Page No:
Date:

Introduction to database systems

- **Data**
  - Raw facts, unprocessed facts
  - Refers to what is actually stored

- **Information**
  - Result of processing raw data
  - Refers to meaning of the data understood by the user

Data management focuses on the generation, storage and retrieval of data.

- **Limitations of file processing systems**
  - Redundancy problems
    - Repetitive data
  - Data inconsistency problems
    - Incorrectness of data
  - Lack of data integration
    - Complex and time consuming

- **Database**

A database is a collection of related data which represents some aspect of the real world. A database system is designed to be built and populated with data for a certain task.

- **Database Management System (DBMS)**

DBMS is a software for storing and retrieving user's data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating

Page No:
Date:

Types of databases:

Based on the number of users using it

i) Single User Database System
(At one time only a single user can interact with the database)

ii) Multi User Database System
(At one time multiple users can interact)

Based on the location of the database

i) Centralized Database System
(Data located at a single place)

ii) Distributed Database System
(Data distributed across different locations)

Advantages of DBMS over file system

(i) Restricting unauthorized access
(ii) Providing multi user interface
(iii) Providing backup and recovery
(iv) Allows data sharing
(v) Enforcing integrity constraints
(vi) Solving data isolation
(vii) Controlling redundancy & inconsistency
(viii) Providing economies of scaling

Data Model

Data Model is a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints. It provides a way to describe the design of database

Components of data model

(i) Entity: Anything that exists, living or non living but not imaginary.

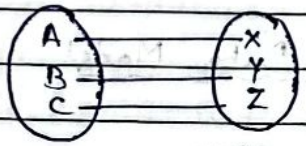
(ii) Entity Set: Set of entities of the same type

(iii) Attributes: Characteristics of an entity

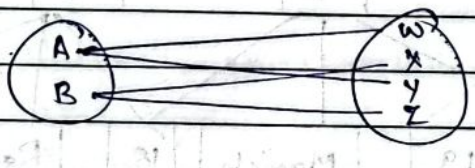
(iv) Constraints: Restrictions placed on a data

(v) Relationships: Similarities or a connection b/w entities

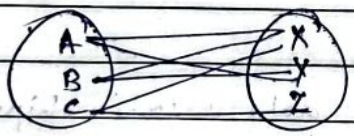
One to One (1:1) Relationship



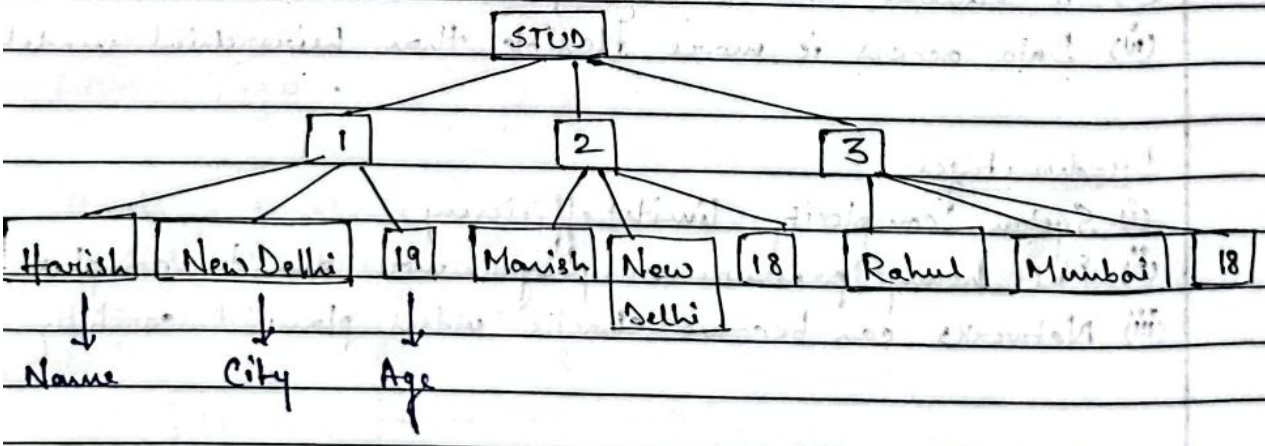
One to Many (1:M) Relationship



Many to Many relationship



Hierarchial Data Model

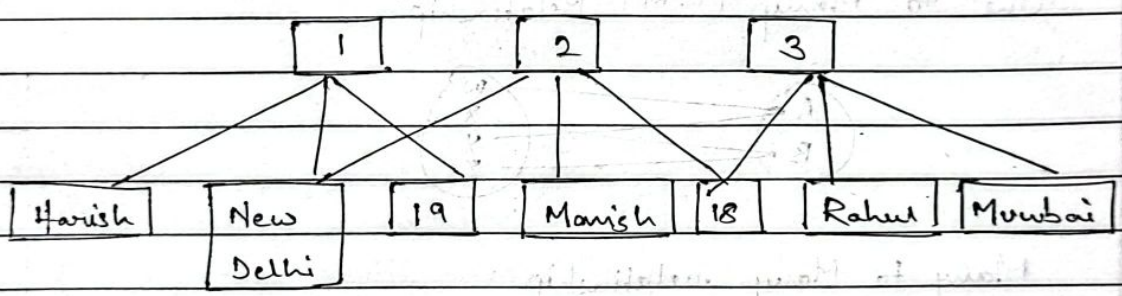


Page No:
Date:

Advantages
(i) It is efficient with 1:M relationships
(ii) Efficient storage for data that have clear hierarchy
(iii) It promotes data sharing

Disadvantages
(i) It is complex to implement and difficult to manage
(ii) It can't represent M:N (many to many) relationships
(iii) There is no DDL and DML

Network Data Model



Advantages
(i) It represents complex data relationships better than hierarchial model
(ii) It includes DDL and DML
(iii) It handles M:N relationships
(iv) Data access is more flexible than hierarchial model

Disadvantages
(i) System complexity limits efficiency
(ii) Put heavy pressure on programmers due to complex structure
(iii) Networks can become chaotic unless planned carefully

Page No:
Date:

Relational Data Model

STUD

roll	Name	City	Age
1	Harish	New Delhi	17
2	Manish	New Delhi	18
3	Rahul	Mumbai	18

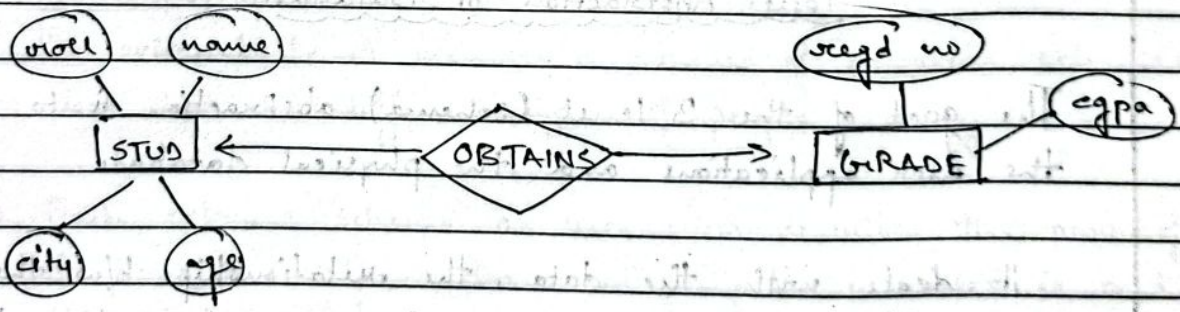
Advantages

(i) Tabular view substantially improves conceptual simplicity
(ii) Changes in table's structure do not affect data access or application programs
(iii) RDMS isolates the end users from physical level details and improves implementation and management simplicity

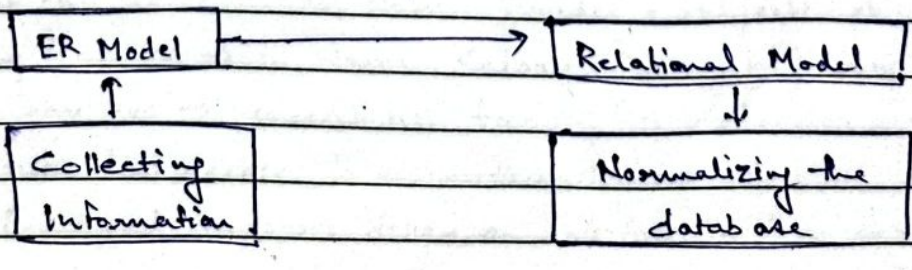
Disadvantages

(i) Conceptual simplicity gives relatively untrained people the tools to use a good system poorly.
(ii) It may promote islands of information problems as individuals and departments can easily develop their own applications.

Entity Relationship (ER) Model



It is a graphical representation of entities and their relationships in a database structure. ER models are basically represented using ER diagram.



Page No:
Date:

Advantages

i) Visual representation makes it an effective communication tool
ii) It is integrated with dominant relational model
iii) Visual modelling yeilds exceptional conceptual simplicity

Disadvantages

i) There is limited constraint representation
ii) There is no DML
iii) There is limited relationship representation.
iv) Loss of information content when attributes are removed from entities to avoid crowded displays.

Explain database architecture (diagram)

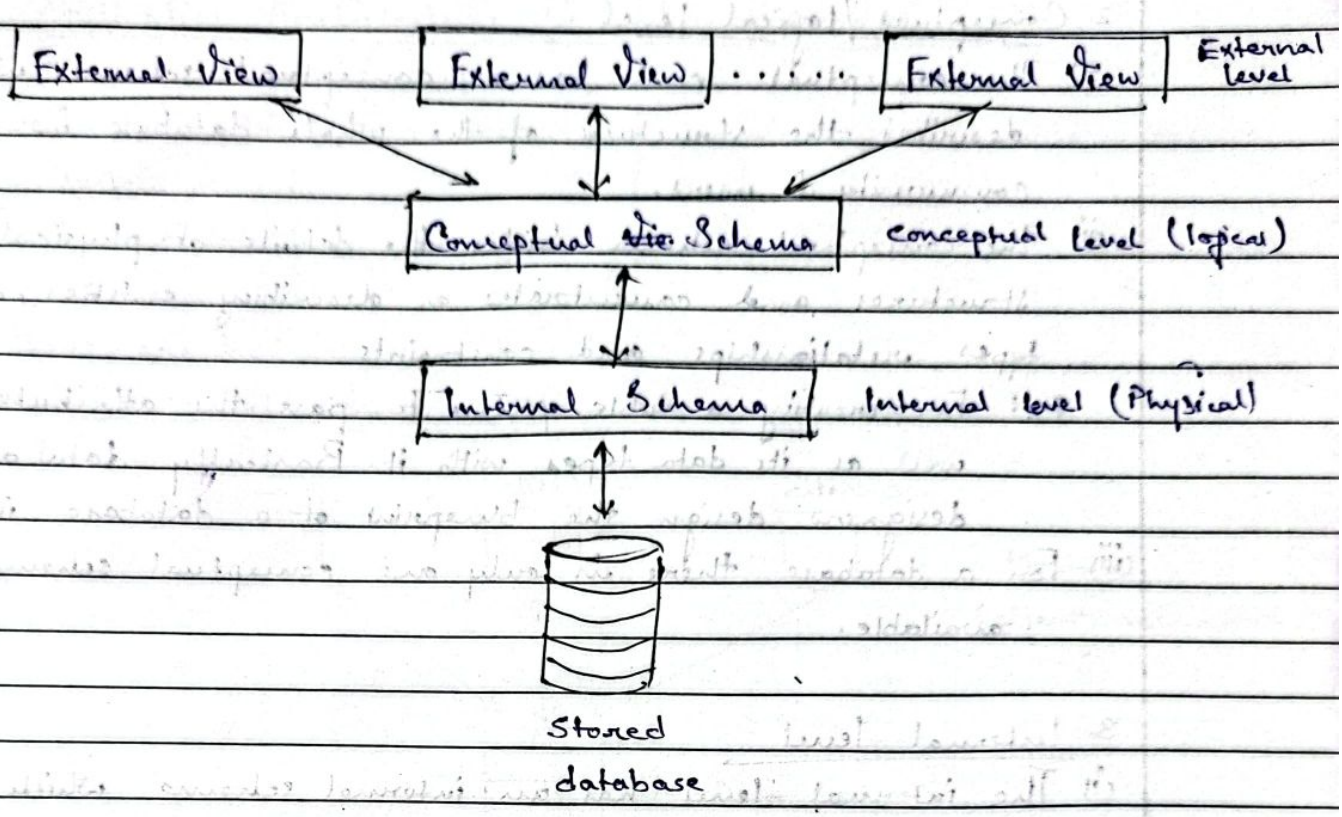
Components of DA
1. Users & DBA
2. Query processor
3. Storage manager
4. Disk storage

Level Abstraction of Database

The goal of the 3 level (schema) abstraction is to separate the user applications and the physical database.

It deals with the data, the relationship b/w them and the different access methods implemented on the database

The logical ~~diagram~~ design of a database is called as schema.



User knows that their data is stored somewhere but they don't know how or where the data is stored. It means the process is kept hidden or abstracted from the user. To achieve this the three level abstraction concept was introduced.

1. External/View level

(i) The external level includes a number of external schemas or user views for different types of people accessing it.

(ii) Each external schema or user view describes the part of the database that a particular user group is interested in and hides the details ~~of the~~ from other users.

Ex: Suppose in a database of university there are three external schemas → Student, Faculty, Dean

Student can see → Roll no, Name, Grades, Subjects etc

Faculty can see → id, Name, Salary, Date of joining etc

Dean can see → student list, faculty list, attendance etc.

* Each user has access to ~~each other~~ different types of data but not each others. Front end developers design an interface in this stage for the user

2. Conceptual / logical level

(i) The conceptual level has a conceptual schema which describes the structure of the whole database for a community of users.

(ii) The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships and constraints

Ex: For creating a table you've to pass the attributes as well as its data types with it. Basically database designers design the blueprint of a database in this stage

(iii) For a database there is only one conceptual schema available.

3. Internal level

(i) The internal level has an internal schema which describes the physical storage structure of the database system.

(ii) Like conceptual schema, there is only one internal/physical schema available

(iii) It is the one which is ~~available~~ closest to the physical storage. In this stage Database Administrators have the access to the whole database and he/she decides where the files should be stored (centralized or decentralized database)

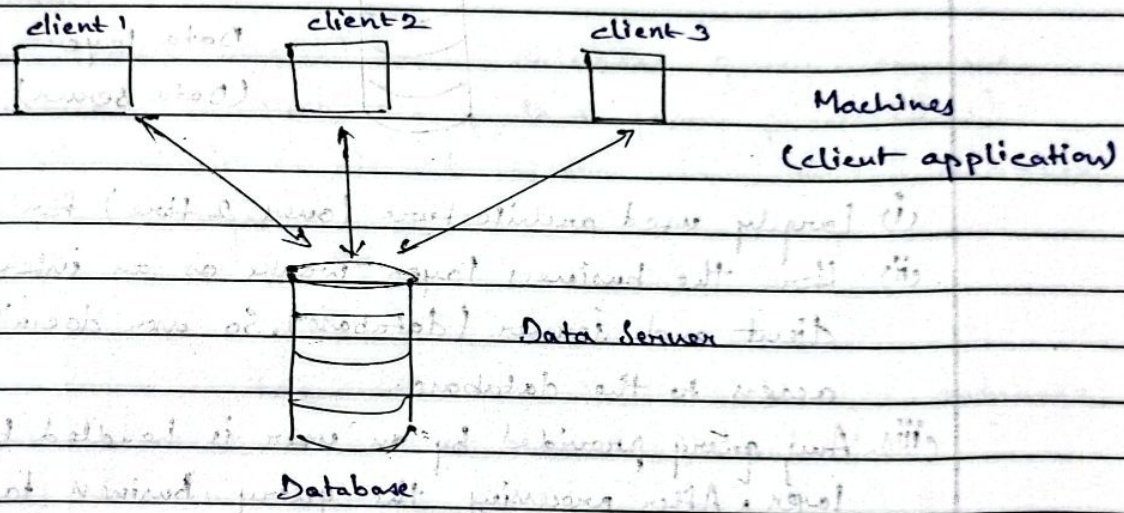
When an user is seeing a data fetched from the database they generally see it in the tabular form. But data aren't actually stored in the database as tabular form. Actually the data's are stored as files and when an user needed it, the file is just converted into tabular format.

Page No:
Date:

Application Architecture Used for Database

(i) Client machines are ~~there~~ those on which the remote database system runs.
(ii) Server machines are those on which the database system runs.

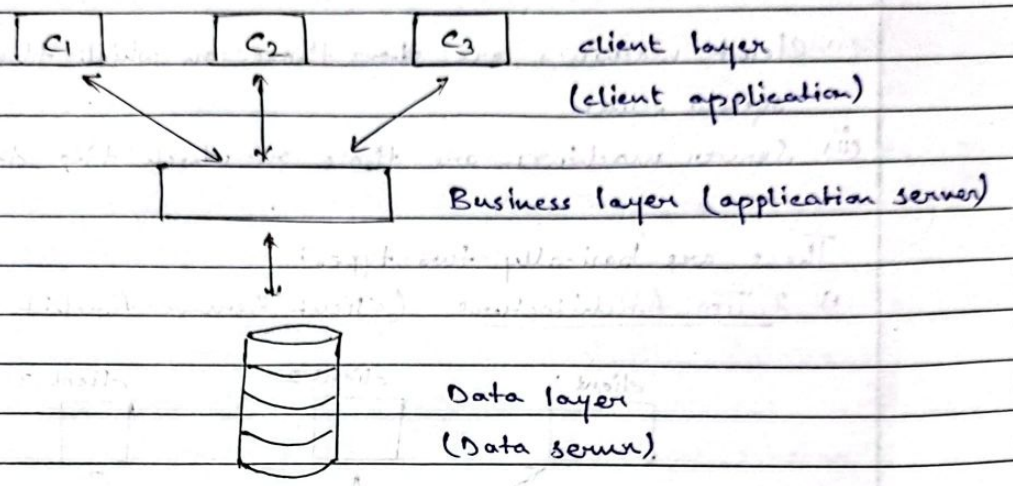
There are basically two types:
1) 2 Tier Architecture (Client Server Architecture)



i) Here the client machine runs an interface or applicatin that helps an user to establish a connection b/w the database and the machine.
ii) User simply writes a query (application program) through the interface and the database then processes it and returns the required data to the client.
iii) Maintainance is easy because of limited and authorized clients.
iv) Fails in terms of scalability. (handling large no. of clients, providing data 24x7)
v) Here user has direct access to the database which can create a huge problem. Fails in terms of security.

Page No:
Date:

2) 3 Tier Architecture



(i) Largely used architecture (over 2 tier) for its advantage.
(ii) Here the business layer works as an intermediate b/w the client and server (database). So user doesn't have direct access to the database.
(iii) Any query provided by an user is handled by the business layer. After processing the query business layer wants the required data from the server and returns it to the user. (In 2 tier all the processes and data transfer things are handled by the server. So its also an disadvantage for 2 tier architecture that it may malfunction because of too much load)

Types of Database Users

① Naive Users

- They are the normal or sophisticated users who interact with the system by invoking application programs that have been written previously.

- The typical user interface for naive users is a form interface where the user can fill in appropriate fields of the form.

② Application programmers
- They are computer professionals who write application programs to access data from the database.
- Application programmers can use different tools to develop user interface.

③ Sophisticated Users
- They interact with the system without creating any application program.
- They form their request in a database query language and submit each such query to a query processor.
- Analysts who submit queries to explore data in the database fall in this category.

④ Specialized Users
- They are sophisticated users who write specialized database applications that don't fit into the traditional data processing framework.

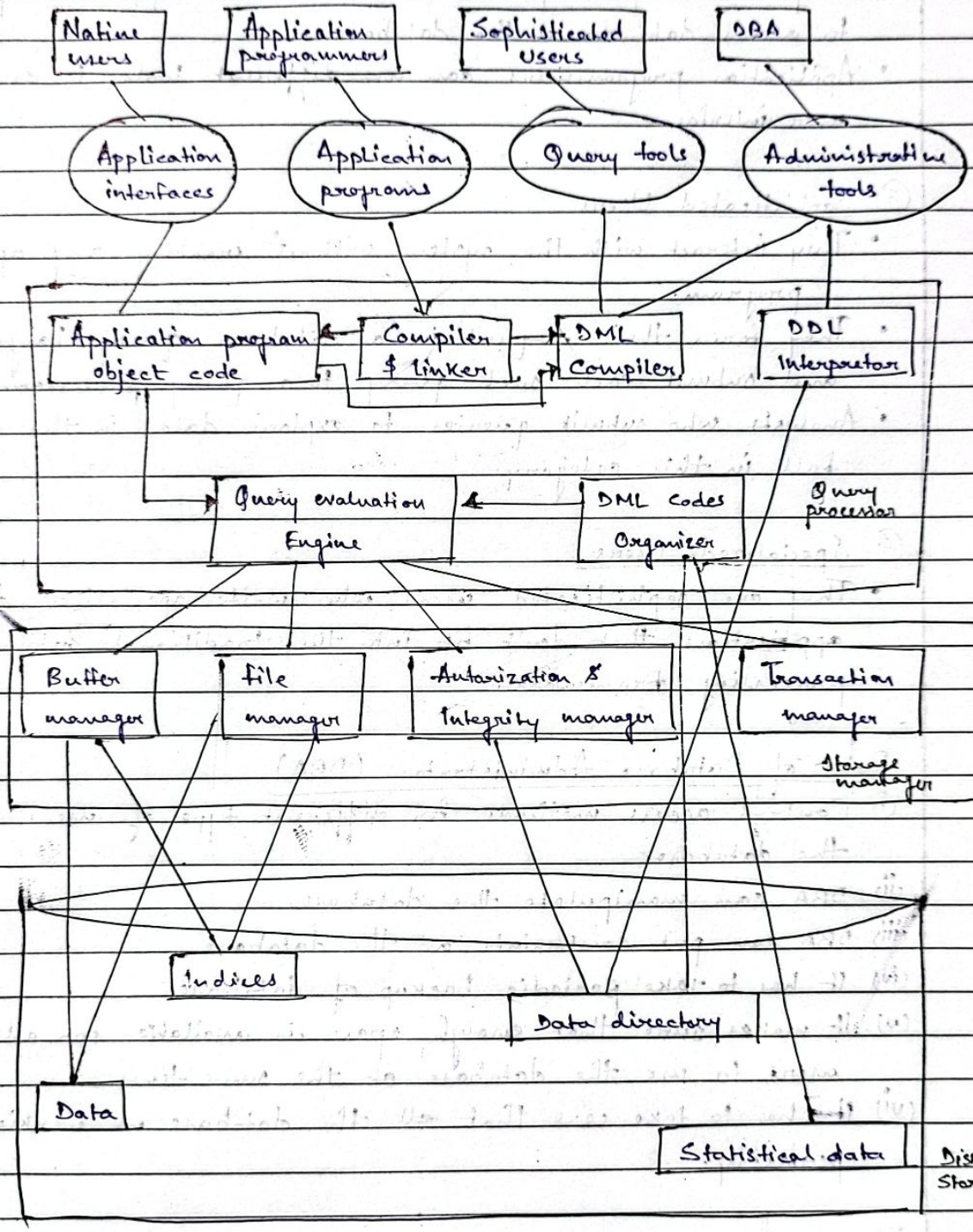
Roles of Database Administrator (DBA)
(i) Control access methods for different type of users to use the database.
(ii) DBA can manipulate the database
(iii) DBA can put constraints on the database
(iv) It has to take periodic backup of database
(v) It makes sure that enough space is available for all the users to use the database at the same time.
(vi) It has to take care that the database is working perfectly.

Physical data independence indicates that the internal schema can be changed without any change to the conceptual schema.

Logical data independence indicates that the conceptual schema can be changed without affecting existing external schemas.

Page No:
Date:

Database Architecture



Page No:
Date:

Disadvantages of DBMS

(i) Large file size
(ii) Increased complexity
(iii) Greater impact of failure
(iv) More difficult to recover.

Chapter 2.

Page No:
Date:

Entity Relationship Model

(i) Collect information about what they want in the database
(ii) Draw the entity relation model for the requirement

Entity

An entity is a thing or object that exist in the real world

Attributes → Oval Shape ⬭
Entities → Rectangle ▭

Relationship

Association b/w entities are referred to relationship

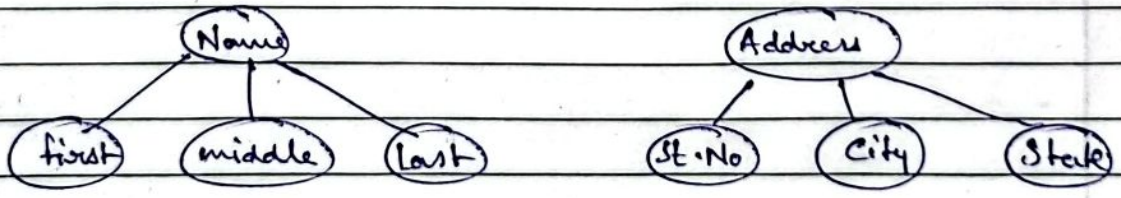
Relationship → Diamond Shape ◇

NULL Value :

When an entity doesn't have a value for it, we can put NULL value for it.

Simple & Composite Attributes

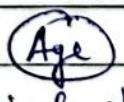
Attributes that can't be sub divided are simple attributes.
Ex: Roll no
Attributes that can be sub divided are composite attributes
Ex: Name (first and last name)



Multivalued attributes

Attributes with more than one value
Ex: Student with more than one mail and phone number

Multi valued



Single Valued

Stored & Derived Attributes

Attribute with independent existence is stored attributes.
Ex: Age of a student

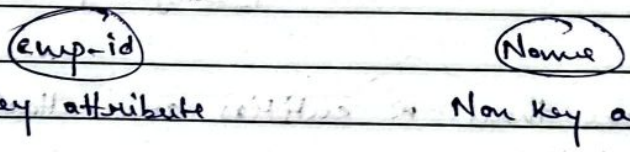
Attribute whose value is depending on other stored attribute is called as derived attribute.
Ex: DOB of a student can be derived from age

Descriptive Attributes

A relationship may also have attributes called as descriptive attribute.

Keys and Non Keys Attribute

Uniquely identified attribute → Key attribute



Complex Attribute

Composite + Multivalued Ex: Address → (current + permanent) → multi valued
→ (street name, city, state) → composite

Types of relationship

(1) M : N relationship

(2) 1 : M relationship

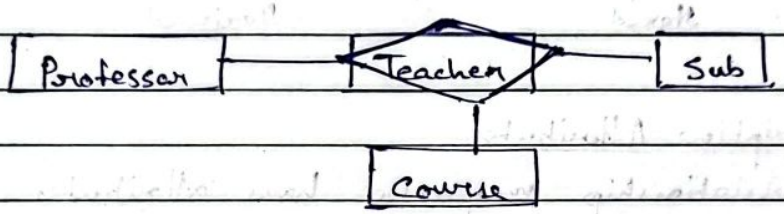
(3) 1:1 relationship

Degree of relationship type

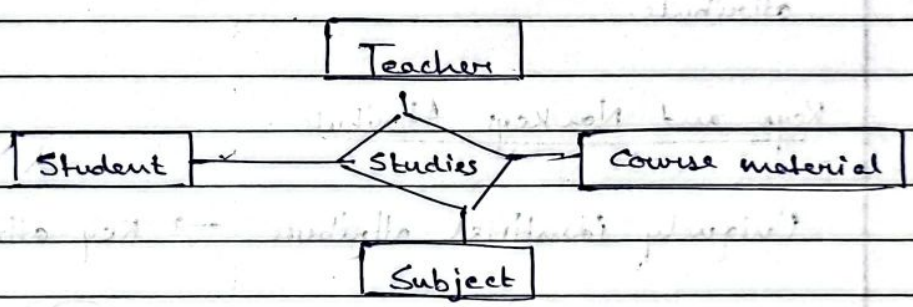
A relationship type is a meaningful association among entities

1) Binary : 2 entities are attached to an attribute

2) Ternary :

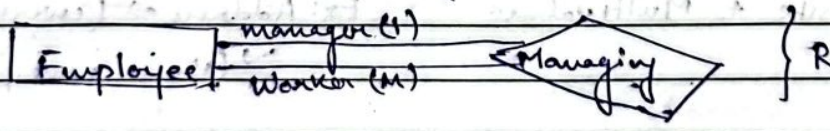


3) Quaternary :



4) n-ary : More than n entities are attached to a single attribute

5) Recursive : Special case of relationship

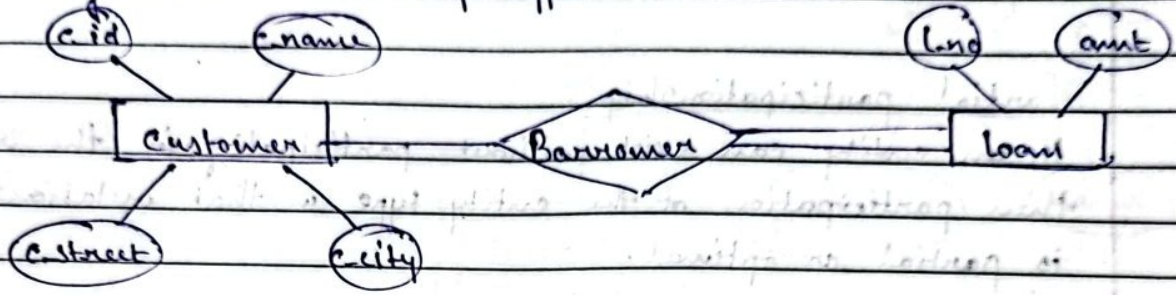


One entity has multiple roles.

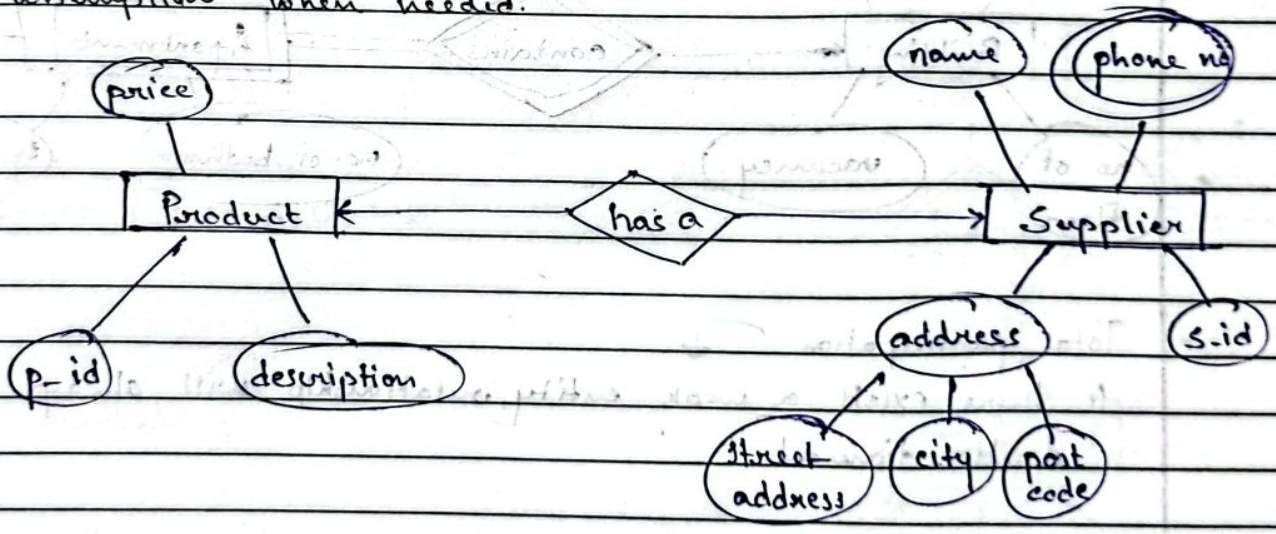
Page No:
Date:

Participation Constraints

The participation constraint for an entity set in a binary relationship type is based on whether an entity of that entity set needs to be related to an entity of the other entity set through this relationship type



Q Construct the ERD to represent information of product and supplier. Each product has a description, a price and a supplier. Suppliers have addresses, phone numbers and name. Each address is made up of a street address, a city and a postcode. Make assumptions when needed.



* s_id is taken as an attribute of the supplier and it is also the primary key

Total participationship

If in order to exist, every entity must participate in the relationship, then participation of the entity set in that relationship type is total or mandatory. It is represented by double lines

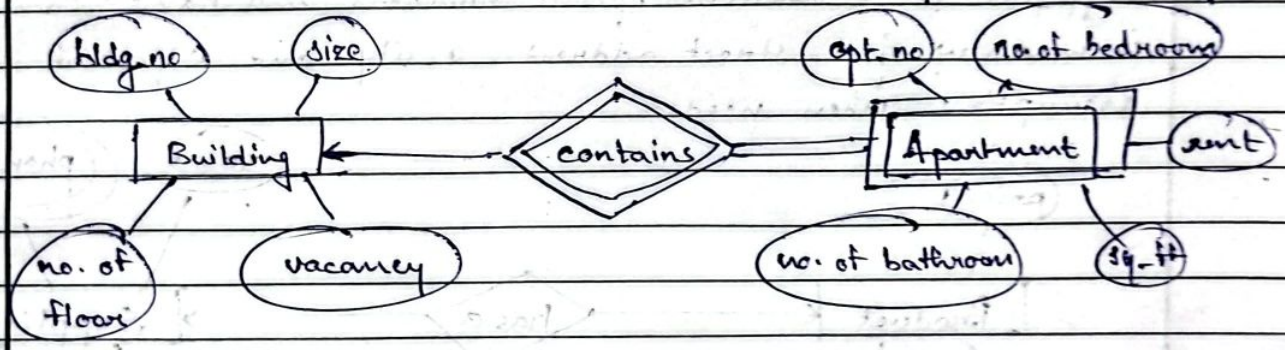
Partial participationship

If an entity can exist without participating in the relationship then participation of the entity type in that relationship type is partial or optional.

Strong & Weak Entity Set

Independent existence → strong entity

Dependent existence → Weak entity → (represented by double rectangle)



Total participation →

If there exists a weak entity, relationship will always be double diamond.

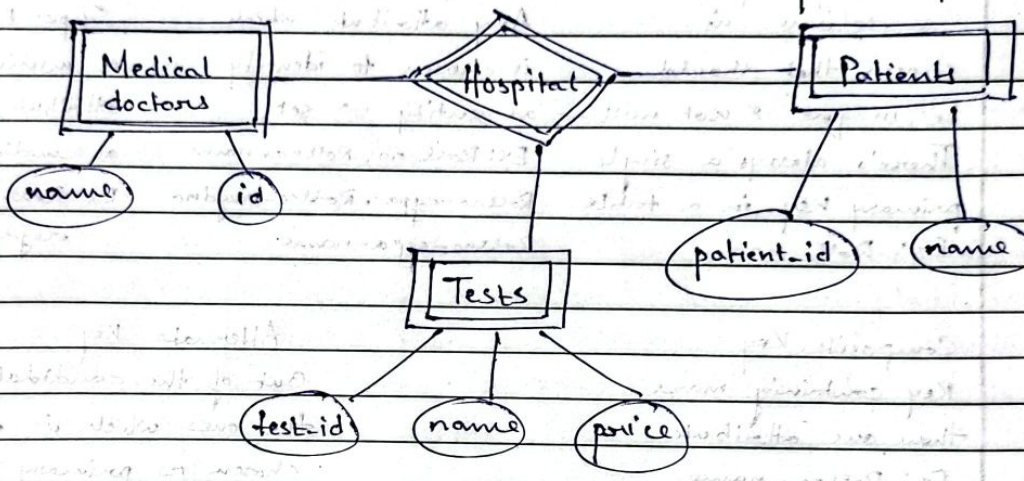
Primary Key → Primary Key of strong entity

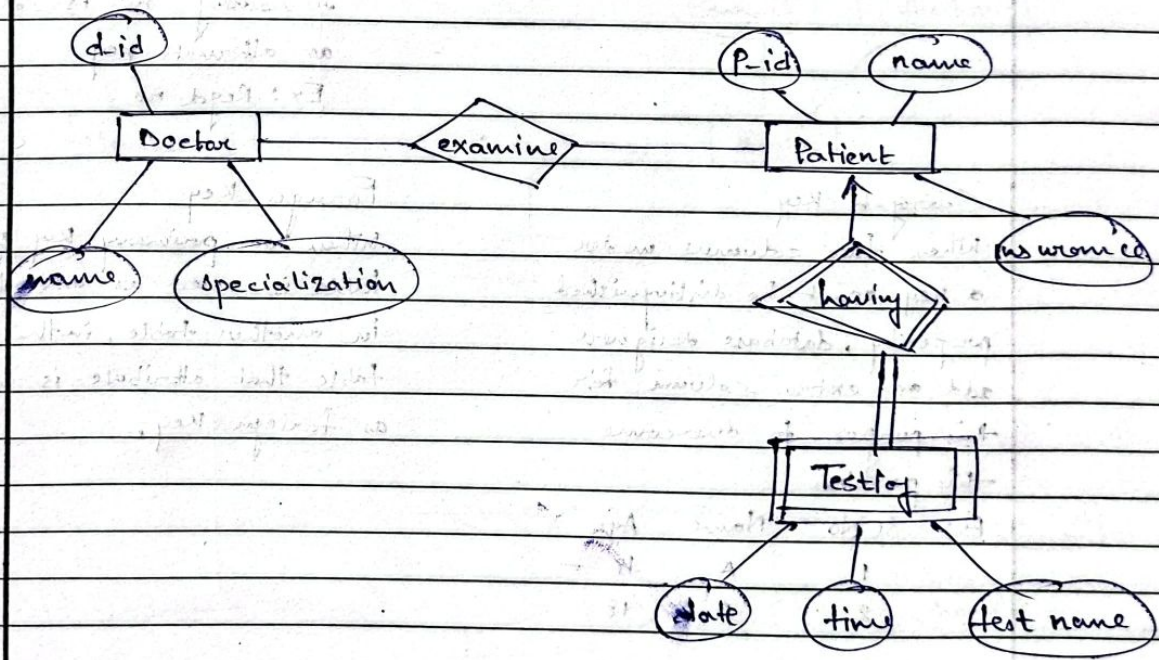
+

distributor Partial Key

Page No:
Date:

Q Construct an ER diagram with a set of patients and a set of medical doctors. Associated with each patient a log of the various tests & examinations conducted. Make neccessary assumptions.





Page No:
Date:

Keys

Primary Key
One column is chosen that should be unique & not null
There's always a single primary key in a table
Ex: Roll no

Super Key
Any attribute which is chosen to identify an entity → set
Ex: Roll no, Rollno-name
Rollno-cgpa, Rollno-regdno
Rollno-cgpa-name

Candidate Key
Super key with a minimal attribute is called as candidate key
Ex: Roll no, regd no.

Composite Key
Key containing more than one attribute
Ex: Rollno-name

Alternate Key
Out of the candidate keys the ones which is not chosen as primary key - the remaining one is called as alternate key
Ex: Regd no

Surrogate Key
When two columns under a key can't be distinguished properly, database designers add an extra column for this purpose to overcome this problem

Ex:

Sl No	Name	Age
1	A	19
2	A	18

Foreign Key
When a primary key of a table is used as an attribute in another table, in the other table that attribute is called as foreign key.

Keys for relationship

- M:N Relationship

The primary key of the relationship set contains the union of the primary keys of the entity set.

Page No:
Date:

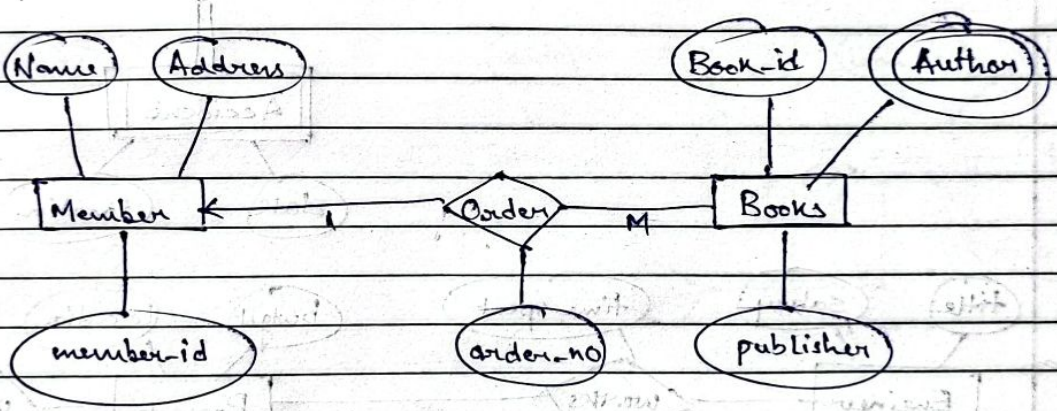
1 : M Relationship

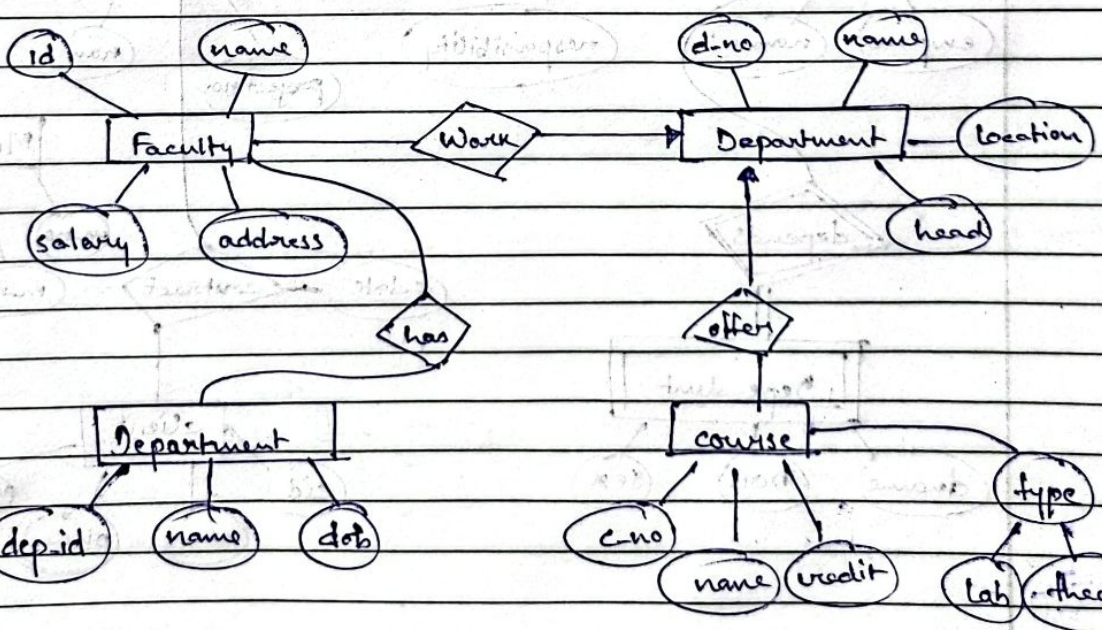
The primary key of the relationship set is the primary key of the many side entity set.

1 : 1 Relationship

The primary key of the relationship set is either the primary key of the entity set

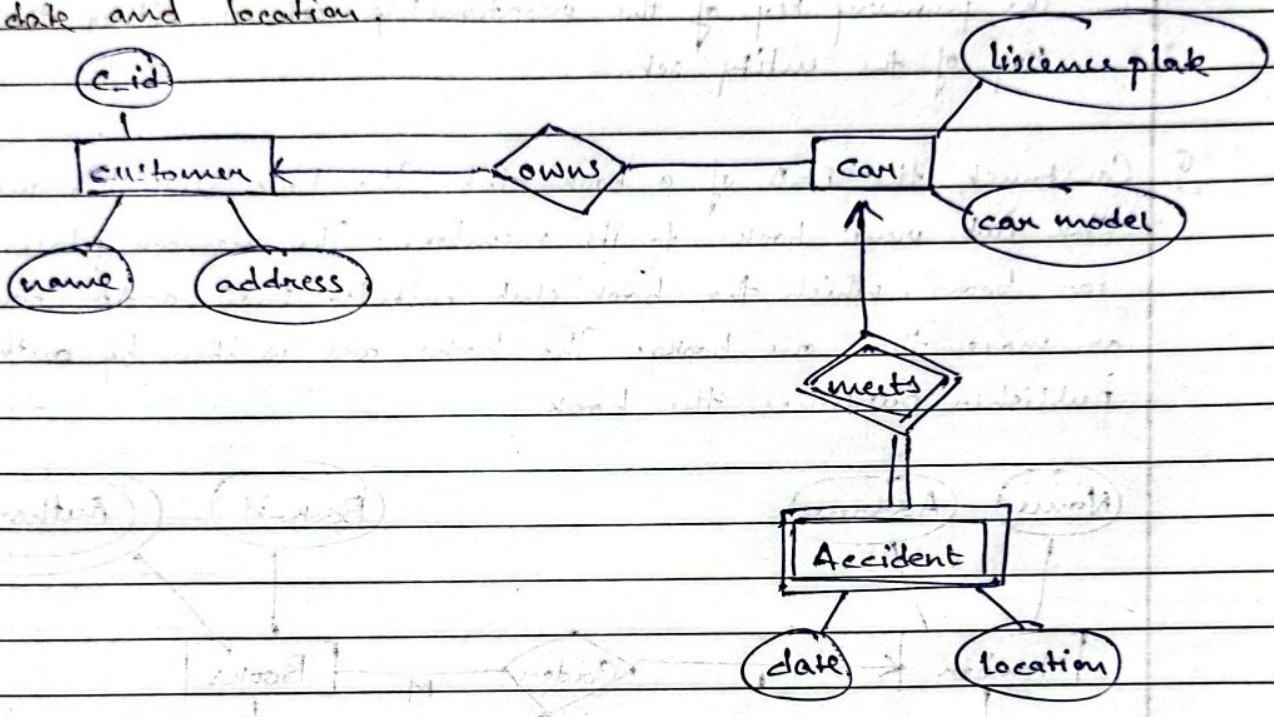
Q Construct the ERD of a book club. The book club has member. The book club rent books to its members. The member places order for books, which the book club fulfills. Each order contains one or more than one books. The books are written by authors. The publisher publishes the book.



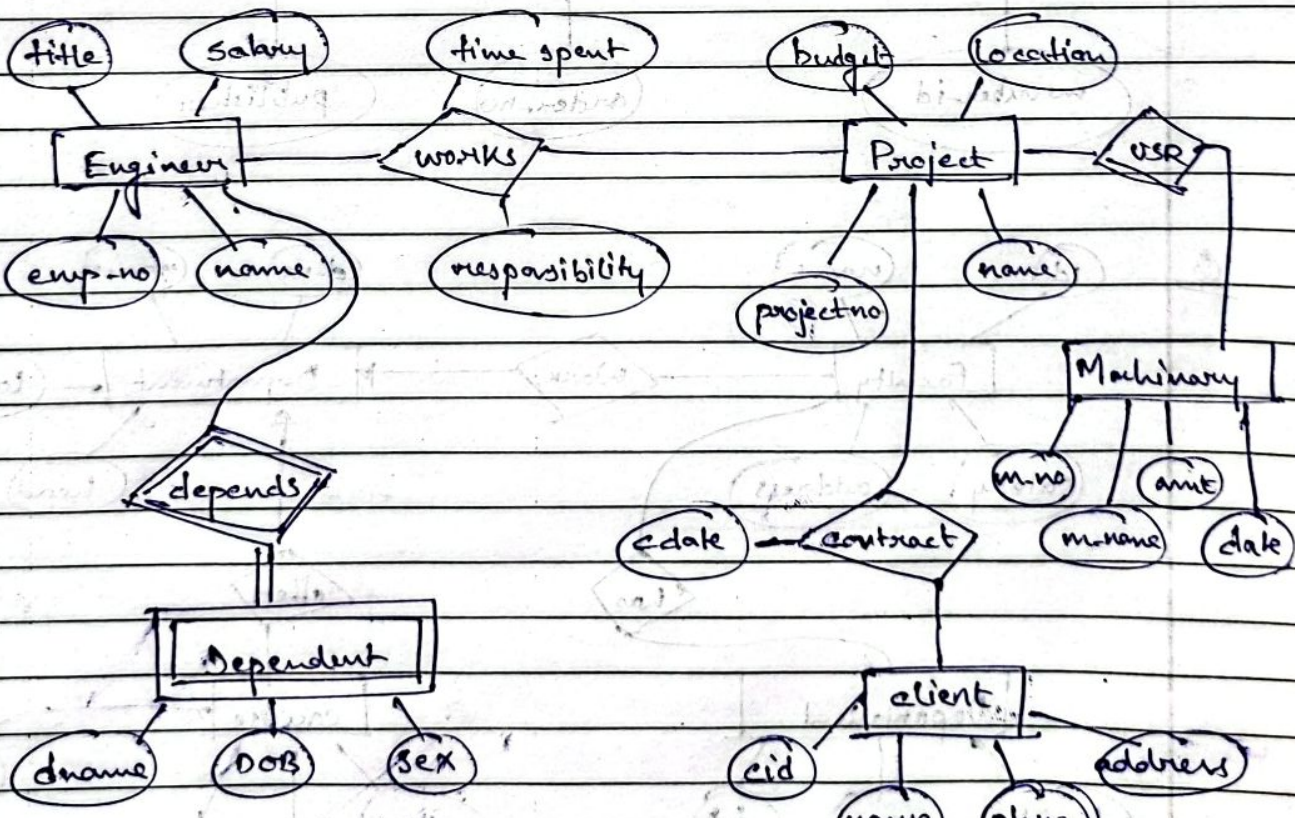


Page No:
Date:

Q Construct an ER diagram for a car insurance company where customers own one or more cars each. Customers are identified by id, name and address. Each car can be identified by car model, liscence plate, car make. Each car is associated with zero to any number of accidents. Accident has accident date and location.



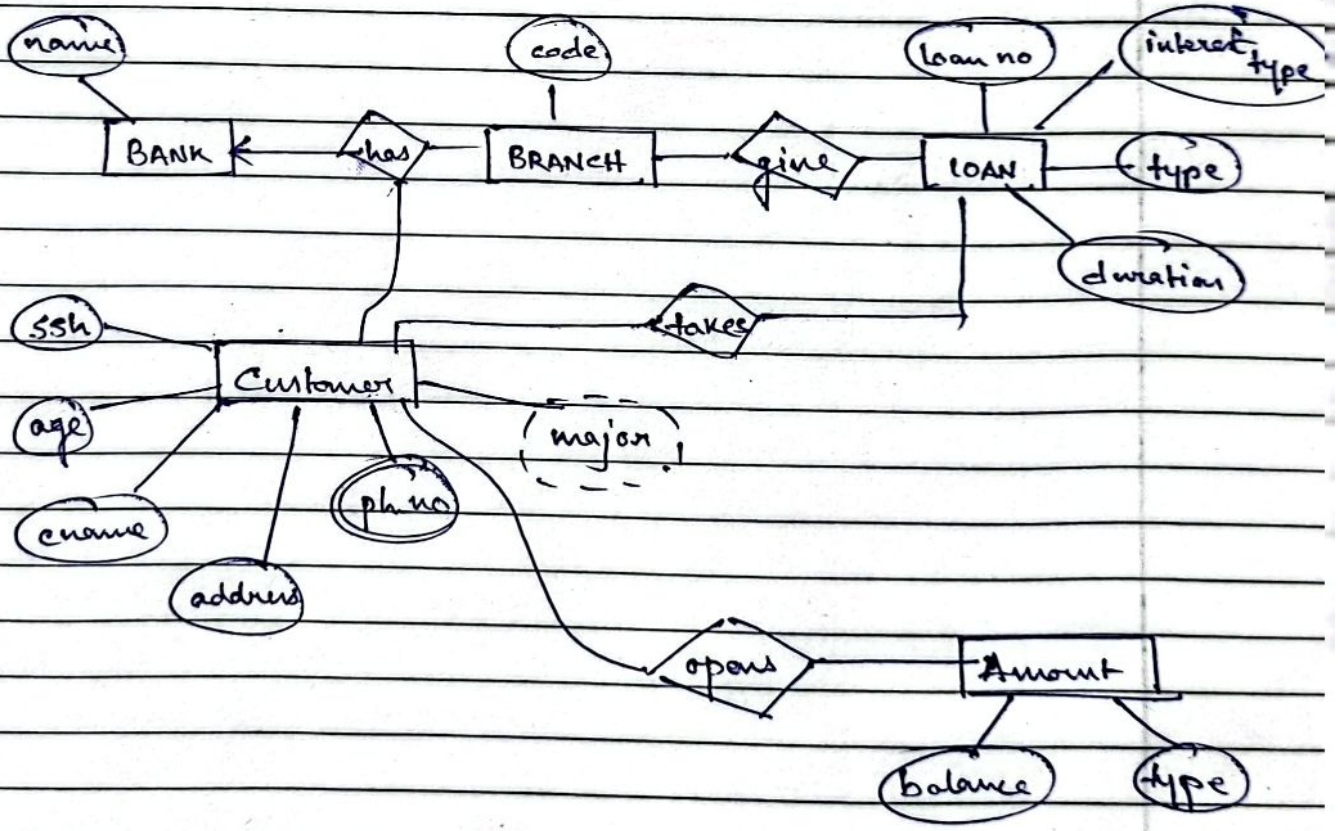
Q



Issues with ER Model

(i) Use of entity set vs attributes
(ii) Use of entity set vs relationship set
(iii) Use of binary vs nary relationship set
(iv) Placement of relationship attributes

- For one-one you can add the descriptive attribute to any side of the entity.
- For one-many you have to add the descriptive attribute to the many side entity.
- For many many you have to keep the descriptive attribute with the relationship.

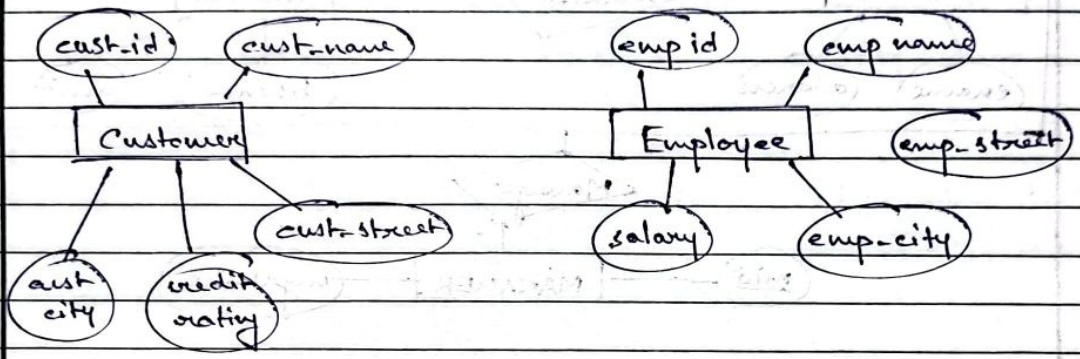


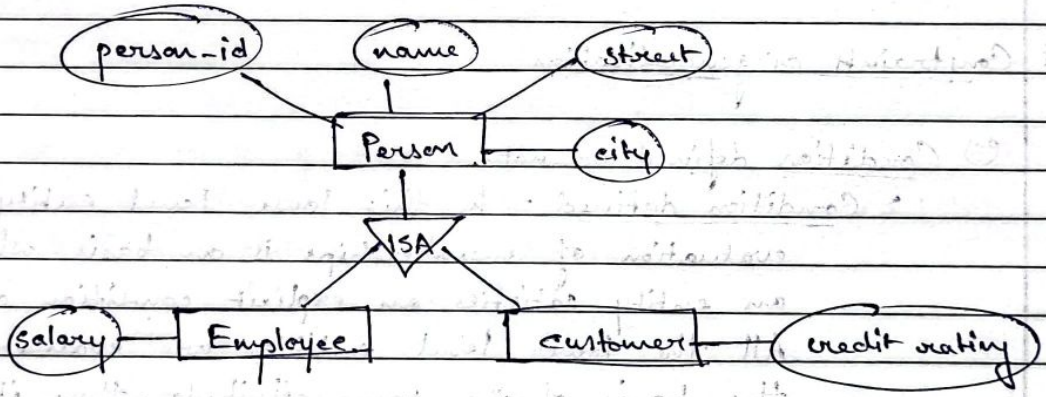
Extended ER Model

1. Specialization → ▽ ISA
2. Generalization ↗
3. Aggregation

Page No:
Date:

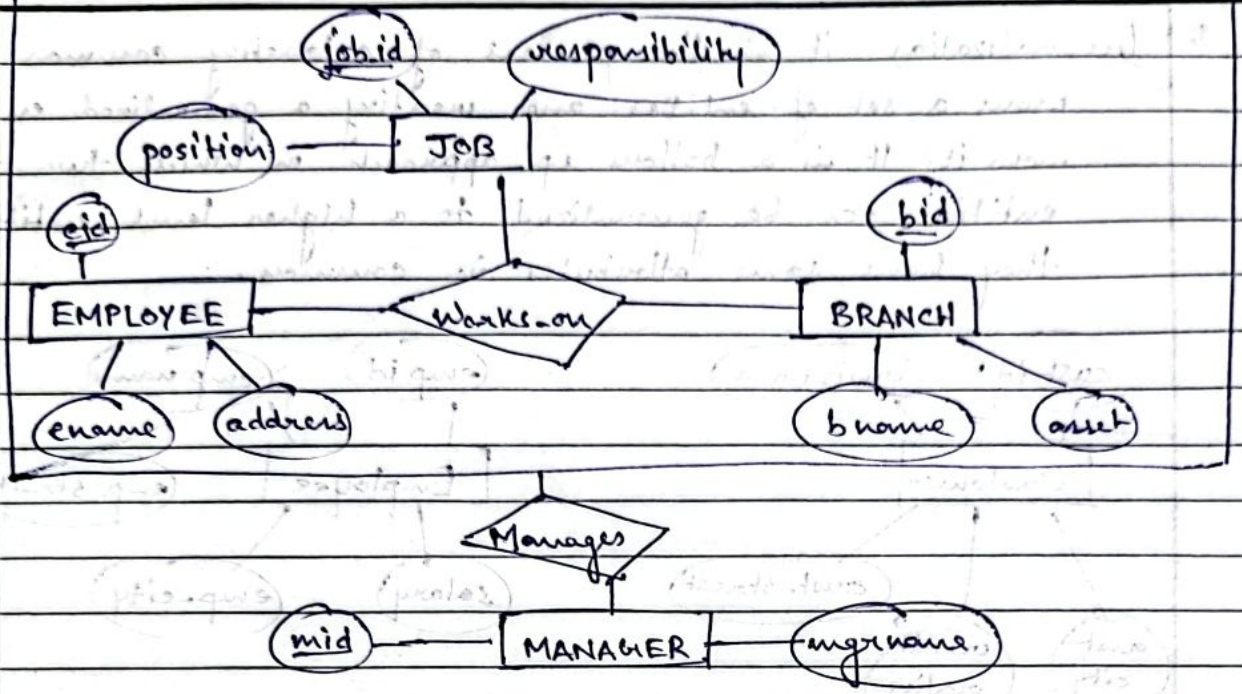
- Generalization It is the process of extracting common properties from a set of entities and creating a generalized entity from it. It is a bottom up approach in which two or more entities can be generalized to a higher level entity if they have some attributes in common.





- Aggregation An ER Model is not capable of representing the relationship b/w an entity and a relationship which may be required in some scenarios. In those cases, a relationship with its corresponding entities is aggregated into a higher level entity.

  Aggregation is an ~~entity~~ abstraction through which we can represent relationships as higher level entity set.



Constraints on generalization

(a) Condition defined or not

- Condition defined - In this lower level entity sets, evaluation of memberships is on basis whether an entity satisfies an explicit condition or not. All the lower level entities are evaluated on the basis of the same attribute, thus it is also referred to as attribute defined.

- User defined - In this lower level entity sets are not constrained by a condition named membership; users of ~~enti~~ database assigns entities to a given entity set.

(b) Disjoint or overlapping

- Disjoint - A ~~disjointe~~ disjointness constraint requires that an entity belongs to only one lower level entity set.

- Overlapping - In overlapping generalizations, the same entity may belong to more than one lower level entity set within a single generalization.

(c) Completeness Constraint

- Total generalization / specialization - According to this constraint, each higher level entity must belong to a lower level entity set.

- Partial generalization / specialization - According to this constraint, some higher level entities may not belong to any lower entity set.

Relational Model

The relational model represents how data is stored in Relational databases. A relational database consists of a collection of tables, each of which is assigned a unique name. Consider a relation STUDENT with attributes ROLL_NO, NAME, ADDRESS, PHONE and AGE shown in the table

Table STUDENT

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI		18

Attributes : Attributes are the properties that define an entity. eg. ROLL_NO, NAME, ADDRESS

Relation Schema : A relation schema defines the structure of the relation and represents the name of the relation with its attributes. EX.: STUDENT (ROLL_NO, NAME, ADDRESS PHONE AND AGE) is the relation schema for STUDENT. If a schema has more than 1 relation, its called as relation schema.

Tuple: Each row in the relation is known as tuple

Relation Instance: The set of tuples of a relation at a particular instance of time is called relation instance

Degree: The number of attributes in the relation is known as degree of the relation.

Cardinality: The number of tuples in a relation is known as Cardinality.

Relation Key: These are basically the keys that are used to identify the rows uniquely or also help in identifying tables.

- **Primary Key:** Is a set of attributes which uniquely identify the tuples in relation or table. The primary key is a minimal super key, so there is one and only one primary key in any relationship

- **Candidate Key:** A candidate key is a set of attributes which uniquely identify the tuples in relation or table. Any attribute can contain NULL value. There can be more than one candidate key.

- **Super Key:** Super key is an attribute that is used to uniquely identifies all attributes in a relation. All super keys can't be candidate keys but the reverse is true. In relation, a number of super keys is more than the number of candidate keys.

- Foreign Key : A foreign key is a column or a group of columns used to identify a row uniquely of a different table

Relational Database

A relational database is a database consisting of multiple relations or tables. The information about an enterprise is broken up into parts, with each relation storing one part of the information.

Constraints in relational model While designing the relational model, we define some conditions which must hold for data present in the database are called Constraints.

- Domain Constraints : All the values that appear in a column of a relation must be taken from the same domain. This constraint can be applied by specifying a particular data type to a column.

- Entity Integrity : The entity integrity rule is designed to assure that every relation has a primary key and that the data values for that primary key are all valid. Usually the primary key of each relation is the first column

  Entity Integrity ensures that every primary key attribute is NOT NULL. Primary key performs the unique identification function in a relational model.

- Referential Integrity: In relational data model, associations b/w tables are defined by using foreign keys. A referential integrity ~~constant~~ constraint is a rule that maintains consistency among the rows of two ~~columns~~ relations.

  The rule states that if there is a foreign key in one relation, either each foreign key must match a primary key value in other table or else foreign key must be NULL

A foreign key that references its own relation is known as recursive foreign key. The linking b/w the foreign key and primary key allows a set of relations to form an integrated database.

- Operational Constraints :- These are the constraints enforced in the database by the business rules or real world limitations.

DATABASE LANGUAGES

① DDL (Data Defination Language)
It is used to define the conceptual schema - The defination includes the information of all the entity sets and their associated attributes as well as the relationships b/w the entity sets.
CREATE, ALTER, DROP, RENAME & TRUNCATE

② DML (Data Manipulation Language)
It is used to manipulate data in the database. A query is a statement in the DML that requests the retrieval of data from the database
SELECT, INSERT, UPDATE & DELETE

③ DCL (Data Control Language)
It allows changing the permissions on database structures
GRANT & REVOKE

④ TCL (Transaction Control Language)
It allows permanently recording the changes made to the rows stored in a table or undoing such changes.
COMMIT, ROLLBACK & SAVEPOINT

Query Languages

Language in which user requests information from the database are :

- Procedural language
- Non procedural language

The categories of different languages are :

- SQL
- Relational Algebra
- Relational Calculus
  - Tuple Relational Calculus
  - Domain Relational Calculus

Relational Algebra

Relational Algebra is a procedural language for manipulating relations. Relational algebra operations manipulate relations. That is, these operations use one or two existing relations to create a new relation.

- Fundamental Operators
  - Unary : SELECT, PROJECT, RENAME
  - Binary : UNION, SET DIFFERENCE, CARTESIAN PRODUCT

- Secondary Operators
  - INTERSECTION, NATURAL JOIN, DIVISION, ASSIGNMENT

The relational schemas used for different operations are :

- Customer (cust_name, cust_street, cust_city)
  - used to store customer details
- Branch (branch_name, branch_city, assets)
  - used to store branch details
- Account (acc_no, branch_name, balance)
  - stores the account details
- Loan (loan_no, branch_name, amount)
  - stores the loan details

- Depositor (cust_name, acc_no)
  - stores the detail about customers account
- Borrower (cust_name, loan_no)
  - used to store the details about the customers loan.

- SELECT OPERATOR (σ)

SELECT Operation is used to create a relation from another relation by selecting only those tuples or rows from the original relation that satisfy a specified condition. It is denoted by sigma (σ) symbol. This predicate appears as a subscript to σ.

The general syntax of select operator is

$$\sigma_{<\text{selection-condition}>}(<\text{relation name}>)$$

Query: Find the details of the loans taken from 'Bhubaneshwar Main' branch.

$$\sigma_{\text{branch\_name} = \text{'BhubaneshwarMain'}}(\text{Loan})$$

- The operators used in selection predicate may be $=, \neq, <, \leq, >, \geq$.

- Different predicates can be combined into a larger predicate by using connectors like : AND ($\wedge$), OR ($\vee$), NOT ($\neg$)

- PROJECT OPERATOR (π)

PROJECT Operation can be thought of as eliminating unwanted columns. It eliminates the ~~unwanted~~ duplicate rows. It is denoted by pie (π) symbol. The attributes needed to be appeared in the resultant relation appear as a subscript to π

The general syntax of project operator is

$$\pi_{<\text{attribute-list}>}(<\text{relation name}>)$$

Composition of relational operators

Relational algebra operators can be expressed together into a relational algebra expression to answer complex queries.

Q Find the name of the customers who live in Bhubaneshwar -

Customer

cust-name	cust-street	cust-city
Rishi	India bnk	New Delhi
Sarthak	MG Road	Bangalore
Manas	Shastri Nagar	Bhubaneshwar
Ramesh	MG Road	Bhubaneshwar
Mahesh	Juhu	Mumbai

$\Pi_{cust\_name}(\sigma_{cust\_city = 'Bhubaneshwar'}(Customer))$

cust-name
Manas
Ramesh

RENAME Operator

The results of relational algebra expressions do not have a name that can be used to refer them. It is useful to be able to give them names; the rename operator is used for this purpose. It is denoted by rho (ρ) symbol.

The general syntax of rename operator is

$\rho_X(E)$

UNION Compatibility

To perform the set ~~intersection~~ operations such as UNION, DIFFERENCE and INTERSECTION, the relations need to be union compatible for the result to be a valid relation.

$R1(a_1, a_2, \ldots a_n)$ and $R2(b_1, b_2, \ldots b_m)$

(i) $n = m$ i.e. both relations have same arity

(ii) $dom(a_i) = dom(b_i)$ for $1 \le i \le n$

Page No:
Date:

UNION OPERATOR (∪)

The Union Operation is used to combine data from two relations. It is denoted by union (∪) symbol. The union of two relation $R_1(a_1, \dots a_n)$ and $R_2(b_1, b_2 \dots b_n)$ is a relation $R_3(c_1, c_2 \dots c_n)$ such as

$$dom(ci) = dom(ai) \cup dom(bi), \quad 1 \le i \le n$$

DIFFERENCE Operator (−)

The Difference Operation is used to identify the rows that are in one relation and not in another. It is denoted as (−) symbol. The difference of two relations $R_1(a_1, a_2, \dots a_n)$ and $R_2(b_1, b_2, \dots b_n)$ is a relation $R_3(c_1, c_2, \dots c_n)$ such that

$$dom(ci) = dom(ai) - dom(bi), \quad 1 \le i \le n$$

$R_1 - R_2$ is a relation that includes all tuples that are in R_1 but not in R_2.

Cartesian Product Operator (×)

The Cartesian Product of two relations $R_1(a_1, a_2 \dots a_n)$ with cardinality i and $R_2(b_1, b_2 \dots b_m)$ with cardinality j is a relation R_3 with

- degree $K = n + m$
- cardinality $i * j$
- attributes $(a_1, a_2, \dots a_n, b_1, b_2, \dots b_m)$

$R_1 \times R_2$ is a relation that includes all the possible combinations of tuples from R_1 and R_2.

Intersection Operator ($\cap$)

The intersection operator is used to identify the rows that are common to two relations. It is denoted by ($\cap$) symbol. The intersection of two relations $R_1(a_1, a_2, \ldots a_n)$ and $R_2(b_1, b_2, \ldots b_n)$ is a relation $R_3(c_1, c_2, \ldots c_n)$ such that

$$dom(c_i) = dom(a_i) \cap dom(b_i), \quad 1 \le i \le n$$

$R_1 \cap R_2$ is a relation that includes all tuples that are present in both R_1 and R_2

$$R \cap S = R - (R - S)$$

Join Operator ($\bowtie$)

The join is a binary operation that is used to combine certain selections and a cartesian product into one operation. It is denoted by $\bowtie$ symbol.

The join operation forms a cartesian product of its two arguments, performs a selection forcing ~~equity~~ equality on those attributes that appear in both relations and finally removes the duplicate attributes

Query: Find the names of customers who have a loan at the bank, along with the loan number and the loan amount

(i) $\Pi_{cust\_name, loan.loan\_no, amount}(\sigma_{Borrower.loan\_no = loan.loan\_no}(Borrower \times Loan))$

(ii) $\Pi_{cust\_name, loan\_no, amount}(Borrower \bowtie Loan)$