in SQL,

Pre defined
- i) NOT NULL
- ii) UNIQUE
- iii) PRIMARY KEY (NOT NULL + UNIQUE).
- iv) FOREIGN KEY
- v) CHECK (Checking the condition), (note 18).
- vi) DEFAULT

User defined
- vii) DOMAIN CONSTRAINTS,

i) NOT NULL

→ NULL means empty, i.e the value is not available OR not applicable.

→ Whenever a table's column is declared as NOT NULL, then the value for that column cannot be empty for any of the table's record.

→ There must exist a value in the column to which the NOT NULL constraint is applied.

NOT NULL : (Contd--).

```
CREATE TABLE student (student_ID INT NOT NULL, student_
                      first_name VARCHAR(20));
```

ii) UNIQUE
→ Duplic
. the UN
→ The co
a uniq
→ This
one
U
(
Cons
UN
CREATE
- fir
iii) PRIMAR
→ PRIMAR
UNIQU
→ The co
will
values.
PRI
CRE
→ If tab
primary
AL

ii) UNIQUE :-

→ Duplicate values are not allowed in the columns to which the UNIQUE constraint is applied.

→ The column with the unique constraint will always contain a unique value.

→ This constraint can be applied to one or more than one column of a table which means more than one unique constraints can exist on a single table.

UNIQUE (contd - -).

CREATE TABLE Student.

UNIQUE (contd - -).

CREATE TABLE Student (Student_ID INT UNIQUE, Student_First name VARCHAR(20));

iii) PRIMARY KEY :-

→ PRIMARY KEY constraint is a combination of NOT NULL and UNIQUE constraints.

→ The column to which we have applied primary constraint will not contain a duplicate value and will not allow null values.

PRIMARY KEY (contd.)

CRETE TABLE Student (Student_ID INT PRIMARY KEY);

→ If table does not contain primary key then we can add primary key using :

ALTER TABLE Student ADD PRIMARY KEY (StudentID);

iv) FOREIGN KEY:-

-> A foreign key is used for referential integrity.

-> When we have two tables and one table takes reference from another table, i.e., the same column is present in both the tables and that columns acts as a primary key in one table, that particular column will act as a foreign key in another table.

v) CHECK:-

-> Whenever a check constraint is applied to the table's column and the user wants to insert the value in it, then the value will first be checked for certain conditions before inserting the value into that column.

CHECK (Cond---).

```
CREATE TABLE Student ( Age INT CHECK ( Age <=15));
```

-> If CHECK constraints is not given at the time of creation we can add using.

```
ALTER TABLE Student ADD CHECK (Age <=15);
```

vi) DEFAULT

-> Whereven a default constraint is applied to the to column and the user has not specified the value to be inserted in it, then the default value which was specified while applying the default constraint will be inserted into that particular column.

DEFAULT (cond---).

```
CREATE TABLE Student (Student-Email-ID VARCHAR (40)
                      DEFAULT 'snsahu@gmail.com');
```

vii) Domain constraints :-

→ Domain constraints are user defined data type and we can define them like this.

Domain constraints = data type + constrains (CHECK / NOT NULL ---).

→ Ex - I want to create table "student-info", with "stud-id" field having value greater than 100, I can create a domain and table like this.

```
Create domain id_value int
check (value > 100);

create table student_info (
    stu_id id_value PRIMARY KEY,
    stu_name varchar(30),
    stu_age int
);
```

24/07/25

Cardinality Mapping:-

→ Cardinality means how the entities arranged to each other or what is the relationship structure between entities in a relationship set.

→ In DBMS Cardinality represents a number that denotes howmany times an entity is participating with another entity in a relationship test.

→ In a table the no. of rows or tuples represent the Cardinality.

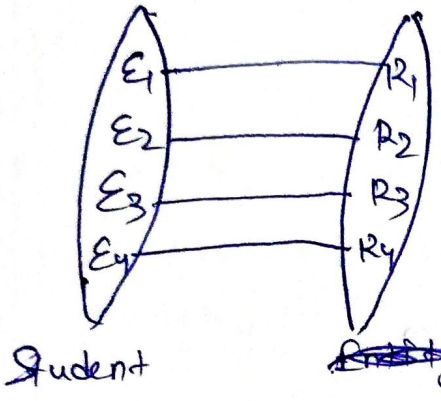
→ Cardinality mapping is known as cardinality ratio, this represents the mapping of one entity set to another entity set in a relationship set.

→ There are 4 types of cardinality mapping.

1) One to One relationship :-

→ One to one cardinality mapping can be represented as 1:1.

→ Here that is almost 1 relationship from 1 entity to another entity.



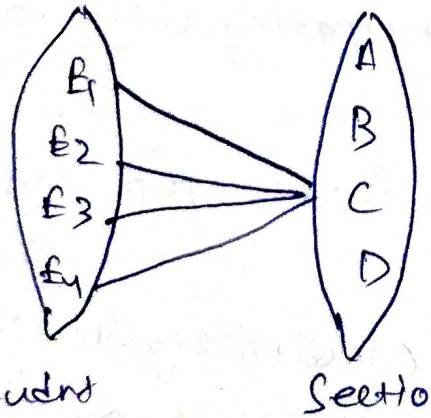
→ Ex- A student can have one roll no. only.

ii) Many to One

→ This mapping can be represented as n:1 / m:1.

→ In this mapping from set-1 there can be multiple entity, that make relationship with single entity of set-2.

→ Ex- There are many students belongs to one section.

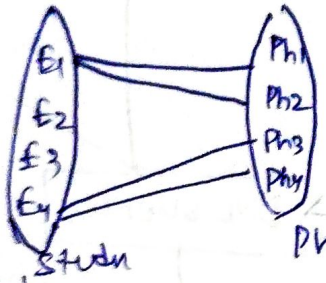


iii) One to Many

→ This mapping can be represented as 1:n / 1:m.

→ In this mapping from set-1 there can be maximum single set can make relationship with single or more than one entity of set-2.

→ Ex- One person can have multiple phone number, email.

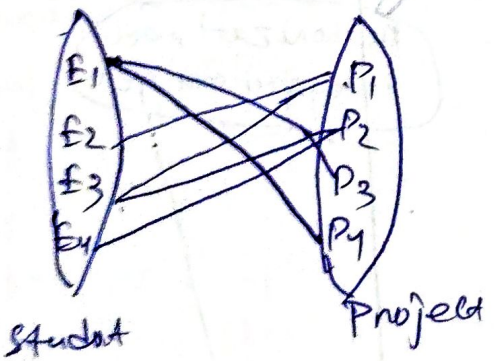


Many to Many:

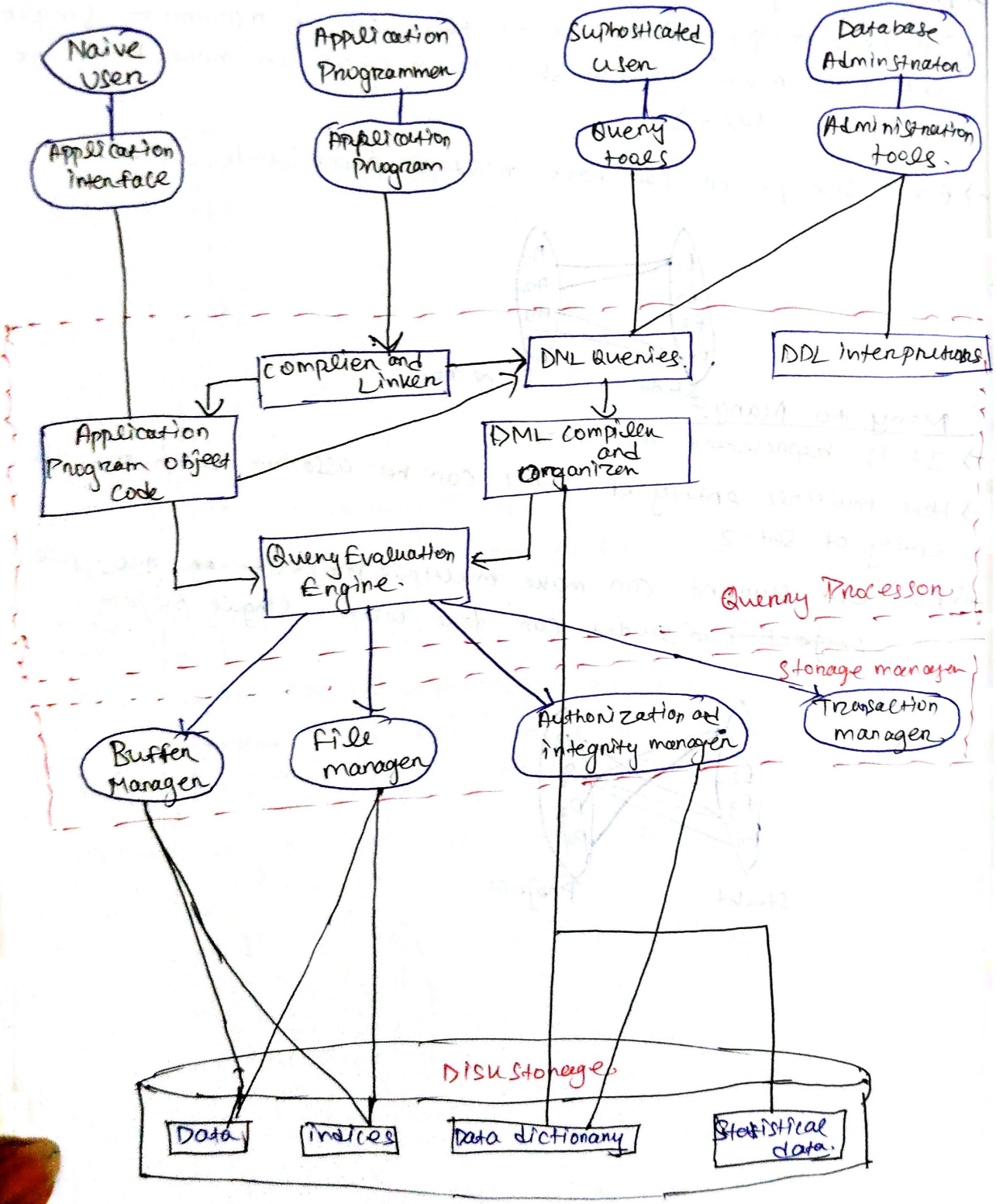
→ It is represented as m:n.

→ Here multiple entity of set-1 can be associate with multiple entity of set-2.

→ Ex- One student can make multiple project. or multiple ~~project can~~ student can done with single project.



Data base system architecture :-



Query P

-> Query P
access

-> The user

-> The int
via a

-> It also
DML c

Query

DDL int

-> It inte
table

DML (c

-> It tran
into S

-> A que
of al
result.
that is
the al

Compile

-> It pro
Program

Query e

-> It execu

Query Processon:-

-> Query Processon helps the database system, simplify and facilitate access to data.

-> The work of query processon is to execute the query successfully.

-> It interprets the request (queries) received from end user via an application program into instructions.

-> It also execute the user request which is received from DML compiler.

Query processon components:-

DDL interpreter:

-> It interprets the DDL Statement (like schema defination) into a table containing data (data about data).

DML Compiler:-

-> It translate the DML statements (like select, insert, update, delete) into low level instruction, so, that they can be executed.

-> A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs query optimization, that is it picks the lowest cost evaluation plan from among the alternatives.

Compiler and Linker:-

-> It processes DML Statements embedded in a application Program into procedural calls.

Query evaluation Engine:

-> It executes the low level instruction generated by DML compiler.

Storage Manager components :-

Authorization Manager and integrity Manager :-

→ It checks the authority of user to access data and checks the integrity constraints when the database is modified.

Transaction manager :-

→ It ensure that the database remains in a consistent state despite system failures, and that concurrent transaction executions proceed without conflicting.

File manager :-

→ It manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

Buffer manager :-

→ It responsible for fetching data from disk storage into main memory and deciding what data to cache in main memory.

Disk storage :-

→ The storage manager implements the following data structure as a part of physical system implementations.

Data file :-

→ It store the database itself.

Data dictionary :-

→ It contains the information about the structure of any database, or, it store metadata about the database, in a particular the schema of the database.

Indices :-

→ It provides faster retrieval of data item.

Statical data :-

→ It stores statical information about the data in database. This information is used by the query processor to select sefficient way to execute query.

Data models in database :-

→ Data models define how the logical structure of a database is modelled and defines how data will be stored, accessed, and updates in database system. ~~Data~~

→ Data model can be defined as an integrated collection of concepts for describing and manipulating data, relationship between data, and constraints on the data in an organization.

→ The purpose of data model is to represent data and to make the data understandable.

→ We have different types of data models they are widely used data models.

- Hierarchical Model
- Network model.
- Entity - Relationship model.
- Relationship model

1) Hierarchial Model :-

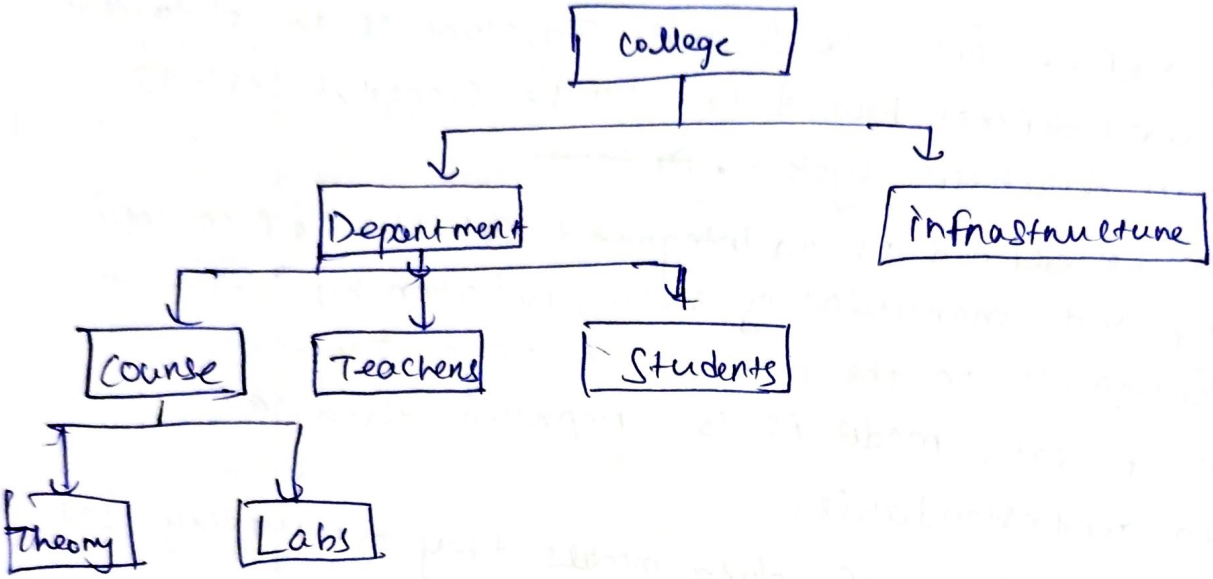
→ The Hierarchial model arranges records in hierarchy like organizational chart.

→ Each record type in this model is called node or segment.

→ A node represents a particular entity the top most node is called root. Each node is a subordinate of the node that is at the next higher level

→ A higher level node is called parent and lower level is called child. A parent node can have one or more child nodes, but child can have only one parent node.

→ This kind of structure is ~~of~~ often called inverted tree.



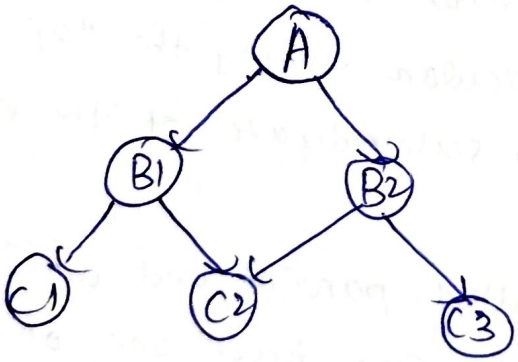
Drawbacks

→ Relationship that are complex are not supported.

→ Because it only supports one parent per child node, if we have a complex relationship in which a child node needs to have two parents, we won't be able to describe it using this model.

→ When a parent is removed, the child node is also removed.

2) Network Model



→ The network model is similar to hierarchial model, the data are organized like graph.

→ The difference is that child node can have more than one parent nodes.

→ The child nodes are represented by arrows in network model.

→ It also provides more flexibility than hierarchial model.

Drawbacks:-

→ It is slow complem and more difficult to maintain.
→ It requires more complem diagram to represent a database.

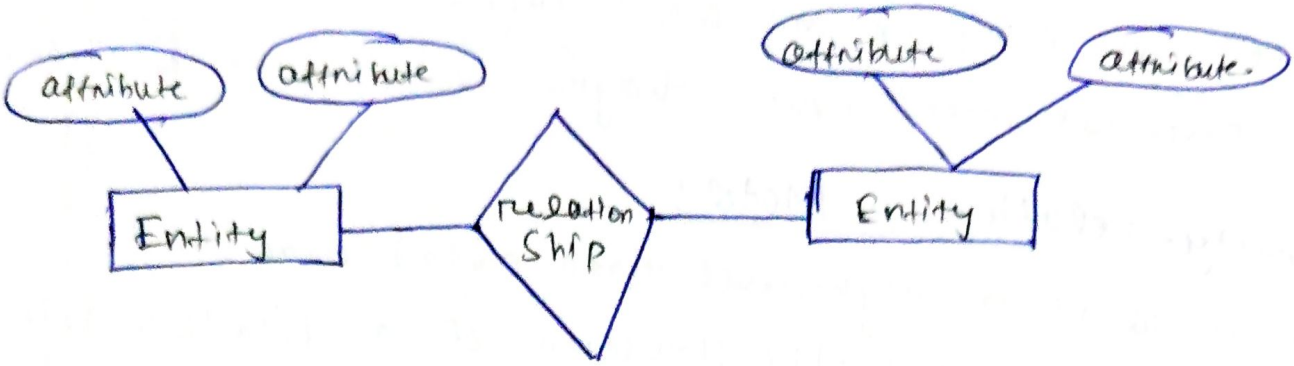
3) Entity-relationship Model:-

→ ER model is a high level data model diagram.
→ ER model describes the structure of a database with the help of a diagram, which is known as entity relationship Diagram (ER diagram).
→ An ER model is a design or blueprint of a database that can later be implemented as a database.
→ It is based on the notation of real world entities and relationship among them.

ER model follows 3 components

- Entity:- Entity is an real world things or object.
  → It can be a person, place or even a concept. It can be represented as rectangular shape
  Ex- Teacher, student, course, building
- Attributes:- An entity contains real world property called attribute.
  → This is the characteristics of that attributes.
  → This can be represented as eclipse
  → Ex- The entity teacher has property like teacher-id, name, salary, age etc.
- Relationship:- → Relationship tells how the entities are related.
  → Diamond or rhombus is used to represent the relationship.
  → Ex- Teacher works for department.

12/09/2025



4) Relational Model :-

→ This model was initially described by Edgar F. codd in 1969.

→ Most widely used model by commercial data processing applications.

→ It uses collection of tables for representing data and the relationship among those data.

→ Data is stored in tables called relations.

→ Each table is a group of column and rows where column represents attributes of an entity and rows represented records on tuples.

→ Attribute on field :- Each column in a relation is called an attribute. The values of an attribute should be from the same ~~domain~~ domain.

→ Tuples on record :- Each row in the relation called ~~row~~ tuple defines a collection of attribute values.
So each row in a relation contains values.

→ Each row has all the information about any specific individuals.

S-id	name	age
1	AKon	17
2	BKon	18
3	CKon	17
4	DKon	18

Sub-id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr C. Hangs
4	PHP	Mr. PHP

S-id	Sub-id	marks
1	1	98
1	2	78
2	1	76
3	2	88

28/7/25

ER Modeling Constructs :-

→ The basics of Entity-Realtionship modeling

1. Entities
2. Attributes
3. Relationships

1. Entity :-

→ An entity may be a 'thing' or 'object' with a physical existance - a particular person, car, house, or employee or it may be an object with conceptual existance - a company, a job, a university course.

→ Entity can be any thing that has an independent existance and about which we collect data - It is knowns as entity type.

→ Entities are represented by means of rectangles.

→ Entities have attributes that gives them identity.

→ Entities became table in relational model.

Entity set :-

→ A entity set is a collection of similar type of entities, that share same attributes.

→ Ex - A Student Set may contains all the student of a school.

2. Attributes:-

→ Attributes are the properties that define the entity type.

→ For example: Roll-no, Name, age, address, and phone-no are the attributes that define entity type Student.

→ For each attribute there is a set of permitted values, called domain of that attribute.

→ In ER diagram, the attribute is represented by an oval or ellipse.

Types of attributes:-

1. Key attribute
2. Composite attribute
3. Multivalued attribute
4. Derived attribute
5. Simple attribute

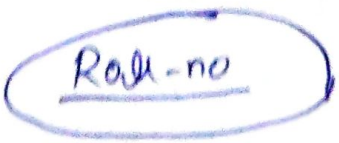
1) Key attribute:-

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

→ It represents a primary key.

→ Key attribute is represented by an ellipse with underlying lines

→ Ex. Roll-no



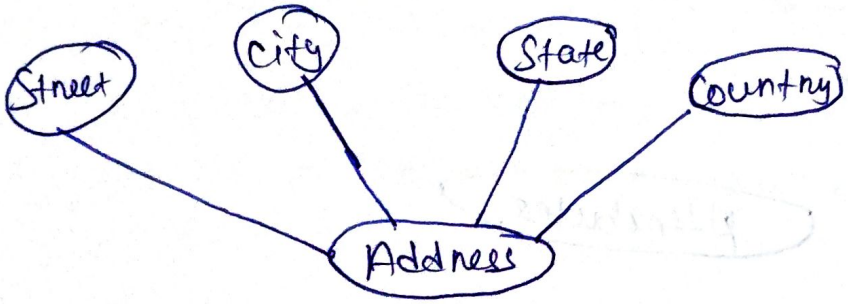
2. Composite Attribute :-

→ Composite attributes are made of more than one simple attri

→ A composite attribute is divided in a tree like structure.

→ Ex - A student's complete name may have first-name and last-name.

→ An address may have street, city, state, country and pin code.

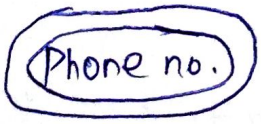


3. Multivalued datatype :-

→ Multi-valued attributes may contain more than one values.

→ Represented by double-ellipse

→ Ex - A person can have more than one phone number, email-address, etc.



4. Derived attributes :-

→ Derieved attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database.

→ Derived attributes are depicted by dashed ellipse.

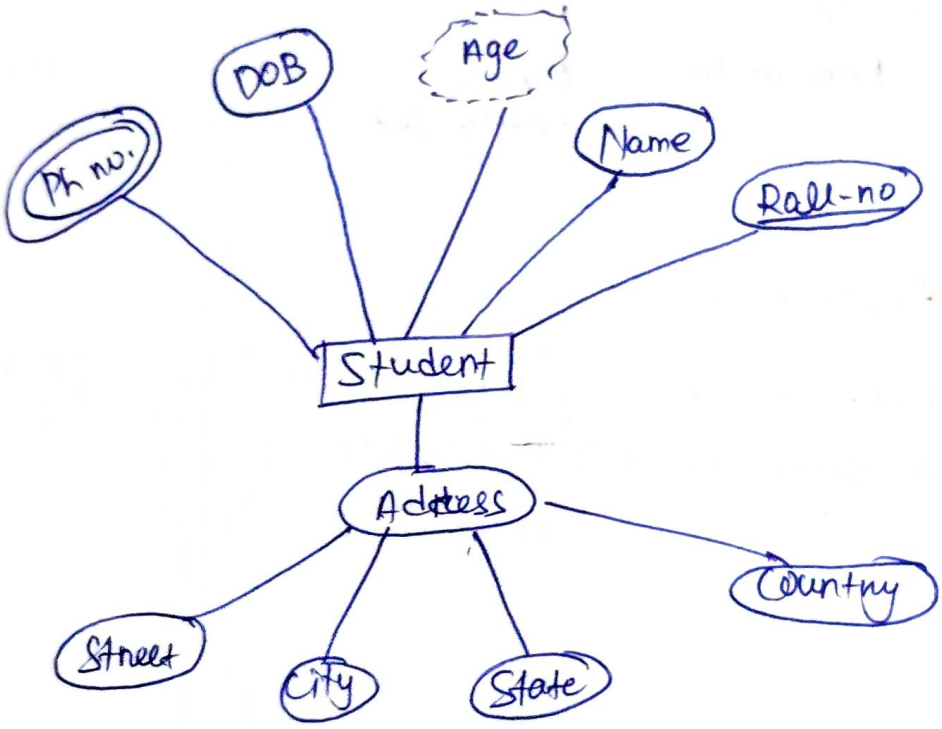
→ For example Age can be derived from date-of-birth.

5. Simple attribute :-

→ Simple attributes are atomic values, which cannot be divided fruther.

→ Ex- Gender, DOB.

ER Diagram Representation :-



3) Relationship :-

→ When an entity is related to another entity they are said to have a relationship.

→ A relationship is an association among entities.

→ Ex- An employee works-at a department.

An student enrolls in a course.

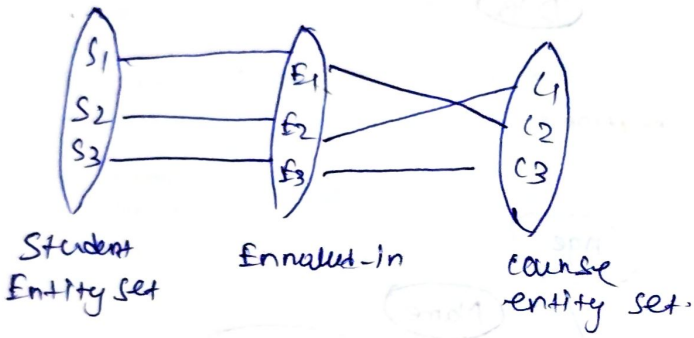
Here works_at and enrolls are called Relationship.

→ Relationship are represented by Diamond-Shape box.

Relationship Set :-

→ A set of similar type of relationship is called relationship set.

→ The following relationship sets Enrolls (E_1, E_2, E_3) depicts.



Degree of Relationship Set :-

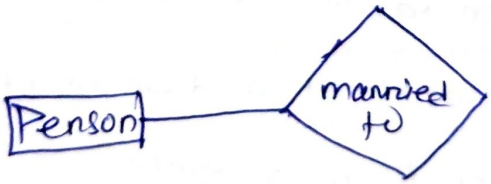
→ The no. of different entity sets participating in a relationship set is called degree of relationship set.

1) Unary
2) Binary
3) N-ary

1) Unary Relationship (degree = 1)

→ A unary relationship is only one entity set participate. in a relationship, the relationship is called as unary relationship.

→ Ex - One person is married to only one person.



ii) Binary

→ A binary relationship

→ Ex - Stu

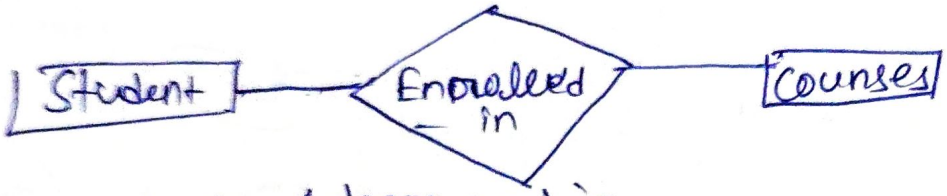
iii) N-ary

→ When

ii) Binary Relationship (degree = 2) :-

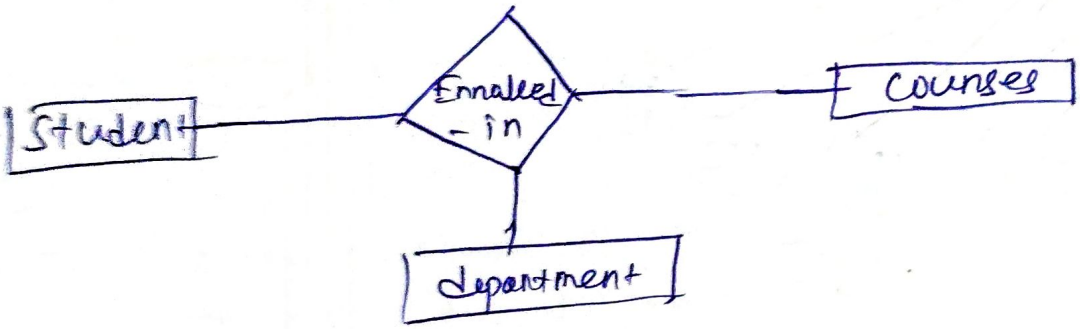
→ A binary relationship is when two entities participating in a relationship and is the most common relationship degree.

→ Ex- Student is enrolled in course.

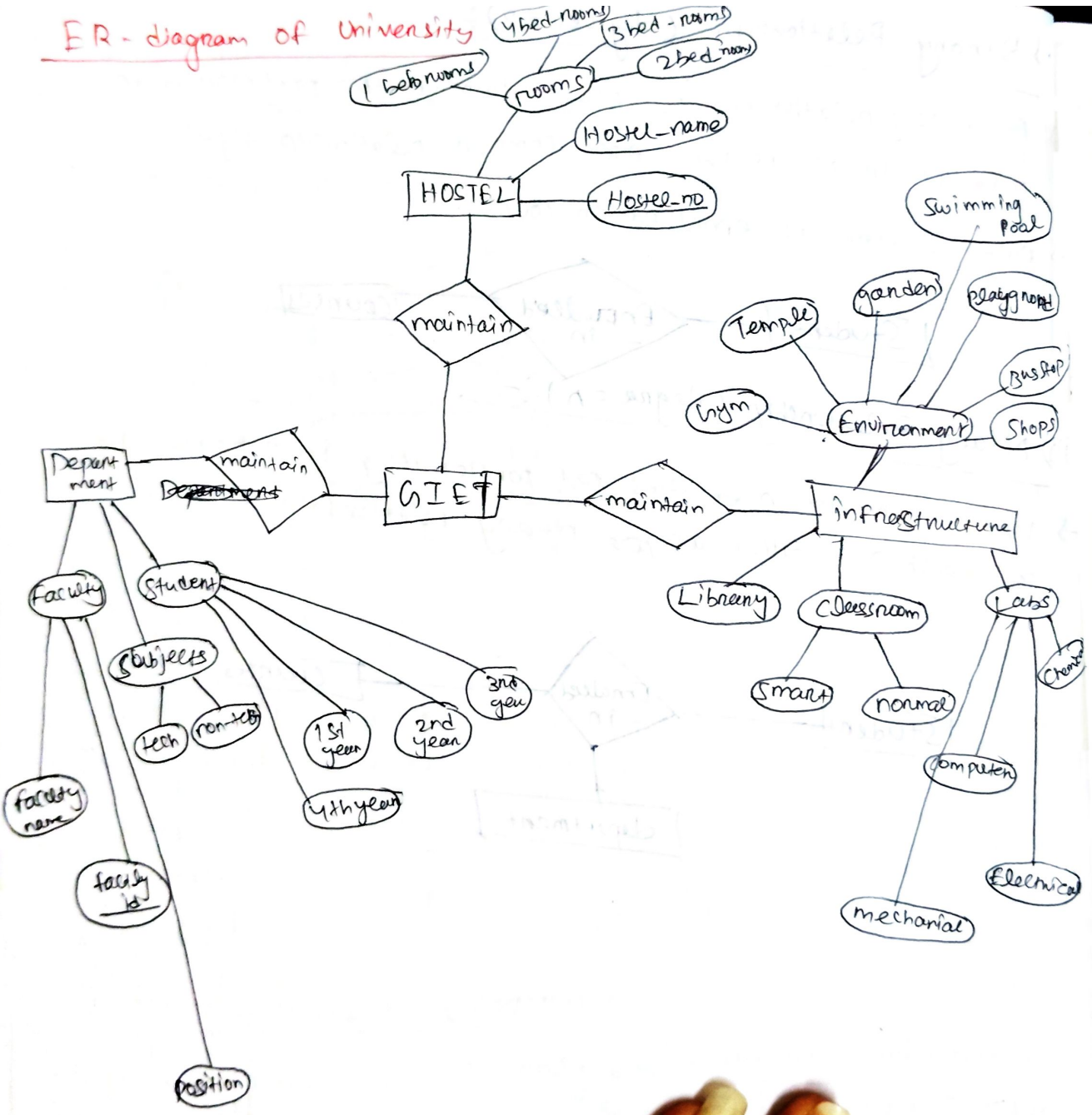


iii) N-ary Relationship (degree = n) :-

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



ER - diagram of University



Participat

→ It is ap

→ It has

i)

2)

1) Total

→ Each e

relations

2) Partia

→ The en

the re

→ Ex - I

Pa

Key

→ A ke

Participation Constraint:-

-> It is applied on the entity participate in relationship set.

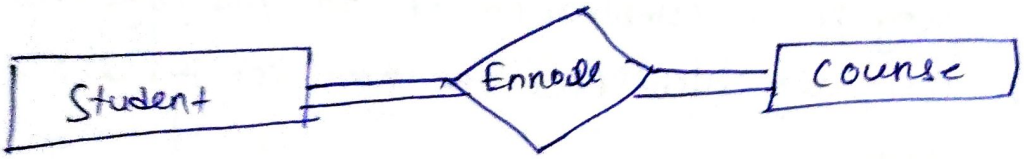
-> It has 2 types.

i) Total participation

2) Partial participation.

1) Total participation:-

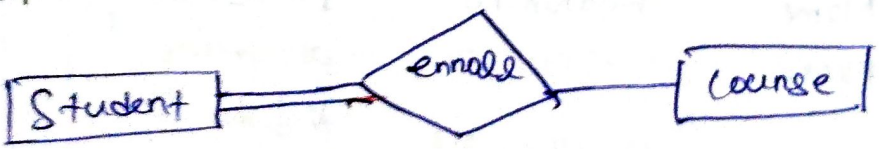
-> Each entity in the entity set must be participate in the relationship.



2) Partial participation:-

-> The entity in the entity set may or may not participate in the relationship.

-> Ex- If some course is not enroll by all the student, then Participation of course is partial.



Keys:-

-> A key is an attribute or set of attributes that uniquely identifies any record (or tuple) from the table.

-> key is used to uniquely identify any record or row of data from the table.

-> It is also used to establish and identify relationship between table.

Types of keys in DBMS

1. Super key
2. Candidate key
3. Primary key
4. Alternate key
5. Foreign key
6. Composite key

1. Super key :-

-> A Super key is a combination of all possible attributes that can uniquely identifies the rows (or tuple) in the given relation

-> A super key is superset of candidate key.

-> A table can have many super key.

-> A super key may have additional attribute that are not needed for unique identity.

Emp_Id	Name	Aadhar-no	Email-id	Dept-no
1	Akash	778942312546	a@gmail.co	1
2	Namita	123456789012	b@gmail.com	2
3	Namita	234567890123	c@gmail.com	2
4	Vimal	345678910 34	d@gmail.com	3

Super keys:

1. {Emp_Id}
2. {Aadhar-no}
3. {Email-id}
4. {Emp-Id, Aadhar no}
5. {Aadhar-no, Email-id}.
6. {Emp-Id, Email-Id}. etc ...

2. Candidate key:

-> A candidate key is an attribute or set of attribute which can uniquely identify a tuple.

-> A candidate key is a Minimal super key, or a super key with no redundant attributes.

-> Candidate keys are defined as distinct set of attributes from which primary key can be selected.

-> candidate key are not allowed to have NULL values.

-> Ex- {Emp-id}
{Email-id}
{Aadhar-no}.

3. Primary key:-

-> A primary key is one of the candidate key chosen by the database designer to uniquely identify the tuple in the relation.

-> Ex- {Emp-id}

4. Alternate key:-

-> Out of all candidates keys, only one gets selected as primary key, remaining keys are known as alternate keys.

-> Ex- In the employee table
Emp-id is best suited for the primary key, rest of the attributes like aadhar-no and email-id are considered as alternate key.

5. Foreign keys :-

→ A foreign key is used to link two tables togather.

→ An attribute on set of attribute in one table that refers to the primary key in another table.

→ The purpose of the foreign key is to ensure refrential integrity of the data.

→ Ex.

Employee

E-id	Name	email	Dept no (f.K.)
1			1
2			2
3			2
4			3
5			3
6			

Department

Dept no (P.K)	D-name
1	x
2	y
3	z

6. Composite key :-

→ A key that has more than one attributes is known as composite key. It is also known as compound key.

→ Ex- { Student-id, Subject-name }.

Student

S-id	Name	Sub-name	mark
101	x	Java	89
102	y	Python	79
101	x	C++	69
103	z	DS	89

Composite key

Types of entity:-

1. Strong entity:-

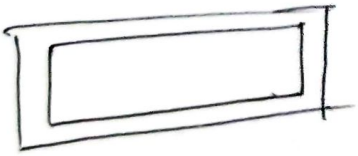
-> A strong entity is a type of entity that has a key attribute.
-> Strong entity does not depend on other entity in the schema.
-> It has a primary key, that helps in identifying it uniquely.
-> It is represented by a rectangle, These are called Strong entity

->

2. Weak entity:-

-> The weak entity does not have a sufficient attributes to form a primary key, i.e weak entity do not have a primary key.
-> A weak entity is dependent on a strong entity to ensure its existance.
-> Weak entity is represented by double rectangle.

->



Extended ER Model:-

→ The Extended Entity-Relationship (EER) model is a conceptual (or semantic) data model, capable of describing the data requirements for a new information system in a direct and easy to understand graphical notation.

→ As the complexity of database increases day by day, we need to use Extended ER model to represent it.

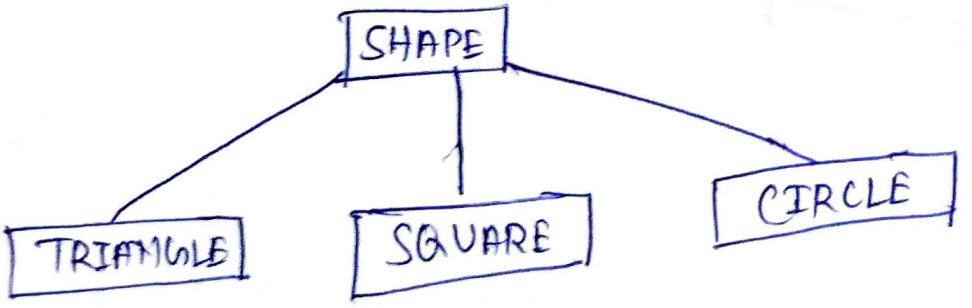
Concepts used in EER Model:-

EER Model uses the concept of,

→ Subclass
→ Superclass
→ Specialization
→ Generalization
→ Aggregation

① Subclass and Superclass

→ Subclass are the group of entities with some unique attributes. Subclass inherits the properties and attributes from super class.



② Generalization:-

→ Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it.

→ Generalization is a "bottom-up approach (Starts with entities and moves upward to a general one)" in which two or more entities can be combined to form a higher level

entity if they have some attributes in common.

→ Generalization is used to emphasize the similarities among lower-level entity set and to hide differences in the schema.

→ Car, truck, bus.

③ Specialization :-

→ Specialization is opposite of Generalization.

→ In specialization an entity is broken down into sub-entities based on their characteristics.

→ Specialization is "Top-down approach". where higher level entity is specialized into two or more lower level entities.

→ Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.

→ Specialization can be repeatedly applied to define the design of schema.

→ Employee.

④ Inheritance :-

→ Inheritance is an important feature of generalization and specialization.

→ Attributes inheritance allows lower level entities to inherit the attribute of higher level entities.

→ This also extends to participation inheritance in which relationship involving higher level entity set are also inherited by lower-level entity sets.

5) Aggregation:-

→ Aggregation is used when we want to treat a relationship itself like an entity. So that it can be participate in another relationship.

→ Aggregation is used when we need to express a relationship among relationships.

→ Aggregation is abstraction through which relationship are treated as higher level entities.

→ Aggregation is a process when a relationship between two entities is considered as a single entity has a relationship with another entity.

→ Employee works_on project ⟶ (relationship: works_on).

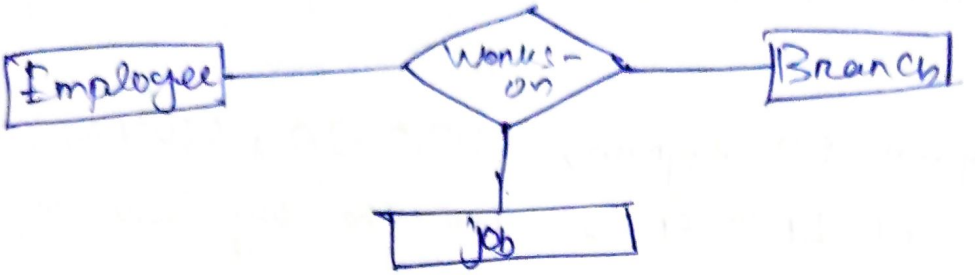
→ Now, suppose a Manager supervises the whole works_on relationship.

→ Here we aggregate (Employee + works_on + project) into a single unit, and then connect it with Manager using another relationship.

Example

→ Basic E-R model can't represent relationship involving other relationships.

→ Consider a tennary relationship works_on between employee, branch and job.



→ Suppose we want to assign a manager for jobs performed by an employee at a branch.

Reduction of ER diagram to Table / Schema :-

→ The database can be represented using the notations and these notation can be reduced to a collection of tables.

→ In the database every entity set on relationship set can be represented in tabular form.

Rules :-

→ There are some points for converting the ER diagram to table :

- Entity type becomes a table.

→ In the given ER diagram Lecturer, Student, Subject and course forms individual tables.

- All single valued attribute becomes a column for the table.

→ In the STUDENT entity, STUDENT-NAME, and STUDENT-ID form the column of STUDENT table. Similarly, COURSE-NAME and COURSE-ID form the column of COURSE table so on.

- A key attribute of the entity type represented by the primary key

→ In then given ER diagram, COURSE-ID, STUDENT-ID, SUBJECT-ID and LECTURE-ID are the key attribute of the entity.

- The multivalued attribute is represented by a separate table

→ In the Student table, a hobby is multivalued attribute. So, it is not possible to represent multiple values in a single column of STUDENT table. Hence we create a table STUD-HOBBY with column name STUDENT-ID and HOBBY. Using both the column, we create a composite key

- Composite attribute represented by components.

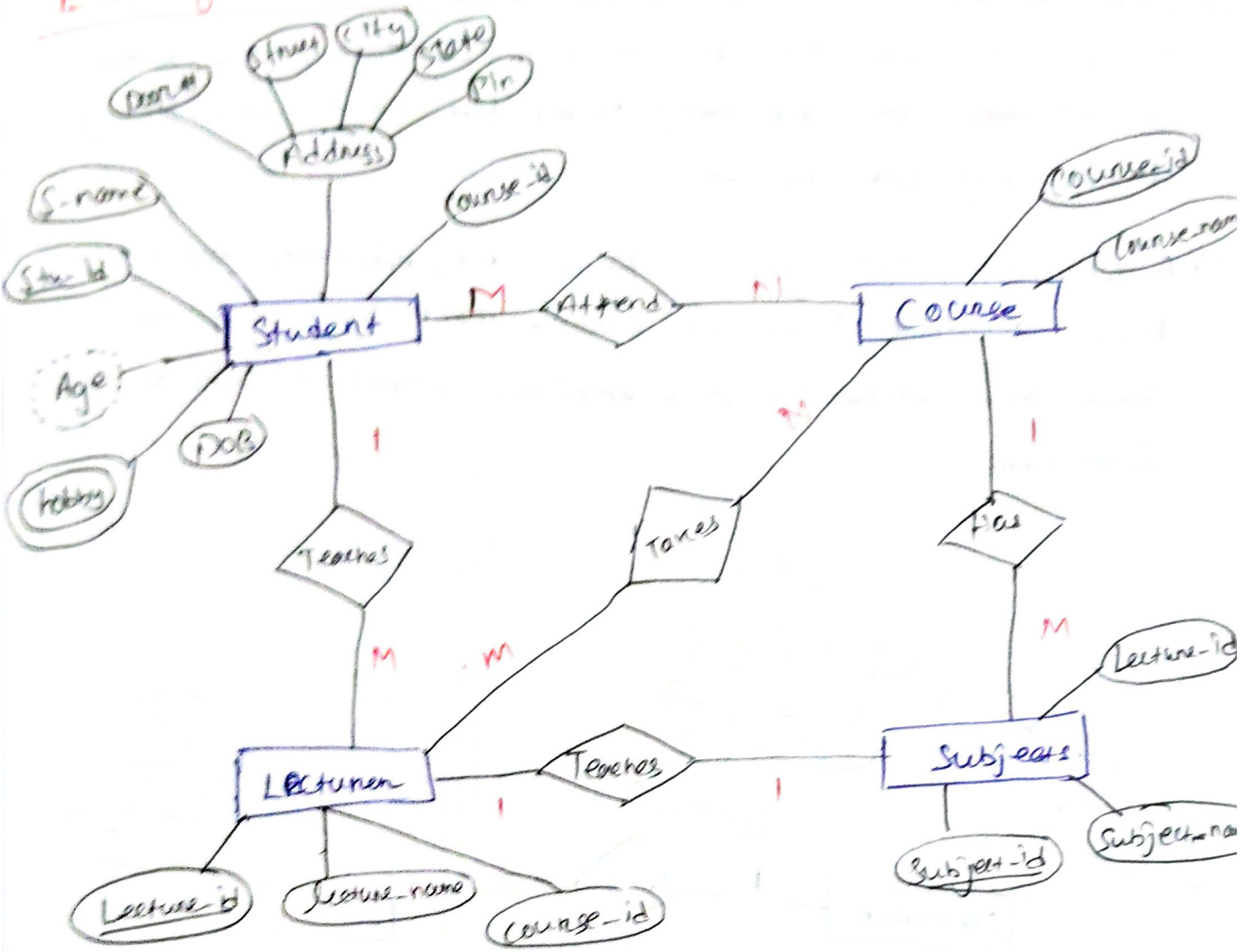
→ In the given ER diagram, Student address is a composite attribute, It contains CITY, PIN, DOOR#, STREET, and STATE. In the student table, these attributes can merge as an individual column.

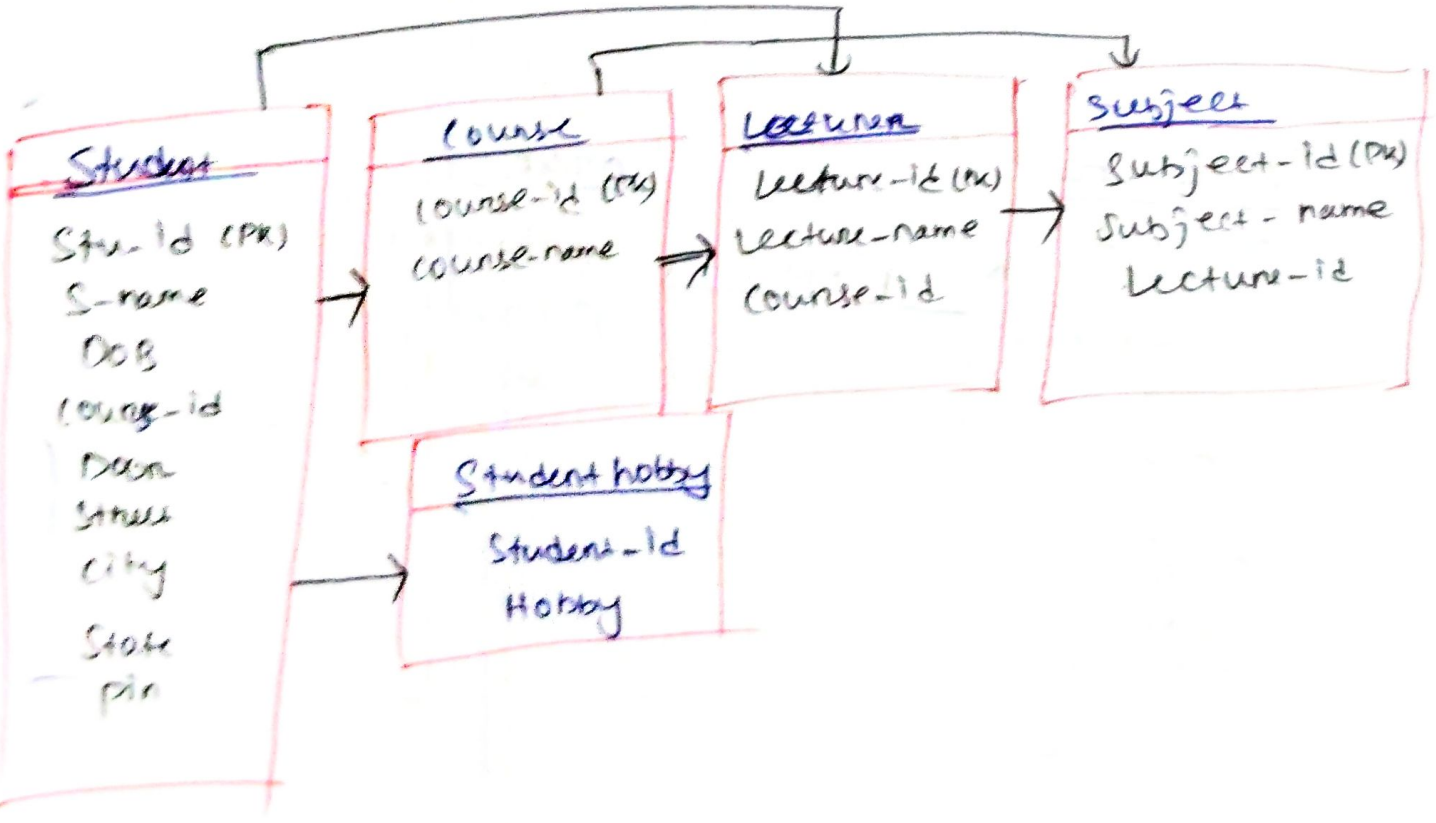
- Derived attributes are not considered in the table.

→ In the STUDENT table, Age is the derived attribute.

→ It can be calculated at any point of time by calculating the difference between current date and Date of birth.

ER diagram of University



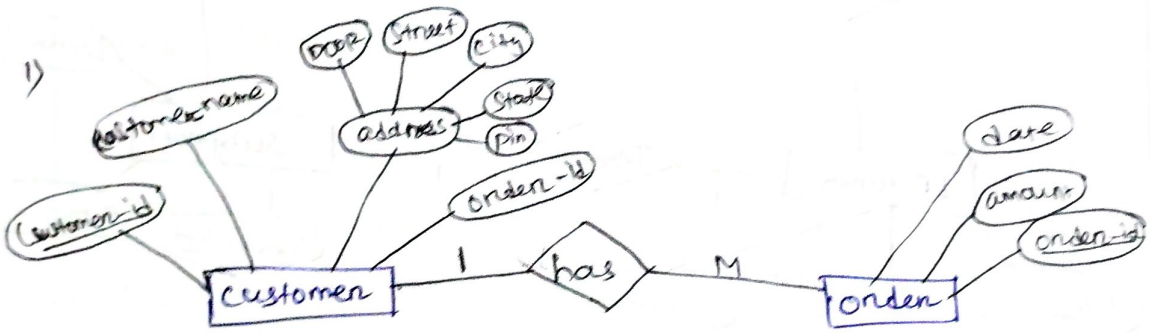


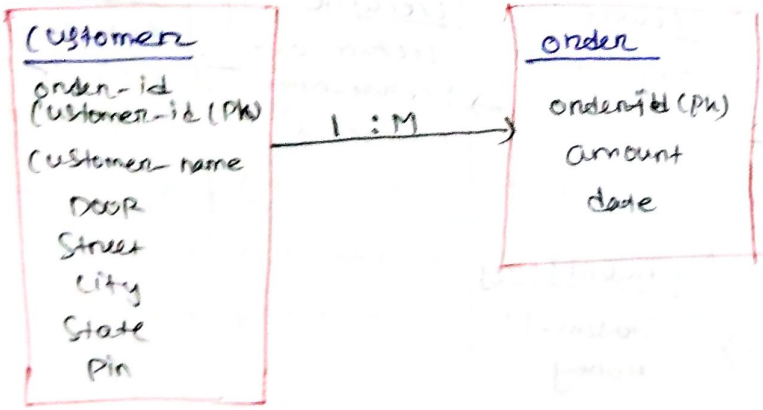
a) Draw an ER diagram of customer having attribute Id, name address, and order having attributes id, amount, date where a customer can place many orders but each order belongs to exactly one customer.

b) A department has attribute having Id, dep-name, and emp having attribute id, name, salary each department can have many employees, each employee works in only one department.

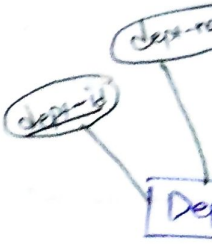
Ans

1)



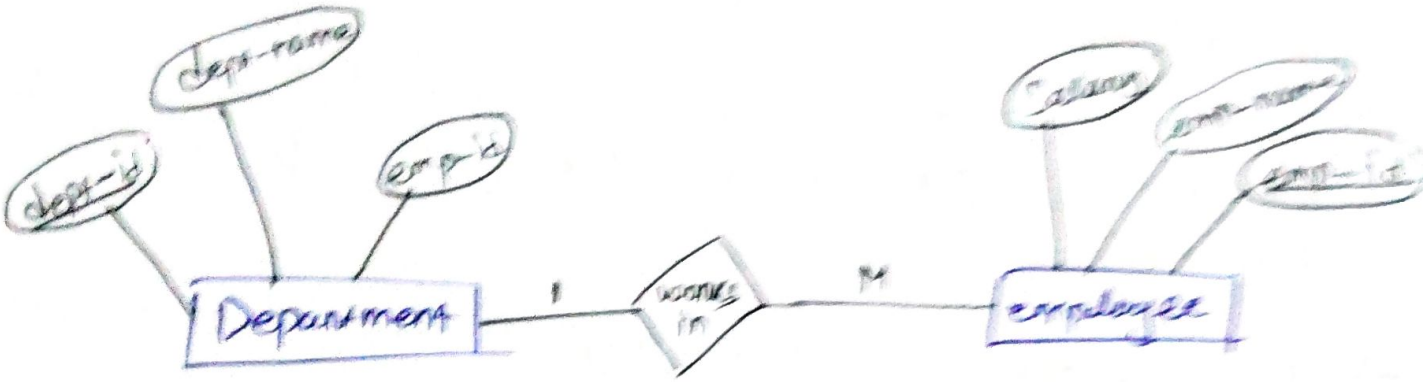


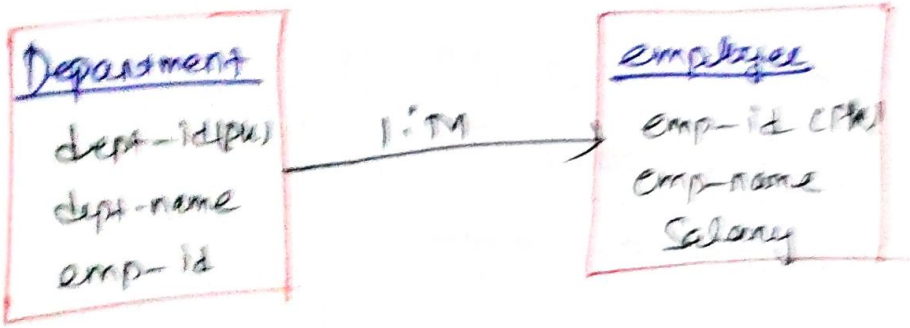
2



Departmen
dept-id
dept-nam
emp-id

(2)


dept-name
dept-id
emp-id
Department
1
works in
M
employee
salary
emp-name


Department
dept-id(pk)
dept-name
emp-id
1:M
employee
emp-id (PK)
emp-name
Salary

Relational Algebra

Relational Query Language:-

→ Relational database systems are expected be equipped with a query language that assist the users to query the database instance.

→ Query language is a language in which users requests information from the database.

→ There are 2 types of query Language.

- Procedural query language.
- Non-procedural query language.

1. Procedural Query Language:-

→ In procedural query language, user instructs the system to perform a series of operations to produce the desired results.

→ User tells what data to be retrived from database and how to retrive it

→ Ex - Relational Algebra

2) Non-procedural Query Language:-

→ In non-procedual query language, user instructs the system to produce the desired results without telling the step by step process.

→ User tells what data to be retrived from database and does not tell how to retrive it

→ Ex - Relational calculus.

Relational Algebra:-

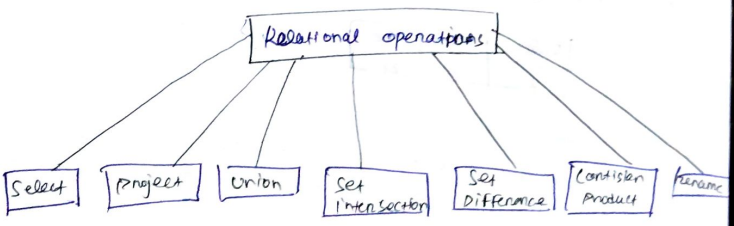
Definations:-

→ Relational Algebra is a procedural query language.

→ It gives a step by step process to obtain the result of the query

→ It uses operations to perform queries.

Types of relational operations:-



1. Project operation:-

→ This operation shows list of those attributes that we wish to appear in the result, rest of the attributes are eliminated from table; it is denoted by Π.

→ Notation:- $\Pi A_1, A_2 \dots A_n (R)$

→ Ex- Π Rollno, name (student)

2) Select operations:-

→ The select operation select tuples that satisfy a given predicate / condition.

→ It is denoted by Sigma (σ)

→ Notation : $\sigma_{P(R)}$

Rollno	Name	Age
1	A	20
2	B	21
3	A	24

-> Ex- Retrive all the details whose roll no is 2.

A- $\sigma_{rollno=2}(\pi_{rollno, age, name}(Student))$

Rollno	name	age
2	B	21

3) Union operation :-

-> Suppose there are two relations R and S. The union operation contains all the tuples that are either in R or S or both in R and S.

-> It eliminates the duplicate tuples.

-> It is denoted by $\cup$.

-> Notation $R \cup S$.

-> A union operation must hold the following condition:

a) R and S must have the same number of attribute.

b) Duplicate tuples are eliminated automatically.

Deposition Relation

Customer name	Acc-no
Ram	A-101
Shyam	A-123
hari	A-131
~~Ram~~ hari	A-143
Ram	A-231
Sai	A-234

Borrowen Relation

customer name	loan-no
~~Ram~~	L-501
Sritam	L-201
~~Radha~~ Sai	L-321
Sritam	L-526
Raja	L-354
Jaga	L-768

Ex- $\pi_{customername}(Deposition) \cup \pi_{customername}(Borrowen)$

customer name
Ram
Shyam
hari
Semin
Sai
Sritam
balu
joga
Raja

4) Intersection Operation :-

-> Suppose there are two relations R and S. The set intersection operation contains all tuples that are in both R and S.

=> It is denoted by $\cap$.

-> Notation : $R \cap S$

-> we can use Deposition and Borrowen table for this operation.

-> Ex- $\pi_{customername}(Deposition) \cap \pi_{customername}(Borrowen)$

customer name
Ram
Sai

5) Set difference :-

-> Suppose there are two relations R and S. The set intersection operation contains all tuples that are in R but not in S.

-> It is denoted by intersection minus (—).

-> Notation : R-S

-> Using the above deposition table and Borrowen table

Ex- $\Pi_{customername}(Deposition) - \Pi_{customername}(Borrowen)$.

Customer name
Shyam
Rani
Hari

6) Cartesian Product

-> The Cartesian product is used to combine each row in one table with each row in the other table.

-> It is also known as cross product.

-> It is denoted by X.

-> Notation E X D.

Employees

empid	E-name	Dept
1	Smith	A
2	Henry	B
3	John	C

Department

Dept no	Dept-name
A	marketing
B	Sales
C	Legal

-> Ex- Employee X Department

empid	e-name	dept	dpptno	Dept-name
1	Smith	A	A	marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Henry	B	A	marketing
2	Henry	B	B	Sales
2	Henry	B	C	Legal
3	John	C	A	marketing
3	John	C	B	Sales
3	John	C	C	legal

Rename Operation :-

-> The rename operation is used to rename the output relation

-> It is denoted by rho (ρ).

-> Ex- we can use the rename operation to rename Student relation to Student1.

-> ρ (Student1, Student).

Branch (branch-name, branch-city, assets)
Customer (cus-name, Street, city)
Account (acc-no, branch-name (FK), balance)
Loan (loan-no, branch-name (FK), amount)
Deposition (customername (FK), acc-no (FK))
Borrowen (cus-name (FK), loan-no (FK)).

① Find all the loan made at gunupur branch.

② Find all the loans over 2 thousand.

③ find all the tuple who have taken loan of more than 2000 and made ~~at~~ gunupur branch.

④ Find all the loan_no and amount of the loan.

⑤ Find the loan_no for each loan of an amount greater than 3000.

⑥ find the name of all customer who have a loan, an account. or both from the bank.

⑦ find the name of all customer who have a loan and account in bank.

⑧ Find the name of all customer who are having an account but not taking any loans from the bank.

⑨ Find the name of all customer who have a loan at gunupur branch.

Ans

① Π loan-no (Loan)

② σ branch-name = "gunpur" (Loan)

③ σ amount >= 2000 (Loan)

④ σ amount >= 2000 and branch-name = "gunupur" (Loan)

⑤ Π loan-no, amount (Loan)

⑥ $\sigma_{amount >= 3000}(\Pi$ loan-no (Loan))

⑥ $\Pi_{customer\_name}$ (Depositer) $\cup$ $\Pi_{customer\_name}$ (Borrower)

⑦ $\Pi_{customer\_name}$ (Depositer) $\cap$ $\Pi_{customer\_name}$ (Borrower)

⑧ $\Pi_{customer\_name}$ (Depositer) $-$ $\Pi_{customer\_name}$ (Borrower)

~~④ $\sigma_{branch\_name = "gunupur"}(\Pi_{customer\_name}$ (Borrower $\times$ Loan))~~

~~⑨ $\sigma_{Borrower.loan\_no = Borrower.loan\_no}$~~

⑨ $\Pi_{customer\_name}(\sigma_{branch\_name = "gunupur"}(\sigma_{Borrower.loan\_no = Loan.loan\_no}$ (Borrower $\times$ Loan)))

Joins:-

→ It is a binary operation which takes 2 relation as an input.

→ The join operation will check the key attribute is present in both the relation or not, if present it will be retrive the required information from both the tables by comparing the value and common key attribute.

→ It is denoted by $\bowtie$.

→ A join operation combine related tuples from different relation. if and only if a join condition is satisfied.

→ Its two types

① Inner join
- Theta join (θ)
- Equi join (=)
- Natural join ($\bowtie$)

② Outer join
- Left outer join ($\bowtie$)
- Right outer join ($\bowtie$)
- Full outer join ($\bowtie$)

1) Inner join :-

⇒ An inner join includes these tuples that satisfy the matching criteria, while rest of the tuples are excluded.

a) Theta join / Conditional Join :-

→ A join operation with such general condition is called Theata join / conditional join.

→ $<, >, <=, >=, ==, !=$

S1

Sid	Name	Age
22	x	45
31	y	55
58	z	35

R1

sid	bid	Date
22	101	1818125
58	103	19/8/25.

S1 X R1

Sid	Name	Age	sid	bid	Date
22	x	45	22	101	1818125
22	x	45	58	103	19/8/25
31	y	55	22	101	1818125
31	y	55	58	103	19/8/25
58	z	35	22	101	1818125
58	z	35	58	103	19/8/25

S1 $\bowtie_c$ R1

S1 $\bowtie_{S1.Sid < R1.sid}$ R1

S1.Sid	Name	Age	R1.Sid	bid	Date
22	x	45	58	103	19/8/25
31	y	55	58	103	19/8/25

b) Equi join (=) :-

→ when the join condition involve with equality comparision then, it is known as equi join.

→ S1

→ S1 ⋈ R1

S1 $\bowtie_{S1.sid = R1.sid}$ R1

S1.Sid	Name	Age	R1.Sid	bid	Date
22	x	45	22	101	1818125
58	z	35	58	103	19/8/25.

c) Natural join (⋈) :-

→ Natural join can only be perform, if there is atleast one common attribute that exist between two relation.

→ In addition the attribute must have same name and domain.

→ Natural join doesnot use any comparison operation.

→ It is same as equi join, which occurs implicitly by comparing all the common attributes in both the relation, but difference is natural join common attribute appears only once. The resulting schema will be change.

→ A ⋈ B Notation.

→ The result of natural joint is set of all combination of tupples in two relations A and B that are equal on their common attribute name.

S1 ⋈ R1

Sid	Name	Age	bid	Date
22	x	45	101	1818125
58	z	35	103	19/8/25

2) Outer join:-

→ The outer join operation is an extension of joins operation, it returns all the records from both tables that satisfies join condition.

→ In other word these join not only ~~thes~~ returns the matching records but also returns all unmatched rows from one or both table.

a) Left outer join (⟕):-

→ The left outer join retrive all the records from left hand table and matching row from ~~from~~ right hand table

→ It will return null when no matching record is find in right hand table

→ Cross Product is important for all joins during exam.

Employee

Ename	Street	City
Ram	N.V line	Mumbai
Shyam	H.V line	Kolkata
Ravi	A.V line	Delhi
Hari	L.V line	Goa

worker

Ename	Branch	salary
Ram	infosys	10000
Shyam	wipro	20000
Kabir	HCL	30000
Hari	TCS	40000

→ Ex- Employee ⟕ worker

Ename	Street	city	Branch	Salary
Ram	N.V line	mumbai	infosys	10000
Shyam	H.V line	kolkata	wipro	20000
Ravi	A.V line	Delhi	~~Null~~	~~Null~~
Hari	L.V line	Goa	TCS	40000

b) Right outer join (⟖):-

→ The right outer join retrive all the records from right hand table and matching row from left hand table.

→ It will return null when no matching record is found in left hand table

→ Ex- Employee ⟖ worker

Ename	Branch	salary	Street	city
Ram	infosys	10000	N.V line	mumbai
Shyam	wipro	20000	H.V line	kolkata
~~Kabir~~	HCL	30000	~~Null~~	Null
Hari	TCS	40000	L.V line	Goa

c) Full outer join (⟗)

→ The full outer join returns that include all the rows from both the tables.

→ The column of right hand table null, when no matching record are found in left hand table and vice versa

→ Ex- Employee ⟗ worker

~~Ename~~	~~Street~~ ~~Branch~~	salary	~~Branch~~ ~~street~~	city
Ram	N.V line	10000	infosys	Mumbai
Shyam	H.V line	20000	wipro	kolkata
Ravi	A.V line	~~Null~~	Null	Delhi
Hari	L.V line	40000	TCS	Goa
Kabir	Null	30000	HCL	Null

Division operation (÷ /)

→ The division operation in relation algebra find all the value in one table that are related to every value in another table

→ The division operation in relational algebra is a binary operation used to retrive tuples from one relation (dividend) that are associated with all or every tuples in another relation.

→ R1 ÷ R2 is possible if and only if R2 subset of R1 (R2 ⊂ R1)

→ R1 ≠ R2 and every attribut of R2 should be present in R1.

→ In result, the relation return by division operation have attributes = all attributes of R1 − all attributes of R2

Student (R1)

S-name	coursenam
Tom	DBMS
John	DS
Tom	DS
Tom	CN
John	DBMS
amy	CN
amy	DBMS
amy	DS

Course (R2)

coursename
DBMS
DS
CN

→ Ex- Student (R1) ÷ course (R2)

S-name
Tom
amy

Relational Calculus:

→ Relational calculus is a non-procedural language, it uses mathematical predicate calculus on first order logic, instead of algebra.

→ The relational calculus tell what to do but never explain How to do.

Types of Quantifier:

i) Universal Quantifiers (∀) is read as for all means in a given set of tuples all the tuples satisfies a given condition

ii) Existential Quantifiers (∃) is read as there exist, which means in a given set of tuples, atleast one operands should satisfies the condition.

Types of Calculus:

→ It is of 2 types
1) Tuple Relational calculus.
ii) Domain Relational calculus

1) Tuple relation calculus : (TRC)

→ It is specified to select the tuples in a relation.

→ In TRC filtred variable uses the tuple of a relation.

→ The result of relation can have one or more tuples

$$\{ t \mid P(t) \}$$

P(t) → Predicate / condition

t → Resiet tuple

It is a set of tuple 't' such that predicate P is true for t. where 't' is resulting tuples, P(t) denotes Predicate on condition used to fetch the tuples.

Predicate Calculus formulas:

a) Set of attribute constraint
b) Set of comparision operators (<, >, <=, >=)
c) Set of connectivities (∨, ∩, ¬)
d) Implication (x => y)
e) Quantifiens (∀, ∃)

Branch(Branch-name, city, assets)

customer (customer-name, street, city)
account (acc-no, branch-name, balance)
loan (loan no, branch-name, amount)
Depositer (customer-name, acc-no)
Borrower (customer-name, loan no)

① Find the loan no, branch-name, amount for all the loan over 2000.

A- $\{ t \mid t \in loan \cap t[amount] > 2000 \}$

② find the loan no of each loan of amount more than 2000.

A- $\{ t \mid \exists s \in loan \; t[loan\,no] = s[loan\,no] \cap s[amount] > 2000 \}$

③ Find the name of customer who have a loan at gunpur branch.

A- $\{ t \mid \exists b \in borrower \; (t[customer\_name] = b[customer\_name]) \cap \exists s \in loan \cap l[branch\_name] = "gunpur" \cap (l[loan\_no] = b[loan\_no]) \}$

④ Find the name of all customer having a loan, account or both at the bank

A- $\{ t \mid \exists b \in Borrower \; (t[customer\_name] = b[customer\_name]) \vee \exists d \in Depositer \cap (d[customer\_name] = b[customer\_name]) \}$

2) Domain Relational calculus :- (DRC)

-> Domain Relational calculus is non-procedural query language.
-> In DRC filtering variable uses the domain of attributes
-> Domain relational calculus uses same operator as TRC.
-> Domain relational calculus uses logical connectivity (∨, ∩, ¬) and Quantifiens (∀, ∃)

$\{ \langle a_1, a_2 \cdots a_n \rangle \mid P \langle a_1, a_2, \cdots a_n \rangle \}$

$\langle a_1, a_2 \cdots a_n \rangle$ → Resulting tuple
$a_1, a_2, \cdots a_n$ → Domain Variables

-> P represents a predicate similar to predicate calculus
$a_1, a_2 \cdots a_n$ are the relation on n no of attributes. a_1, a_2 upto an domain variable and domain constraint

① Find the loan no, branch-name, amount for loan over 2000.

A- $\{ \langle l, b, a \rangle \mid \langle l, b, a \rangle \in loan \cap a > 2000 \}$

② Find the loan no of each loan of amount more than 2000.

A- $\{ \langle l \rangle \mid \exists b, a \; \langle l, b, a \rangle \in loan \cap a > 2000 \}$

③ find the name of all customer who have a loan of 2000

A- $\{ \langle c \rangle \mid \exists l, b, a \; \langle c, l \rangle \in Borrower \cap \langle l, b, a \rangle \in loan \cap a > 2000 \}$

(branch name taken, from loan table)

③ find the name of the customer who have a loan at gunupur branch.

A- $\{ \langle c \rangle \mid \exists l, b, a \; \langle c, l \rangle \in Borrower \wedge \langle l, b, a \rangle \in loan \wedge b = \text{"gunupur"} \}$

④

UNIT-3 Functional Dependency :-

→ Functional dependency is a relationship that exist between two attributes. on two set of attributes.

→ Functional dependency ~~between~~ exist between primary key ~~and~~ or key attribute and non-key attribute.

$$x \longrightarrow y$$

where

x = determinant

y = depended.

Rule :-

If $t_1.x = t_2.x$

then

$t_1.y = t_2.y$

Ex- Student

Rollno	Name	marks	Dept	course
1	a	78	CS	c_1
2	b	60	EE	c_2
3	a	78	CS	c_2
4	b	60	EE	c_3
5	c	80	IT	c_3
6	d	80	ECE	c_4
10				

1) Rollno → Name (exist)
2) Name → Rollno. (not exist)
3) Rollno → marks (exist)
4) Dept → course (not exist)
5) marks → Dept. (not exist)
6) Rollno, Name → marks (exist)
7) Name → marks (exist)
8) Name, marks → dept (exist)

9) Name, marks → Dept, course (not exist).

Types:-

i) Trivial FD
ii) Non-trivial FD.

i) Trivial FD:-

→ A→B has trivial FD if 'B' is the subset of A
→ ~~If any~~ attributes can determine itself. (A→A)
→ {Rollno, Name} → Name.

ii) Non-Trivial FD:-

→ A→B has non-trivial FD. if 'B' is not a subset of A with A∩B = ∅, then A→B is called non-trivial FD.

Armstrong Axioms / Inference rule:-

→ Armstrong axioms are the basic inference rules.
→ It is used to conclude functional dependency on a relational data base.
→ It can apply to a set of functional dependency to derive the other functional dependency
→ Using this rule we can derive additional function dependency from initial states.

i) Reflexivity:-

→ In reflexive rule $y \subseteq x$ then $x \rightarrow y$.
→ x can determine itself.

$$x \rightarrow y, \quad y \subseteq x.$$

ii) Transitivity:-

→ If x determine y and y determine z, then x determine z.

$$x \rightarrow y, y \rightarrow z \text{ then } x \rightarrow z$$

iii) Augmentation:-

→ if $x \to y$ then $xz \to yz$.

iv) Union:-

→ union rules says if x determines y and x determines z then x must determine y and z.

if $x \to y$, $x \to z$ then $x \to yz$.

v) Decomposition:-

→ It is also known as project rule, it is the reverse of union rule.

→ This rules says if x determines y and z, then x can determine y and x can determine z.

if $x \to yz$ then $x \to y$, $x \to z$.

vi) Pseudo Transitivity:-

→ Here if x determines y and wy determines z, then wx can determine z.

if $x \to y$, $wy \to z$ then $wx \to z$.

① R (A, B, C, D, E)

FD: { A→B, B→C, C→D, D→E }.

Trick for e.x

A B C D E

↑ ↑ ↑ ↑

Reflexivity:-

A→A, B→B, C→C, D→D, E→E.

Transitivity:-

A→B, B→C. then A→C, A→D, A→E, A→ABCDE

B→C, C→D, then B→D, B→E,

C→D, D→E, then C→E

Agumentation:-

$A \to B$ $\Rightarrow AC \to BC$

$\Rightarrow AD \to BD$

$\Rightarrow AE \to BE$

$\Rightarrow ACD \to BCD$

$\Rightarrow ADE \to BDE$

$\Rightarrow ACE \to BCE$

$\Rightarrow ACDE \to BCDE$

$B \to C$ $\Rightarrow BA \to CA$

$\Rightarrow BD \to CD$

$\Rightarrow BE \to CE$

$\Rightarrow BAD \to CAD$

$\Rightarrow BAE \to CAE$

$\Rightarrow BDE \to CDE$

$\Rightarrow BADE \to CADE$.

$C \to D$ $\Rightarrow CA \to DA$

$\Rightarrow CB \to DB$

$\Rightarrow CE \to DE$

$\Rightarrow CAB \to DAB$

$\Rightarrow CAE \to DAE$

$\Rightarrow CBE \to DBE$

$\Rightarrow CABE \to DABE$

$D \to E$ $\Rightarrow DA \to EA$

$\Rightarrow DB \to EB$

$\Rightarrow DC \to EC$

$\Rightarrow DAB \to EAB$

$\Rightarrow DAC \to EAC$

$\Rightarrow DBC \to EBC$

$\Rightarrow DABC \to EABC$

Closure of attribute (X^+):-

→ It will contain set of attribute determined by X.

$A^+ = \{A, B, C, D, E\}$.

$(AD)^+ = \{AD, B, C, E\}$.

→ Super key is the set of attribute whose closure contain all the attributes of given relation.

→ candidate key is the minimal super key whose proper subset is not a superkey.

(1) R (A, B, C, D, E)

F.D = { A → B, D → E } . find every closure attribute.

$A^+ = \{A, B\}$

$B^+ = \{B\}$

$C^+ = \{C\}$

$D^+ = \{D, E\}$

$E^+ = \{E\}$

$ABD^+ = \{A, B, D, E\}$

$ABC^+ = \{A, B, C\}$

$ABCDE^+ = \{A, B, C, D, E\}$ → SK.

$ABDE^+ = \{A, B, D, E\}$ → not SK

$ACDE^+ = \{A, C, D, E, B\}$ → SK.

$ACD^+ = \{A, B, C, D, E\}$ → SK

→ Candidate key.

A B C D E

A D

ACD (C.K).

Proper subset

$A^+ = \{A, B\}$

$C^+ = \{C\}$

$CD^+ = \{C, D, E\}$

$AD^+ = \{A, B, D, E\}$

$AC^+ = \{A, B, C\}$.

$CD^+ = \{C, D, E\}$.

∴ ACD^+ is a candidate key because its proper subset can not a superkey.

(1) Relation R having attribute

R(A, B, C, D).

FD : {A → B, B → C, C → A}.

A B C D

↑ ↑ ↑

A- $(ABCD)^+$ = {A, B, C, D}.

$(ABC)^+$ = {A, B, C}

$(ABD)^+$ = {A, B, C, D}

$(AB)^+$ = {A, B, C}

$(AD)^+$ = {A, B, C, D}.

$(A)^+$ = {A, B, C, D}.

$(D)^+$ = {D}

$(ABCD)^+$ = {A, B, C, D} SK

$(ACD)^+$ = {A, B, C, D} SK

$(AD)^+$ = {A, B, C, D} SK → candidate key.

↓ Proper subset

$(A)^+$ = {A, B, C}

$(D)^+$ = {D}

$(CD)^+$ = {C, A, B, D} → SK

$(C)^+$ = {C, A, B} → candidate key.

$(D)^+$ = {D}

$(BD)^+$ = {B, C, A, D} → SK

$(B)^+$ = {B, C, A} → candidate key.

$(D)^+$ = {D}

→ After finding one CK we can determine any other C.K is exist or not we have to apply a rule, i.e.

if attribute of the C.K is present at right side of the FD then we can replace that attribute with FD attribute, by following this we can determine other C.K.

(2) Relation R having attribute

R(A, B, C, D)

FD :- {AB → CD, D → B, C → A}.

AB CD

↑ ↑ ↑ ↑

A. $ABCD^+ = \{A, B, C, D\}$

$CD^+ = \{A, C, D, B\}$

$AD^+ = \{A, D, B, C\}$

↓

Proper subset

$C = \{A, C\}$

$D = \{D, B\}$

CD^+ is a C.K.

$CD \rightarrow$ C.K.

$AB \rightarrow$ C.K

$AD \rightarrow$ C.K.

$BC \rightarrow$ C.K

Prime attributes and Non-prime attributes

① R (A, B, C, D, E, F)

FD: { $AB \rightarrow C$, $C \rightarrow DE$, $E \rightarrow F$, $D \rightarrow A$, $C \rightarrow B$ }:

A. $ABDEF^+ = \{A, B, C, D, E, F\}$. → SK.

~~$ABDEF^+ = \{A, B, C, D, E, F\}$. → SK~~

$ABDE^+ = \{A, B, C, D, E, F\}$.

$BDE^+ = \{B, D, E, A, C, F\}$.

$ABEF^+ = \{A, B, C, D, E, F\}$ → SK

$ABF^+ = \{A, B, C, D, E, F\}$. → SK.

Proper subset

$A^+ = \{A\}$

$B^+ = \{B\}$

$F^+ = \{F\}$

$AB^+ = \{A, B, C, D, E, F\}$ → SK. → $A^+ = \{A\}$, $B^+ = \{B\}$

$AF^+ = \{A, F\}$

$BF^+ = \{B, F\}$.

Here we get AB^+ is a Candidate key.

→ Then often candidate key are (by applying right side rules).

AB

DB

(or) ~~X~~AC → $A^+ = \{A\}$

$C^+ = \{D, E, F, A, B\}$ → $C = \emptyset$

(or) C

~~Prime~~

∴ ~~For~~ Candidate key ~~are~~ AB, DB, C.

Prime attributes are {A, B, C, D}.

Non-prime attributes are {E, F}.

② R(A, B, C, D, E)

FD: {A → B, B → E, C → D}

A. $AB^+ = \{A, B, C, D, E, F\}$

$ACDE^+ = \{A, B, C, D, E, F\}$.

$ACE^+ = \{A, E, F, B, C, D\}$.

$ACF^+ = \{A, B, C, D, E, F\}$.

$A^+ = \{A, B, E\}$

$C = \{C, D\}$

$F = \{F\}$

$AC^+ = \{A, C, B, E,$

AF^+

CF^+

$ABCDE^+ = \{A, B, C, D, E\}$

$ACDE^+ = \{A, B, C, D, E\}$

$ACE^+ = \{A, B, C, D, E\}$.

$AC^+ = \{A, B, E, C, D\}$.

Proper subset

$A^+ = \{A, B, E\}$

$C^+ = \{C, D\}$.

∴ Here AC is a candidate key

Often candidate key are

AC (only)

Prime Attributes { A, C }

Non prime { B, E, D }.

Normalisation:-

-> Normalisation is the process of organizing the data in database.

-> Normalisation is used to minimize the redundancy from a relation or set of relations.

-> It is also used to eliminate undesirable characteristics like insertion, updation and deletion anomaly.

-> Normalisation divides the larger relation / table into smaller relation / table and link them using relationships.

-> The normal form is used to reduce redundancy from the database table.

Data modification Anomalies (Problems)

Insertion Anomaly:

-> Insertion anamoly refers to when one cannot insert a new tuple into a relationship due to lack of data.

Deletion Anomaly:

-> The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.

Updation anomaly:

-> The update anomaly is when an update of a single data value requires multiple row of data to be updated.

Insertion anomaly

S-ID	name	Age	D-name	D-code	HOD
1	Tom	19	Math	101	Wani
2	Katie	19	math	101	Wani
3	Aleem	20	math	101	Wani
4	Nejati	18	Physics	102	Steve
5	John	21	Physics	102	Steve
6	Amen	19	Chem	103	Bisi
			Geology	115	Ganiy

NOT allowed.

Delation anomaly:-

ero	ename	bdate	title	Sal	Superno	dno	dname	mgrno.
E1	Doe	01-05	EE	3000	E2	null	null	E[illegible]
E2	Smith	06-04	SA	5000	E5	D1	Accou	E5
E3	Lee	7-5	ME	4000	E7	D2	consult	E7
E4	Casey	9-1	PR	2000	E6	D3	Accoun	E5
E5	chu	12-25	SA	5000	E8	D3	Accun	E5
E6	Davis	11-30	EE	3000	E7	D2	consult	E7
E7	miller	04-8	ME	4000	E8	D1	manage	E8
E8	Jones	10-11	SA	5000	null	D1	manage	E8

Consider the delation anomaly:-

-) Delete employee E3 and E6 from the database.

-) Deleting those two employees remove from from the database; And we know have lost information about department D2!

Updation Anomaly

Employee

EID	Name	Salary	D-ID	Department
1	Paul	10000	1-4	A/c
2	John	6000	1	A/c
3	Shoona	7000	3	Sales
4	mine	6000	2	married
5	hudson	10,000	2	married
6	smith	20,000	1	A/c
7	paul	50,000	2	m/u
8	maithili	15,000	1	A/c

-> D-ID updated 1 to 4.

Why normalization required

-> The main reason for normalization the relation is removing these anomalies.

-> Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows.

-> Normalization consist of a series of guidelines that helps to guide you in creating a good database structure.

Types of Normal forms

Normal form	Description
1NF	A relation is in 1NF, if it contains an atomic values.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.

Normal form	Description
BCNF	A stronger definition of 3NF is known as Boyce codd's normal form.
4NF	A relation will be in 4NF, if it is in Boyce codd's normal form and has no multi valued dependency.
5NF.	A relation is in 5NF, if it is in 4NF and and does not contain any join dependency, joining should be lossless.

Advantages of Normalization

-> Normalization helps to minimize data redundancy.

-> Data consistency with in the data base.

-> Much more flexible database design.

-> Enfonces the concept of relational integrity.

Disadvantage

-> You cannot start building the database before knowing what the user needs.

-> The perfonmance degrades when normalizating the relations to higher normal forms i.e 4NF, 5NF.

-> It is very time-consuming and difficult to normalize relation of a higher degree.

-> Careless decomposition may lead to a bad database design, leading to serious problems.

First Normal Form

→ A relation will be 1 NF if it contains atomic values.

→ It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute

→ First normal form disallows the multi-valued attribute and their combinations.

→ Ex.

Roll no	Name	course
1	Akash	C, DBMS
2	Satish	Python
3	Dev	PHP, ~~Python~~ java

Is it in 1 NF ? No, as it contains multiple values.

→ we can represent the table in 1 NF as follows:

Option 1

Roll no	Name	course (PK)
1	Akash	C
1	Akash	DBMS
2	Satish	Python
3	Dev	PHP
3	Dev	~~Python~~ java.

Option 2

Roll	name	course1	course2
1	Akash	C	DBMS
2	Satish	Python	NULL
3	Dev	Php	python.

Option 3

Roll	name
1	Akash
2	satish
3	Dev

Roll	Name
1	C
1	DBMS
2	Python
3	PHP
3	Java.

Second Normal form:-

→ In 2nd Normal form, relation must be in 1NF.

→ In 2NF all non-key attributes fully functional depedent on candidate key.

→ That should not be any partial dependency.

Ex- R(A, B, C, D, E, F)

FD: { C → F, E → A, EC → D, A → B }

PPD PFD

A ABCDEF$^+$ = {A, B, C, D, E, F} → S.K

AB CDE$^+$ = {A, B, C, D, E, F} → S.K

ABCDE$^+$ = {B, C, D, E, F, A} → S.K

BCE$^+$ = {B, C, E, F, A, D} → S.K.

CE$^+$ = {C, E, F, A, B, D} → C.K.

{C}$^+$ = {C, F}

{E}$^+$ = {A, E, B}

~~AB~~?

other C.K

CE

PA = {C, E}

NPA = {A, B, D, F}

→ This table is not in 2NF, we have to convert in 2NF by spliting.

for 2NF

R1 (C, F)

R2 (E, A, B)

R3 (E, C, D).

① R(A, B, C, D, E)

FD: { AB → C, D → E } .
PFD

A- $ABCDE^+ = \{A, B, C, D, E\}$.

$ABD^+ = \{A, B, D, C, E\}$

$A^+ = \{A\}$ $\quad AB^+ = \{A, B, C\}$

$B^+ = \{B\}$ $\quad BD^+ = \{B, D, E\}$

$D^+ = \{D, E\}$ $\quad AD^+ = \{A, D, E\}$

Hence ABD is a C.K.

Other C.K

ABD.

PA = { A, B, D }

NPA = { C, E }.

→ If is not in 2NF, we have to convert 2NF by splitting.

→ R_1 (A, B, D)

R_2 (D, E)

R_3 (AB C)

② R (A, B, C, D, E)

FD: { A → B, B → E, C → D }.

③ R (A, B, C, D, E, F, G, H, I, J)

FD: { AB → C, AD → GH, BD → EF, A → I, H → J }

④ R (D, B, C, D)

FD: { AB → CD, C → A, D → B }.

Third Normal Form : (3NF) :-

-> In 3NF table must be in 2 NF and doesnot contain any transitive FD.

-> 3NF is used to reduce data duplication, it is also used to achieve data integrity.

-> If there is no tranisitive dependency for non-primeary attributes, then the relation must be in 3NF.

-> If the FD is
NPA -> NPA then it is not in 3NF.

-> If the FD is
PA -> PA, P -> NPA, NPA -> P is in 3NF.

(1) R(A, B, C, D)
FD : { AB -> C, C -> D }

A- $AB^+ = \{A, B, C, D\}$

A = {A}
B = {B}

AB is the C.K

PA = {A, B}
NPA = {C, D}.

-> It is in 2NF.

-> As it is not in 3NF, we have decompose the table C(C->D) NPA->NPA

R_1 (A B C)

R_2 (C D)

② R (A, B, C, D, E)

FD { A → B, B → E, C → D }.
PFD PFD

A- CK = AC.

PA = {A, C}

NPA = {B, D, E}

→ It is not in 2NF, we have split it.

R1 (A C)

R2 (A, B, E)

R3 (C, D)

→ It is not in 3NF, we have to decompose it.

R1 (A, C)

R21 (A, B)

R22 (B, E)

R3 (C, D)

③ R (A, B, C, D, E, F, G, H, I, J)

FD { AB → C, A → DE, B → F, F → GH, D → IJ }.
DFD PFD

A- Here AB is the C.K.

PA = { A, B }

NPA = { C, D, E, F, G, H, I, J }.

→ It is is not in 2NF.

R1 (AB C)

R2 (A DE IJ)

R3 (B F GH)

→ It is not in 3NF.

R1 (AB C)

R21 (A DE)

R22 (D IJ)

R31 (B F)

R32 (F GH)

BCNF :- Boyce codd Normal Form

→ BCNF is the advanceversion of 3NF, it is Stricten than 3NF.

→ A table is In BCNF if every FD, $x \rightarrow y$, x is the Superkey of the table. OR L.H.S should be superkey.

① R (ABC)

FD :- {AB → C, C → B}

A- $AB^+ = \{A, B, C\}$

AB is C.K.

AC is C.K

$PA = \{A, B, C\}$

→ It is in 2NF, also in 3NF.

→ not in BCNF

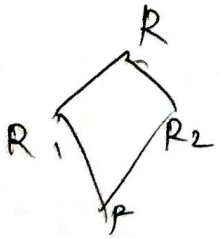
R_1 (ABC)

R_2 (C B)

Decomposition :-

→ The process of breaking or dividing a single relation into 2 or more sub relation is called, decomposition of a relation.

→ Whenever the relation not in proper normal from, we need to decompose it.



Rules

→ Attribute of R_1 ∪ Attribute of R_2 = Attribute of R.

→ Attribute (R_1) ∩ Attribute (R_2) ≠ ∅

→ Attribute (R_1) ∩ Attribute (R_2) ⟶ Attribute(R_1) OR Attribute(R_2)

FDependency preserving Decomposition:

→ It is a technique use in DBMS to decompose a relation into a smaller relation, while preserving the functional dependencies between the attributes

→ The goal is to improve the efficiency of the database by reducing redundancy and improving query performance.

→ In dependency preservation at least one decomposed table must satisfies every dependency.

→ If a relation R is decomposed into R_1 and R_2, then dependencies of R either must be part of R_1 or R_1 or must be derieveable from the combination of functional dependency of R_1 and R_2..

① find the FD exist in.

R

A	B	C
1	1	1
2	1	2
3	2	1
4	2	2

$t_1.x = t_2.x$ the $t_1.y = t_2.y$

A- FD: { A → BC , BC → A } ∴

R_1

A	B
1	1
2	1
3	2
4	2

FD1 {A → B}

R_2

B	C
1	1
1	2
2	1
2	2

FD2 {X}

FD1 ∪ FD2 ≡ FD

→ It is lossy decomposition.

(7) R(A, B, C, D, E)

FD{ A→B, B→C, C→D, D→A }.

Ans- R_1(ABC)

R_2(C, D, E)

For FD of R_1

R_1(ABC)

A^+ = {A, B, C}

B^+ = {B, C, A}.

C^+ = {C, A, B}

AB^+ = {A, B, C}

BC^+ = {B, C, A}

CA^+ = {C, A, B}

(trivial is not allowed)

These are duplicates

FD1:- A→BC, B→CA, C→AB, (AB→C, BC→A, AC→B)

For FD of R_2

R(CDE)

C^+ = {C, D, A, B}

D^+ = {D, A, B, C}

E^+ = {E}

CD^+ = {C, D, A, B}

CE^+ = {C, E, D, A, B}

DE^+ = {D, E, A, B, C}.

Duplicates.

FD2:- { C→D, D→C, CE→D, DE→C }.

FD1 ∪ FD2 ≡ FD

→ Here all the functional dependency are preserve.

4NF

→ The relation will be 4NF, if it is in BCNF and has no multivalued dependency.

multivalued dependency

→ Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on third attribute.

→ If A →→ B (read as A multi determines B), exists, for a single value of A, more than one value of B exists.

→ Table should have at least 3 columns.