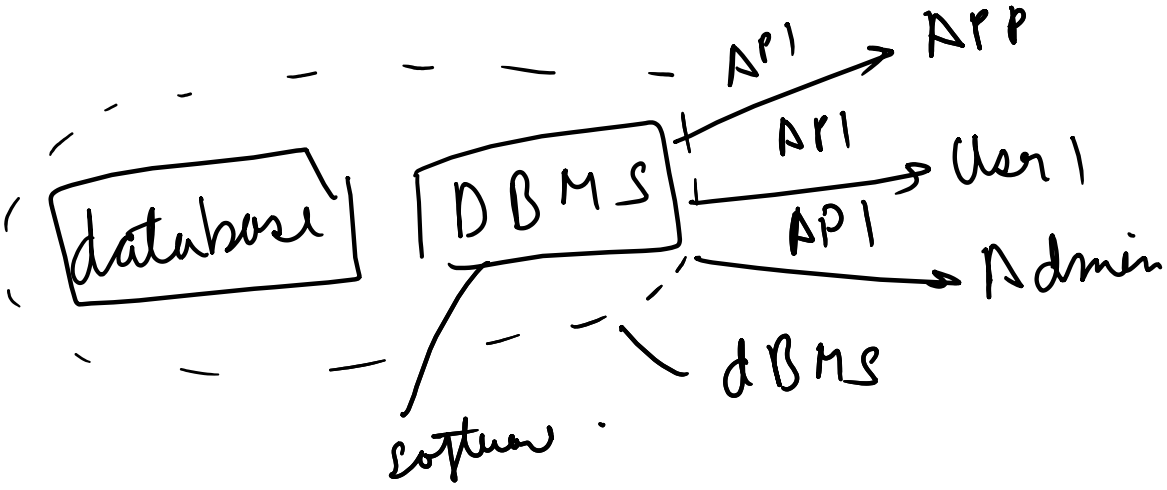
DBMS

➔ By Rahul Kumar

Introduction:

- **Data:** Data is a collection of raw, unorganized facts and details like text, observations, figures, symbols, and descriptions of things etc. In other words, data does not carry any specific purpose and has no significance by itself. Stored in form of bits &bytes. Quantitative-> numeric value, Qualitative -> Descriptive.
- **Information:** Process data which contains special meaning, enables decision making. Ex data of country when processed provide its population, sex -ration, literacy rate etc.
- **Database**: Database is a place when data is stored in a way where it can be easily **accessed, crated, read, updated and deleted**.
- **DBMS:** It is collection of interrelated data and set of program to access those data. Its goal is to store and retrieve database information in both convenient and efficient way. A DBMS is the database itself, along with all the software and functionality. It is used to perform different operations, like addition, access, updating, and deletion of the data.



Advantages of DBMS :-

1. **No Data Redundancy and inconsistency** :- Data Redundancy means occurance of same data more than two places. Inconsistency means within a database multiple data doesn't match.
2. Easy of accessing data
3. **Data abstraction:** The major purpose of a database system is to provide users with an abstract view of the data. Hiding underlying complexities.
4. **Data sharing:** A DBMS provides a platform for sharing data across multiple applications and users
5. **Security**: DBMS provides a high level of data security by offering user authentication and authorization etc.
6. **Atomicity:** Operation either completed or failed, nothing in between.

File System VS DBMS

File System	DBMS
Redundant data can be present in a file system.	In DBMS there is no redundant data.
There is no efficient query processing in the file system.	Efficient query processing is there in DBMS.
There is less data consistency in the file system.	There is more data consistency because of the process of normalization.
File systems provide less security in comparison to DBMS.	DBMS has more security mechanisms as compared to file systems.
Only one user can access data at a time.	Multiple users can access data at a time.
It gives details of storage and representation of data	It hides the internal details of Database
Integrity Constraints are difficult to implement Example: C++	**Integrity constraints** are easy to implement Example :- MYSQL, Oracle.

TRANSACTION IN DBMS

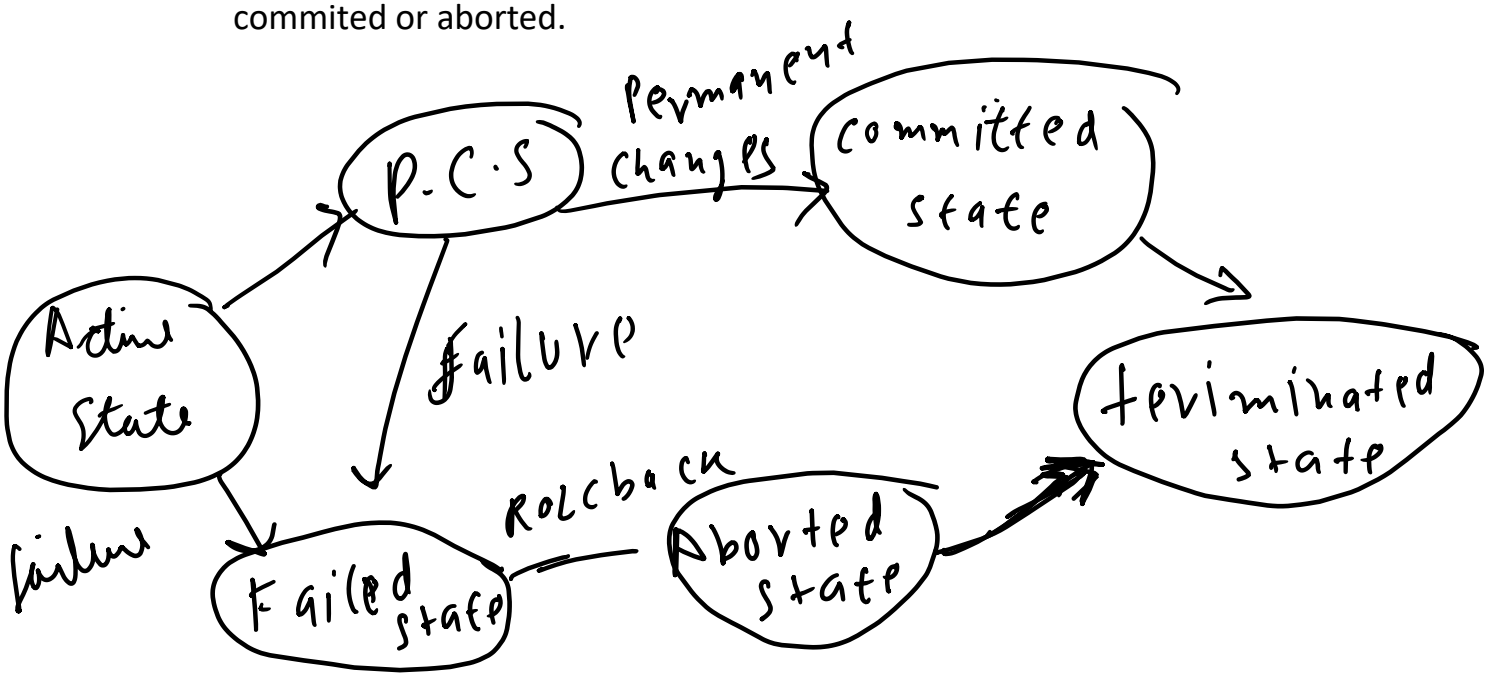
Work done in DBMS, more than one SQL command executed. Ex a bank Transaction.

Each Transaction in DB should follow **ACID properties** to ensure data integrity.

1. **Atomicity**: Each transaction to be atomic i.e either successful or failed nothing should be in between.
2. **Consistency**: Integrity constrains must be maintained before and after transaction. DB must be consistent after each transaction.
3. **Isolation:** Even though multiple transaction can occur concurrently the system must grantee that each transaction should be isolated.
4. **Durability**: After transaction completes, the changes should be made in the database even if the system failure occurs.

Transaction states

1. Active state : All rad and write operation is performed.
2. Partially Committed state: After transaction is executed the changes is saved in the buffer in the main memory.
3. Committed state : When updates are made permanent to DB. Rollback cannot be done.
4. Failed state: When any failure occurs.
5. Aborted state: When T reaches to failed state, the changes made in buffer is reverse. After that T rollbacks completely.
6. Terminated state: A transaction is said to have terminated if has either commited or aborted.

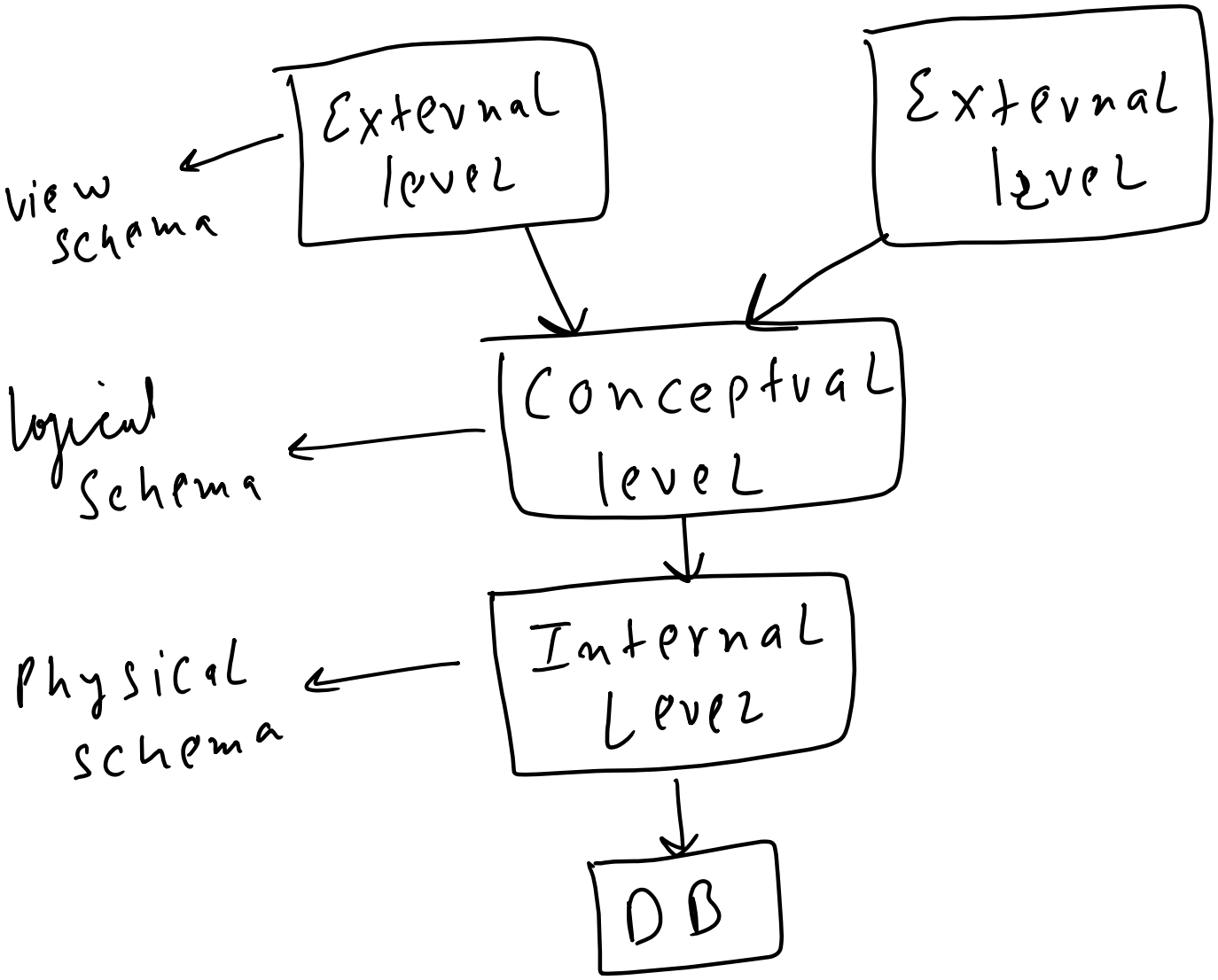


Implement Atomicity and Durability:

Recovery mechanism component of DBMS support atomicity and durability.

1. Shadow-copy Scheme: Here DB copy is made and the transaction happens in the copied DB. If transaction is successful than old db is deleted and new db is used for future. If transaction is failed than new db gets deleted and old db is not affected and used for future. This is not efficient as copy of db is made at each transaction.
2. Log-based recovery method: Log of each transaction is maintained here and if any failure happens than log help us to restore the original db state.

Three Schema Architecture



Ex:- 3 tive architecture

1. **Physical Level/ Internal Level:** Physical schema which tell how data is store data efficiently . Data compression encryption etc are done here.
2. **Logical level/ Conceptual Schema**: It describes what data is stored in db, and what kind of relationship they share. It help us to easily handle the data. They don't care how at physical level data is stored.
3. **View level/ external level**: It has view schema, which shows a particular database part to a user and hide remaining. Different view to different end user. Multiple Schema can be present know as subschema. Provide security.

Entity – Relation Model

- **Data Model:** Collection of conceptual tools for describing data, data relationship, consistency constrains.
- **E-R Model:** It is a high level data model, based on real world that consists of a collection of object called entities and relationship among them.

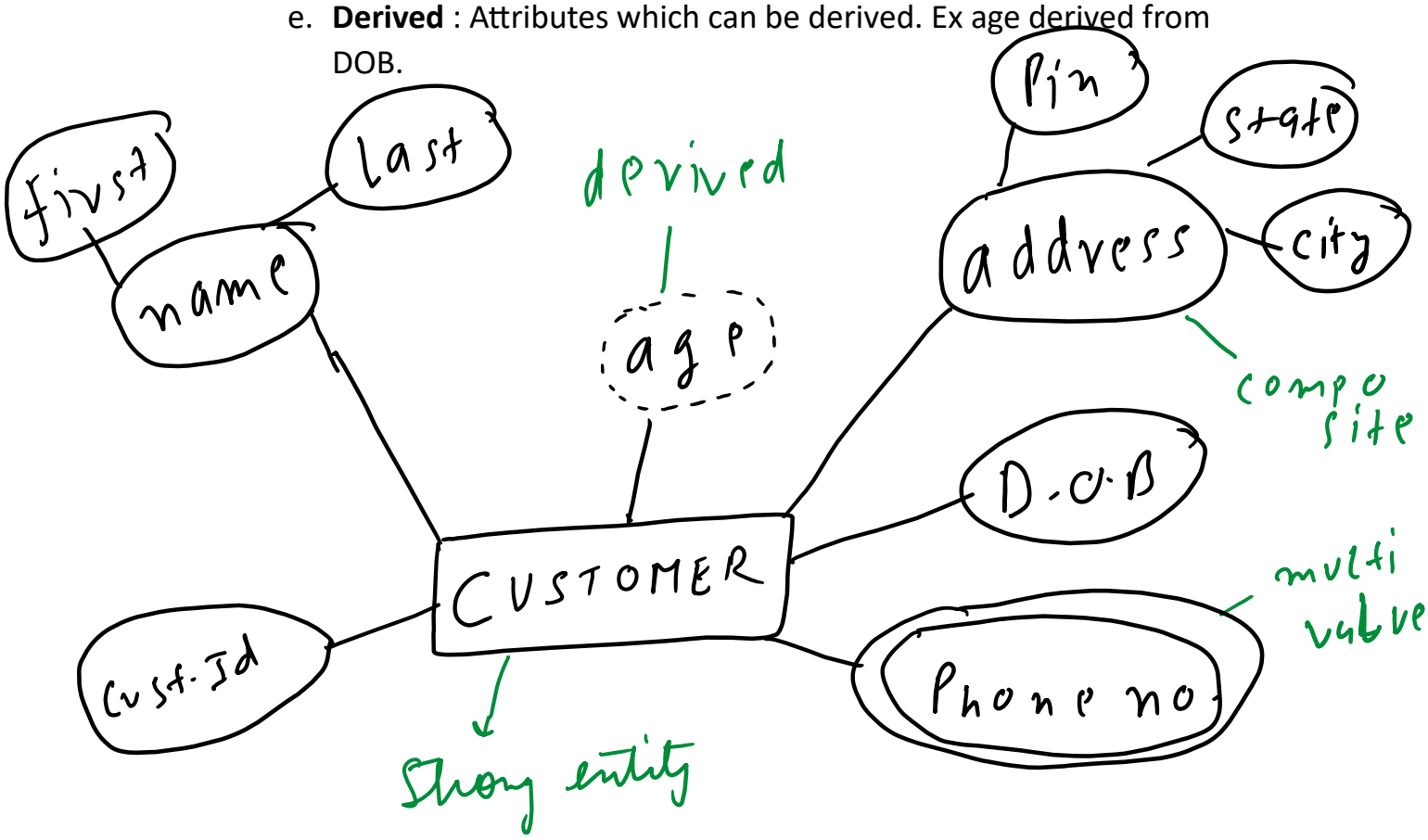
Terms in ER Model

1. **Entity:** Object, which is different from other object. It can be uniquely identified. They can be of two type :
   a. ***Strong Entity***: A strong entity is not dependent on any other entity in the schema. Represented by a rectangle.
   b. ***Weak Entity***: weak entity is dependent on a strong entity to ensure its existence.

   Ex: Customer borrows loan. Loan strong entity and loan weak entity.

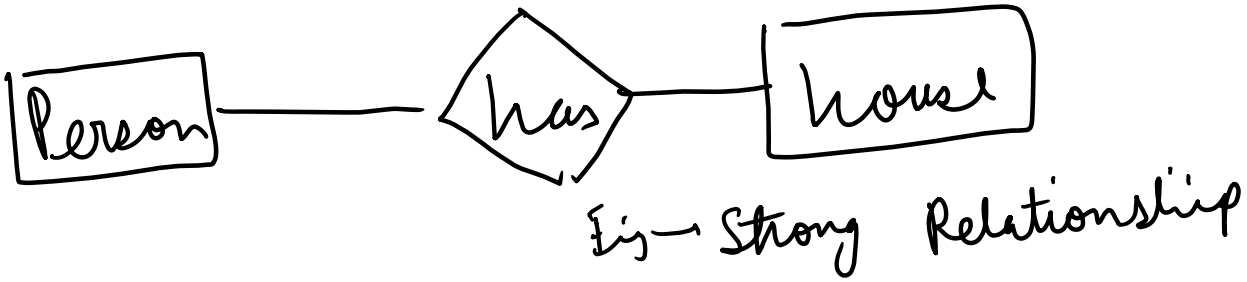
   Entity set : Collection of same type of entity.

2. **Attributes**: An entity is represented by set of attributes. Types of attributes are :
   a. **Simple attributes**: They cannot be divided. Ex: Reg no.
   b. **Composite attributes**: Can be divided. Ex Address, name etc.
   c. **Single valued**: Only one value.
   d. **Mulit value**: Having multiple value. Ex Phone no.
   e. **Derived** : Attributes which can be derived. Ex age derived from DOB.



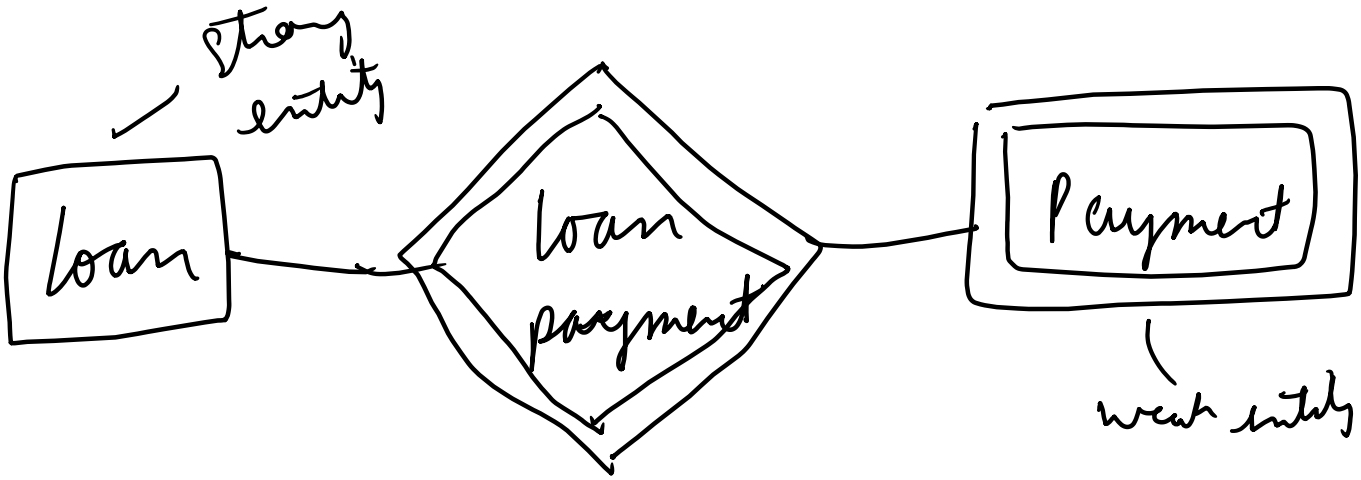
Eg Strong entity with attributes

3. Relationship: Associate two or more entities. Ex. Person has a car. Parent has a child.
   a. **Strong relationship**: Relation between two or more strong entities. Represented by a diamond shape.
      Example:



Eg- Strong Relationship

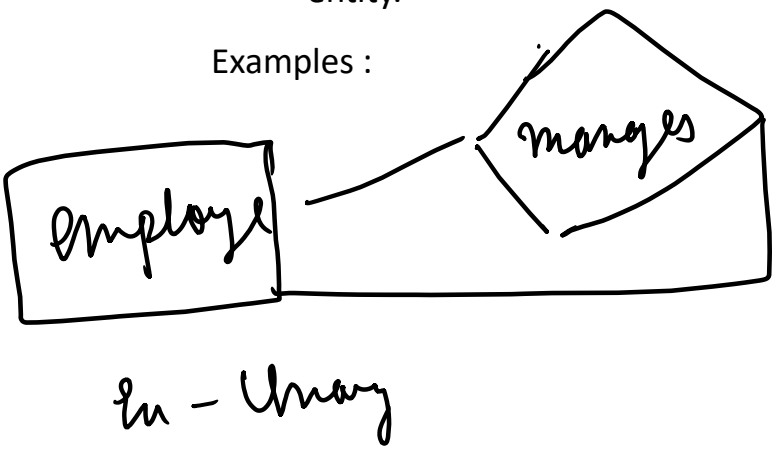
   b. **Weak relationship**: Between one strong and one weak entities. Represented by a double diamond shape.



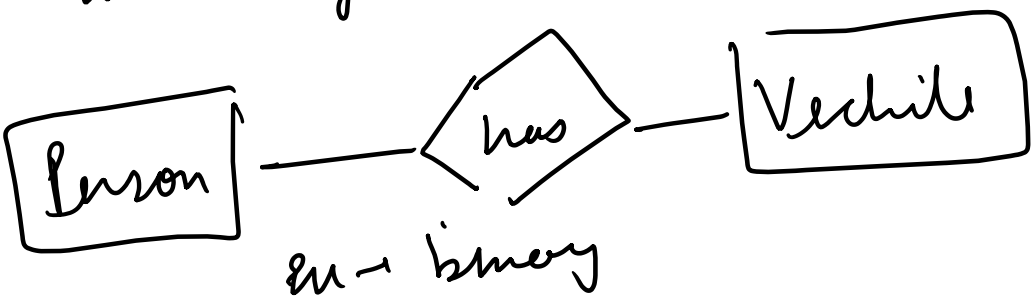
Eg weak relationship

c. Types of Relationship:
   i. **Unary relationship**: Relationship between same entity. Ex employee manages employee.
   ii. **Binary relationship**: Relationship between two different entity. Person has a car.
   iii. **Ternary relationship:** Relationship between three different entity.

Examples :



Eg - Unary



Eg - binary

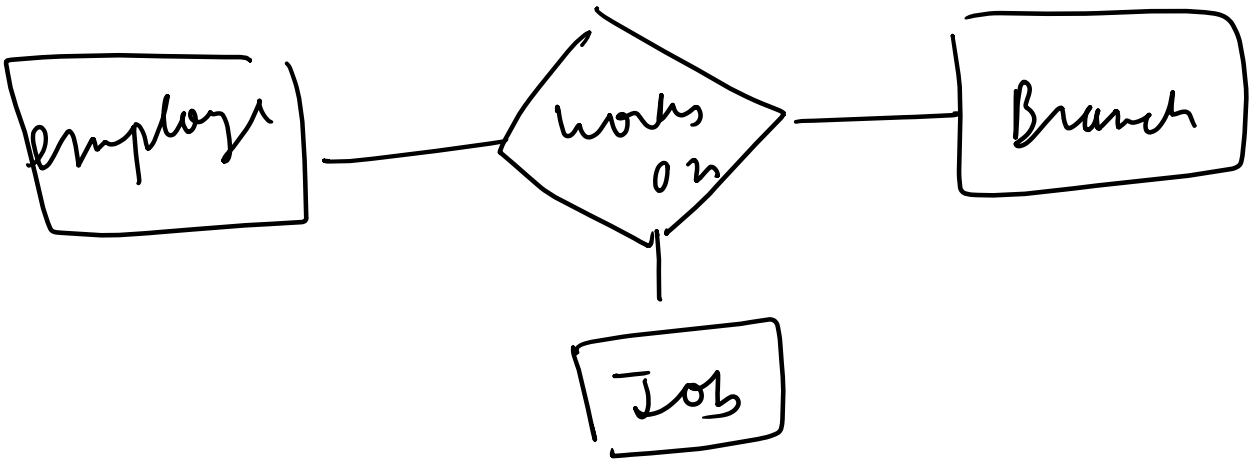
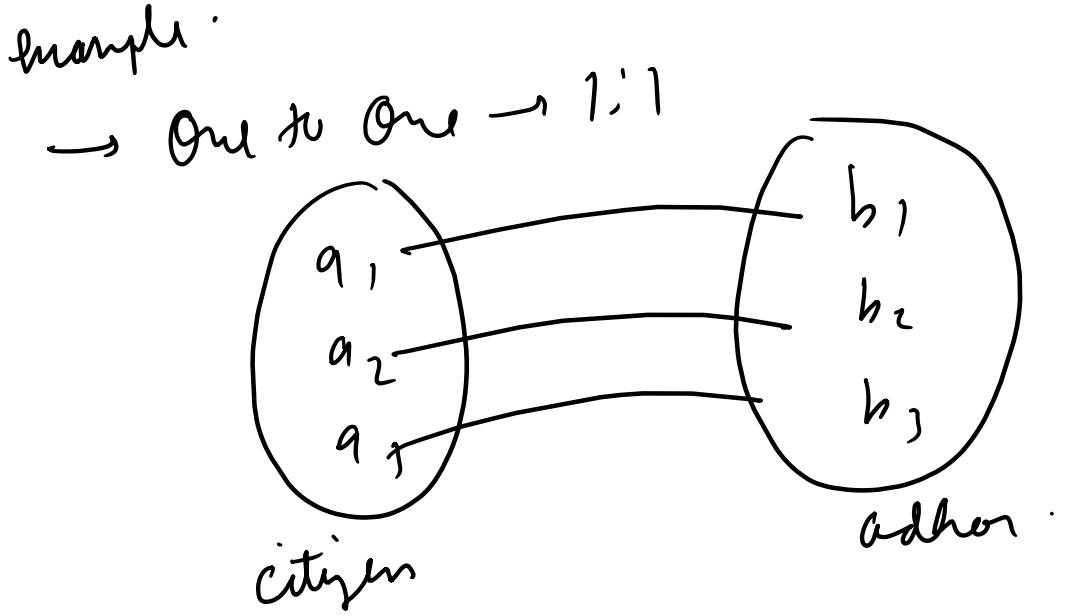
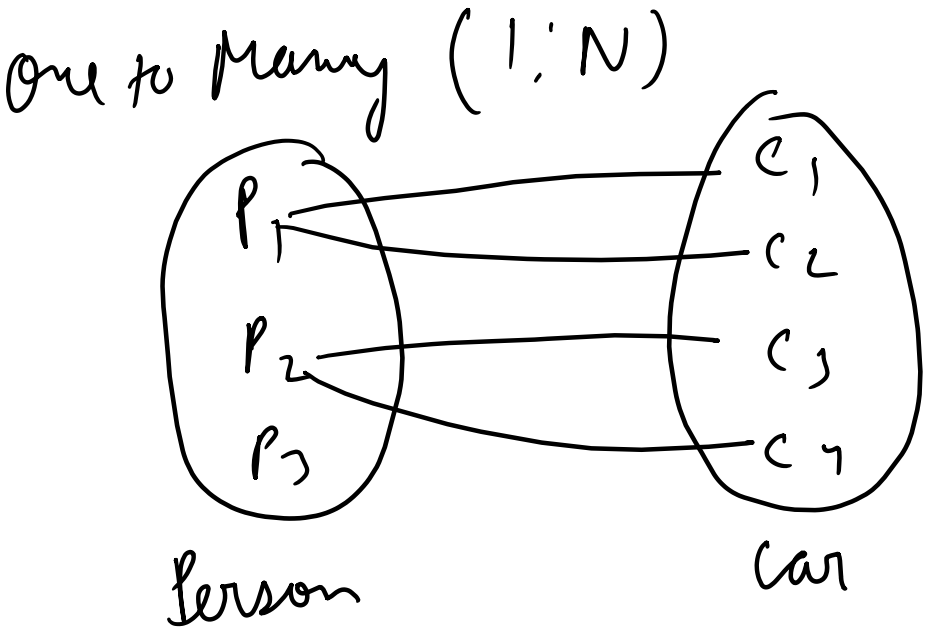


Fig ternary relationship

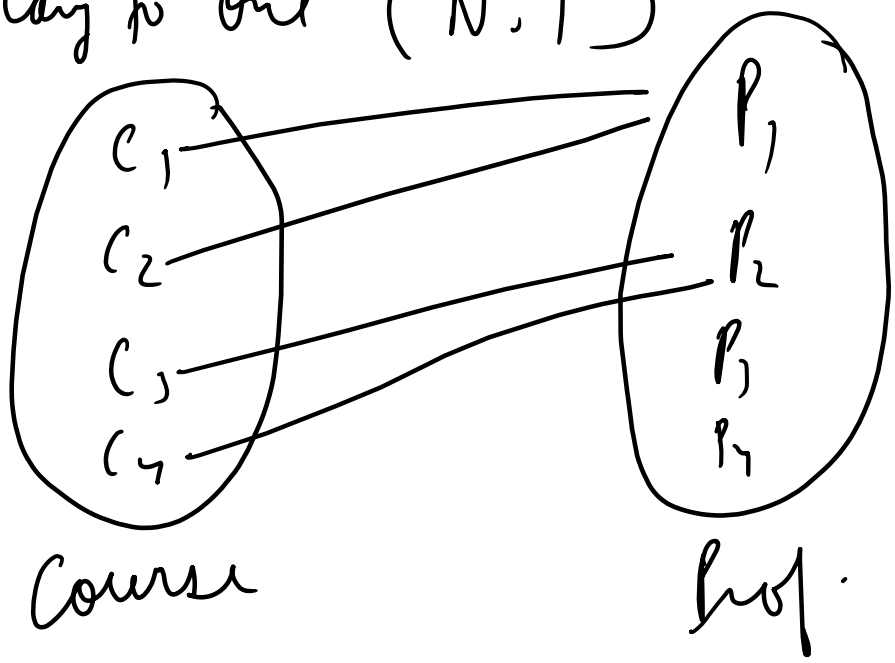
Note: Number of entity in relationship is called degree of relationship.

4. Relationship Constraints
   a. **Maping cardinality**/ cardinality ratio : Number of entity to which another entity can be associated via relationship.
      i. One to One -→ 1:1
      ii. One to Many -→ 1:N
      iii. Many to One -→ N:1
      iv. Many to Many → N:M
   b. Participation Constraint:
      i. Partial participation: Not all entity are involved in relationship
      ii. Total participation: Each enity must involved in relationship.

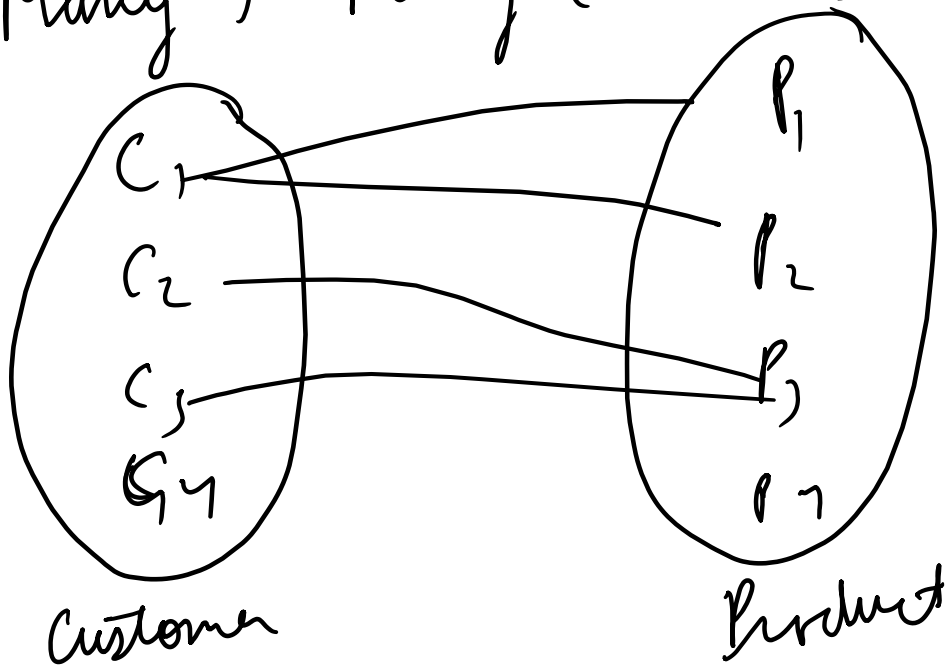


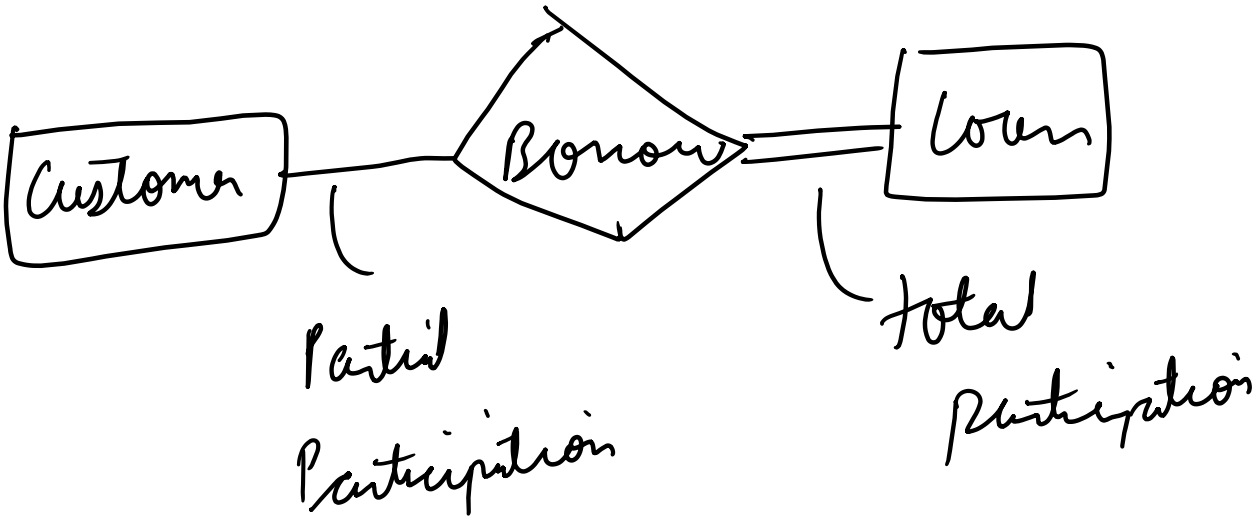


→ Many to One (N:1)



→ Many to Many (N:M)

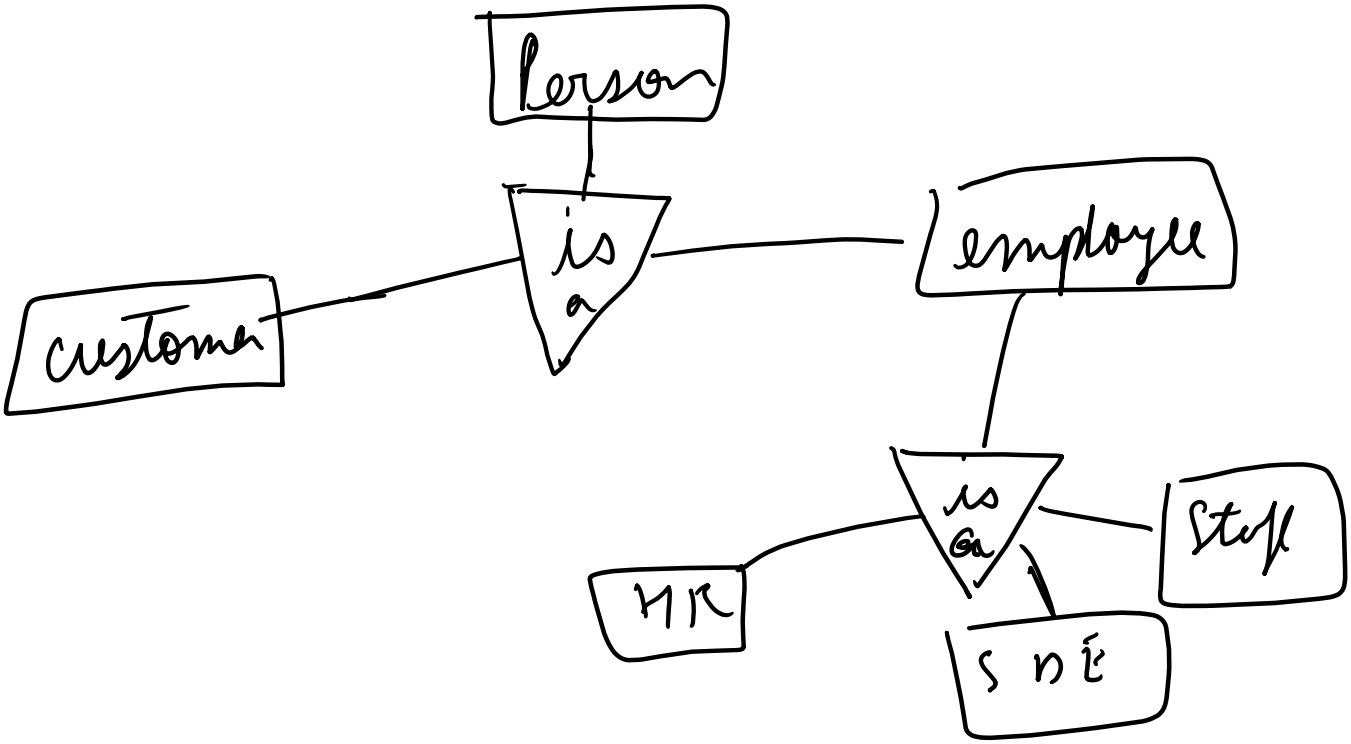




Extended ER features :

1. **Specialisation**: It splits entity set into sub entity set, based on functionalities. Top-down approach. It removes redundancy and refines the db.

   Example :



2. **Generalisation**: Reverse of Specialisation. Db designer may Encounter certain properties of two or more entity are overlapping, he can make new entity set, will be a super class. Ex car, bike ,jee → vechile. Bottom top approach.
3. **Attribute Inheritance**: The attribute of higher level entity is inherited by lower level entity.
4. **Participation Inheritance**: If parent entity set participates in relationship then its child will also participate.