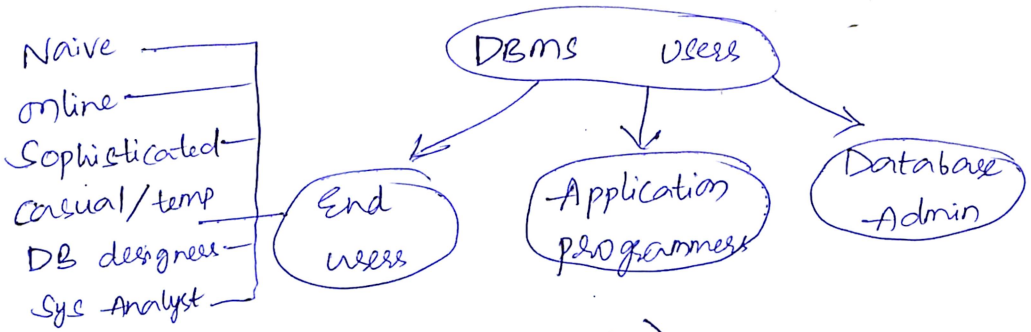


i). DBA (Database Administrators)

↳ person/team – who destines the schema

→ controls 3 levels of DB users

→ creates new account id & password for the user to access the DB

→ provides security (only authorised users can access the DB).

→ provides back up and recovery

→ we can call it as Super users account

ii). Application programmer

↳ programmers who writes the code for Application programs or User Interface

→ written in programming lang.

iii). End users:

a) Naive users: Unsophisticated – who doesn't have any DBMS knowledge but they frequently use the DB applications → Parametric users

Eg: Ticket booking users.

b) online users: may communicate with DB directly through online terminal or indirectly through User Interface.

c) Sophisticated users: (SQL programmers) – who are going to deal directly with DB using queries

d) Casual / Temporary users: – who occasionally use/access the DB. But they access DB, when they need new information.

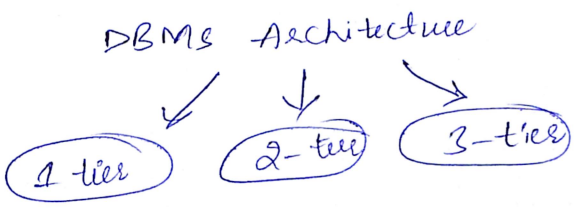
e). DB designers:

- who designs the structure of DB { tables, views, constraints ... }
- Defines What data has to store

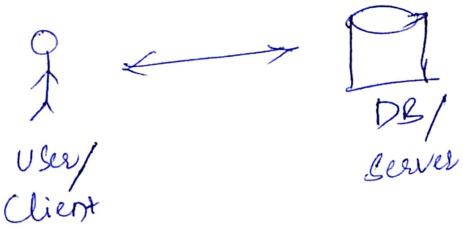
f) System Analyst:

- who analyzes the requirements of naive/parametric end users
- Check whether all the requirements of end users are satisfied.

* DBMS Architecture:

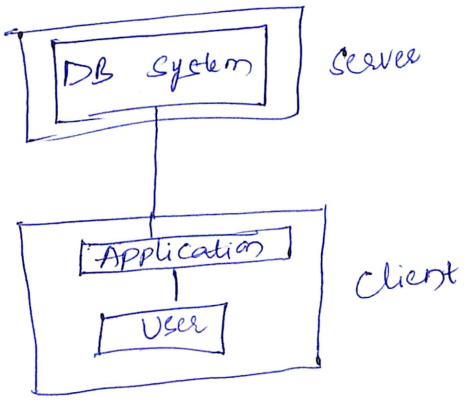


(i). 1 tier:



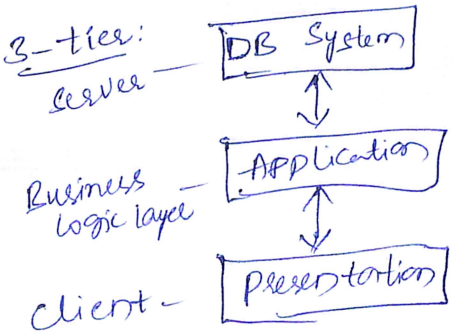
- Any changes done by user, that will directly be done on DB itself
- Eg: local application.

(ii). 2-tier:



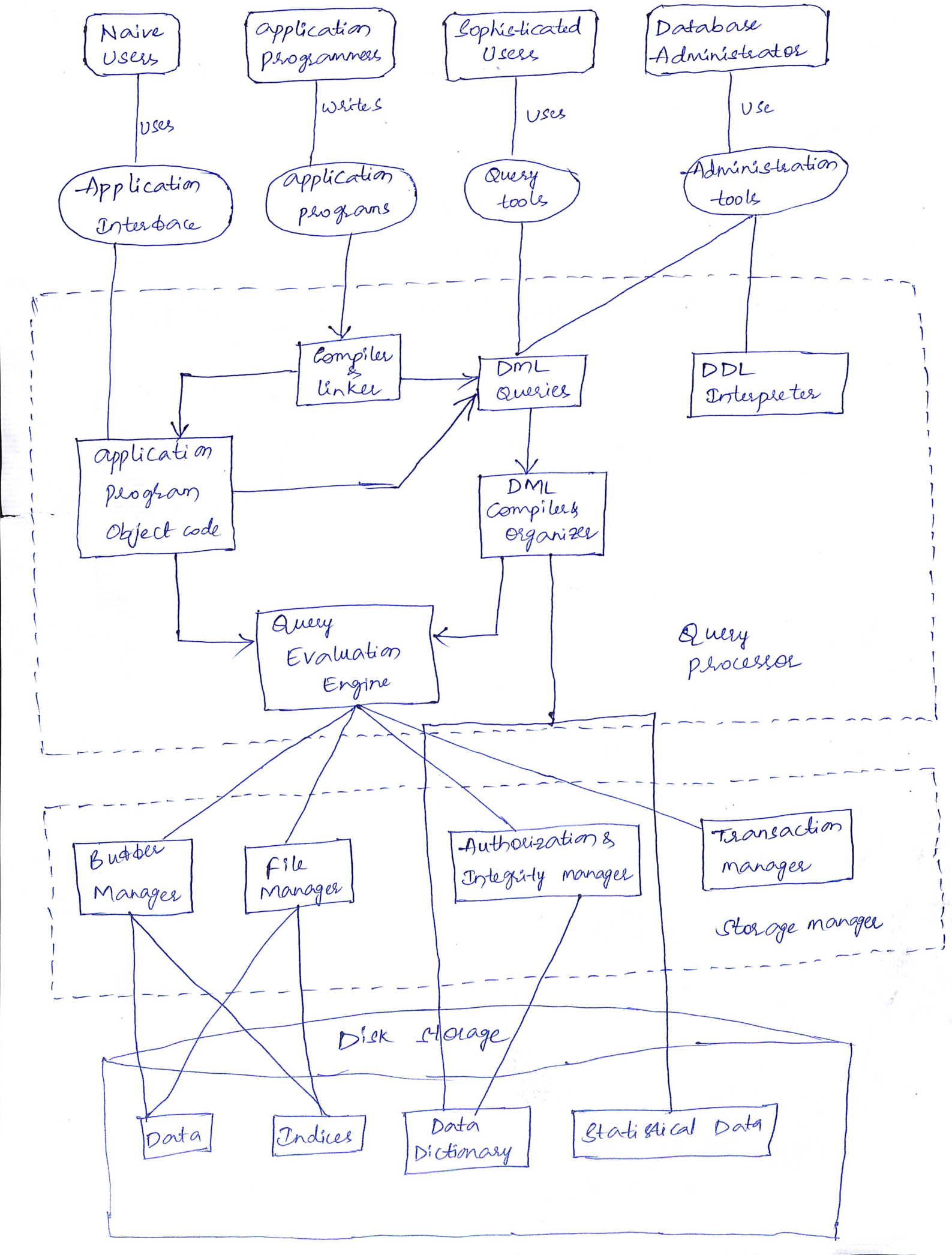
- Interact application with server ODBC, JDBC are used.
- UI /app program runs on client side
- To communicate with server, UI will establish the connection

(iii). 3-tier:



1, 2, 3, 5, 6, 14, 16, 20, 22, 26, 27, 28, 31, 34, 35, 36, 38, 40, 42, 43, 45, 47, 48, 49, 50, 51, 53, 55,

*Database System Architecture.



* Storage Manager: is a program b/w DB & Query processor

- It is also known as DB control System.
- It maintains the consistency & Integrity of the DB using DCL/TCL commands

(i) Buffer manager:

- monitors how SQL Server uses memory to store data pages
- It is responsible for cache memory & transfer data b/w

secondary & main memory

↓ store data permanently eg: Hard drive, CD ...

↓ Primary ↓ store data temp eg: RAM, ROM

- If user request a particular block & if the block is available in the buffer, buffer manager provides the block address in main memory.

(ii). File manager:

- Responsible for creation, deletion, modification of files
- managing files access & ~~provides~~ security & resources used by files.
- Keep track of each file in the system through directories. which contains file name, file location.
- Allocating each file when user is granted to access them.
- Deallocate the files when their use is finished or not needed.

iii> Authorization & Integrity Manager:

- Tests for the satisfaction of Integrity constraints & checks the authority of users to access the data.

- It maintains consistency & Integrity of DB. Integrity means that the data is protected from unauthorized changes to ensure that it is reliable and correct. (3)

(iv) Transaction Manager:

- It controls concurrent access by performing the operations in a scheduled way that it recieves the transaction.
- It ensures that the DB remains in the consistent state before & after the execution of a transaction.

Eg: Transfer 50/- from Account A to Account B.

Initially If they have A = 500/-, B = 800/-

R = Read
W = write

Steps: i) Read (A)
A = A - 50
W(A)
R(B)
B = B + 50
W(B)

— Now, A = 450/- , B = 850/-

* Disk Storage: Storing of data in storage medium.

(i) Data files: Stores the data in the form of files.

(ii) Indices: Indexing used to optimize the performance of DB.

- Quick data access using index.

(iii). Data dictionary:

- It contains metadata (Data about the DB).
views, constraints (PK, FK), what is in DB; where is it stored, who can access.
- It can be handled by DB Admin.

(iv) Statistical data:

- Collection of numerical data, used for statistical analysis purpose.

* Query Processor:

(i) DDL Interpreter:

- Interpretes DDL statements & records the definition in the data dictionary.
- Interpreter means, analyse & executes the program line by line.

(ii). Compiler & linker:

- Compiler generates machine language code from source code. (ie., obj file)
- Linker combines all object code files together & gives executable file.

(iii). Query Evaluation Engine:

- Responsible for generating O/p for the given query.
- Query Evaluation plan. — Another name.

* Relational Databases:

- RDBMS is Relational Data Base management System, which stores the data in tabular form. Which have records & fields.

DBMS eg: XML	RDBMS eg: mySQL, PostgreSQL, SQL Server, Oracle.
(i). Stores data as file - Low security	(i). Stores data as table form - Higher security
(ii). Single data can be accessed at a time.	(ii). Multiple data elements can be accessed at same time.
(iii). No relation blw data	(iii). tables are related to each other
(iv). Normalization is not present	(iv). Normalization is present.
(v). Doesn't support distributed DB ie., (Single user)	(v). Supports distributed DB. ie., (multi users)
(vi) Stores data in hierarchical form	(vi) Stores data in tabular form. Header are column names remaining rows corresponding values
(vii) deals with small quantity of data	(vii) Large amount of data
(viii) Data redundancy is common	(viii). No Data redundancy (keys & index)
(ix). Data fetching is slow for large data	(ix) Fast

* Overview of the Design process:

(1)

(i). Requirement collection & Analysis:

- what data is needed
- DB designers interact with the domain Experts & Users.

(ii). Choosing a data model:

- Applying all concepts of chosen data model.
- Translates these requirements into conceptual schema of the DB.
- Fully developed conceptual schema indicates the functional requirements of the enterprise.
- Describe the kind of operations (or transactions) that will be performed on the data.

(iii) Moving from an abstract data model to the implementation of DB.

a) Logical design: Deciding on the DB schema
b) physical design: " " physical layout of the DB.

- while designing DB, we should avoid Redudancy, incompleteness.

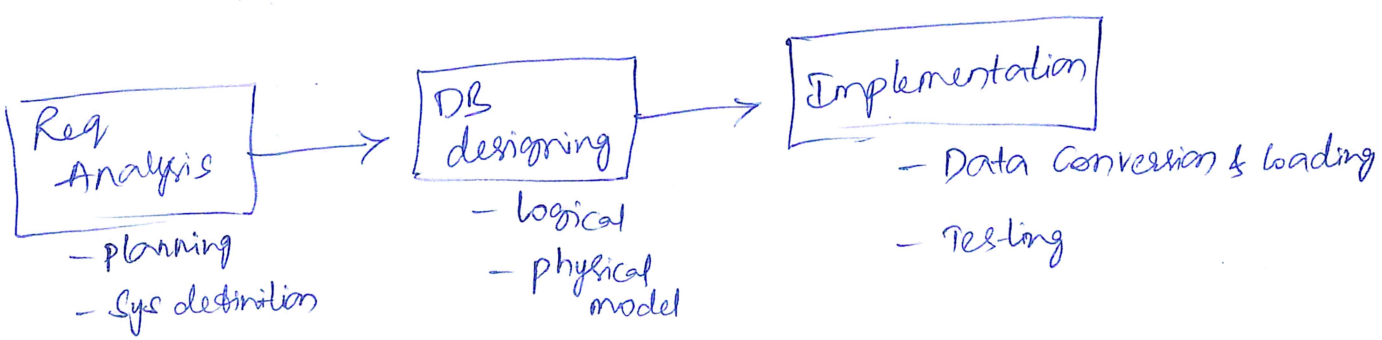
* ~~Data models~~:

- ~~Defines how the logical structure of a DB is modelled.~~

(i). Requirement collection & Analysis:

a) planning: DDLC (DB Development Life Cycle).

b) System definition: boundaries & scope of DB after planning



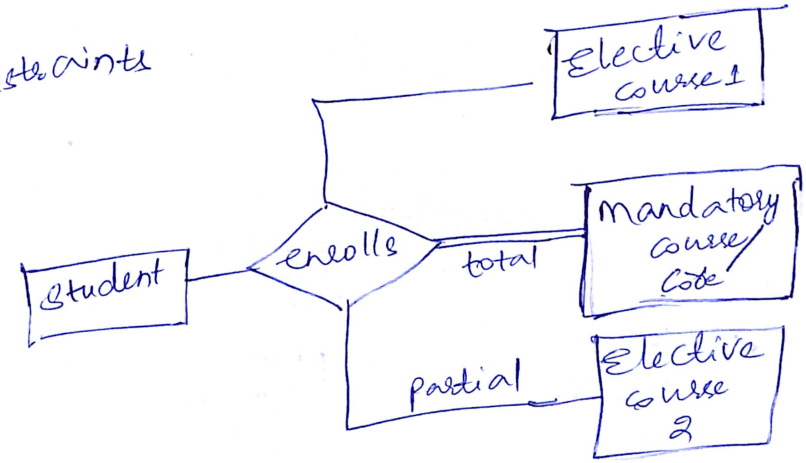
* Constraints: modelling limitations on the relations b/w entities

(i) mapping Cardinality or Cardinality ratio
- one-to one - one to many - many to one - many to many

(ii) participation constraints

total participation :

partial "



* Data Models:

Data model is the modelling of the data description, data semantics, Consistency constraints of the data.

types of data models:

(i) Relational data model:

Relational model uses tables for representing data & in between relationships Tables also called as relations. primarily used by commercial data processing applications.

(ii) Entity-Relationship Data model:

ER model is the logical representation of data as objects & relationships. These objects are called as Entities. Relationship is an association among these entities. It is used in DB designing. A set of attributes describes the entities.

Eg: student: name, rno, address, DOB, age.

(iii) Object based data model:

Extension of ER model with notations of functions, encapsulation & obj. identity. Here, objects are the data carrying its properties.

(iv). Semi structured data model:

It allows the data specifications at places where the individual data items of same type may have different attribute sets.

XML is widely used for representing the semistructured data.

(v) Hierarchical data model:

It is in the form tree i.e., parent, child.

Parent to child → 1:N

Child to parent → 1:1.

Doesn't support n:n relationship.

* ER model Design Issues:

* (v) N/w model:

It is graph structure. It allows more than one parent.

* Stages of data modelling: (i) conceptual (ii) logical (iii) physical.

(i) Conceptual Data model:

Describes the database at very high level & It is useful to understand the needs or requirements of the database.

Eg: ER model, Object oriented model.

(ii). Logical Data Model: (Representational data model)

It is used to represent only logical part of DB. Doesn't represent the physical structure of the DB. It focuses on design part.

Eg: Relational model, Hierarchical, N/w.

(iii). physical Data model:

All data in DB stored physically on secondary storage device such as disks & tapes. Stored in the form of files.

* E-R Design issues:

(i) Choosing of Entity set Vs Attributes:

Depends on the structure & semantics associated with its attributes.

Eg: Entity set student with attributes name & sno.

Now, sno can be an entity with attributes class & section.

(ii). Choosing of Entity set Vs Relationship sets:

It is difficult to examine which one to choose. If any new requirement arise in future, choose entity.

Eg:

A person takes loan from bank.

person & bank are 2 entities. Loan is a relationship

If disburse a joint loan — here loan should be entity set

(3)

(iii). Choosing of binary vs n-ary relationship:

- If we have 2 entity sets — binary relationship
- If we have more than 2 entity sets — n-ary relationship.

we can convert n-ary relationship using multiple binary relationships

Eg: Relationship b/w 4 family members - mother, Father, son, daughter

Father - mother → spouse

son - Daughter → Sibilings

(Father - mother) - children → child

(iv) placing relationship attributes:

Cardinality ratio helps to place relationship attributes.

↓

1:1 or 1:n

Eg: If an entity determined by combination of participating entity sets.

Better to represent it as many-to-many.

* Extended ER features:

ER diagram is a graphical diagram of overall logical structure of DB.

Rectangles — entity

Ellipses — attributes

Diamonds — Relationships

Lines — links attributes to entity & entity to relationship.

Double ellipse — multivalued attribute

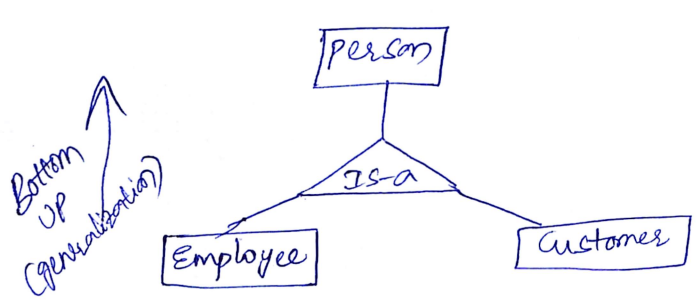
Dashed ellipse — Derived attribute

Double lines — total participation of an entity in a relationship

Double rectangle — weak entity sets.

(i). Specialization: (is-a) - △ - (superclass - subclass)

- Top down approach.
- one higher level entity can be broken down into 2 lower level entities.
- used to identify the subset of an entity set that shares some distinguishing characteristics.

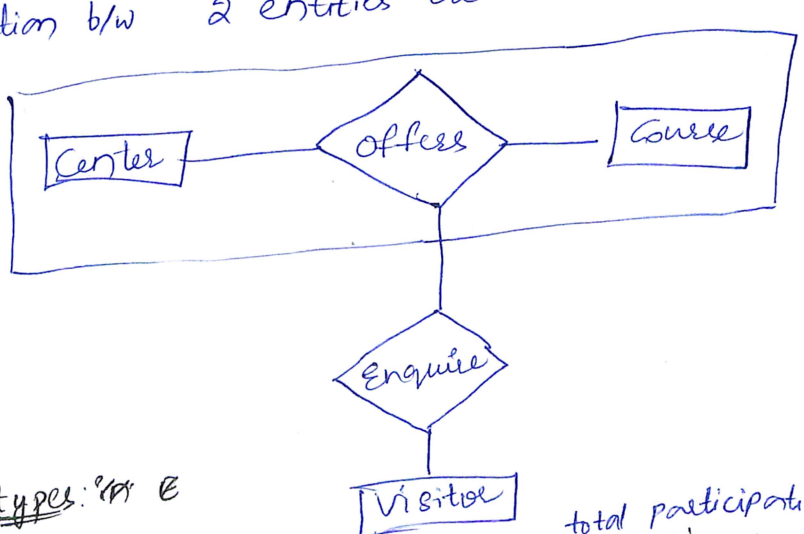


(ii). Generalization: (Sub class - super class)

- Bottom up approach
- Two or more lower level entities will be combined & forms higher level entity.

(iii). Aggregation:

Relation b/w 2 entities treated as a single entity.



* Entity types:

(i). Strong entity: may have or may not have total participation

- Doesn't depend on any other entity in schema
- It will have primary key
- represented by single Rectangle
- relationship of 2 strong entities by single diamond

(ii). Weak entity - always have total participation

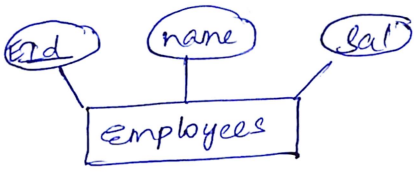
- Depends on strong entity
- Doesn't have any PK.
- Double Rectangle
- one strong, one weak relationship double diamond.

* Reduction to Relation schemas:

ER to relation:

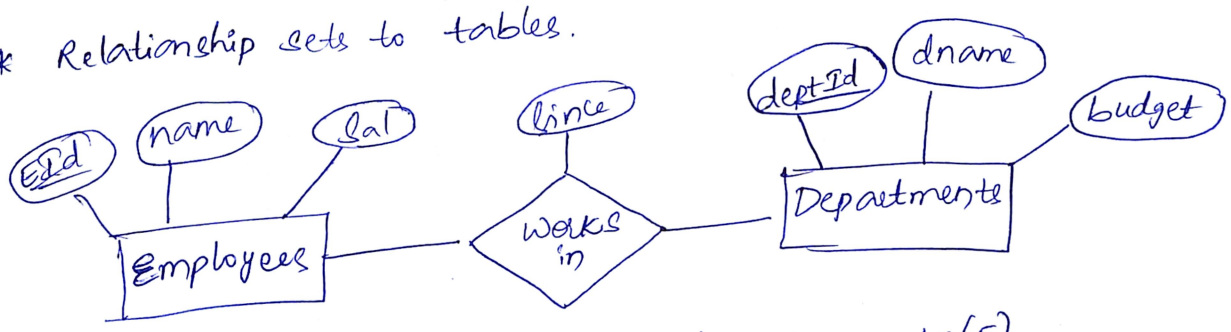
* Entity sets to tables.

Eg:



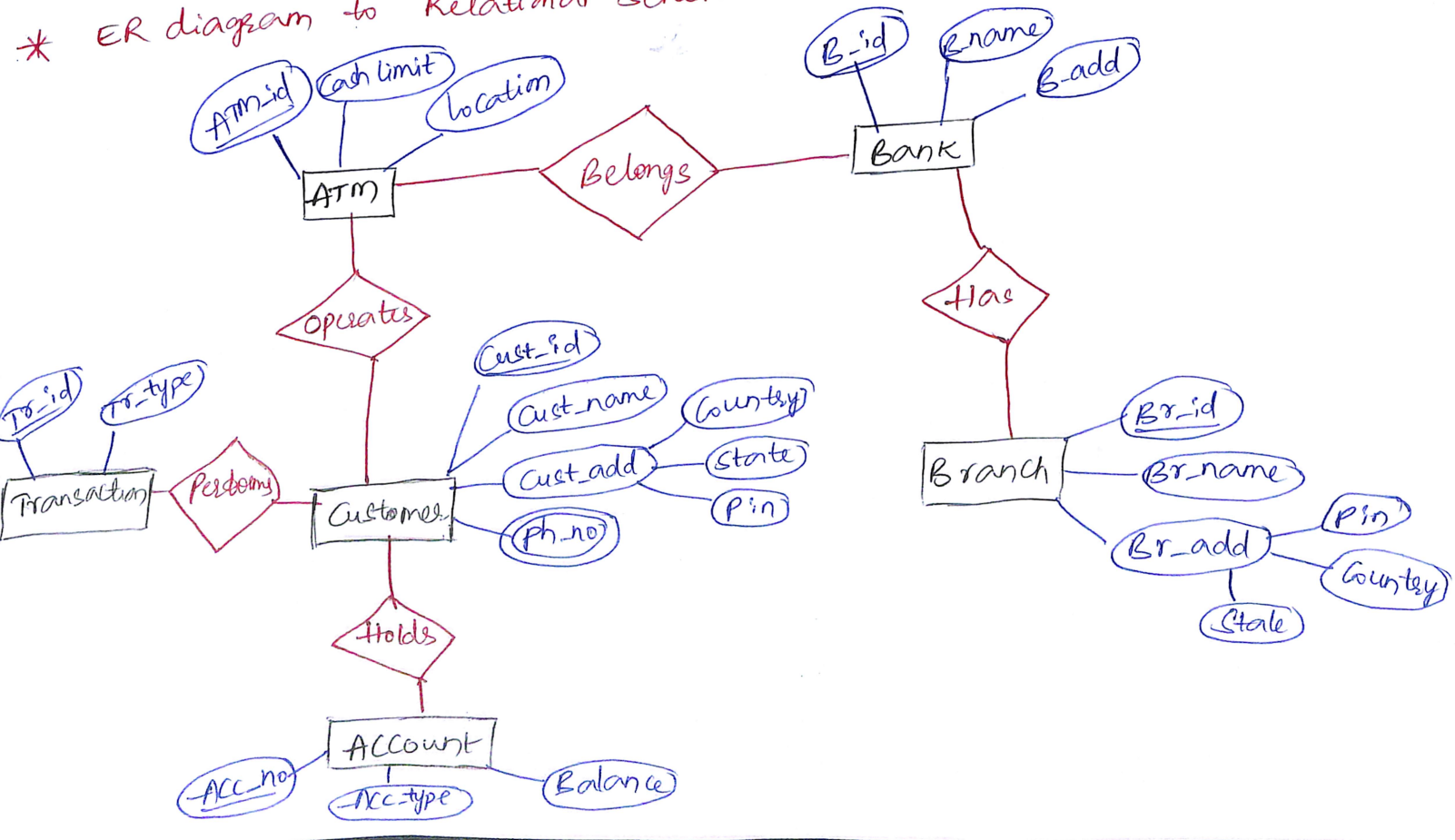

```
create table Employees (
    Eid number(5),
    name varchar(20),
    sal number(10, 2),
    primary key (Eid))
```

* Relationship sets to tables.




```
create table works_in ( Eid number(5),
    dept_id int,
    since date,
    primary key (Eid, dept_id),
    foreign key (Eid) references Employees,
    foreign key (dept_id) references department)
```

* ER diagram to Relational schema:



* Strong entity sets:

ATM (ATM_id, Cash limit, location)

Bank (B_id, B_name, B_add)

Branch (Br_id, Br_name, Br_add) — Br_add: state, country, pin

Transaction (Tr_id, Tr_type)

Customer (Cust_id, Cust_name, Cust_add, ph_no) — Cust_add: state, country, pin

Account (Acc_no, Acc_type, Balance)

* Composite: include all sub attributes into the table/schema.
* multivalued: create seperate table. It should have PK.

Customer_phone (Cust_id, ph_no)

* Relationship sets:

① Belongs: 1 to n 1:n — Bank to ATM

one bank can have many ATM's

An ATM can belongs to atmost one Bank.

② Has: 1:n → Bank to Branch

one bank can have many branches

1 Branch can belongs to only one bank.

③ Operates: n:n → ATM to customer

ATM can be operated by many customers

A customer can operates many ATM's

④ performs: n:1 → transaction to customer

A transaction can be done by one customer.

one customer can perform any no of transactions

⑤ Holds: n:n → Customer to Account.

A customer can have any no of accounts

An account can be maintained by more than one customer (joint account).

* For many to many (n:n) relationship, we need to create ⑤ separate schema by including the PK of participating entity sets as attributes. ④

* For other relationships, PK of one side has to be included as the FK of other side. No need to create seperate table/schema.

① Belongs relationship: ATM (ATM_id, cash limit, location, B_id)
include B_id in many side.

② Has:
Branch (Br_id, Br_name, state, Country, Pin, B_id)

③ Operates: Operates (ATM_id, cust_id)

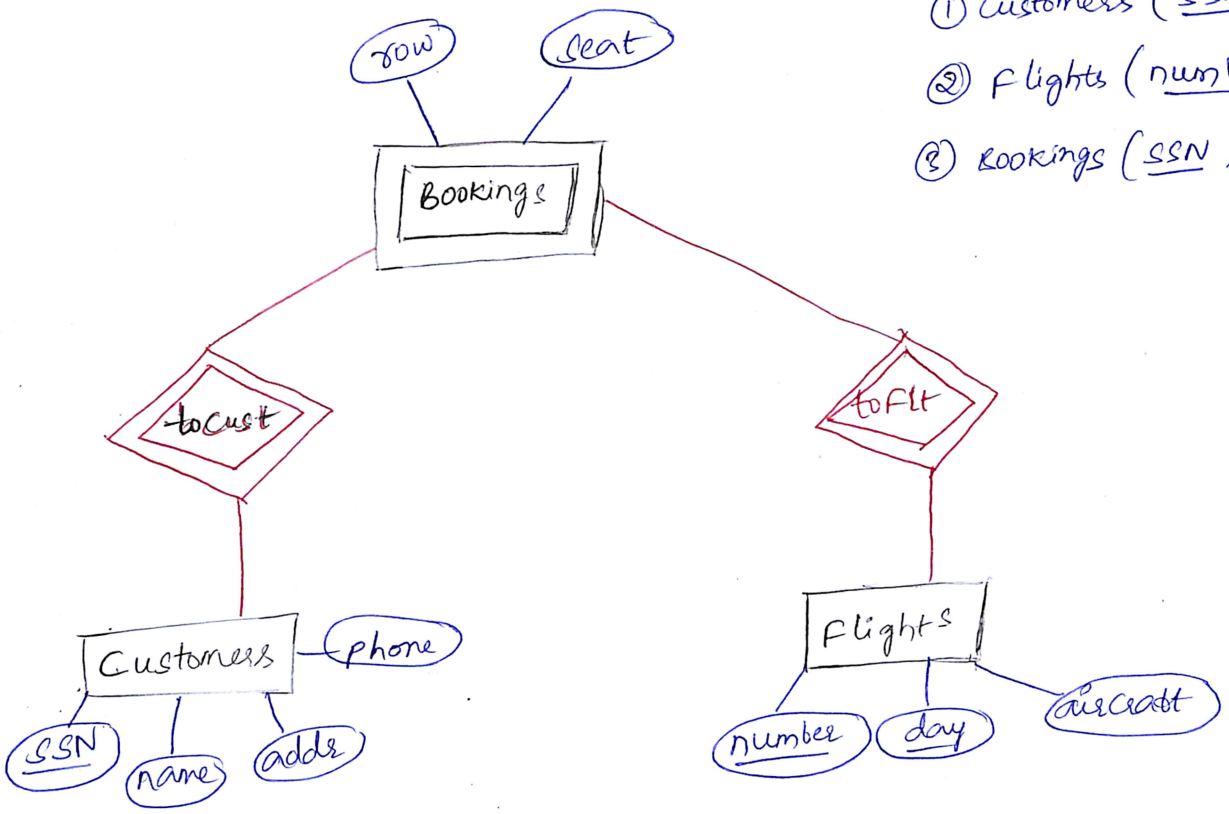
④ performs: Transaction (Tr_id, Tr_type, cust_id)

⑤ Holds: Holds (cust_id, ACC_no)

So, final relational schema after reduction is,

① Bank (B_id, B_name, B_add)
② Customer (cust_id, cust_name state, Country, Pin)
③ Customer_phone (cust_id, ph_no)
④ Account (ACC_no, ACC_type, Balance)
⑤ ATM (ATM_id, Cash limit, location, B_id)
⑥ ~~Has~~ Branch (Br_id, Br_name, State, Country, Pin, B_id)
⑦ Transaction (Tr_id, Tr_type, cust_id)
⑧ Operates (ATM_id, cust_id)
⑨ Holds (cust_id, ACC_no)

* Example 2:

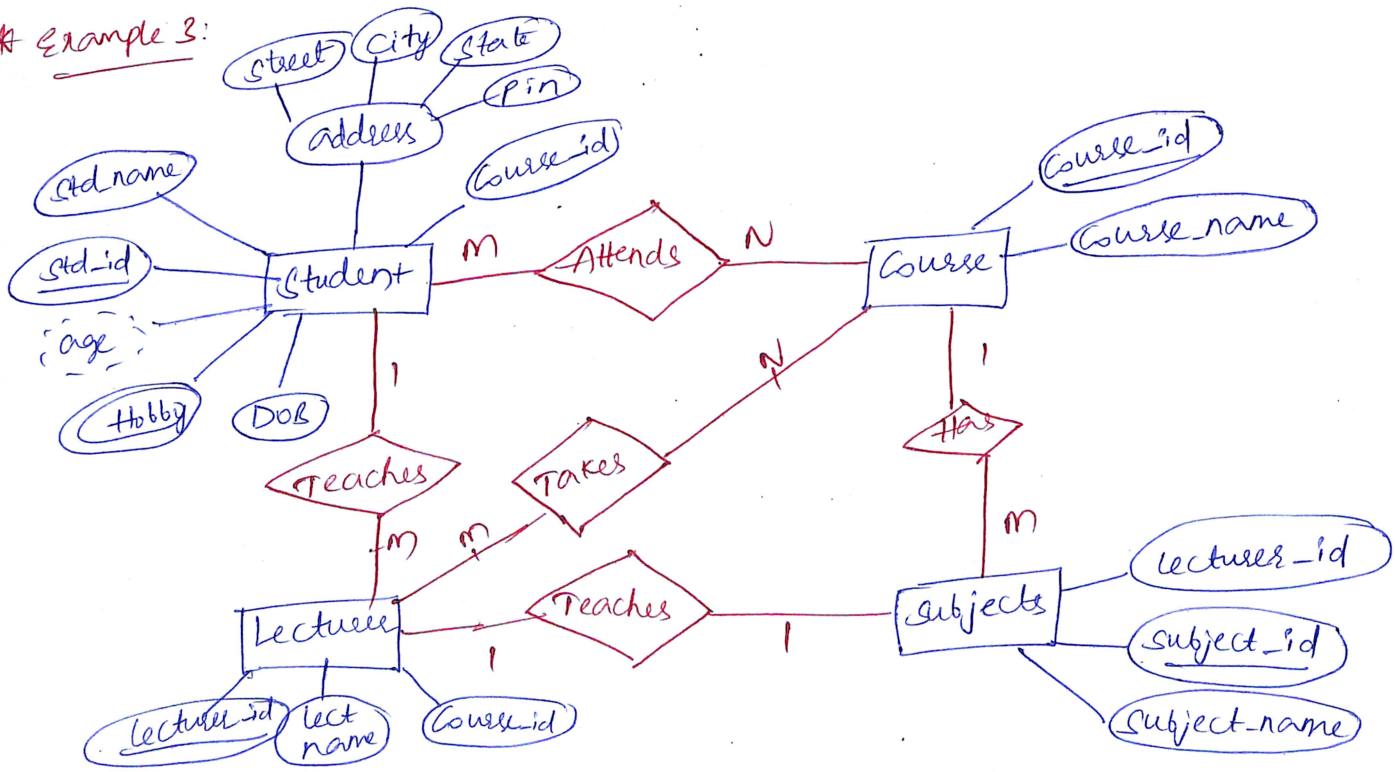


① Customers (SSN, name, addr, phone)

② Flights (number, day, aircraft)

③ Bookings (SSN, number, day, row, seat)

* Example 3:



① student (std_id, std_name, street, city, state, pin, course_id, dob)

② Course (course_id, course_name)

③ Subjects (sub_id, lecturer_id, subject_name, course_id)

④ lecturers (lecturer_id, lect_name, course_id, std_id, subject_id)

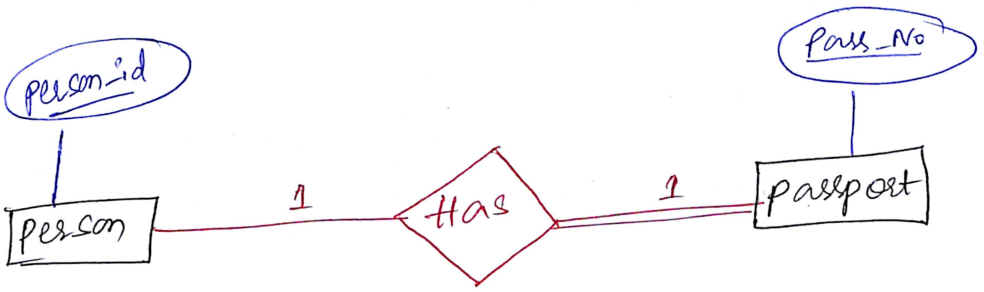
⑤ Attends (std_id, course_id)

⑥ Takes (lecturer_id, course_id)

⑦ std_Hobby (std_id, hobby)

Neglect derived attribute.

* Example: Binary Relation 1:1 Cardinality with total participation of an entity.

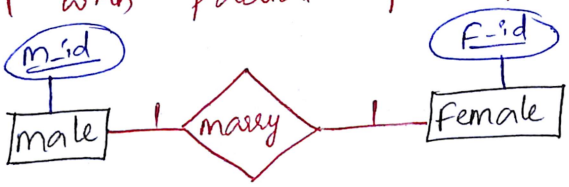


① person (person_id) ✓

② passport (pass_NO) ✓

③ Has (person_id, pass_no) — only this table..

* 1:1 with partial participation:



① Male (M_id) ✓

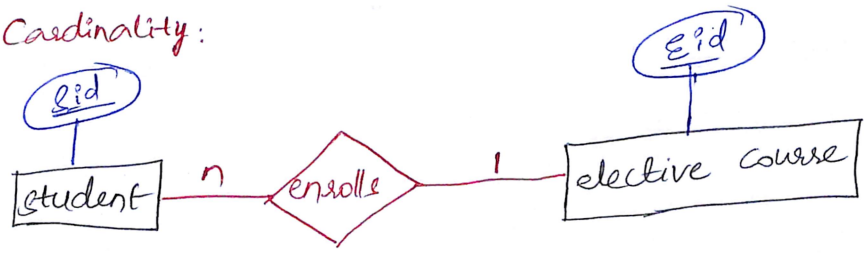
② Female (F_id)

③ marry (M_id, F_id)

→ 2 tables

Binary relationship with 1:1 will have 2 tables if partial participation of both entities. If atleast 1 entity has total participation, no. of tables required will be 1.

* n:1 Cardinality:



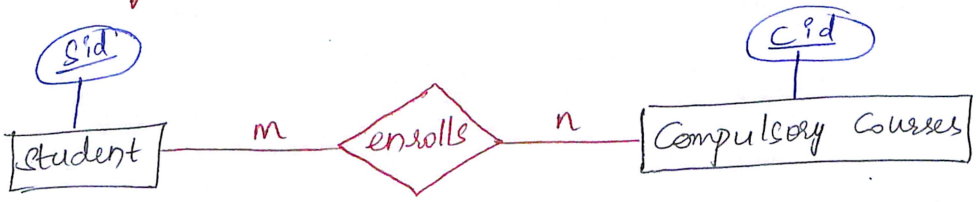
① student (S_id) ✓

② elective_course (E_id)

③ enrolls (S_id, E_id)

→ 2 tables.

* m:n Cardinality:



① student (S_id)

② Compulsory courses (C_id)

③ enrolls (S_id, C_id)