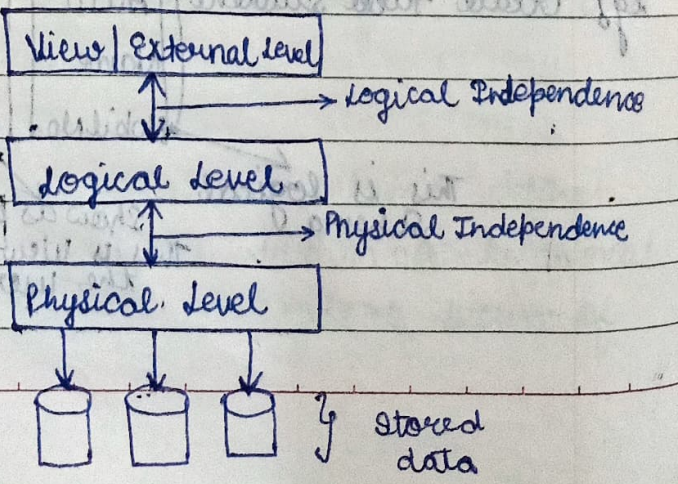
⇨ Database Instance?

- It is important that we distinguish these two terms individually. Database schema is the skelton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.
- A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instance tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints and conditions that the database designers have imposed.

⇨ Database Independence:

There are two types of independence physical and logical independence.



It refers to the capacity to change the schema of the database without affecting the application programs or user views that access the data.

⇨ Disadvantages of DBMS:

1. Cost of Hardware and Software: A processor with high speed of data processing and memory of large size is required to run the DBMS software. It means that you have to upgrade the hardware used for file based system. Similarly, database software is also very costly.

2. Complexity: The provision of the functionality that is expected from a good DBMS makes the DBMS an extremely complex piece of software. Failure to understand the system can lead to bad design decisions, which can have serious consequences for an organisation.

3. Cost of staff training: Mostly DBMS are often complex systems so the training for user to use the database is required. The organization has to pay a lot of amount for the training of staff to run the DBMS.

4. Appointing Technical staff: The trained technical persons such as database administrator,

application programmers etc are required to handle the database. You have to pay a lot of amount to these persons. Therefore the system cost increases.

5. Database Failure: In most of the organizations, all the data is integrated into a single database. If database is corrupted due to power failure or it is corrupted on the storage media, then our valuable data may be lost or whole system stop.

⇨ Advantages of DBMS:

1. Reduction in data Redundancy: The duplication of data refers to data redundancy. DBMS cannot make separate copies of the same data. All the data is kept at a place and different applications refer to data from centrally controlled system.

2. Better interaction with users: In DBMS, the availability of upto-date information improves the data to be access or respond as per user requests.

3. Improvement in Data Security: DBMS can allow the means of access to the database through the authorised channels. To ensure security,

DBMS | Page No.
Date / /

DBMS provides security tools, i.e username and password.

4. Maintenance of data integrity: Data integrity ensures that the data of database is accurate. In DBMS, data is centralised and used by many users at a time, it is essential to enforce integrity controls.

5. Ease of application Development: The application programmers need to develop the application programs according to the user's need. The other issues like concurrent access, security, data integrity, etc are handled by database itself. This makes the application development an easier task.

6. Backup and recovery: The DBMS provides backup and recovery subsystem that is responsible to recover data from hardware and software failures.

Database independence:

Data independence is the property of DBMS that helps you to change the database schema at one level of a database system without requiring to change the schema at the next higher level.

DOMS | Page No.
Date / /

- **Logical Data Independence:**

Logical data is data about database, that it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constrains applied on that relation.

Logical data independence is kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

Examples of changes under Logical Data Indepen

Due to logical independence, any of the below change will not affect the external layer:

1. Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs.
2. Merging two records into one.
3. Breaking an existing record into two or more records.

- **Physical Data Independence:**

All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

DOMS | Page No.
Date / /

Examples of changes under Physical Data Indepen—
Due to physical independence, any of the below change will not affect the conceptual layer.

1. Using a new storage device like Hard Drive or Magnetic Tapes.
2. Modifying the file organization technique in the database.
3. Switching to different data structures.
4. Changing the access method.
5. Modifying Indexes.
6. Changes to compression techniques or hashing algorithms.
7. Change of location of database from say C drive to D Drive.

⇨ **Attributes:** Attributes are an important component of database management systems. They refer to the specific characteristics or properties of an entity. Attributes describe the various features of an entity, such as its size, shape, colour, weight and so; it helps to distinguish it from other entities. A patient for eg; can be an entity in a hospital database, and attributes can include the patient's name, age, gender, blood type, medical history, and so on.

DOMS | Page No.
Date / /

Types of attribute:

1. Simple Attributes: Simple attributes are single-valued traits that cannot be further split. In a student database, for eg. AUID can be simple attribute.

2. Composite Attributes: composite qualities can be further subdivided into smaller sub-parts. A student address, for eg. can be further subdivided into street, city, state and zip code.

3. Multivalued Attributes: Have numerous values for the same entity. for eg. many Ph. no. emails.

4. Key attributes: Those that uniquely identify a database entity. for eg. a AUID can be the key property that uniquely identifies each student.

5. Derived attributes: Derived attributes are qualities that originate from other attribute in the database. The date of birth feature, for eg. can be used to calculate a student's age.

⇨ Entity:

When an object becomes uniquely identifiable we call it an entity.

An entity is a real-world thing which can be distinctly identified like a person, place or a concept. It is an object which is distinguishable from others. If we cannot distinguish it from others then it is an object but not an entity.

An entity can be of two types:

1. Tangible Entity: Tangible entities are those entities which exist in the real world physically eg - Person, car etc.
2. Intangible entity: These are those entities which exist only logically and have no physical existence. eg - Bank account etc.

Example: If we have a table of a Student (Roll_no, Student_name, Age, Mobile_no) then each Student in that table is an entity and can be uniquely identified by their Roll Number i.e Roll_no.

Entity Type: It is a collection of the entity having similar attributes. In the below student table example, we have each row as an entity and they are having common attributes i.e each row has its own value for attributes Roll_no, Age, Student_name and Mobile_no. So we

Date / /

Can define the STUDENT table as an entity type because it is a collection of entities having the same attributes. So, an entity type is an ER diagram is defined by a name (here, STUDENT) and a set of attributes (here Roll_no, Student_name, Age, Mobile_no). The table below shows how the data of different entities (different students) are stored.

Student → Entity Type

Roll_no	Student_name	Age	Mobile_no
1	Ram	18	893128069
2	Sham	19	3489 10010
3	Geeta	20	546789012

→ entity

Roll_no 1, 2, 3 are entity set.

We use rectangle to represent an entity type in the E-R diagram, not entity.

Student

* Types of Entity type

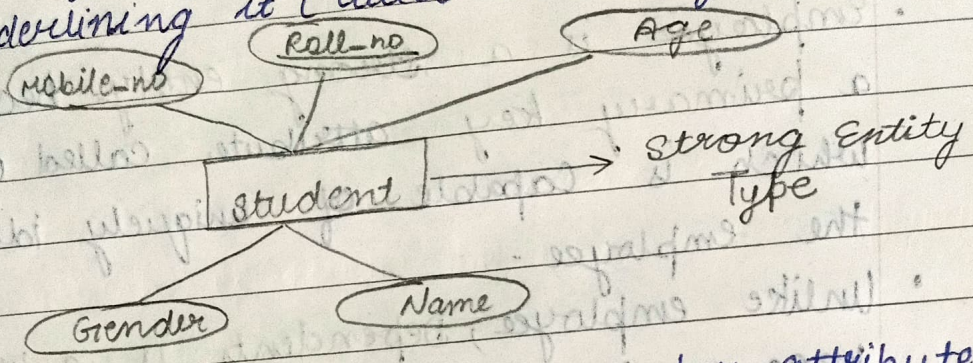
- Strong Entity type: These are those entity types which has a key attribute. A strong entity is nothing but an entity set having a primary key attribute or a table that consist of of a primary key column.

The primary key helps in identifying each entity uniquely. It is represent by a rectangle. In the above ex. Roll_no identifies each element of

table uniquely and hence we can say that STUDENT is a strong entity type.

Representation

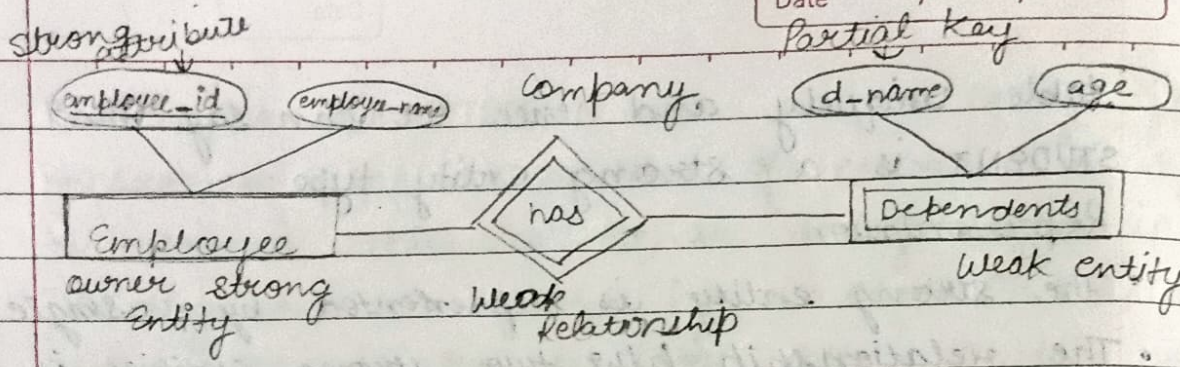
- The strong entity is represented by a single rect.
- The relationship b/w two strong entities is represented by single diamond.
- The primary key of the strong entity is represented by underlining it (called as strong attribute)



- **Weak Entity Type:** It doesn't have a key attribute. Weak entity type can't be identified on its own. It depends upon some other strong entity for its distinct identity. Simply a weak entity is nothing but an entity that does not have a primary key attribute

Representation:

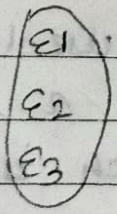
- A double rectangle is used for representing a weak entity set.
- A double diamond symbol is used for representing the relationship b/w a strong entity and a weak entity which is known as identifying relationship.



Example :

- In ER diagram we have two entities Employee and Dependents.
- Employee is a strong entity because it has a primary key attribute called employee no. which is capable of uniquely identifying all the employee.
- Unlike employee, Dependents is a weak entity because it does not have any primary key.
- D_name along with the employee_no can uniquely identify the records of dependents. So here D_name (dependents name) is partial key.

* Entity Set : It is a collection or set of all entities of a particular entity type at any point in time. The type of all the entities should be the same.



Entity Set

Example : The collection of all the students from the Student table at a particular instance of time is a example of entity set.

- The collection of all the employees from the employee table at a particular instant of time is an example of an entity set.

Relationship: A relationship in a DBMS is primarily the way two or more data sets are linked. Any association between two entities is known as a relationship between those two entities. This relationship in DBMS is represented using a diamond shape in the entity relationship diagram. An entity-relationship diagram is a graphical representation of entities and the relationships that exist between them. for eg: The driver drives a car.

In the above eg, the driver and the car are entities whereas the word drives is a relationship between those entities.

One of the huge advantages of a relational database is that, once you have your data held in clearly defined tables, you can connect or relate the data held in different tables. When analyzing table relationships, you need to look at the relationships from both sides. When creating table relationships you always work with two tables at a time. One table is called the primary or parent table and the other is the related or child table.

DOMS | Page No.
Date / /

One-To-One Relationship :

A one to one (1:1) relationship means that each record in Table A relates to one, and only one, record in Table B relates to one, and only one, record in Table A.

Eg:

Personal

EmployeeId	First Name	Last Name	Address	City	State	Zip
EN1-10	Ram	Kumar	Ajeet Road	Bbl	Pb	0151
EN2-20	Sham	Dev	Deep nagar	Mansa	Pb	0281
EN3-30	Seeta	Rani	Bibi wala road	Kotbra	Pb	0891
EN4-40	Geeta	Devi	kotshamir	Rampura	Pb	450

Payroll

EmployeeId	Pay Rate
EN1-10	25000
EN2-20	26000
EN3-30	27000
EN4-40	30000

Each record in the Personal table is about one employee. That record relates to one, and only one, record in the Payroll table. Each record in the Payroll table relates to one, and only one, record in the Personal table.

Another example:
If there are two entities, 'Person' (Name, age, address, contact no.) and 'Aadhar card' (Name, Aadhar no.). So each person can have only one aadhar card, and the single aadhar card belongs to only ~~only~~ one person.

This type of relationship is used for security purposes. In the above example, we can store the aadhar card number in the entity 'Person' but we created another table for the 'Aadhar card' because the aadhar card no. may be sensitive data and should be hidden from others.

One-To-Many Relationship:
A one-to-many (1:N) relationship means a record in Table A can relate to zero, one or many records in Table B. Many records in Table B can relate to one record in Table A. The potential relationship is what's important; for a single record in Table A there might be no related records in Table B or there might be only one related record, but there could be many.

Customer id	Customer name	Customer no.
111	X	12345678
222	Y	90123456
333	Z	78901234

Order id	Order amount	Customer id
10001	1200	222
10002	2000	333
10003	4500	222
10004	1000	111
10005	3500	222

The Customer table holds a unique records for each customer. Each Customer can place many orders. Many records in the Orders Table can relate to only one record in the Customer table. This is the one-to-many relationship (1:M) between the customer table and the orders table.

In a one-to-many relationship, the table on the one side of the relationship is the primary table and the table on the many side is the related table.

Many-To-Many Relationship:

Examine the sample data below. These tables hold data about employees and the projects to which they are assigned. Each project

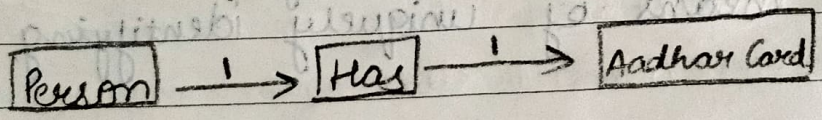
can involve more than one employee and each employee can be working on more than one projects. This constitutes a many-to-many (N:N) relationship.

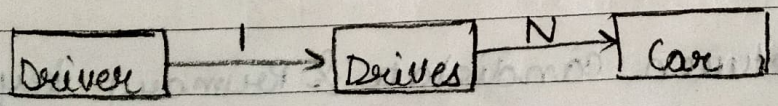
Employees

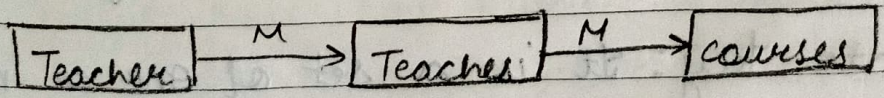
Employee Id	Last Name	First Name	Project Num
ENI-26	Kumar	Ram	30A
ENI-26	Kumar	Ram	34A
ENI-33	Rani	Seeta	30A
ENI-35	Devi	Geeta	35B

Projects

Project No	Project Title	Employee Id
30A	C++	ENI-26
30A	C++	ENI-33
34A	C	ENI-26
35B	Python	ENI-35







DOMS | Page No.
Date / /

Keys:

- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.
- Helps you to enforce identity and integrity in the relationship.

- Candidate key: A candidate key in a database management system (DBMS) is a unique identifier for a record within a table that can be chosen as the primary key. It possess the essential characteristics required for a primary key: uniqueness and minimal redundancy. While multiple candidate keys may exist within a table, the chosen primary key becomes the definitive means of uniquely identifying each record

Difference between candidate & Primary key

Candidate key: It is a set of one or more attributes that can uniquely identify a record in a table. Multiple candidate keys may exist within a table, and each candidate key must ensure uniqueness.

DOMS | Page No.
Date / /

Primary key: It is a specific candidate key chosen as the main means of uniquely identifying records within a table. Unlike candidate keys, a table can have only one primary key, and it ensures uniqueness and non-null values.

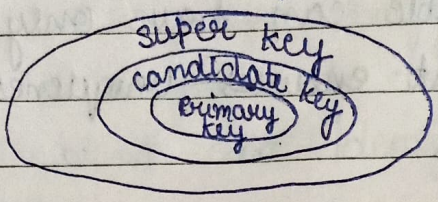
Primary key: It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.

- In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_no and Passport_no as primary keys since they are also unique.
- For each entity, the primary key selection is based on requirements and developers.

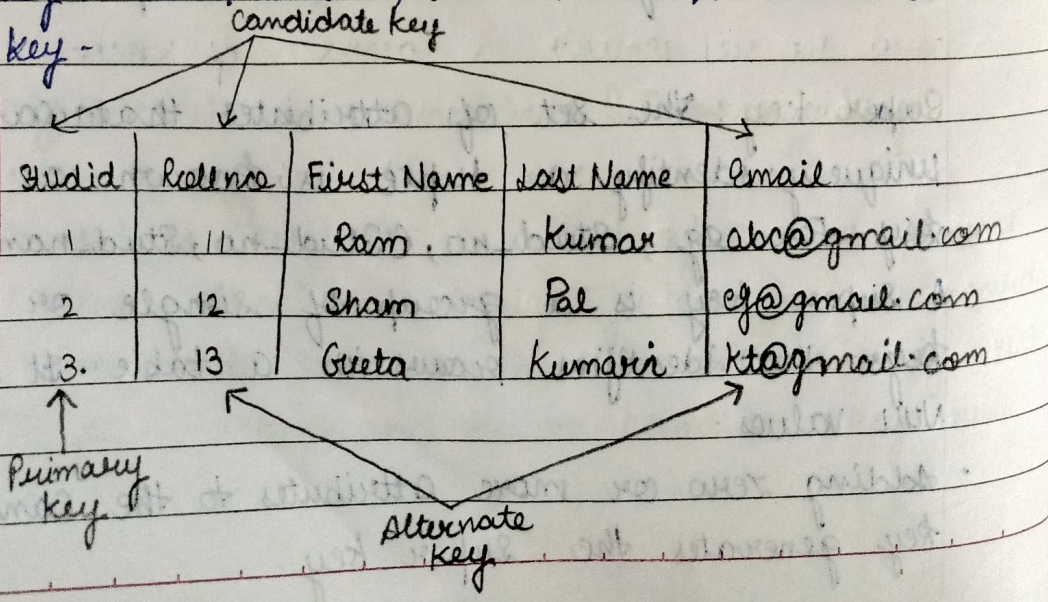
Super key: The set of attributes that can uniquely identify a tuple is known as Super key. For eg, Stud_no, (Stud_no, Stud_name) etc. A Super key is a group of single or multiple keys that identifies rows in a table. It supports NULL values.

- Adding zero or more attributes to the candidate key generates the super key.

- A candidate key is a super key but vice versa is not true.
- Super key values may also be NULL.

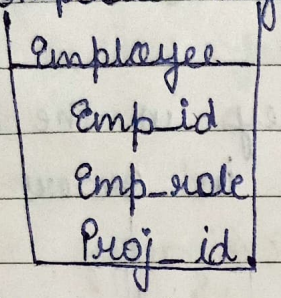


Alternate key: There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, it termed the alternate key. In other words, the total no. of the alternate keys is the total no. of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.



DOMS | Page No.
Date / /

Composite key: Whenever a primary key consists of more than one attribute, it is known as a composite key.



For eg, in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_id, Emp_role and Proj_id in combination. So these attributes act as a composite key since the primary key comprises more than one attribute

Foreign key: If an attribute can only take the values which are present as values of some other attribute, it will be a foreign key to the attribute to which it refer. The relation which is being referenced is called referenced relation and the corresponding attribute is called referencing attribute. The referenced attribute of the referenced relation should be the primary key to it.

- It is a key it acts as a primary key in one table and it acts as secondary key in another table.

- It combines two or more relations (tables) at a table.
- They act as a cross-reference between the tables.
- For eg, id is a primary key in the Customer table and a foreign-key in Orders table.

Table: Orders

Foreign key ↑

Order id	Product	Total	Customer id
1	Paper	500	4
2	Pen	10	2
3	Marker	120	3
4	Books	1000	1

Table: Customer

id	Name	age	country
1	X	30	USA
2	Y	21	India
3	Z	28	UK
4	A	29	USA

The entity relationship diagram explains the relationship among the entities present in the database. ER models are used to model real world objects like a person, car, company and the relation between these real-world objects. In short, the ER diagram is the structural format of the database.

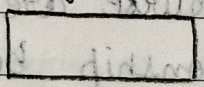
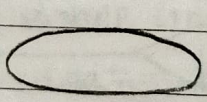
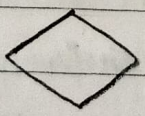
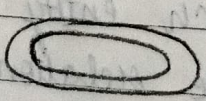
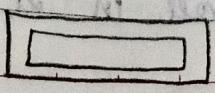
DOMS | Page No.
Date / /

Why use ER diagrams in DBMS?

- It helps you conceptualize the database and lets you know which fields need to be embedded for a particular entity.
- It gives a better understanding of the information to be stored in a database.
- It reduces complexity and allows database designers to build database quickly.
- It helps to describe elements using entity relationship models.
- It allows users to get a preview of the logical structure of the database.

Symbols Used in ER Model:

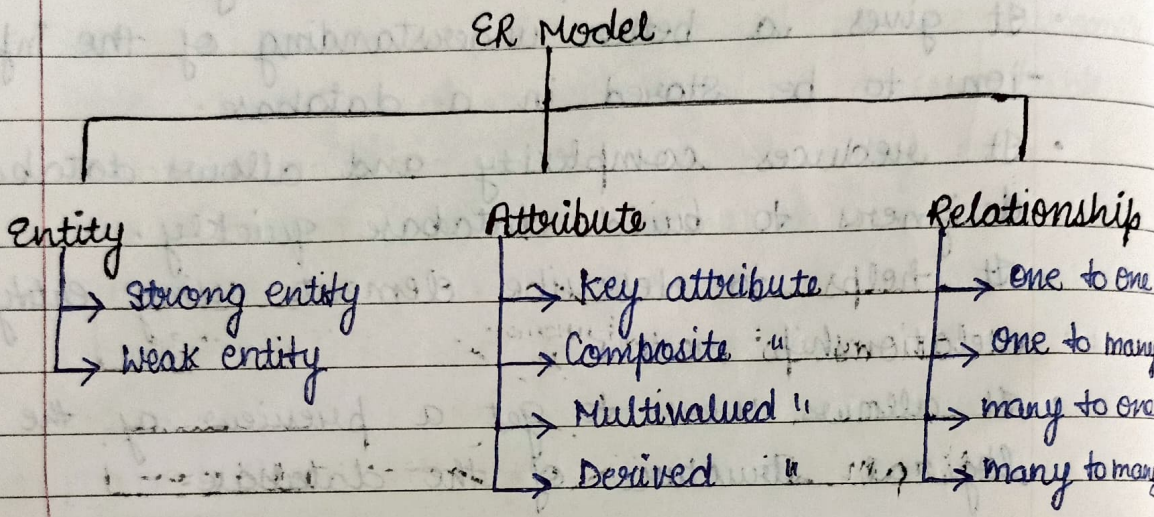
ER Model is used to model the logical view of the system from a data perspective which consist of these symbols:

Figures	Symbols	Represents
Rectangle		Entities in ER Model
Ellipse		Attributes
Diamond		Relationships among entities
Line		Attributes to entities & entity sets with other relationship type
Double ellipse		Multi-valued attributes
Double rectangle		weak entity

DOMS | Page No.
Date / /

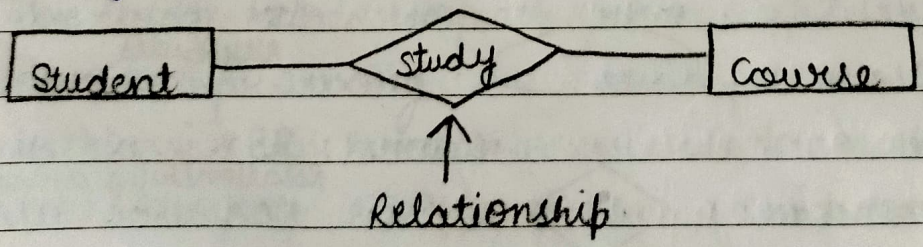
Components of ER Diagram:

ER model consists of Entities, Attributes and relation-hips among entities in a database system.



Relationship

The diamond shape showcases a relationship in the ER diagram. It depicts the relationship between two entities. In the eg. below both the Student & the Course are entities and study is the relationship between them.



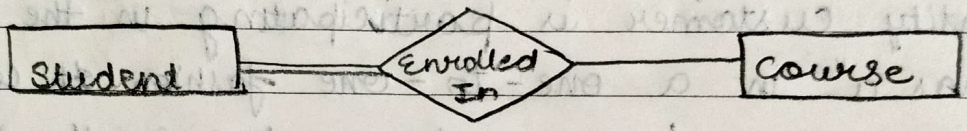
Participation Constraint:

1. Total Participation: Each entity in the entity set must participate in the relationship. If each Student must enroll in a course, the participation

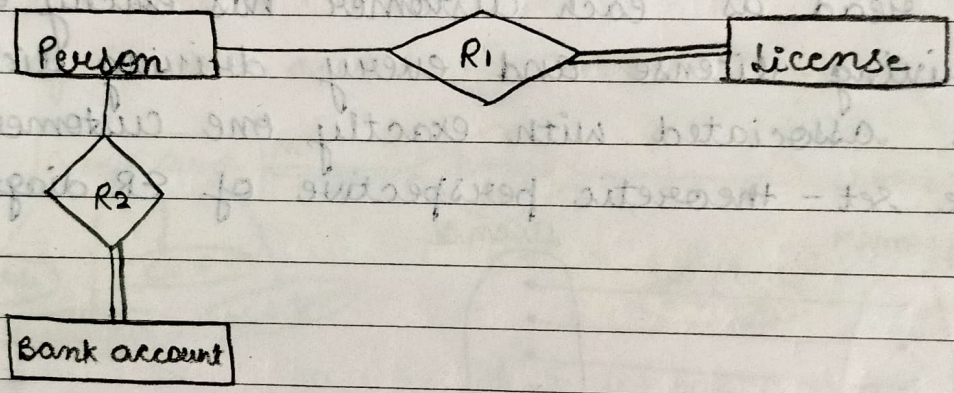
of students will be total. The total participation is shown by double line in ER diagram.

2. Partial Participation: The entity in the entity set may or may not participate in the relationship. If some courses are not enrolled by any of the students, the participation in the course will be partial.

eg!



eg:

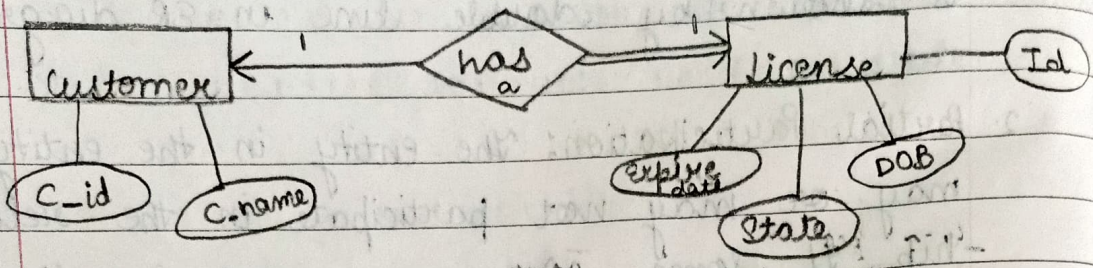


Cardinality: It refers to the maximum number of times an instance in one entity can relate to instances of another entity. Ordinality, on the other hand, is the minimum number of times an instance in one entity can be associated with an instance in the related entity.

1. One to One relationship (1:1): The cardinality ratio and participation constraint together is called structural constraint of the relationship type.

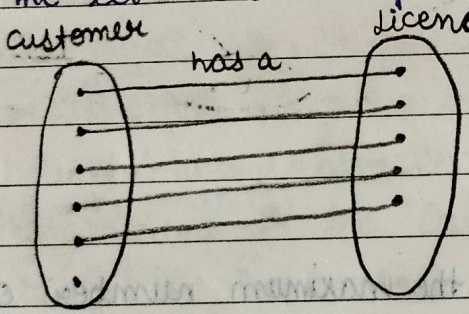
It is represented using an arrow (←, →)

eg



In this ER diagram both entities customer and license having an arrow which means the entity customer is participating in the relation "has a" in a one-to-one fashion. It could be read as 'each customer has exactly one driving license and every driving license is associated with exactly one customer.

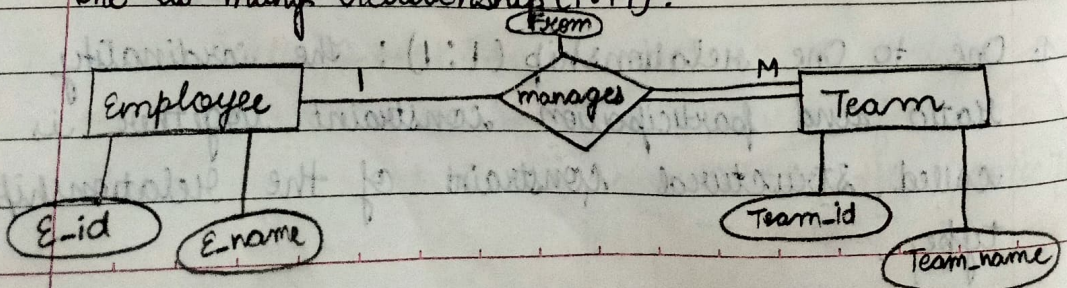
The set-theoretic perspective of ER diagram is



There may be customers who do not have a license, but every license is associated with exactly one customer. Therefore, the entity customer has total participation.

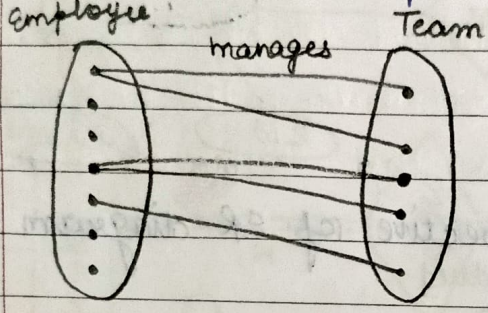
One to many relationship (1:M):

eg



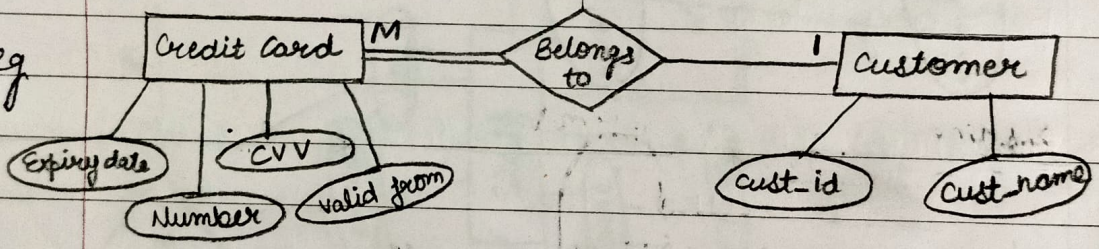
This relationship is 1:M because "There are some employees who manage more than one team while there is only one manager to manage a team."

The set-theoretic perspective of the ER diagram is:



Many to one relationship (M:1):

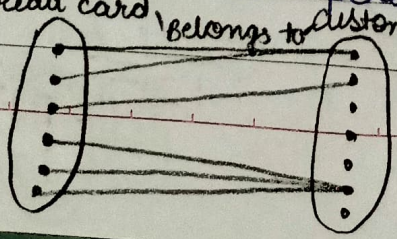
eg



It is related to 1:M relationship but the difference is due to perspective.

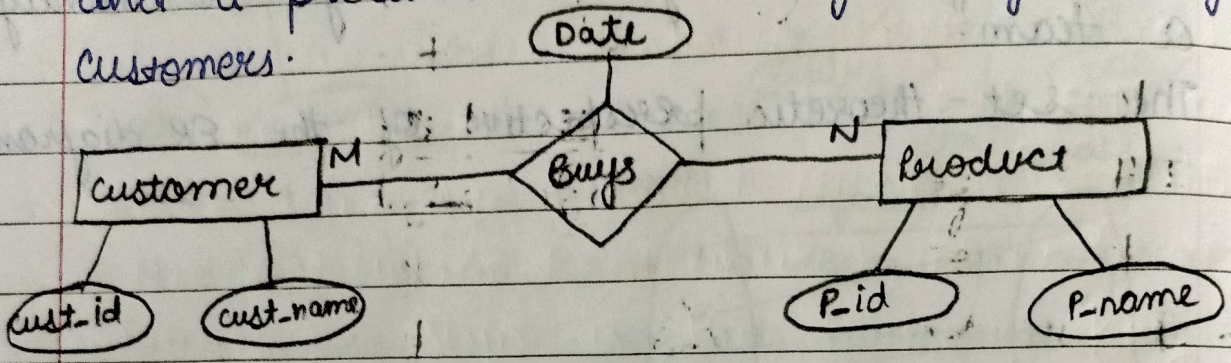
Any no. of credit cards can belong to a customer and there might be some customers who do not have any credit card, but every credit card in a system has to be associated with an employee (i.e total participation). While a single credit card can not belong to multiple customers.

The set-theoretic perspective of ER diagram is:

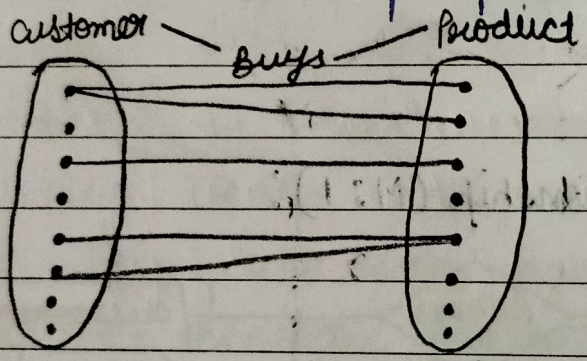


Many to Many relationship (M:N):

Eg: A customer can buy any number of products and a product can be bought by many customers.

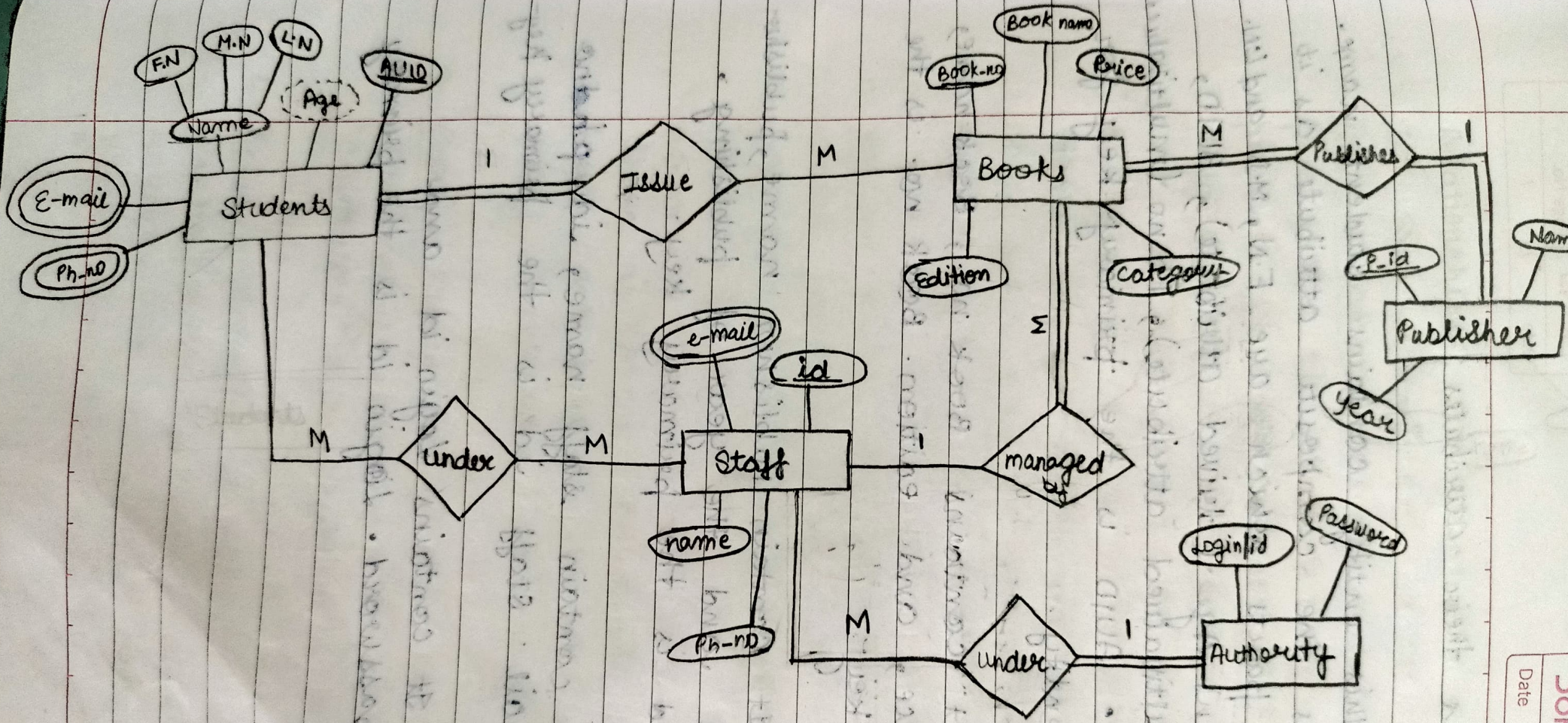


The set-theoretic perspective of ER diagram is:



ER Model on Library Management:

The library E-R diagram gives the information about library. It include entities such as students, Books, Staff, publisher etc. This ER diagram is used to understand the relationship between all the entities.


F.N
M.N
L.N
Name
Age
AUID
E-mail
Ph-no
Students
1
Issue
M
Book name
Book-no
Price
Books
M
Publishes
1
Name
P_id
Publisher
Year
Edition
Category
Σ
e-mail
id
M
Under
M
Staff
1
managed
name
Ph-no
M
Under
1
Login id
Password
Authority

Entities and their attributes:

1. Students: This entity contains Student name. Name is the composite attribute as it has sub parts which are F.N, M.N and L.N. It Contains Age (derived attribute), AUID, E-Mail (Multivalued attribute), Ph_no (multivalue attribute). AUID is the primary key for student entity.

2 Books: It contains Book no., Book name, Category, price, and edition. Book no. is the primary key.

3. Publisher: It contain Publisher name, publisher id and the year of publishing. Publisher id is the primary key.

4 Staff: It contain staff name, id, phone and email. Staff id is the primary key

5. Authority: It contains Login id and password. Login id is the primary key.

Relationships :

1. Between Books and students : There is partial participation of students and partial participation of Book. Because not all the students issue the book as well as not all the books get issued by students.
There is 1 : M cardinality as 1 student can issue many books but 1 book is not issued by M students.

2. Between Books and Publishers : There is total participation of books and total participation of publishers. As all the books published by any publisher whose name is present in system.
There is M : 1 cardinality because many books get published by 1 publisher but many publishers cannot publish 1 book.

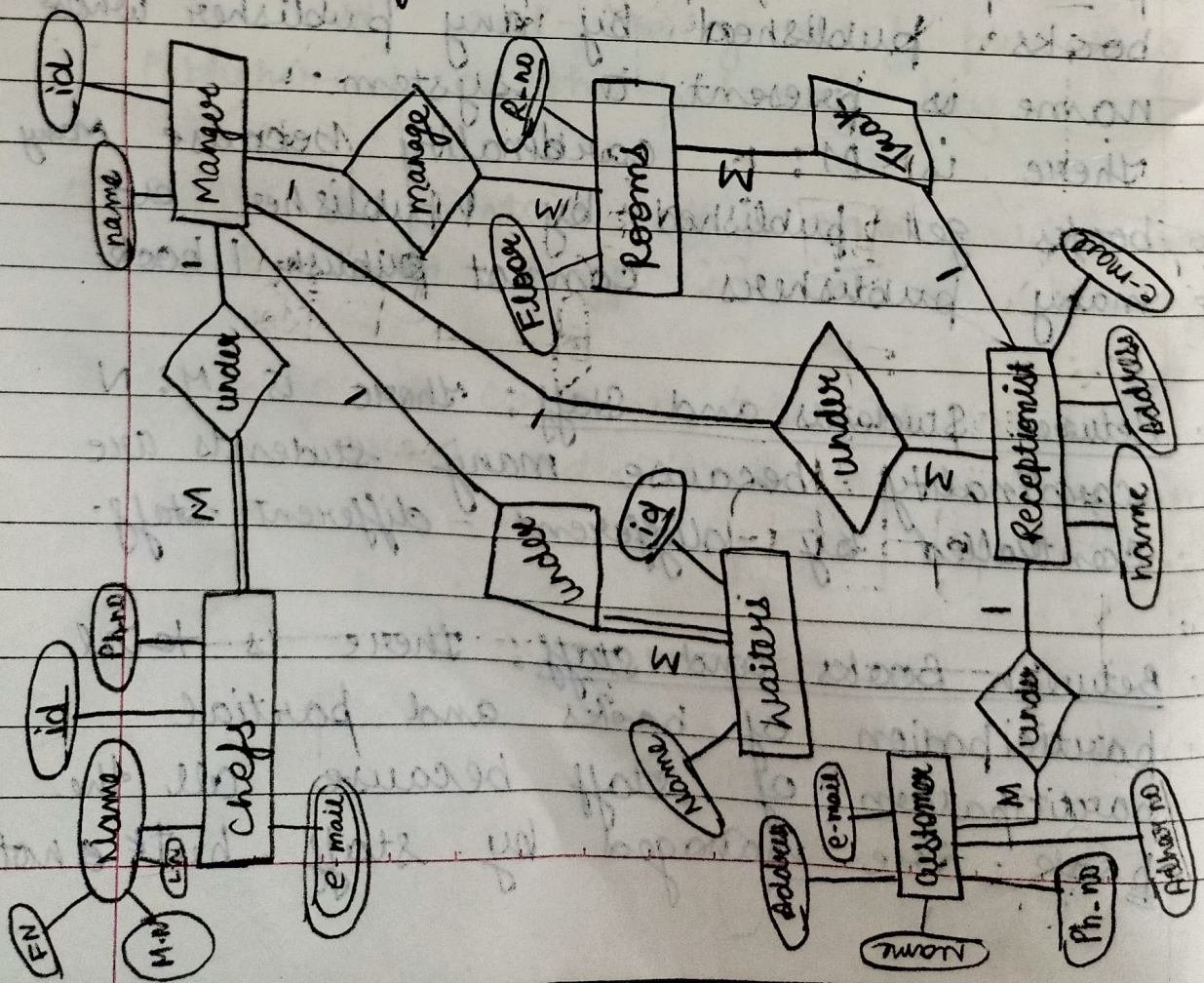
3. Between Students and Staff : There is M : N cardinality because many students are controlled by different - different staff.

4. Between Books and Staff : There is total participation of books and partial participation of staff because all the book are managed by staff but not

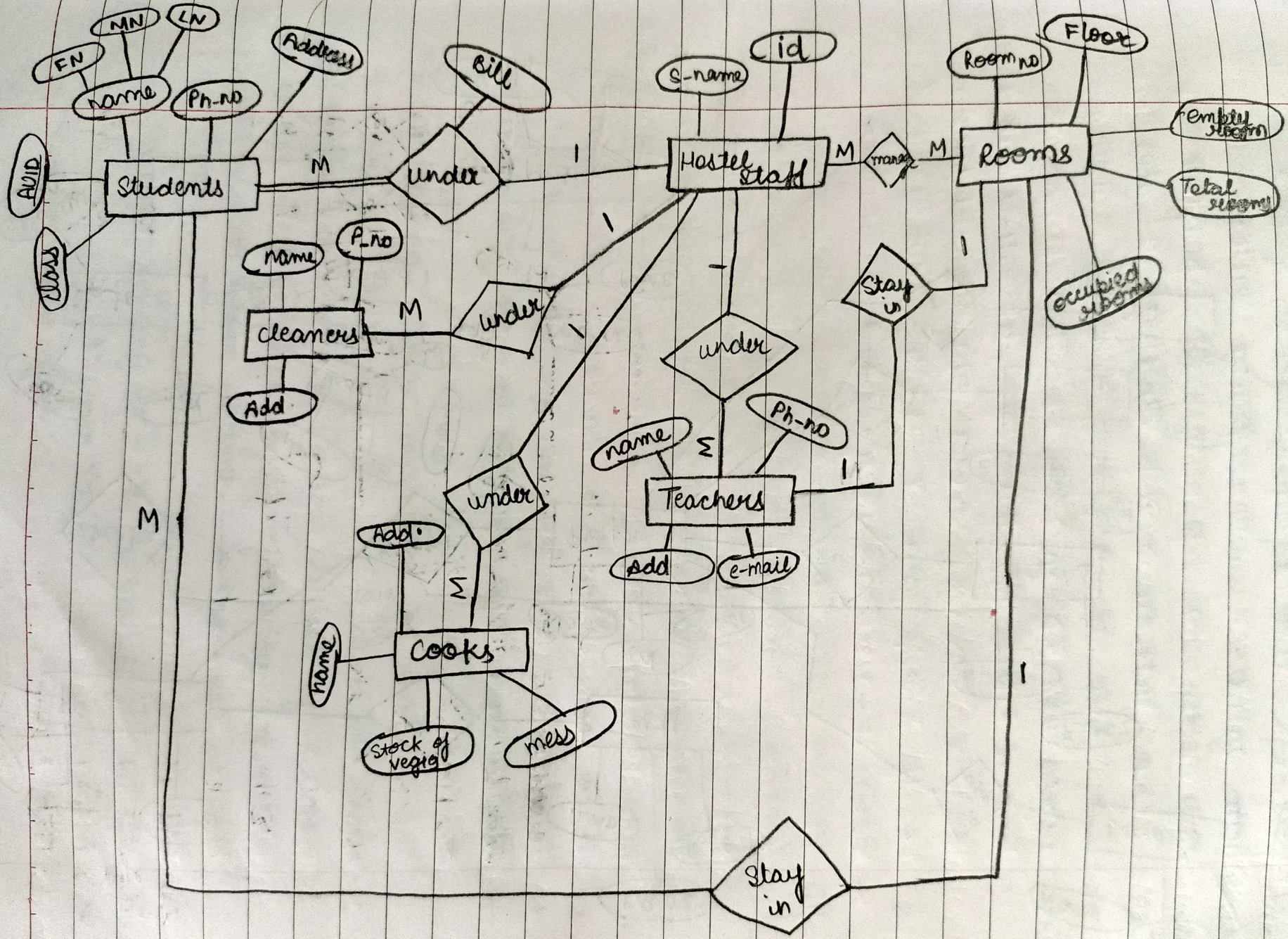
by each and every staff.
There is M:1 Cardinality because many books are managed by 1 staff.

5. Between Staff and authority: There is total participation of authority and partial participation of staff. Because not every staff member is the part permanent part of laboring or they are the simple staff members.
There is M:1 caridinality. Because many staff members are managed by only 1 authority.

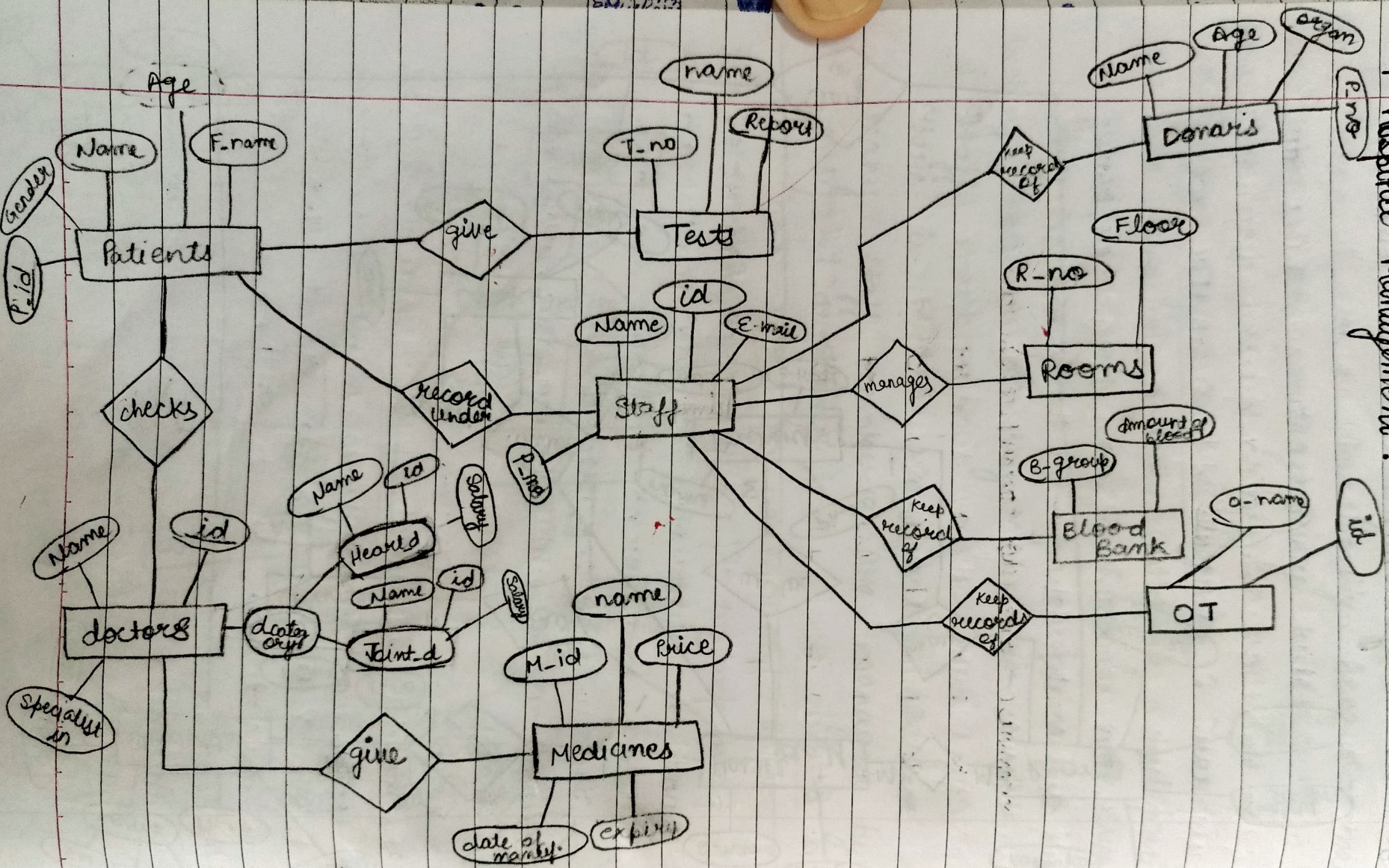
2. Hotel Management:



3. Hostel Management :



4. Hospital Management :



DOMS
Page No.
Date / /