PAGE No
DATE / /

DBMS - GB

1. DBA - DataBase Administrator controls, maintains, coordinates

Database Administrator (DBA) is a person / team who defines the schema and also controls the 3-levels of database. (Physical, Conceptual, ~~Internal~~ External).
The DBA will then create a new account id and password for the user if he/she needs to access the database.

Types of DBA: Cloud DBA, Administrative DBA, Data Warehouse DBA, Application DBA, Database Analyst.

Roles of DBA.

1. DataBase Design:
DataBase administrator has the responsibility of designing a database that meets the demand of users.

2. DataBase availability:
Responsibility of ensuring data accessibility to users from time to time.

3. DataBase Backup:
Responsibility to backup every data in database.

4. DataBase security:
To protect data and ensure adequate security in a organizations database.

5. Capacity planning:
Responsibility of planning for increased capacity, in case of sudden growth in database need.

6. Database monitoring:
Responsibility of monitoring database and movement of data in database.

7. DataMove:
Responsibility of moving a database set, eg. from physical box to cloud box. or from existing app to new app.

8 DataBase Restor:
Responsibility of Restoring a file from a backup stor

Responsibilites of DBA:

1. DBA is responsible for providing security to the data and allow only the authorized users to access the database.

2. Responsible for problems such as security breach and poor system response time.

3. Creating and maintaing database standards & polices.

4. Supporting database design, creation and testing activity

5. DBA monitors the recovery and backup and provide technical support.

6. DBA repairs damage caused due to hardware and software failures.

7 Performing database housekeeping such as tuning, indexing

8 Administrating database objects to achieve optimum utilization

9. Designing database backup, archiving and storage capacity

PAGE No.
DATE / /

Data Independence.

Data Independence is a property of DBMS that help you to change the database schema at one level of a system without requiring to change the schema at the next level. It helps to keep the data seperated from all programs that makes use of it.

There are two types of data independence:

Physical Data Independence:

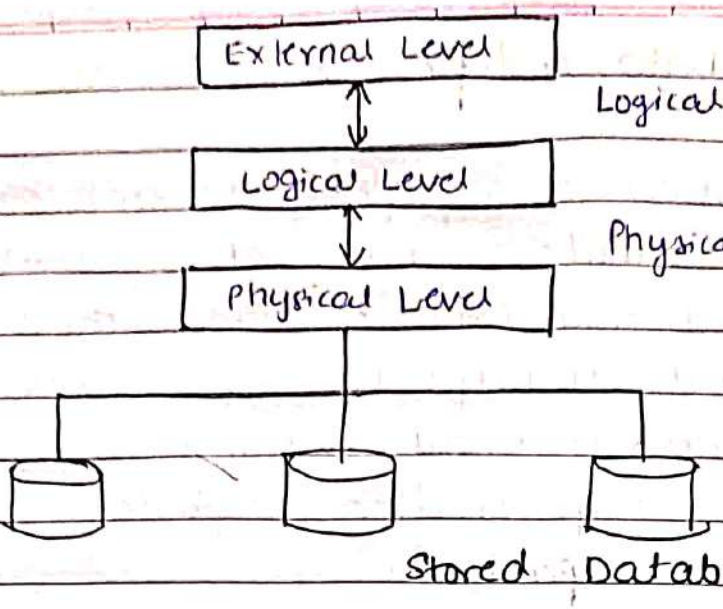
- It refers to the characteristic of being able to modify the physical schema (internal schema) without any change to Logical (conceptual) schema.
- Eg. Conceptual structure of the database would not be affected by any change in storage size of the database system server.
- Eg: Changing from sequencial to random access files.

Modification to the physical structure include:

- Utilizing new storage devices.
- Modifying data structures used for storage.
- Altering index or using alternative file organization techniques etc.

Logical Data Independence:

- It refers to characteristic of being able to modify the logical schema (conceptual) without affecting the external schema.
- The user view of the data would not be affected by any changes to the conceptual view of data.
- Eg. insertion or deletion of attributes, altering table structures entities or relationships to the logical schema, etc.



3 DBMS Architecture:

- A Database Architecture is a representation of DBMS design.
- It helps to design, develop, implement and maintain the DBMS.
- DBMS architecture allows dividing the database system into individual components that can be independently modified, changed, replaced and altered.
- It also helps to understand the components of a database.

A Database stores critical information and helps access data quickly and securely. Therefore selecting the correct Architecture of DBMS helps in easy and efficient data management.

Types of DBMS Architecure.

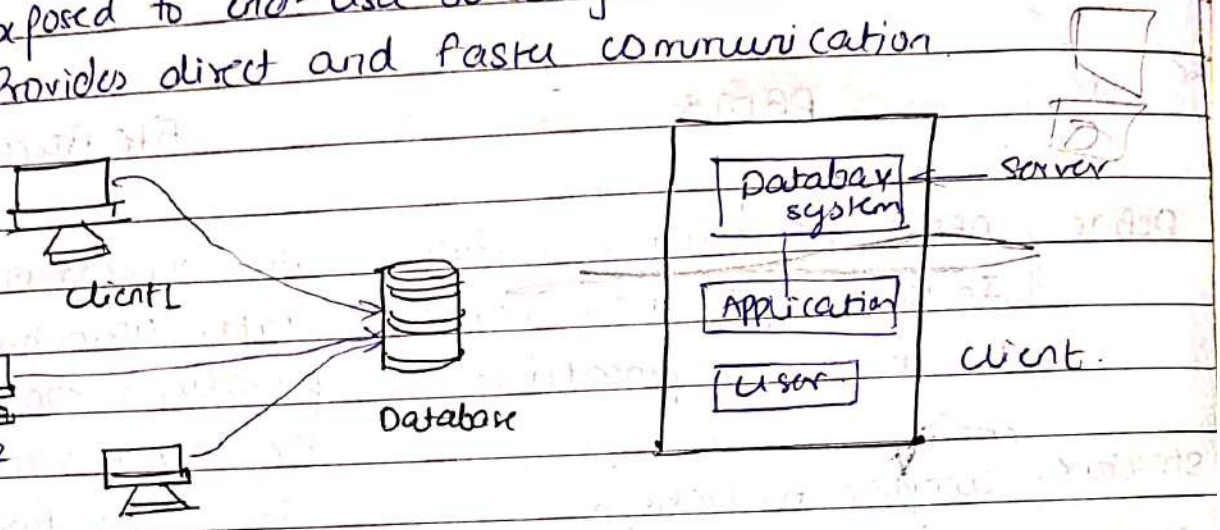
1- Tier Architecture -

- Simplest architecture of Database in which the client, server and Database all reside on the same machine.
- Ex: anytime you install a Database in your system and access it to practice SQL queries.
- Such architecture is rarely used.

PAGE No.
DATE / /

2-Tier Architecture:

- Presentation layer runs on a client (PC, Mobile) and data is stored on a server called second tier.
- Two tier architecture provides security to DBMS as it is not exposed to end-user directly.
- It Provides direct and fastr communication.



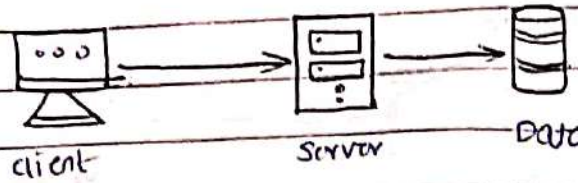
One server is connected to client 1,2,3 contact management system created using MS-Access

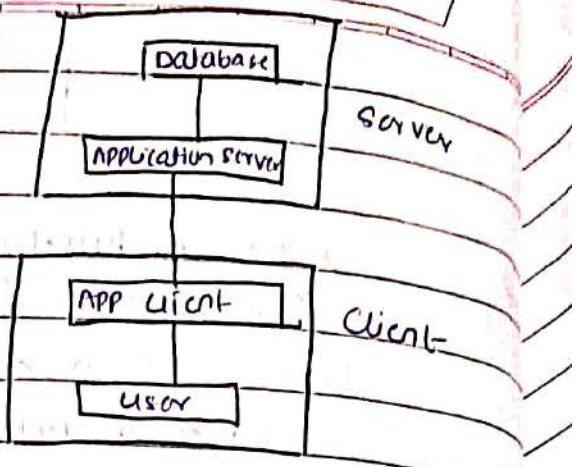
3 tier Architecture:

- most popular client architecture in DBMS in which development and maintainance of functional process, logic, data access, data storage and user interface is done independently.
- Contains Presentation Layer, application Layer & Database server.
- It is an extention of 2 tier client-server architecture.
- Application Layer resides between user and DBMS, which is responsible for communicating the user's request to the DBMS and send response from DBMS to the user.

Goal: seprate user apps & physical DB

support DBMS changes
Program-data independence.





Uses: any large website on internet including google.com

4.	DBMS	File Processing System
Define:	DBMS is a collection of data. In DBMS, user is not required to write the procedures	File system is a collection of data. User has to write the procedures for managing the databases.
Structure:	Complex to Design	Simple Structure
Sharing data.	Due to centralized approach, data sharing is easy	Data is distributed in many files, and of different formats, so it is not easy to share.
Data Abstraction	DataBase Management System gives an abstract view of data that hides the details	Provides the detail of the data representation and storage of data
Security	DBMS provides a good security mechanism.	It isn't easy to protect a file under file system.
Data Redundancy & Inconsistency	Due to centralization of the database, the problems of data redundancy and inconsistency are controlled.	The files and application programs are created by different programmers so there exist a lot of duplication of data which leads to inconsistency
Cost:	Expensive to design	cheaper to design
Ex:	Oracle, SQL server, Sybase etc	Cobol, C++ etc.

PAGE No.
DATE / /

6. ER Diagram Airline Reservation system

ER stands for Entity Relationship model.

It is a high-level data model used to define data elements and relationship for a specified system.

Entity:

i) USER – User Entity: iv) Airline – Airline Entity

ii) Flight – Flight Entity v) Payment – Payment Entity

iii) Ticket – Ticket Entity vi) Class – Class Entity.

Attributes:

USER

Key attribute: Username

Composit Attribute: (Name) → F_Name, M_Name, L_Name

USER

Multivalued attribute: Phone_NO

Derived attribute: DOB, Age

Composit Attribute:

Address → Pin_code, City, State

Gender.

Ticket

Key attribute: PNR_NO

Flight_no, Date_time, Passenger_N, Source, Destination

Seat_NO, Class

PAGE No.
DATE / /

Flight: key attribute: Flight_no.

Flight_Name, Dep_time, Arr_time, Sourc

Destination, Distance, Seat_avail

Class

Class_type, Fare, Flight_no

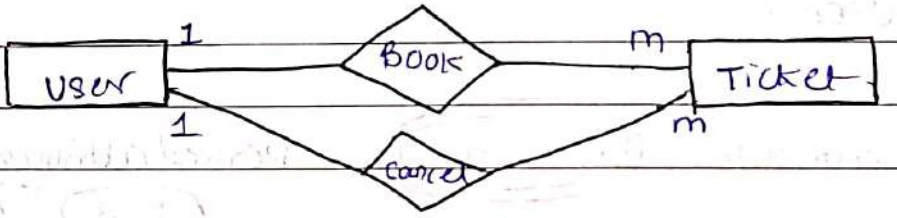
Airline

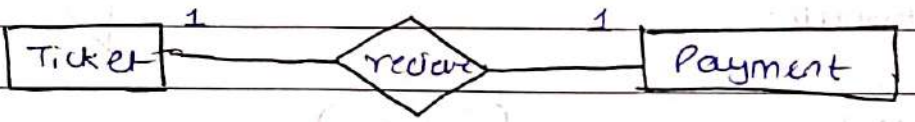
Airline_Name, contact_no ← Multivalued attribute

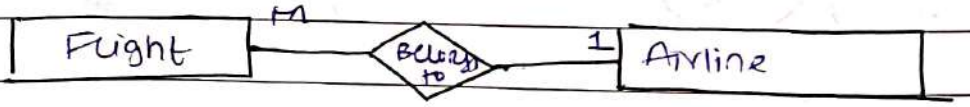
Payment

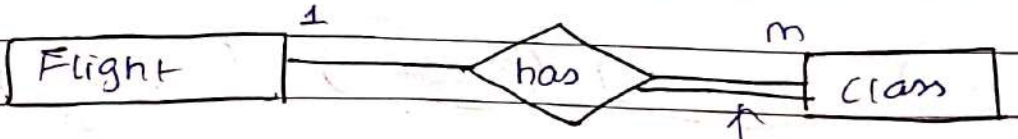
Transaction_id, Amount, Payment_mode

Relationships:

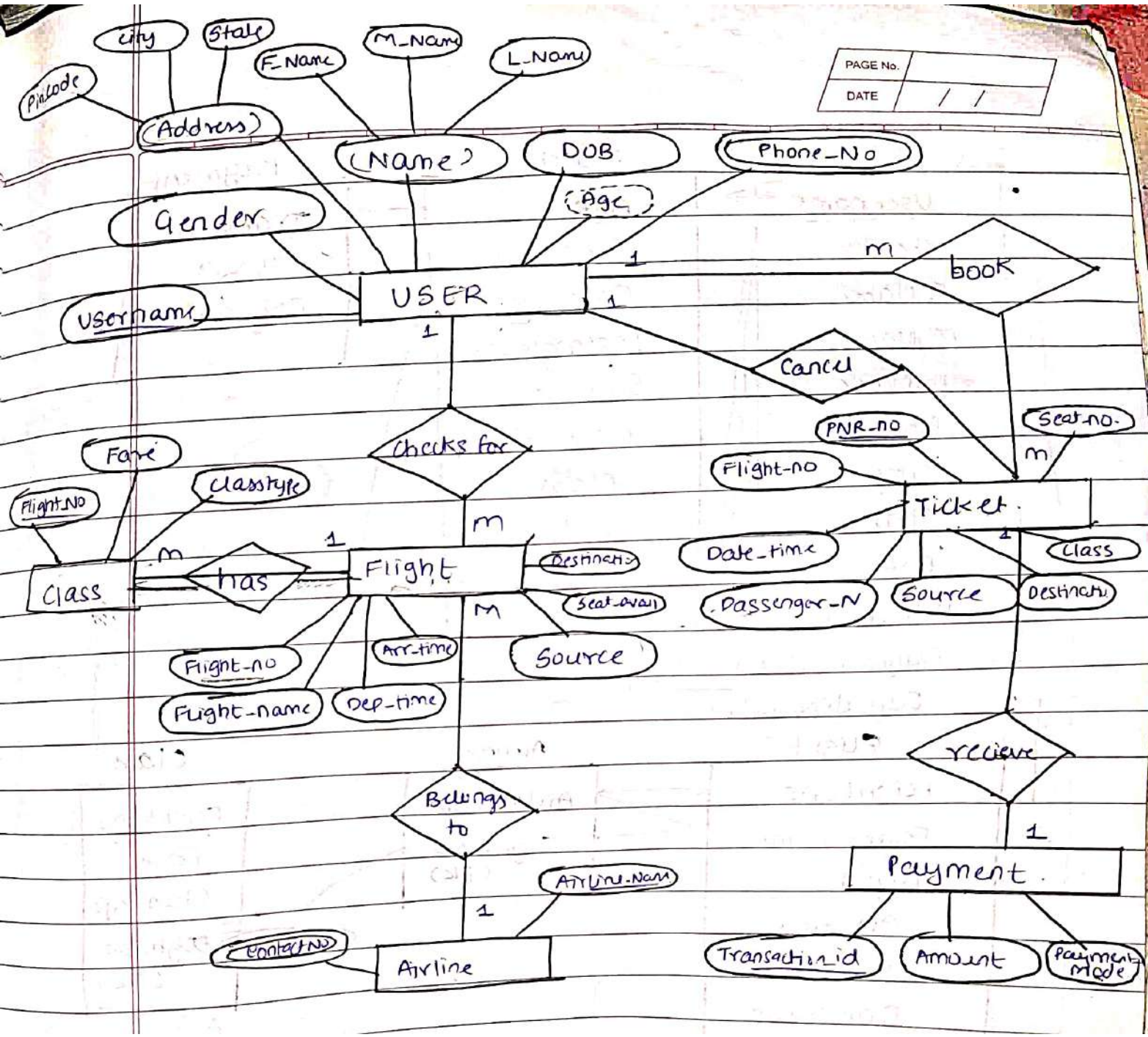


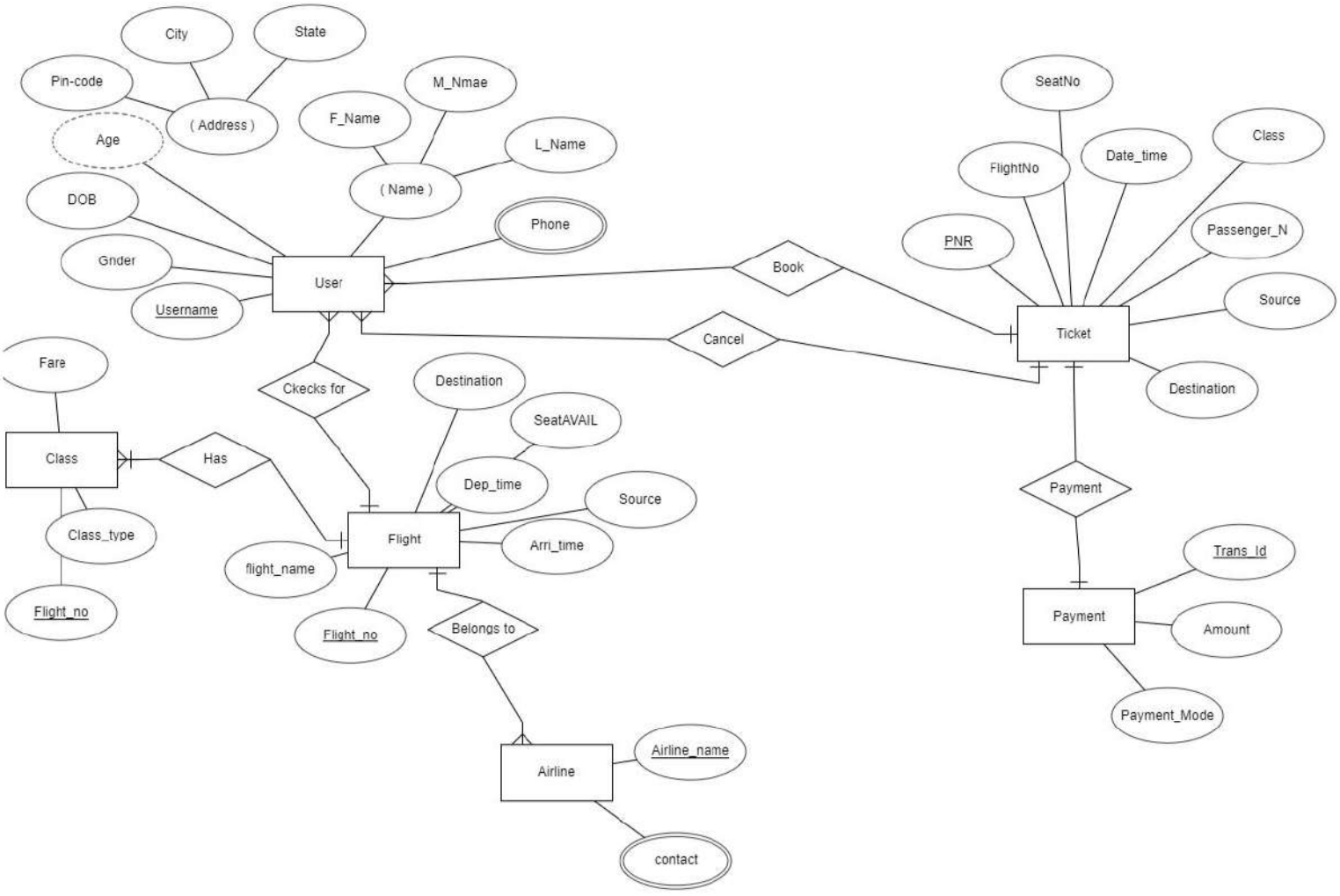


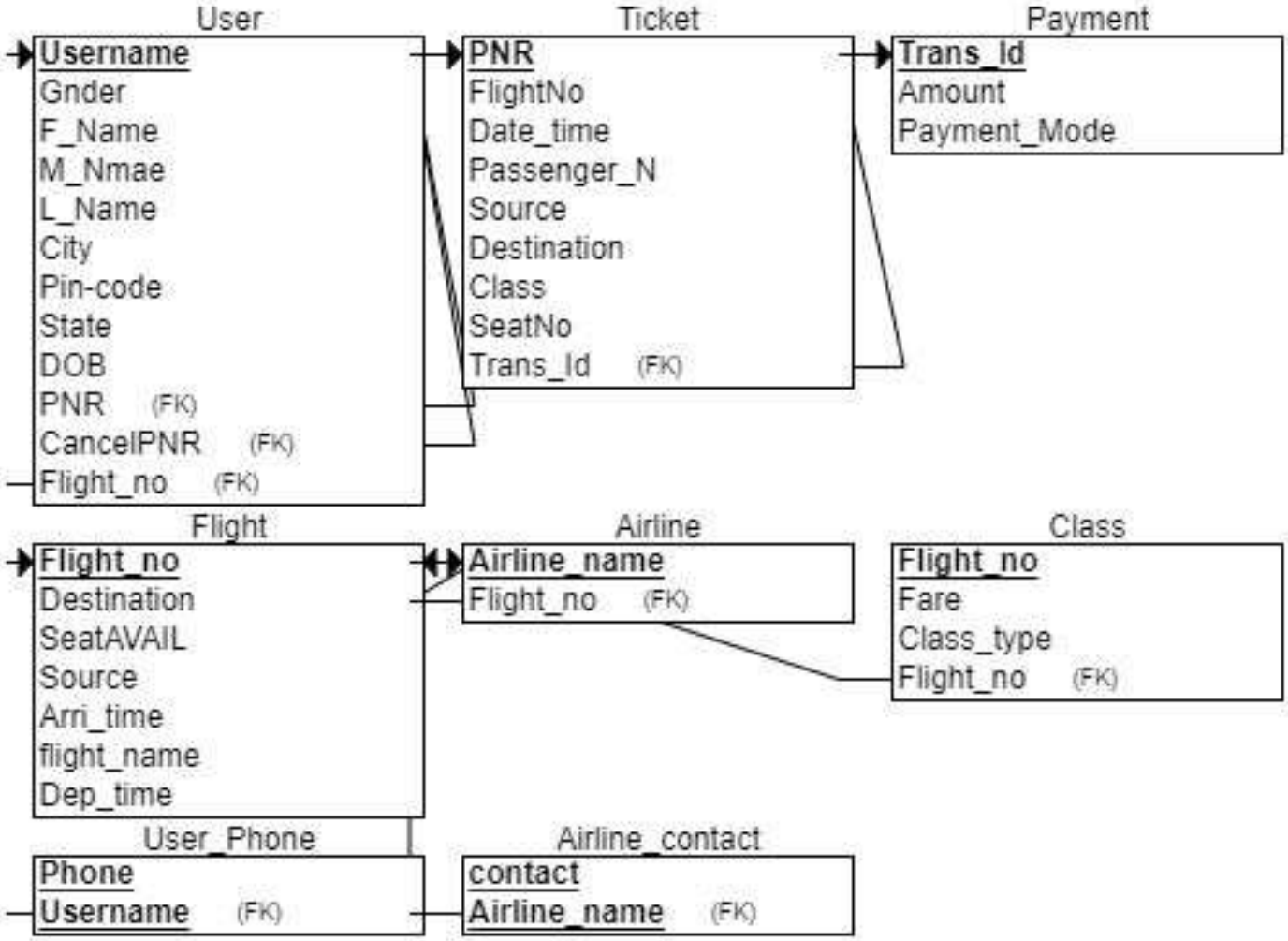




Total participation (must).

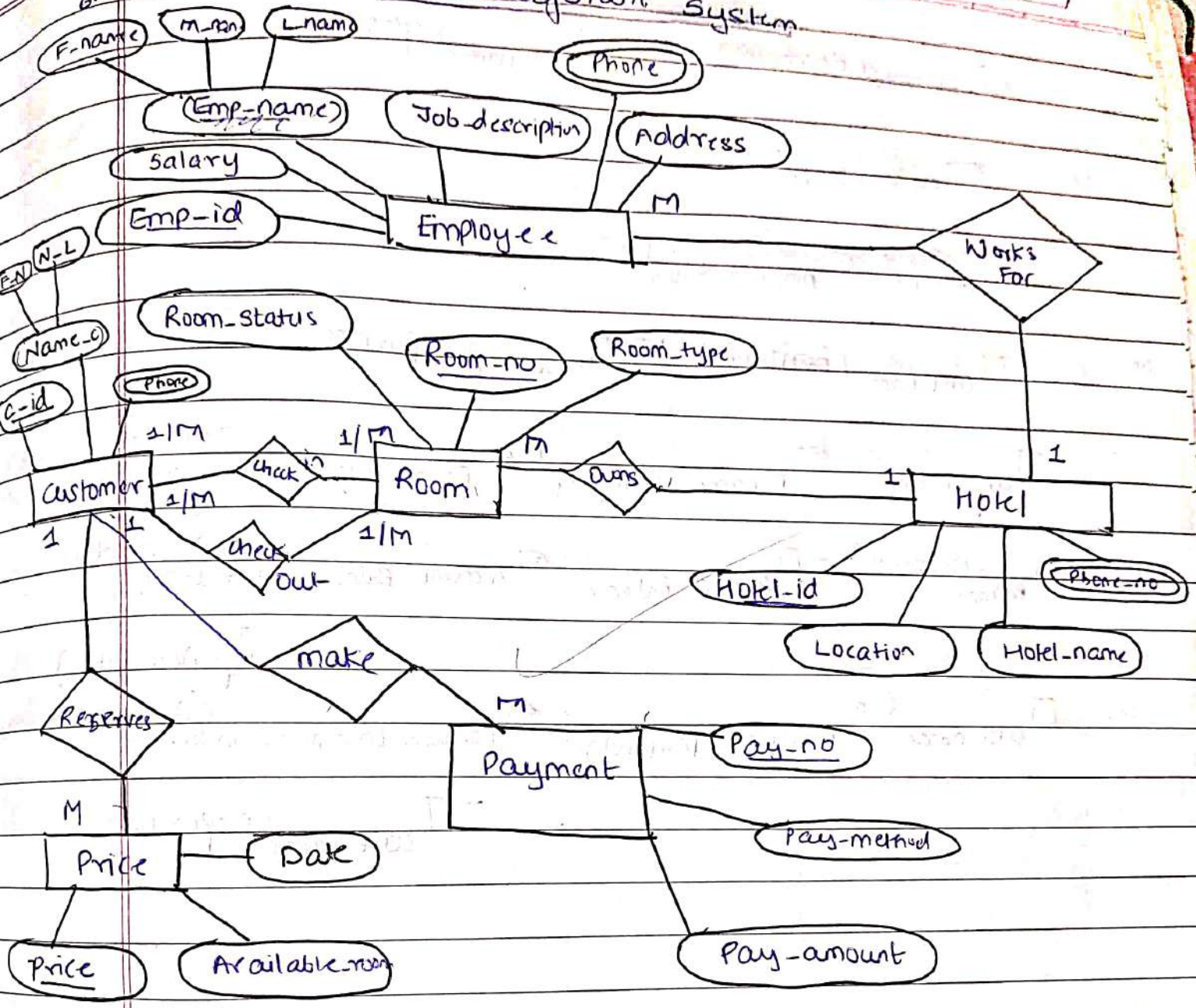

City
State
Pincode
Address
F_Name
M_Name
L_Name
Name
DOB
Age
Phone_No
Gender
USER
Username
book
Cancel
Checks for
Fare
Class type
Flight_NO
Class
has
Flight
Destination
Seat_avail
Source
Arr_time
Dep_time
Flight_no
Flight_name
Belongs to
Airline
Airline_Name
Contact_No
PNR_NO
Seat_no.
Flight_no
Ticket
Date_time
Passenger_N
Source
Destination
Class
recieve
Payment
Transaction_id
Amount
Payment Mode
1
m


City
State
Pin-code
(Address)
Age
M_Nmae
F_Name
L_Name
(Name)
DOB
Phone
Gnder
User
Username
Book
Cancel
SeatNo
Class
FlightNo
Date_time
Passenger_N
PNR
Source
Ticket
Destination
Fare
Ckecks for
Destination
SeatAVAIL
Class
Has
Dep_time
Source
Payment
Class_type
Flight
Arri_time
flight_name
Trans_Id
Flight_no
Flight_no
Belongs to
Payment
Amount
Payment_Mode
Airline
Airline_name
contact

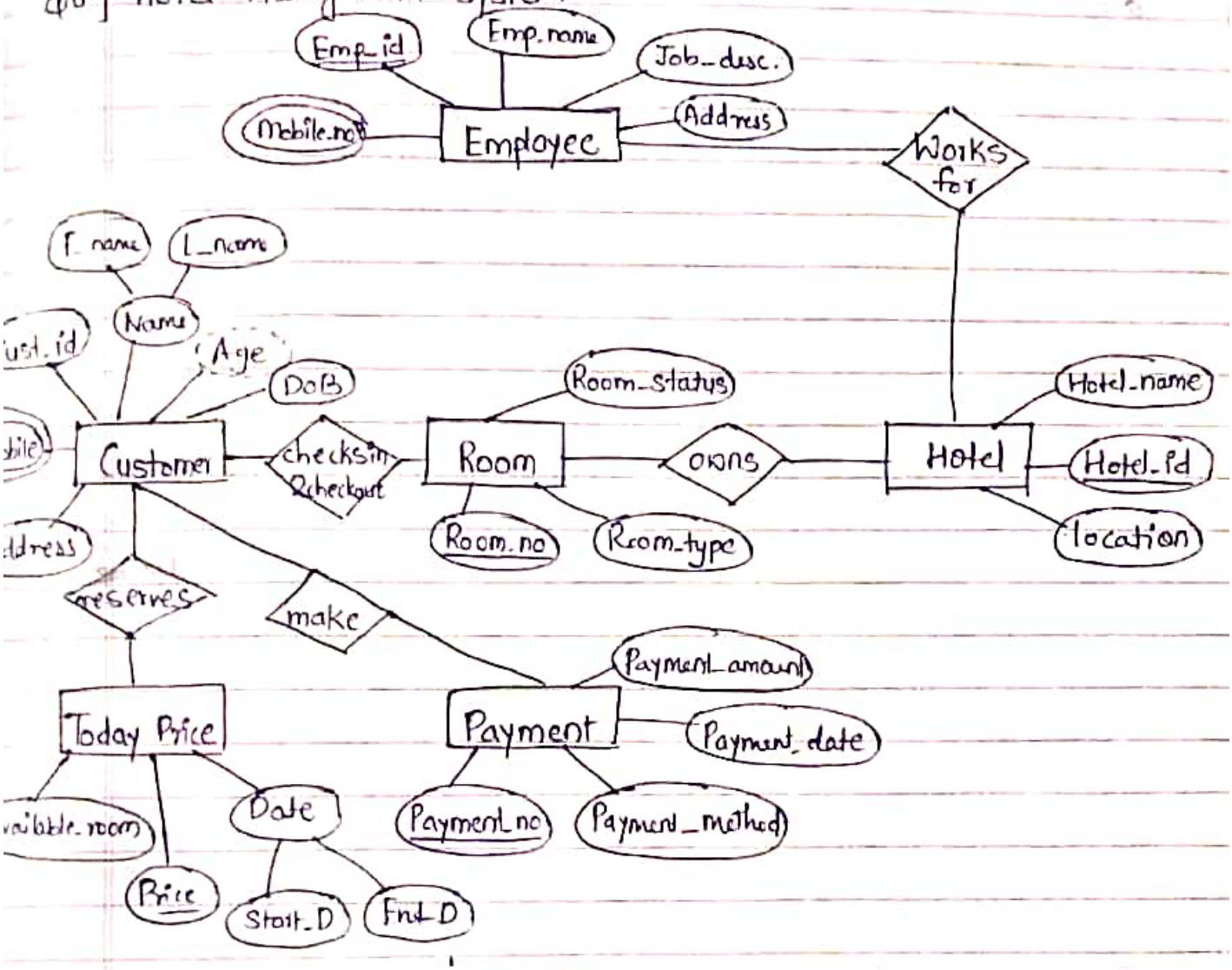

User
Username
Gnder
F_Name
M_Nmae
L_Name
City
Pin-code
State
DOB
PNR (FK)
CancelPNR (FK)
Flight_no (FK)
Ticket
PNR
FlightNo
Date_time
Passenger_N
Source
Destination
Class
SeatNo
Trans_Id (FK)
Payment
Trans_Id
Amount
Payment_Mode
Flight
Flight_no
Destination
SeatAVAIL
Source
Arri_time
flight_name
Dep_time
Airline
Airline_name
Flight_no (FK)
Class
Flight_no
Fare
Class_type
Flight_no (FK)
User_Phone
Phone
Username (FK)
Airline_contact
contact
Airline_name (FK)

6.

Hotel Management System



Q6.] Hotel Management System



Employee (

Hotel

Payment

Room

Customer

Today Price

Employee
Emp_id
Emp_name
Job_desc
Address

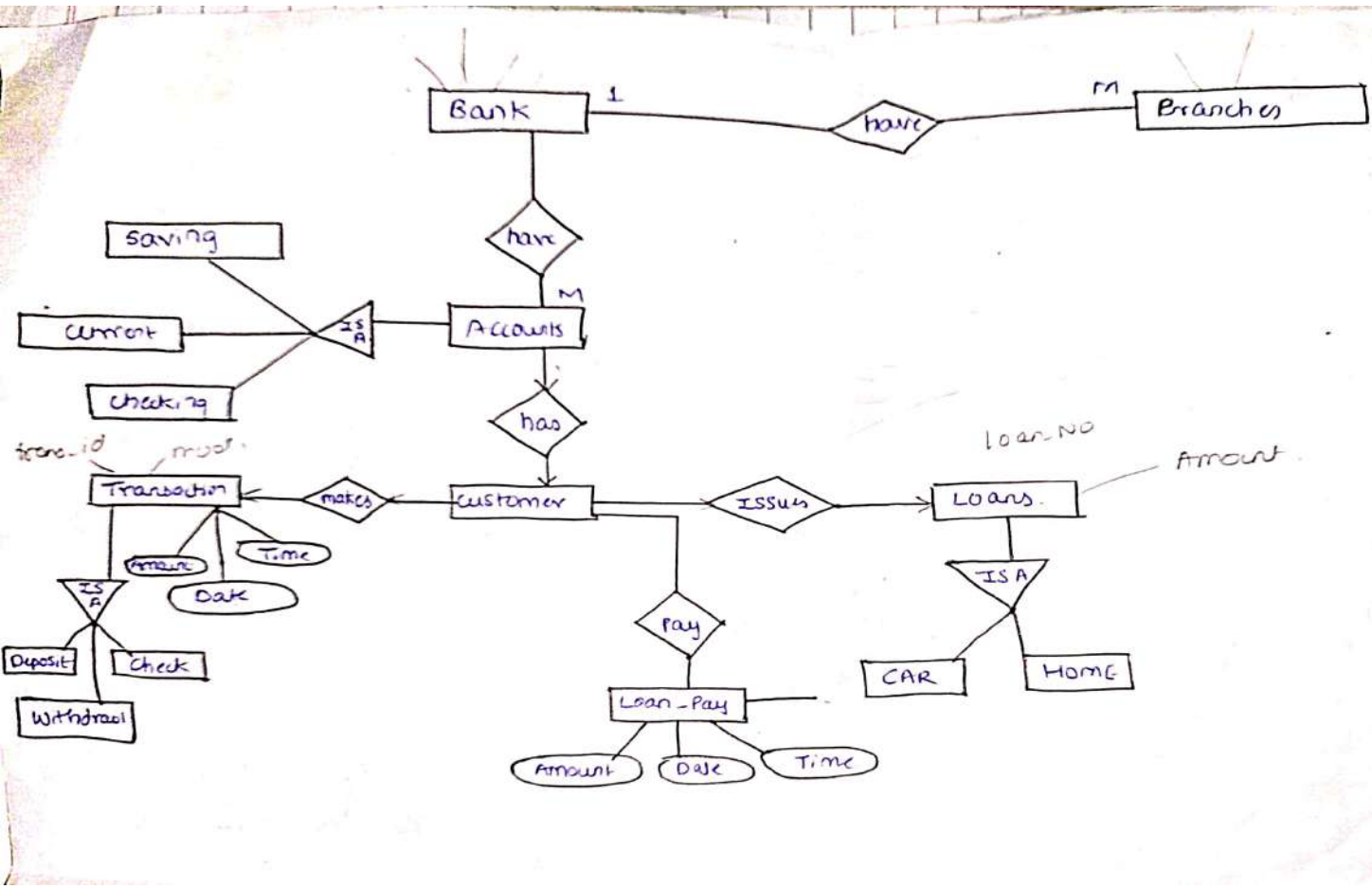
Hotel
Hotel_id
Hotel_name
location

Room
Room_no
Room_typ
Room_stat

Employee mobile
Mobile no
Emp_id (FK)

Today Price
Available room
Price
Start_D
End_D

Payment
Payment_no
Payment_amou
Payment_date


Bank
1
have
M
Branches
have
Saving
Current
IS A
Accounts
Checking
has
trans_id
must
Transaction
makes
Customer
ISSUS
Loan_NO
Amount
LOANS
Amount
Time
Date
IS A
IS A
Deposit
Check
Withdrawal
Pay
Loan-Pay
CAR
HOME
Amount
Date
Time

1. Cartesian Product: (JOIN)

In cartesian product two relations that is on two sets of tuples, it will take every tuple one by one from left set and will pair it up with all the tuples in right set.

so the Cartesian Product of two relation A(R1, R2, R3 .. Rp) with degree p and B(S1, S2, S3 .. Sn) with degree n is a relation C(R1, R2, R3 ... Rp, S1 .. Sn) with degree p+n attributes.

Notation:

S x r

where S and r are the Relation

x denotes cartesian Product.

Ex:

Student(SNo, FName, LName) and Details(RollNo, Age)

SNo	FName	LName	RollNo	Age
1	Sonal	Sonarghare	60	19
2	Tanvi	Panchal	35	18.

Student x Details

SNo	FName	LName	RollNo	Age
1	Sonal	Sonarghare	60	19
2	Tanvi	Panchal	35	18
1	Sonal	Sonarghare	35	18
2	Tanvi	Panchal	60	19

2. Generalized Projection:

Extends the projection operation by allowing arithmetic functions to be used in the projection list.

$$\Pi_{F_1, F_2, \ldots F_n}(E)$$

- E is any relational-algebra expression.
- Each $F_1, F_2 \ldots F_n$ are the arithmetic expressions involving constants and attributes in schema of E.
- Given credit_info(customer-name, limit, credit_balance), find how much more each person can spend:

$\Pi_{customer\_name,\ limit - credit\_balance}(credit\_info)$

3. Natural Join

A natural join is the set of tuples of all combinations in R and S are equal to their common attribute names. It is denoted by ⋈

Ex: Employee Table and Salary table.

Emp-id	Emp-name
101	Stephan
102	Jack

Emp-id	Salary
101	5000
103	2000

103

$\Pi_{Emp\_name,\ salary}(Employee \bowtie Salary)$.

Emp-name	Salary
Stephan	6000
Jack	2000

4. Union.

Union operator finds the result-set or combination of two or more tables.

Let R and S be two relations.

Then.

- R ∪ S is set of all tuples belonging to either R or S or both
- R ∪ S duplicates are removed automatically
- Union operation is both commutative & associative.

Relation R

Id	Name	Age
1	XYZ	20
2	RST	18

Relation S

Id	Name	Age
1	XYZ	20
3	ABC	15
4	IJK	25

R ∪ S

Id	Name	Age
1	XYZ	20
2	RST	18
3	ABC	15
4	IJK	25

5. Set Intersection:

Intersection Operator gives common data values between two tables.

Let R and S be two relations

Then:

1. R ∩ S is set of all tuples belonging to both R and S.
2. R ∩ S duplicates are automatically removed.
3. Intersection operation is both commutative & associative.

R ∩ S.

Id	Name	Age
1.	XYZ	20

6. Aggregation Operation:

Aggregate operation is used to perform the calculation on multiple rows of a single columns of a table.

It returns a single value.

Aggregate operators.

- max(): returns maximum value in a collection
- min(): returns minimum value in a collection.
- count(): counts no. of elements in a collection
- sum(): sums the value in the collection
- Avg(): computes average values in the collection.

Emp

Emp-id	Emp_salary
1	500
2	600

max(Emp_salary): 600

Avg(Emp_salary): 550

Count(Emp_salary): 2

$\mathcal{G}$ max(Emp_salary)(Emp) = 600

$\mathcal{G}$ avg(Emp_salary)(Emp)

$\mathcal{G}$ count(Emp_salary)(Emp)

$\mathcal{G}_{1,2,\ldots,n}\, G_{F_1(A_1), F_2(A_2), \ldots, F_n(A_n)}(E)$ → Entity

attribute

7. Select:

Select operation chooses tuples from a relation that satisfy the provided condition.

Notation is $\sigma_{p(r)}$

σ Stands for selection predicate

r relation

p proportional logic formula that may use connectors such as or, and and not.

and also relational operators such as $=, \neq, \geq, <$

Ex: select tuples from the novels whose subject = 'information'

$\sigma_{subject = "information"}(novel)$

8. Project:

It projects those columns that satisfy given condition.

Notation: $\prod_{B1, B2}(r)$

B1, B2 refers to attribute name

r name of relation.

Ex: selecting and projecting columns named writer as well as subject from Novel

$\Pi_{A,C}(R)$

$\prod_{writer, subject}(Novel)$

PAGE No.
DATE / /

8. Rename:

Rename operator allows to rename relation output.

Notation: $\rho_x(E)$

ρ denotes Rename operator

E relational algebra expression

x new Rename

Reasons to rename a relation:

- To save the result of a relational algebra expression as a relation so that we can use it later.
- To change existing relation into new name.
- To change columns name to new name.

Types:-

eg.

Stud

RNo	Name	Marks
1	A	70
2	B	80
3	C	82.

$\sigma_{marks>70}(Student)$

Resultant.

RNO	Name	Marks
2	B	80
3	C	82.

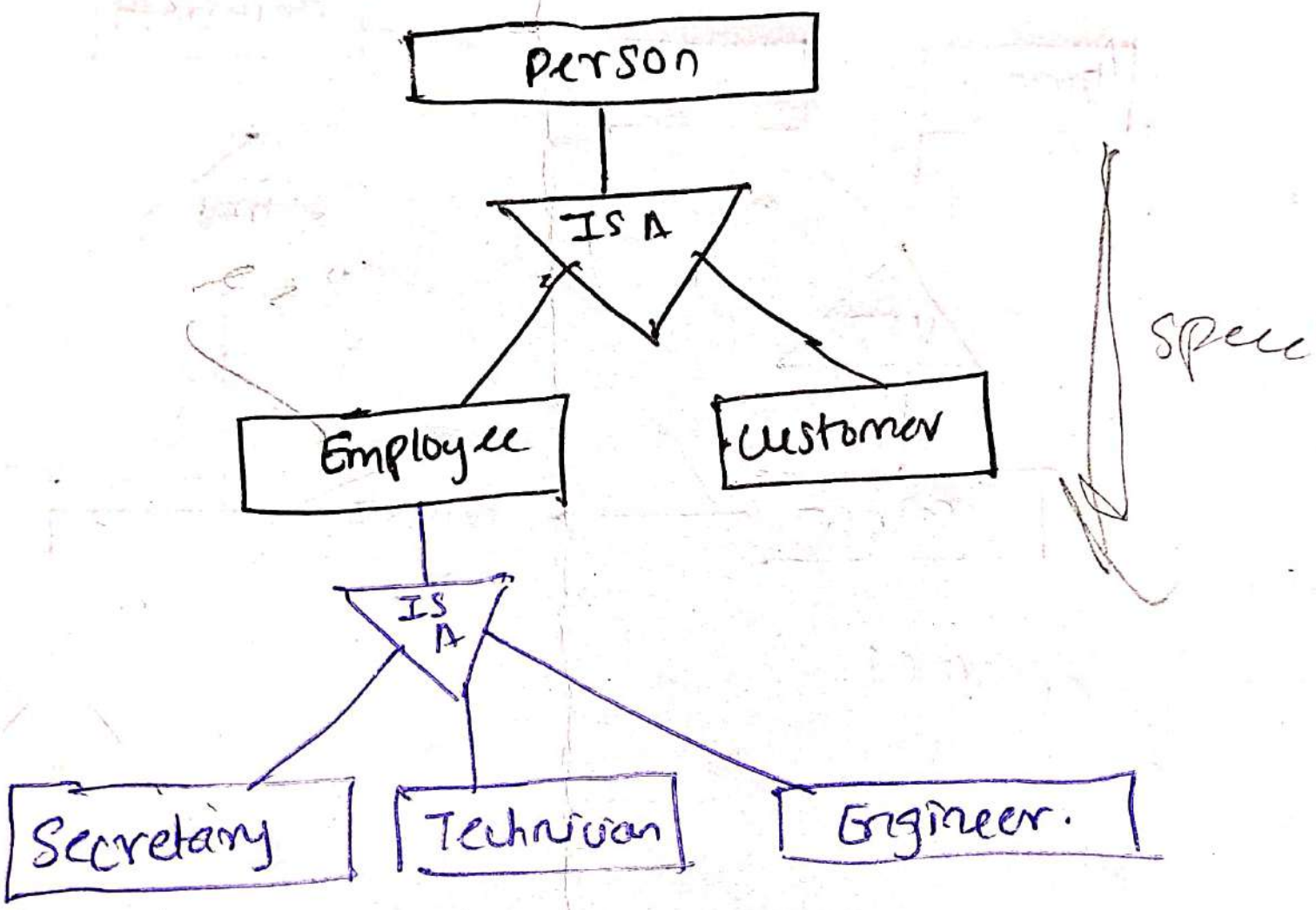
$\therefore \rho_{First\_class}(\sigma_{marks>70}(Stud))$

$\rho_{newname(Stude)} \Rightarrow \rho_{Student(Stud)}$.

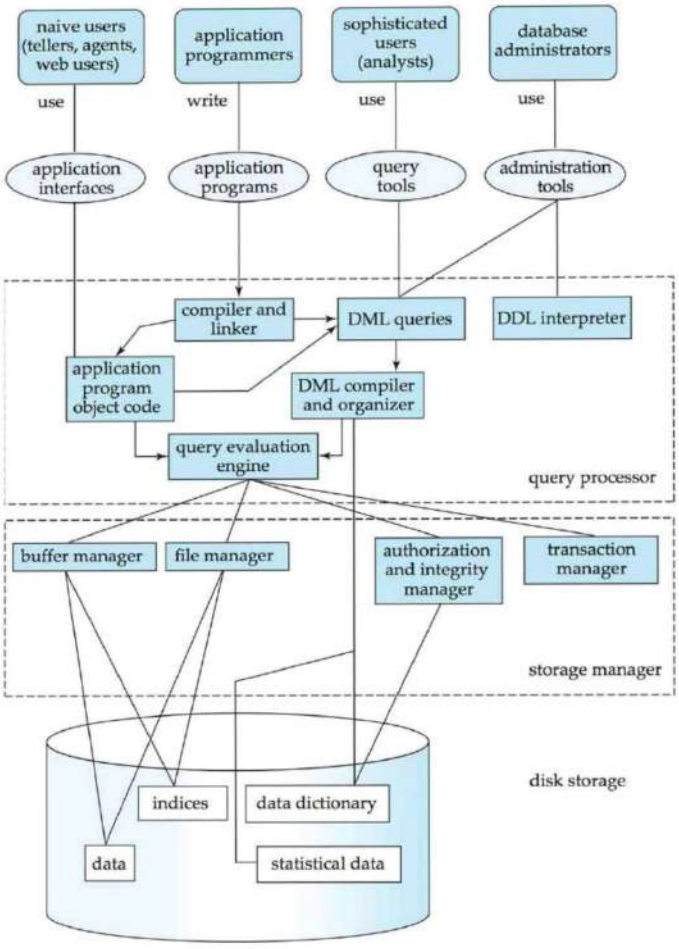
$\rho_{s\_id, s\_name, s\_percentage}(Student)$.

9).

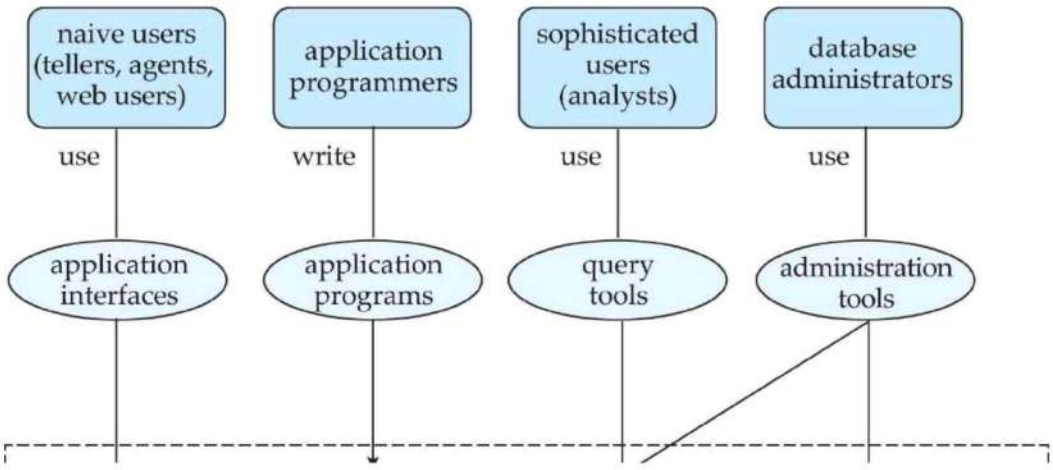
9)


Person
IS A
Employee
Customer
IS A
Secretary
Technician
Engineer.
Spec

Database System structure



Database Users and Administrators



Database Engine

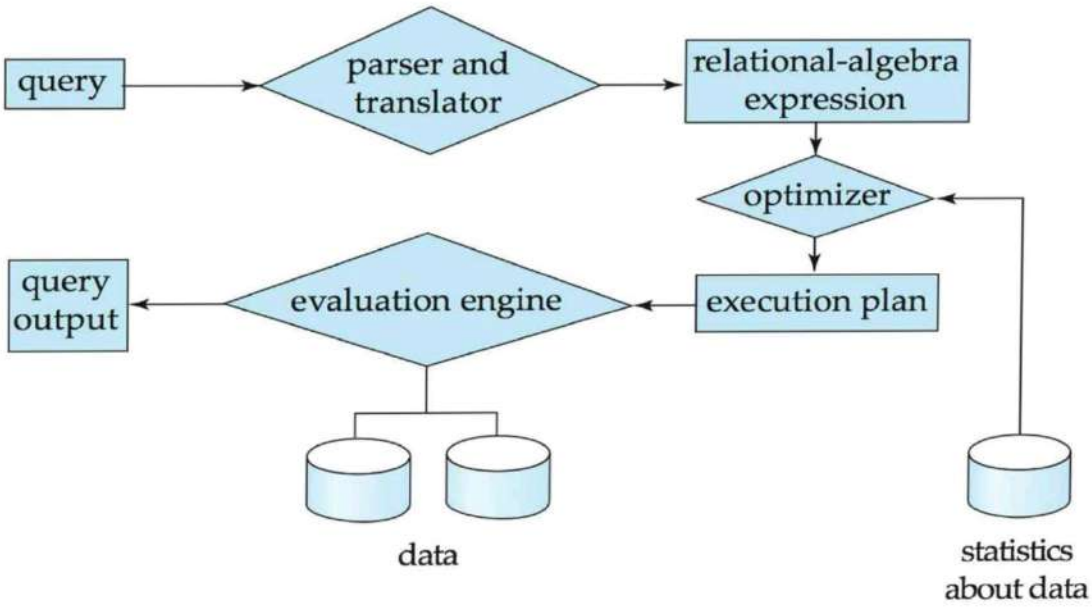
- Storage manager
- Query processing
- Transaction manager

Storage Management

- **Storage manager** is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data
- Issues:
  - Storage access
  - File organization
  - Indexing and hashing

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Query Processing (Cont.)

- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
  - Depends critically on statistical information about relations which the database must maintain
  - Need to estimate statistics for intermediate results to compute cost of complex expressions

Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

PAGE No.
DATE / /

11.

i) $\Pi_{cust\_name}(Borrower) \cap \Pi_{cust\_name}(Depositor)$

ii) $\sigma_{Amount \geq 1200}(Loan)$

iii $\Pi_{loan\_no}(\sigma_{Amount > 1200}(Loan))$

13) i) $\Pi_{cust\_name}(Borrower) \cup \Pi_{cust\_name}(Depositor)$

ii) $\Pi_{cust\_name}(\sigma_{B\_name = "Perryridge"}(\sigma_{Borrower.loan\_no = Loan.loan\_no}(Borrower \times Loan)))$

iii) $\Pi_{Balance}(Account) - \Pi_{Account.Balance}(\sigma_{Account.Balance < d.Balance}(\rho_{d}(Account)) \times (Account))$

iv) $\Pi_{cust\_name}(\sigma_{B\_name = "Perryridge"}(\sigma_{Borrower.Loan\_no = Loan.loan\_no}(Borrower \times Loan))) - \Pi_{cust\_name}(Depositor)$

Attributes:

Attributes are the properties that define the entity type. For example, Roll_No, Name, DOB, Age, Address, Mobile_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.

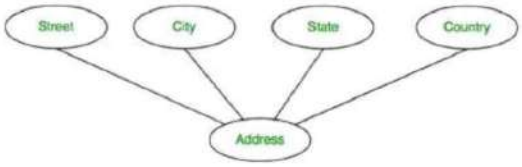
Types of attributes

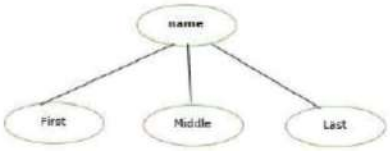
The types of attributes in the Entity Relationship (ER) model are as follows –

- **Key Attribute** – The attribute which **uniquely identifies each entity** in the entity set is called key attribute.For example, Roll_No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.

- **Single value attribute** − These attributes contain a single value. For example, age, salary etc.
- **Multivalued attribute** − They contain more than one value of a single entity. An attribute consisting more than one value for a given entity.
    - For example, Phone_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval. For example, phone numbers.

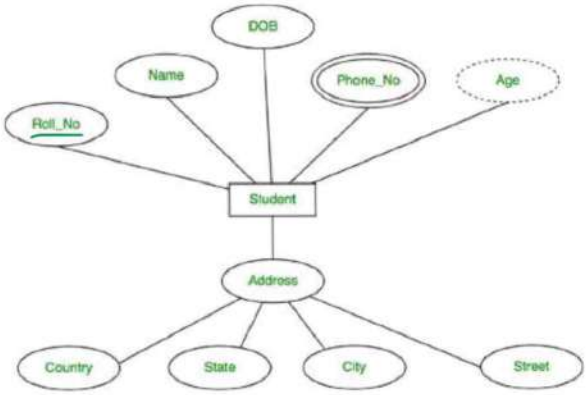
- **Composite attribute** − The attributes which can be further divided. An attribute composed of many other attribute is called as composite attribute.
    - For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals.
    - Name-> First name, Middle name, last name

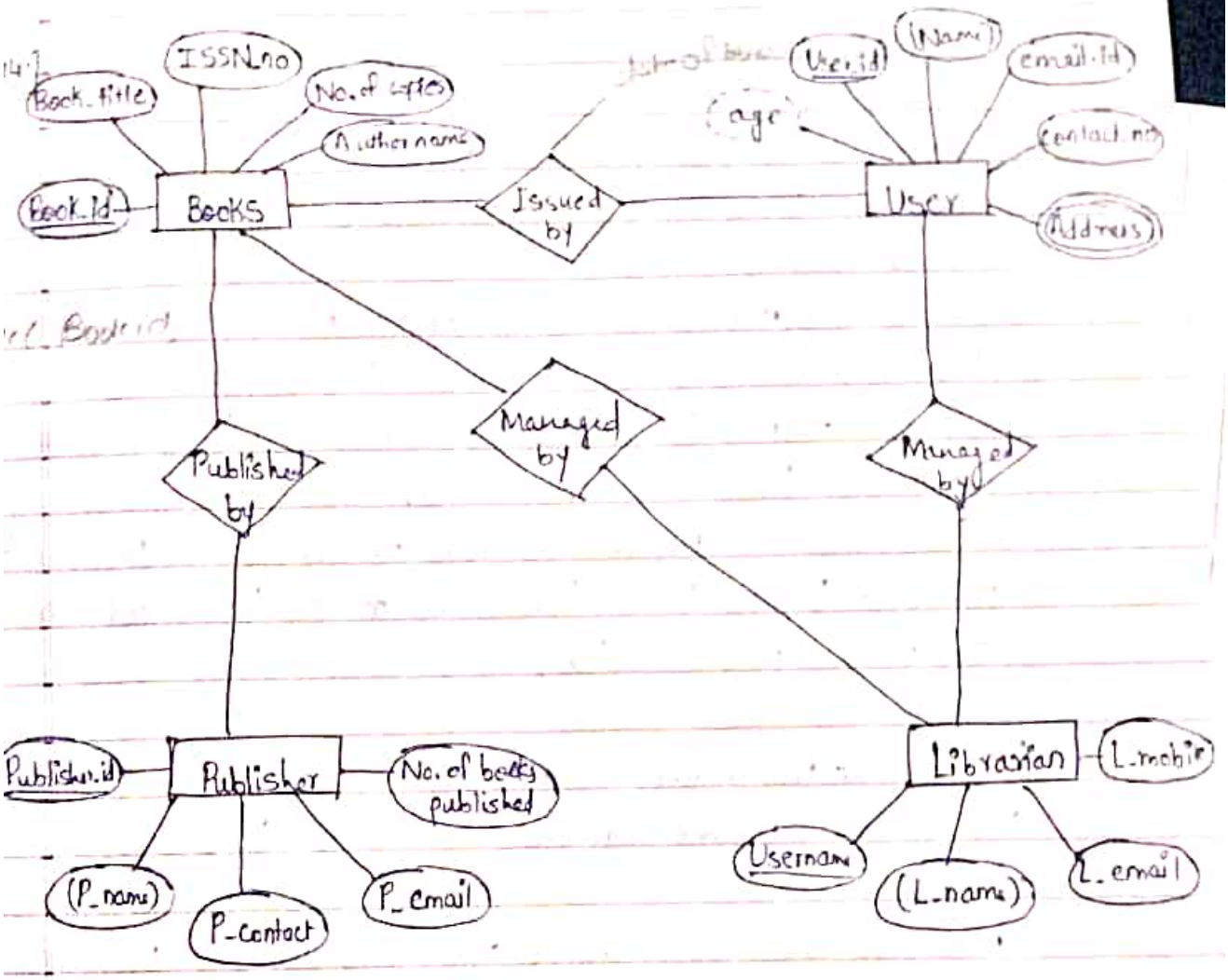




- **Derived attribute** − The attribute that can be derived from others. An attribute that can be derived from other attributes of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, the derived attribute is represented by a dashed oval.

The complete entity type **Student** with its attributes can be represented as:





Books

Book_id
Book_title
ISSN_no
No. of copies
Authorname

User

User_id
Age
Email-id

Publisher

Publisher_id
P_contact
No. of books published
P_email
P_name

Librarian

Username
F_name
L_name
L_email
L_mobile

User Address

Address
User_id (FK)

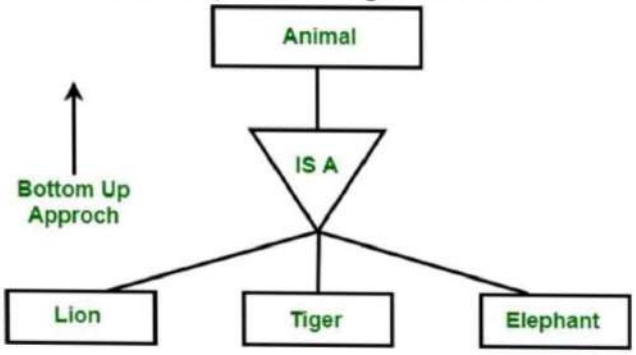
User contact no

Contact_no.
User_id (FK)

Constraints on generalization and specialization

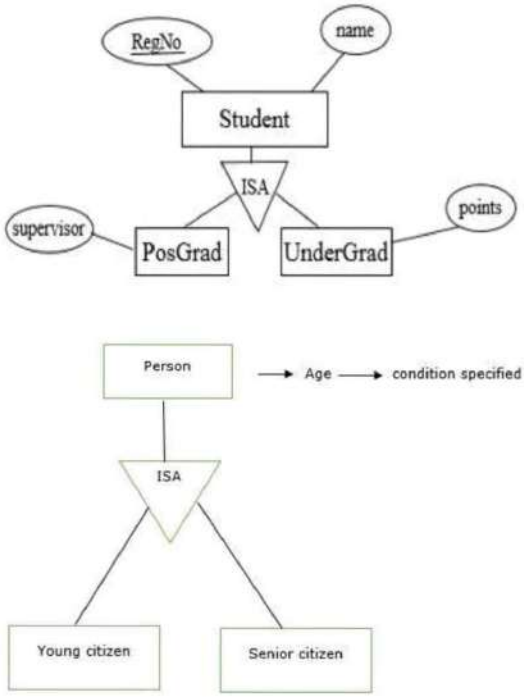
There are three types of constraints on generalization which are as follows:

1. First one determines which entity can be a member of the low-level entity set.
2. Second relates to whether or not entities belong to more than one lower-level entity set.
3. Third specifies whether an entity in the higher level entity set must belong to at least one of the lower level entity set within generalization.

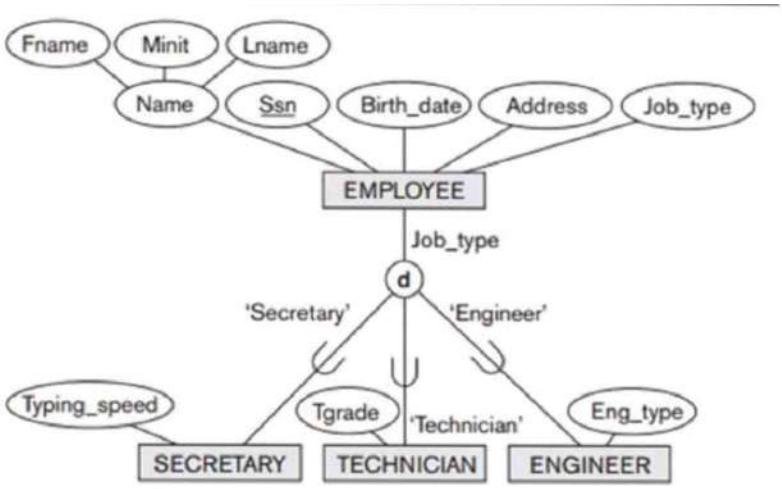


1. First one determines which entity can be a member of the low-level entity set:
Such kind of membership may be one of the following:

- **Condition-defined** –
  In this lower-level entity sets, evaluation of membership is on basis whether an entity satisfies an explicit condition or not. For example, Let us assume, higher-level entity set student which has attribute student type. All the entities of student are evaluated by definition of attribute of student. Entities are accepted by the satisfaction of condition i.e. student type = "graduate" then only they are allowed to belong to lower-level entity set i.e. graduate student. By the satisfaction of condition student type = "undergraduate" then they are included in undergraduate student. In fact, all the lower-level entities are evaluated on the basis of the same attribute, thus it is also referred as attribute-defined.



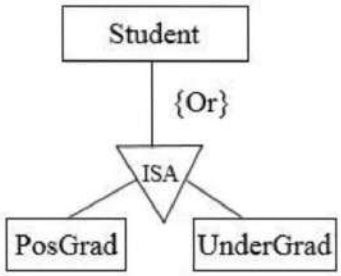
- **User-defined** –
  In this lower-level entity sets are not get constrained by a condition named membership; users of database assigns entities to a given entity set. For example, Consider a situation where after 3 months of employment, the employees of the university are assigned to one of four work teams. For this purpose, we represent teams them as four lower-level entity sets of higher-level employee entity set. On the basis of an explicit defining condition, a given employee is not assigned to specific team entity. User in charge of this decision makes the team assignment on an individual basis. By adding entity to an entity set, assignment is implemented.

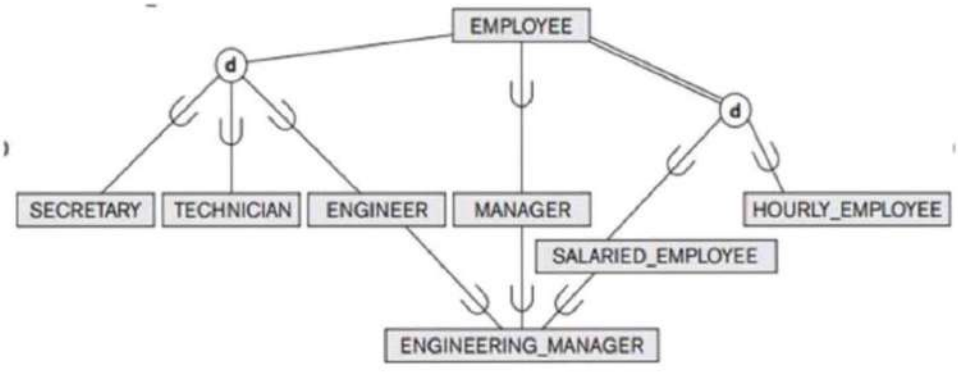


2. Second relates to whether or not entities belong to more than one lower-level entity set :

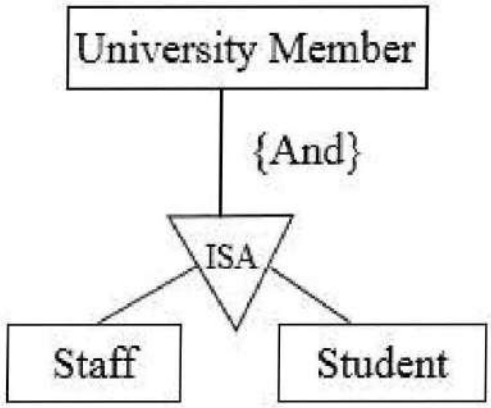
Following is one of the lower-level entity sets:

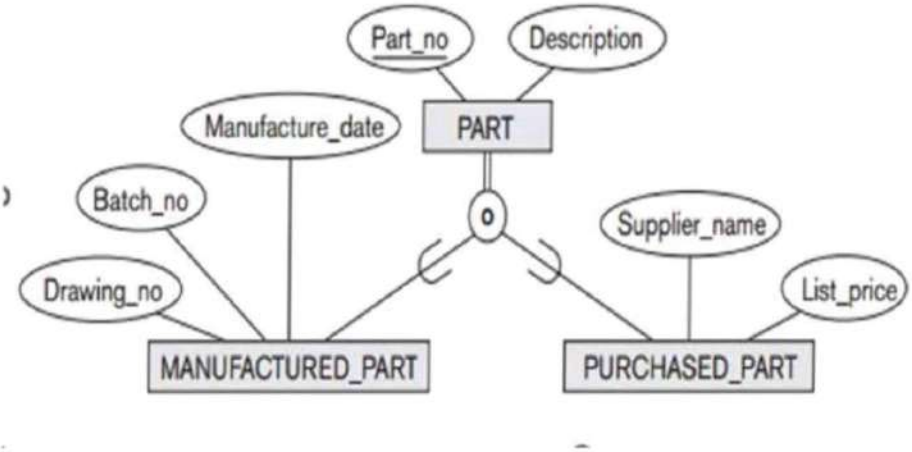
- **Disjoint** –
  The requirement of this constraint is that an entity should not belong to no more than one lower-level entity set. For example, the entity of student entity satisfy only one condition for student type attribute i.e. Either an entity can be a graduate or an undergraduate student, but cannot be both at the same time.





- **Overlapping** –
  In this category of generalizations, within a single generalization, the same entity may belong to more than one lower-level entity set. For example, in the employee work-team assume that certain employees participate in more than one work team. Thus, it offers a given employee that he may appear in more than one of the team entity sets that are lower-level entity sets of employee. Thus, generalization is overlapping.

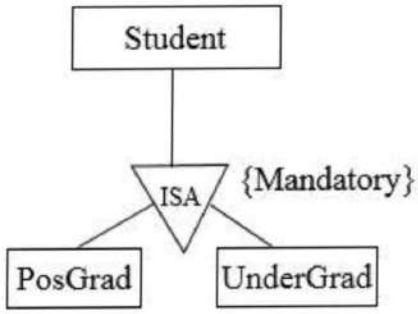




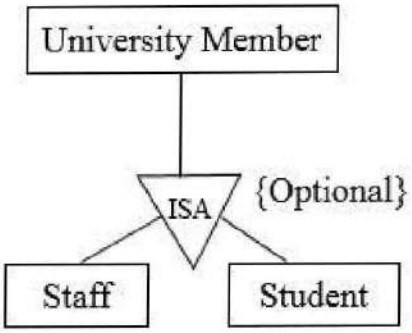
3. Third specifies whether pr not an entity in the higher level entity set must belong to at least one of the lower level entity set within generalization :

This constraint may be one of the following:

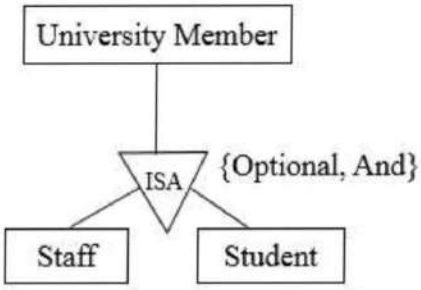
- **Total generalization or specialization** –
  According to this constraint, each higher-level entity must belong to a lower-level entity set.
  - To represent completeness in the specialization/generalization relationship, the keyword "**Mandatory**" is used.



- **Partial generalization or specialization –**
  According to this constraint, some higher-level entities may not belong to any lower-level entity set.
  - The keyword **"Optional"** is used to represent a partial specialization/generalization relationship



We can show both disjoint and completeness constraints in the ER diagram. Following our examples, we can combine disjoint and completeness constraints.



Some members of a university are both students and staff. Not all members of the university are staff and students.