Chapter-2: Data Models

2.1 Three level architecture of Data model.

The goal of three-schema architecture, illustrated in following figure is to separate the user applications and the physical database. In this architecture, schemas can be defined at the following three levels.

i. The **internal level** has an **internal schema**, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

ii. The **conceptual level** has a **conceptual schema**, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures & concentrates on describing entities, data types, relationships, user operations, and constraints. A high-level data model or an implementation data model can be used at this level.

iii. The **external or view level** includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. A high-level data model or implementation data model can be used at this level.

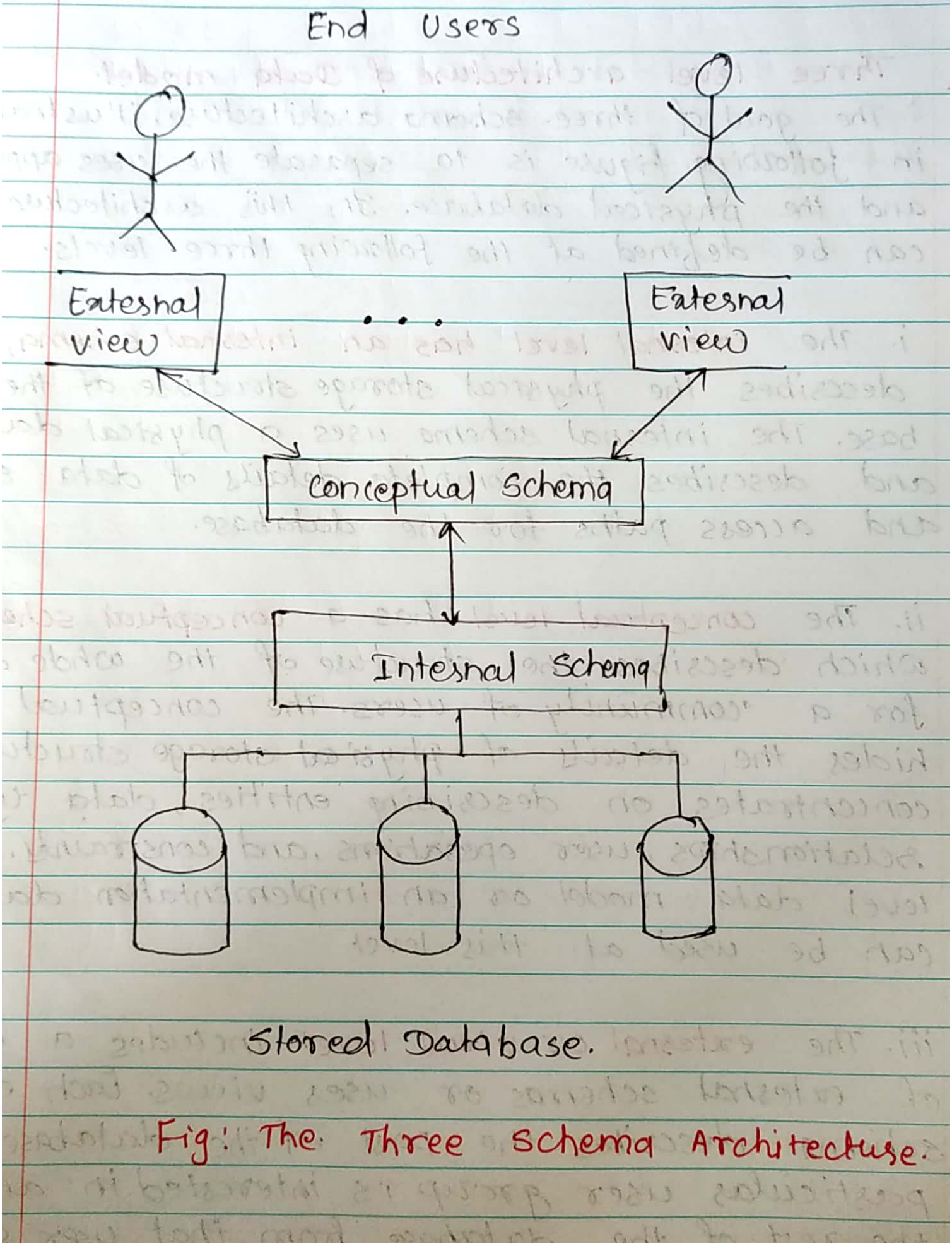


Fig: The Three Schema Architecture.

Data Models.

↳ A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

↳ Data Model is link between user's view of the world and bits stored in computes.

Categories of Data Model.

Object-based Data Model.

↳ Provides concepts that are close to the way many users perceive data. Also called entity-based data model.

↳ It uses concepts such as entities, attributes and relationships.

common types of object-based data models are

i. Entity-relationship
ii. Semantics
iii. Functional
iv. Object-oriented.

Physical Data Model.

↳ Provide concepts that describe detail of how details stored in the computer, representing information such as record structures, record ordering and access paths.

↳ Only of interest to database designers, implementers and maintainess ... not end users.

↳ Must provide a well-defined structure that can be mapped to conceptual schema.

Record-Based Data Model:

↳ Provides concept that fall between the above two, balancing user views with some computer storage details.

↳ Database consists of a number of fixed format records possibly of different types.

↳ Types of Record-based data model:
- i. Relational
- ii. Network
- iii. Hierarchical

Relational Model.

↳ The model was first proposed by Dr. E.F. codd of IBM in 1970

↳ The relational model of data is based on the concept of a mathematical relations.

↳ In the relational model, data and relationship are represented as tables, each of which has a numbes of columns with a unique name.

↳ A Relation is a mathematical concept based on

the ideas of sets, It's simple and easy to maintain.

↳ The model is based on a collection of tables

↳ Relational model is most widely used data model for commercial data-processing.

↳ The relation is formed over the cartesian product of the sets; each set has values from a domain, that domain is used in a specific role which is conveyed by the attributes name.

↳ Formaly, given sets $D_1, D_2, \ldots D_n$ a relation r is a subset of $D_1 \times D_2 \times \ldots \times D_n$ Thus a relation is a set of n-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$.

Example:

if

customes_name = { Ram, shyam, Hasi, Sita, ... }
customes_street = { Buddhanagar, imadol, lagankhel, ... }
customes_city = { kathmandu, Lalitpur, Lalitpur, ... }

Then r = { (Ram, Buddhanagar, kathmandu),
(shyam, imadol, Lalitpur),
(Hasi, lagankhel, Lalitpur) .... }

is a relation over; customes_name * customes.street * customes_city.

Relation schema.

↳ The name of a relation and the set of attributes for a relation is called a schema.

↳ The schema for the relation with the relation name followed by a parenthesized list of its attributes.

↳ $A_1, A_2, \ldots, A_n$ are attributes.

$R = (A_1, A_2, \ldots, A_n)$ is a relation schema.

Example: customer_schema = (customer_name, customes_street, customes_city)

r(R) denotes a relation r on the relation schema R

Example : customes (customes_schema)

↳ The rows of a relation, others than the heades row containing the attribute names are called tuples.

↳ Numbes of tuples present in the relation is called cardinality of relation.

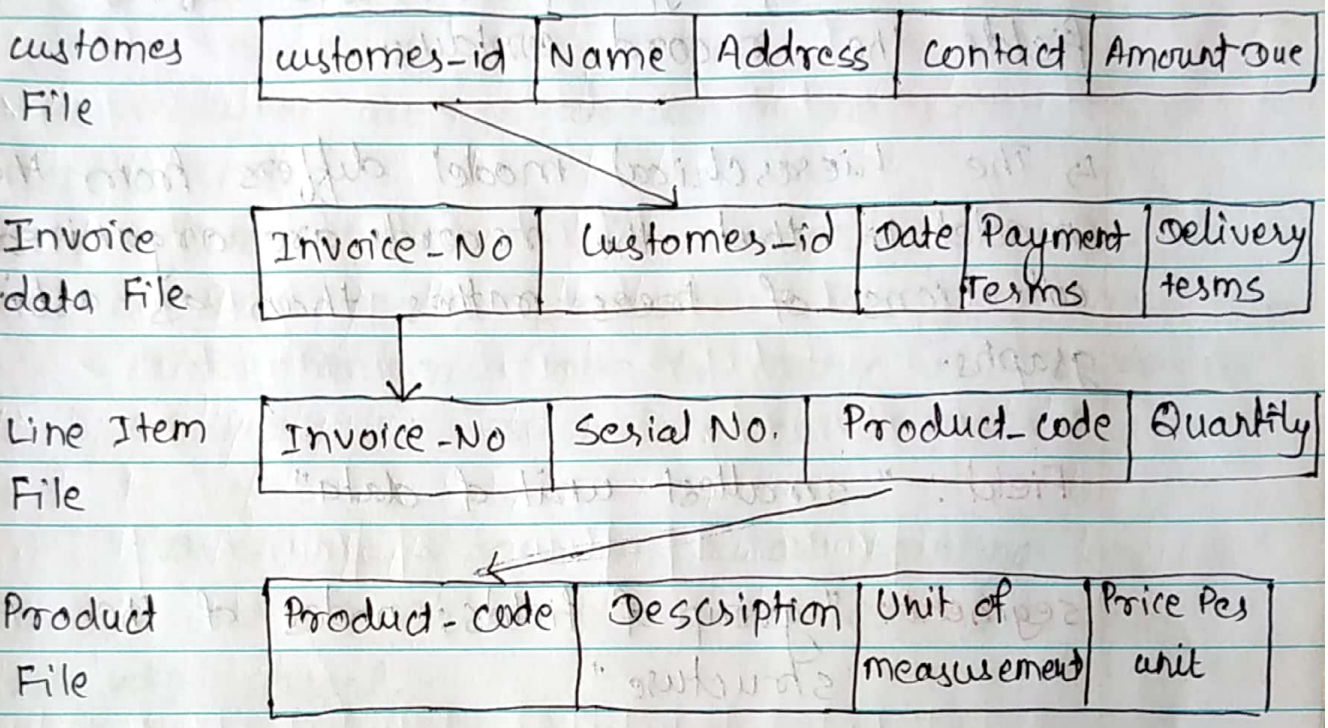
↳ Numbes of attributes / Fields present in the relation is called degree of relation.

↳ If these are fous attributes in the relation,

then degree of this relation is four.

Example:

Relational Data model for customes invoicing system.



Hiesarchical Database Model.

↳ A hiesarchical model can be represented as a tree graph, with records appearing as nodes (also called segments) and sets as edges.

↳ The data is stored as records which are connected to one another through links.

↳ A record is a collection of fields, with each field containing only one value.

↳ A hiesaschical database opesates on the psinciple that a record will contain gsoup of similas objects. These gsoups ase onganized into cascading hiesaschy.

↳ The entity type of a record defines which fields the record contains.

↳ The hiesaschical model differs from the network model in that the records are organized as collections of trees rathes than as asbitrary gsaphs.

Field: "smallest unit of data"

segment: "group of fields; nodes of the tree structure".

Data base: "composed of database records"

Data base description: "how data base records ase defined; set of assembly language macso instructions"

Root: "first segment"

sequence field: "one field in each segment used to ordes the occusances of a given type"

Tree-Structuse Diagrams.

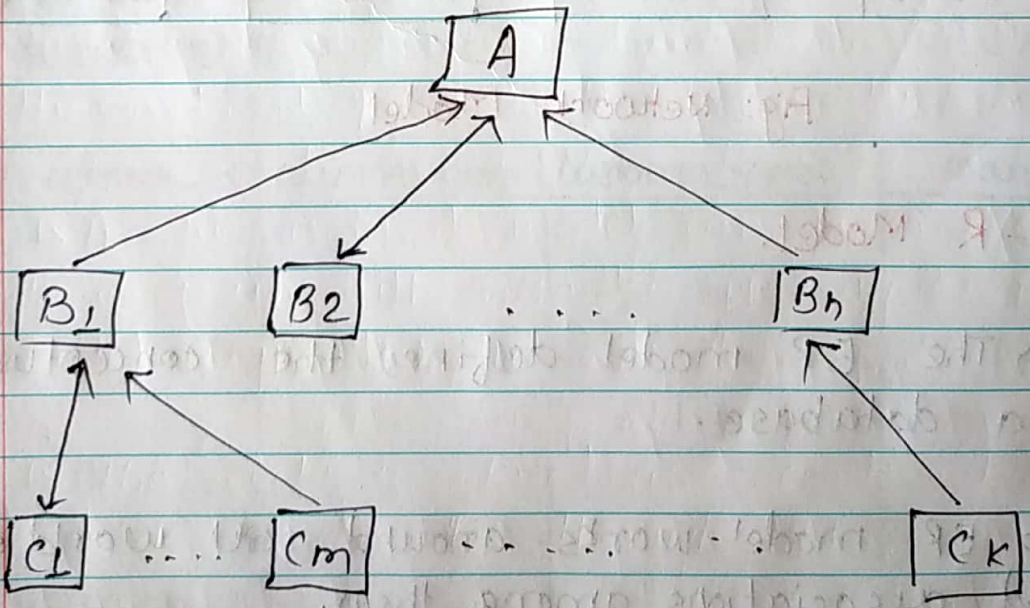
The schema for a hiesarchical database consists of

boxes: which correspond to record types.
lines: which correspond to links

Record types are organized in the form of a rooted tree.

↳ No cycles in the underlying graph.
↳ Relationship formed in the graph must be such that only one-to-many or one-to-one relationships exist between a parent & a child

A parent may have an assow pointing to a child, but a child must have an arrow pointing to its parent.



Network model.

↳ It allowed a more natural modeling of relationships between entities.

↳ These are no levels and a record can have numbes of owners and also can have ownership of sevesal records.

↳ There is no definite path for retrieval of data, the numbes of links is very large and thus network database are compex, slow & difficult to implement.

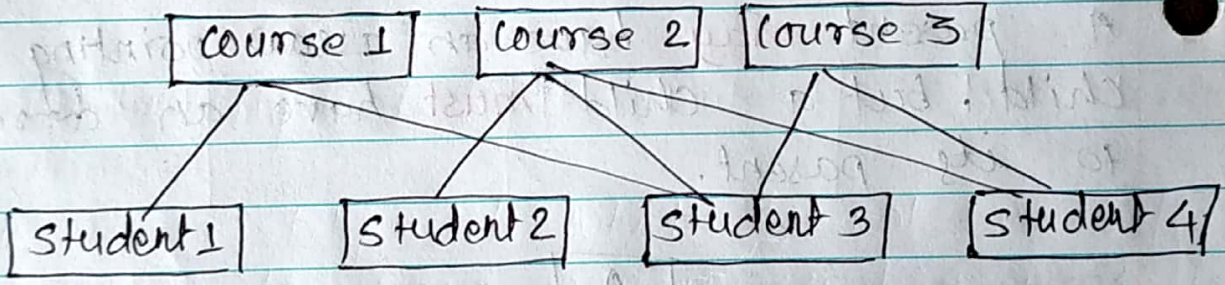


Fig: Network Model

2.2 E-R Model.

↳ The ER model defines the conceptual view of a database.

↳ ER model works around real world entities and associations among them.

↳ At view level, the ER model is considesed

a good option for designing databases.

2.3 Entity:

↳ An entity can be a real world object, that can be easily identifiable.

Example: specific person, company, car or product.

↳ All these entities have some attributes or properties that give them their identity.

↳ An entity set is a collection of similar types of entities.

For example: a students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties.

customes_id	customes_name	customes_street	loan_number	amount

Fig: Entity sets customes & loan.

2.5 Strong Entity set:

↳ An entity set that has a primary key is called as strong entity set.

* ↳ Considers an entity set payment which has three attributes payment-numbes, payment-dade and payment-amount. Although each payment entity is distinct but payment for different loans may shase the same payment numbes.

↳ A membes of a strong entity set is called dominant entity.

2.5 Weak Entity Set:

↳ The entity set which doesnot have sufficient attributes to form a primary key is called as weak entity set.

↳ *

↳ A membes of weak entity set is called subordinate entity.

↳ A weak entity set doesnot have a primary key but we need a means of distinguishing among all those entries in the entity set that depend on one pasticulas strong entity set.

Type of attributes.

Simple attribute:

Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.

Composite attribute:

Composite attributes are made of more than one simple attribute.

For example: a student's complete name may have first_name & last_name

Derived attribute:

Derived attributes are the attributes that donot exist in the physical database, but their values are derived from other attributes present in the database.

For example: average_salary in a department shouldnot be saved directly in the database, instead it can be derived.

Age can be derived from date_of_birth.

Single valued attribute.

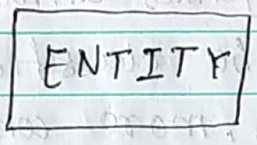
Single value attributes contain single value.

For example: social_security_Numbers.

↳ The discriminator of a weak entity set is a set of attributes that allows this distinction be made.

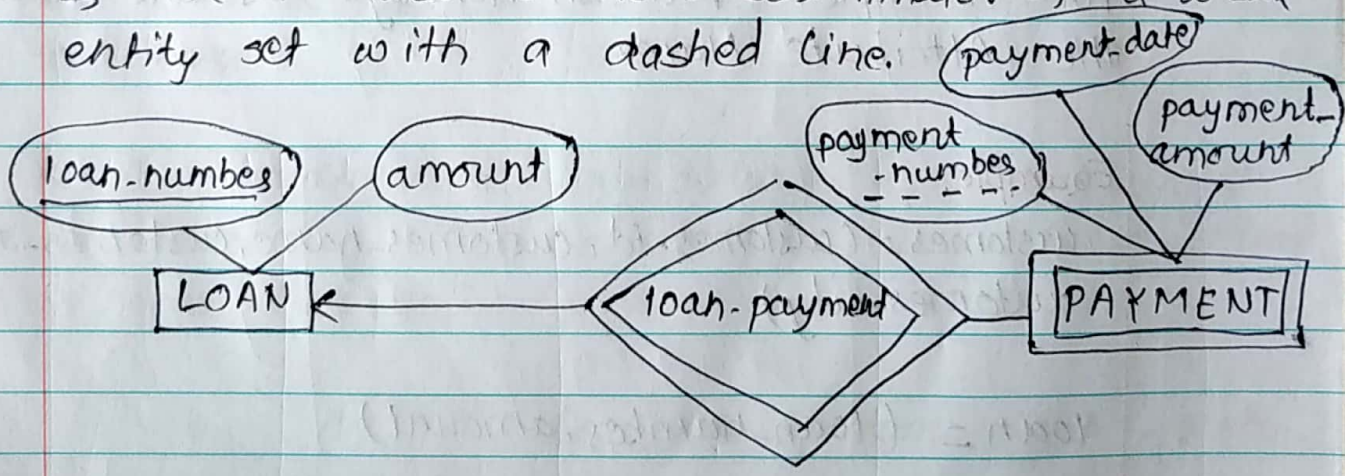
For example: payment_number acts as discriminator for payment entity set. It is also called as the partial key of the entity set.

↳ The entity name is usually written in capital letters and represented by a rectangle that contains the entity's name.



↳ We represent weak entity set by double rectangles

↳ We underline the discriminator of a weak entity set with a dashed line.



payment_numbes - discriminator of payment entity set

Primary key for payment - (loannumbes, payment_numbes)

2.6 Attribute:

↳ Descriptive properties possessed by all members of an entity set.

↳ An entity is represented by a set of attributes.

↳ The domain of an attribute is the collection of all possible values an attribute can have. The domain of Name is a character string.
The domain for a gender attribute consists of only M or F.

↳ Attributes can be classified as identifiers or descriptors. Identifiers, more commonly called keys, uniquely identify an instance of an entity. A descriptor describes a non unique characteristics of an entity instance.

For example: "Ramesh Acharya" is one value of the Attribute Name.

Example:

customes = (customes-id, customes_name, customes_street, customes-city)

loan = (loan_number, amount)

Type of attributes.

Simple attribute:

Simple attributes are atomic values, which cannot be divided fusthes. For example, a student's phone numbes is an atomic value of 10 digits.

Composite attribute:

Composite attributes are made of more than one simple attribute.

For example: a student's complete name may have first_name & last_name

Desived attribute:

Desived attributes are the attributes that donot exist in the physical database, but theis values are desived from othes attributes present in the database.

For example: average_salary in a depastment shouldnot be saved disectly in the database, instead it can be desived.

Age can be desived from date_of_bisth.

Single valued attribute.

Single value attoibutes contain single value.
For example: social_security_Numbes.

Multi-valued attribute.

Multi-value attribute may contain more than one values. For example, a person can have more than one phone numbes, email-address

2.6 Keys

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example: the roll_numbes of a student makes him/hes identifiable among students.

Types of keys:

Supes key:

A set of attributes (one or more) that collectively identifies an entity in an entity set.

Candidate key:

A minimal supes key is called a candidate key. An entity set may have more than one candidate key.

Example: Employee numbes or the social Security numbes.

Primary key:

A psimary key is one of the candidate keys

chosen by the database designer to uniquely identify the entity set.

Foreign key:

↳ A foreign key is a field in one table that must match a primary key value in another table in order to establish the relationship between the two tables.

↳ A foreign key must either match a primary key or else be null.

Secondary key.

↳ a field or combination of fields that can be used to access or retrieve records. example, if you need to access records for only those customers in a specific ZIP code, you would use the ZIP code field as secondary key.

↳ secondary key is refered to as an alternate key.

↳ Used to sort or display records in certain order.

Example:

Student Table.

Foreign key

Primary keys

STUDENT_NUMBER	STUDENT_NAME	TOTAL_CREDITS	ADVISOR_NUMBER	CGPA
5097	Pradip	14	49	3.2
3395	Anmol	9	49	3.6
4018	Rajesh	9	23	2.3

Advisor Table

secondary key

ADVISOR_NUMBER	SOCIAL_SECURITY_NUMBER	ADVISOR_NAME
23	504 91233	Ramesh
49	159 79804	Bhuvan

Course Table

Candidate key

Primary key

COURSE_ID	COURSE_DESCRIPTION	NUMBER_OF_CREDITS
CHM 112	General Chemistry	5
CSE 151	Computer Science-I	3
ENG 161	English Composition	3
MKT 212	Marketing Management	3

Grade Table

Combination primary key

STUDENT_NUMBER	COURSE_ID	GRADE
5097	CSE 151	B
3395 5097	MKT 212	A
[illegible] 5097	ENG 161	B

Relationship & Relationship Sets.

↳ A relationship is an association among several entities.

↳ A relationship set is a set of relationships of the same type.

It is a mathematical relation on $n \geq 2$ entities each taken from entity sets.

R is subset of

$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$

where $(e_1, e_2, e_3, \ldots e_n)$ is a relationship.

Relationship can be binary, ternary and n-array

Example:

Ramesh depositor A-105

customes entity relationship set account entity.

(Ramesh, A-102) $\in$ deposites

1	Ramesh	20
2	Bipin	18
3	Sudasshan	19
4	Amrit	21

Student entity set

B_1	Jivan	Physics
B_2	Madhav	Biology
B_3	Narayan	Biology
B_4	Amas	Chem
B_5	Hemraj	Math
B_6	Sushil	Comp

Book entity set.

Student (roll, name, age)

Book (bkno, author, subject)

issue(roll, bkno)

roll	bkno
1	B3
2	B6
3	B1
4	B5

Set of attributes in Relationship R will be

primary-key(E_1) ∪ primary-key(E_2) ∪ ... ∪ primary-key(E_n)

If the relationship has descsiptive attributes associated with it and if these attributes are $\{a_1, a_2, \ldots, a_n\}$ then the relationship will have the following set of attributes.

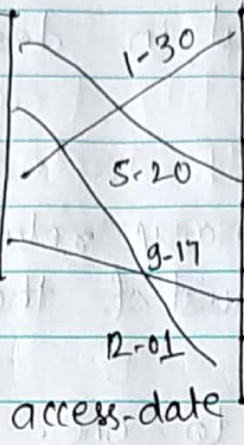
primary-key(E_1) ∪ primary-key(E_2) ∪ ... primary-key(E_n) ∪ $\{a_1, a_2, a_3, \ldots, a_n\}$

issue (roll, bkno, access-date)

For example:

1	Ramesh	20
2	Bipin	18
3	sudarshan	19
4	Amrit	21

Student Entity Set



B1	Jivan	Physics
B2	Madhav	Biology
B3	Narayan	Biology
B4	Amar	Chem
B5	Hemraj	Math
B6	Sushil	Comp

Book Entity set

student (roll, name, age)

Book (bkno, author, subject)

Roll	Bkno	access-date
1	B3	5-20
2	B6	12-01
3	B1	1-30
4	B5	9-17

Kind of Constraints :

Cardinality Constraints
Participation Constraints
Integrity Constraints

Togethes called " structural constraints"

Cardinalites Constraints :

It limits the numbes of entity occurances that are associated in a relationship to the numbes of occurances in anothes

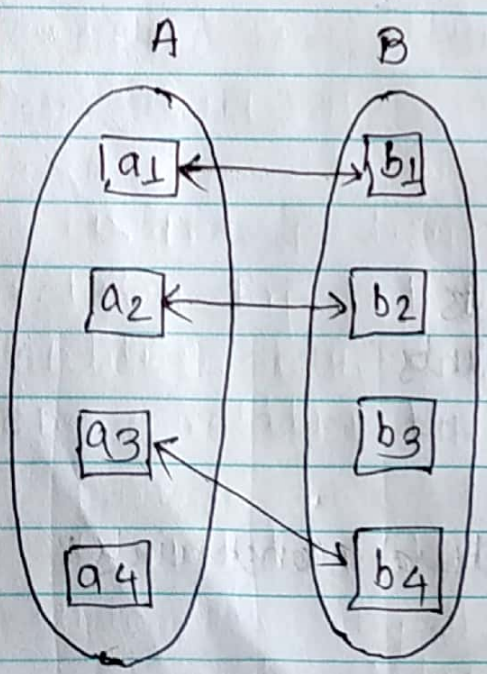
↳ Most useful in describing binary relationship sets.

↳ For a binary relationship set the mapping cardinality must of the following types.

i. One to one.
ii. One to many.
iii. Many to one.
iv. Many to Many.

One to one :

↳ One entity from entity set A can be associated ~~in~~ with at most one entity of set B & vice versa.

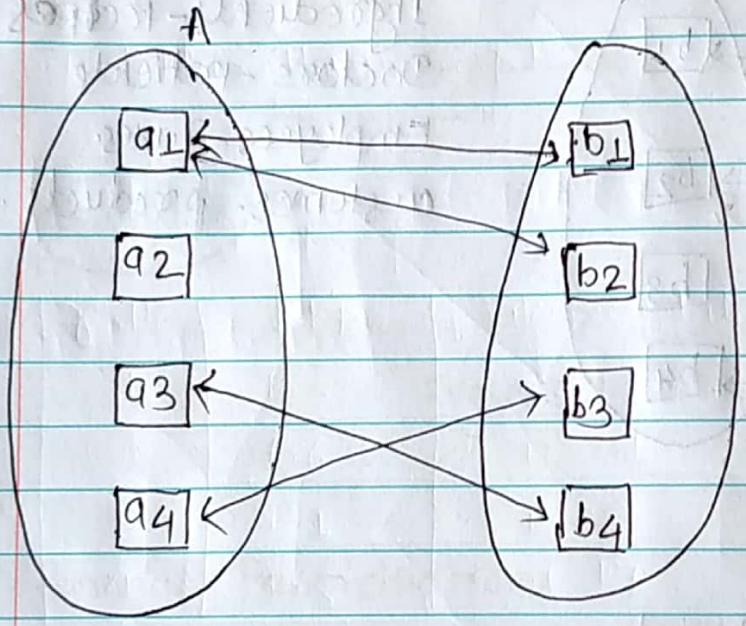


Example:
people - passports
country - Flag

↳ This type of relationship is rarely seen in real world.

One to Many.

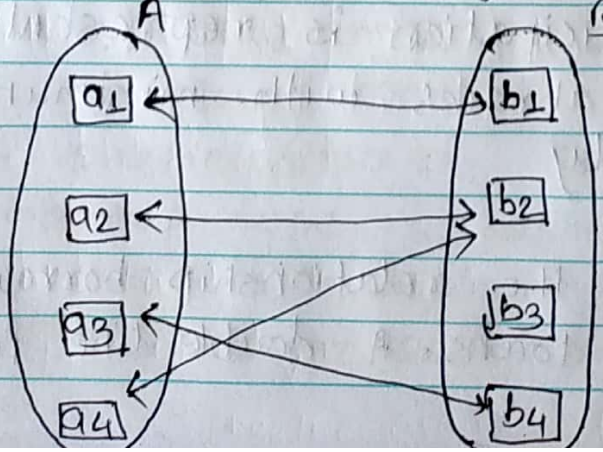
↳ One entity from set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.



Example:
people - address
Owners - pety
Farmes - equipmat

Many to One

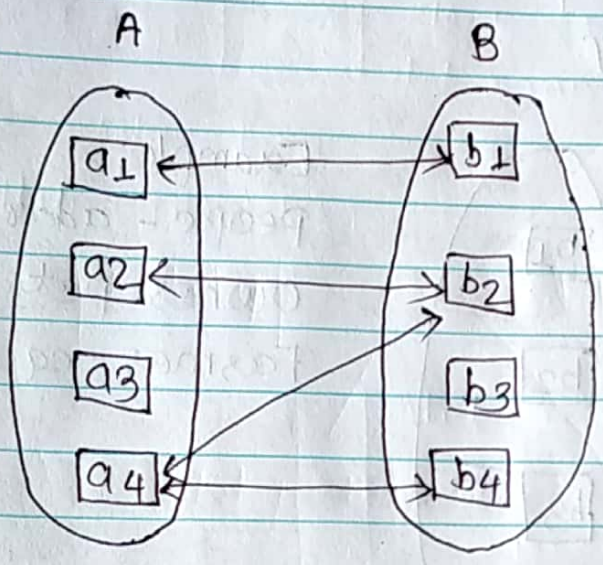
↳ More than one entities from set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.



Example:
employee - department

Many to Many

One entity from entity set A can be associated with more than one entity from entity set B & vice versa.



Example:

Ingredient - Receipe
Doctors - patients
Employees - Taxes
customes - products.

Pasticipation Constraints.

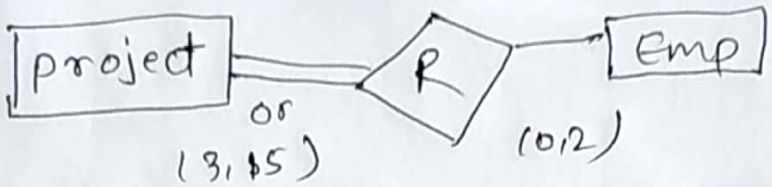
Total pasticipation

↳ The pasticipation of an entity set E in a relationship set R is said to be total if every entity in entity set E pasticipates in at least one relationship in R.

↳ Total pasticipation is represented by double lines or by braces with minimum & maximum cardinality

Example:

Consides the relationship borrowes between customers and loans. A double line from loan to



borrower, as shown in figure below indicates that each loan must have at least one associated customer.

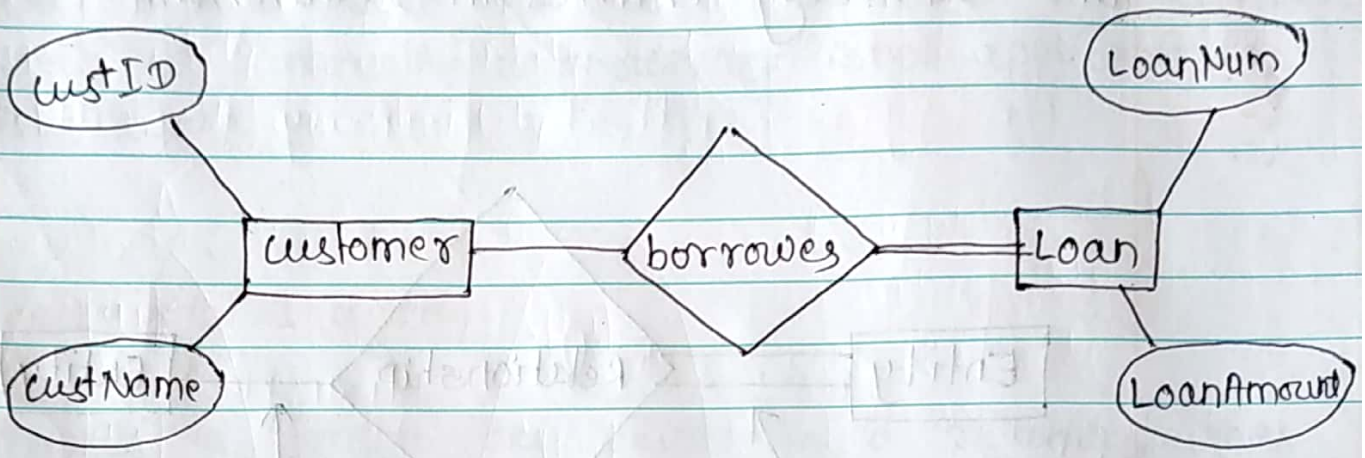


Figure : Total Pasticipation

Partial Pasticipation.

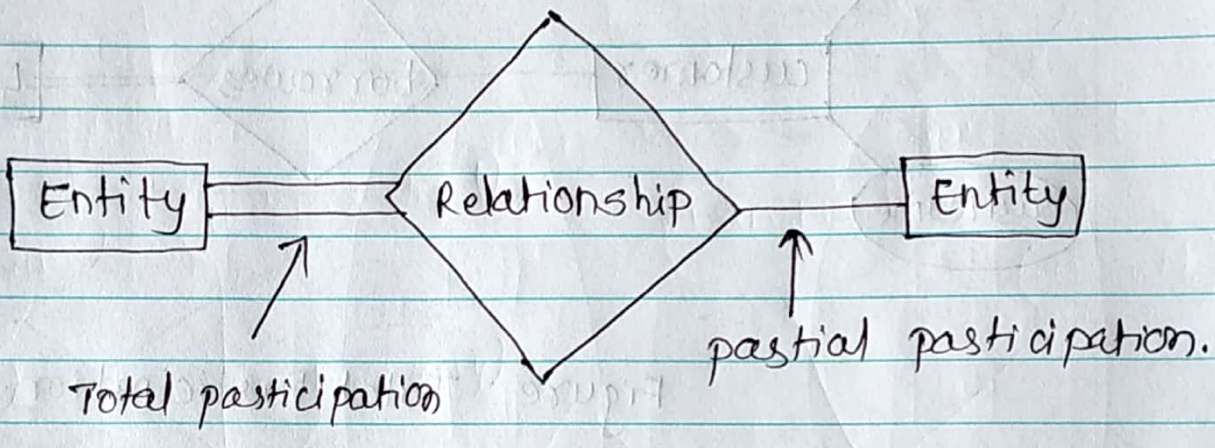
↳ If only some entities in entity set E pasticipate in relationships in R, the pasticipation of entity set E in relationship R is said to be pastial.

Example:

If a company policy states that employee (manages) must manage a depastment, Howevcs every employee maynot manage a depastmcnt, So the pasticipation of EMPLOYEE in the MANAGES relationship type is pastial, meaning that some or past of the employee entities are related to some depastment entity via MANAGES, but not

necessarily all.

↳ Pastial pasticipation is represented by single line connecting entities in relationship.



Integrity Constraint.

↳ Integrity constraints are used to ensure accuracy and consistency of data in a relational database.

↳ A constraint (rule) that must remains true for a database to presesve integrity.

↳ Integrity constraints are specified at database creation time and enforced by DBMS.

Entity Integrity

↳ In base relation, no attributes of a primary key can be null. Null represents a value for an attribute that is currently unknown or isnot applicable for this tuple.

Referential Integrity.

↳ If a foreign key exists in a relation, either the foreign key value must match a candidate key value of some tuple in its home relation or the foreign key value must be wholly null.

E-R diagrams.

An entity relationship diagram (ERD) shows the relationship of entity sets stored in a database.

Rectangles represent entity sets.

Diamonds represent relationship sets.

Lines link attributes to entity sets and entity sets to relationship sets.

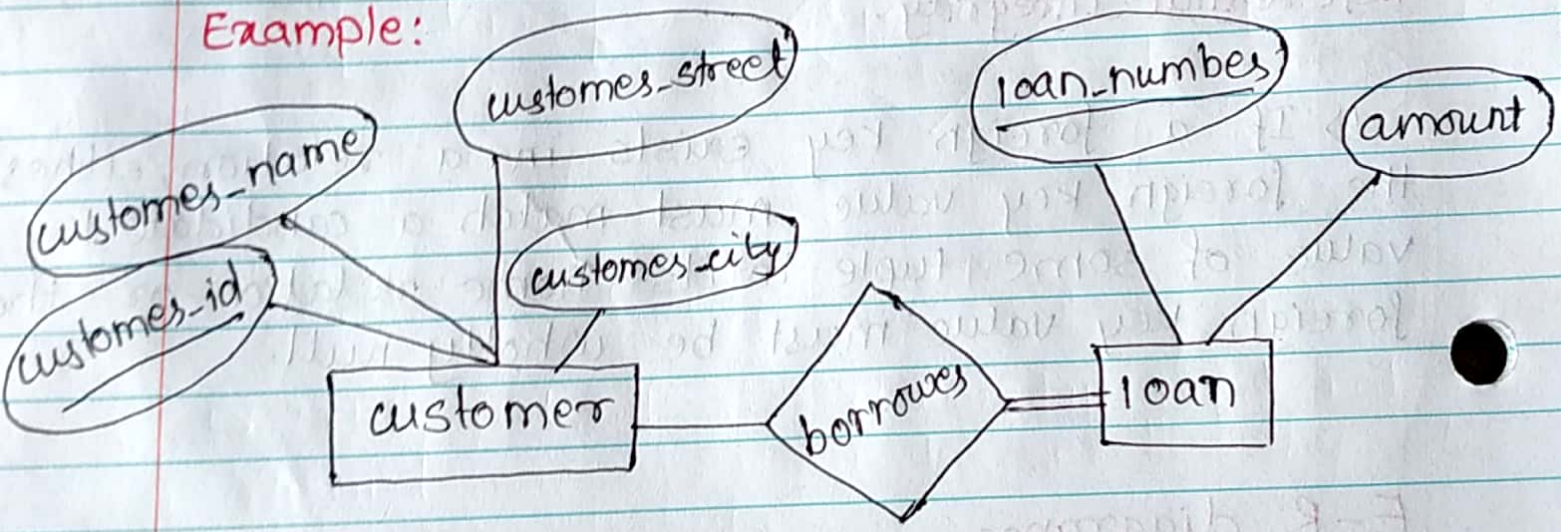
Ellipses represent attributes

Double ellipses represent multivalued attributes

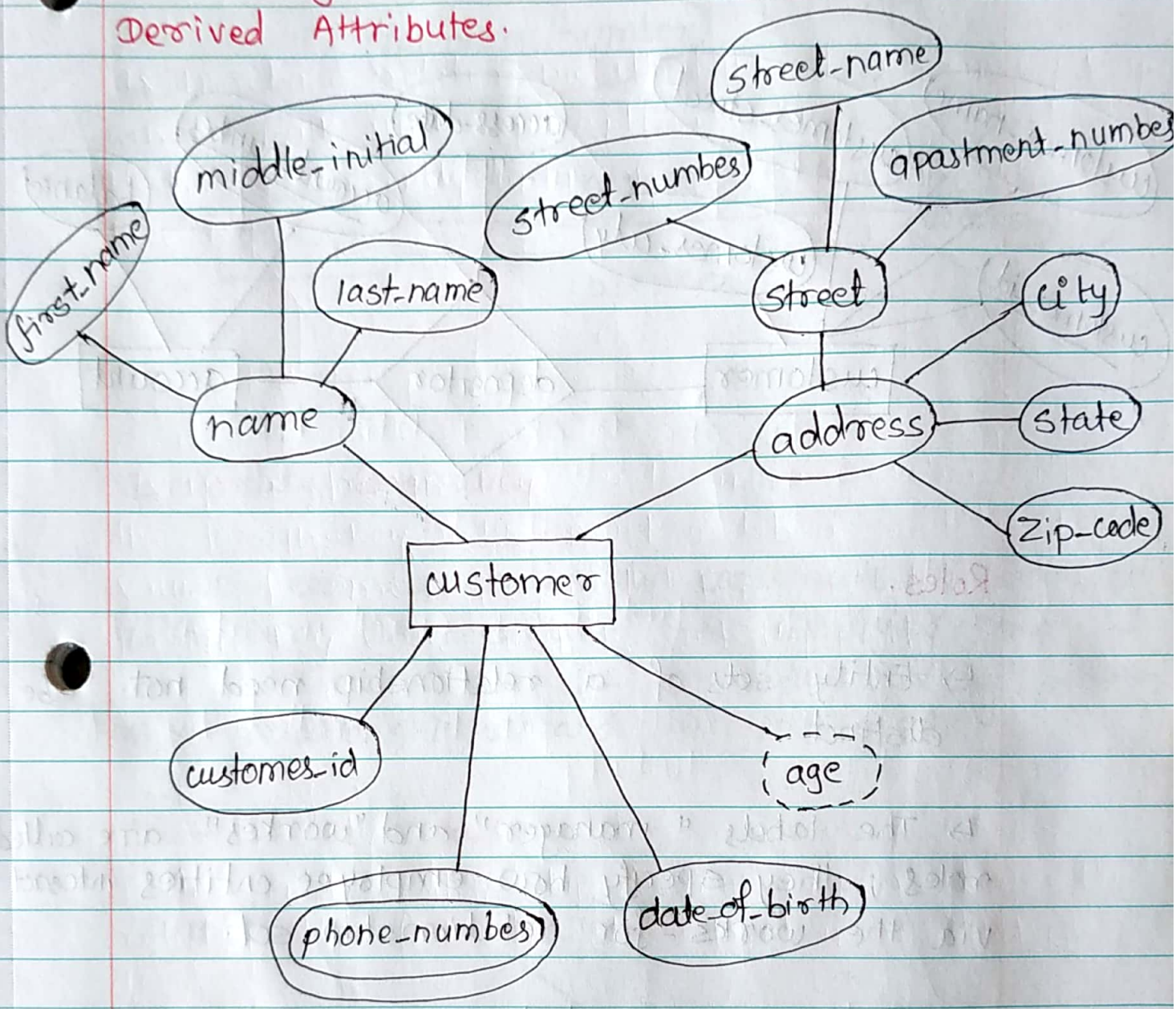
Dashed ellipses denote desived attributes.

Underline indicates primary key attributes

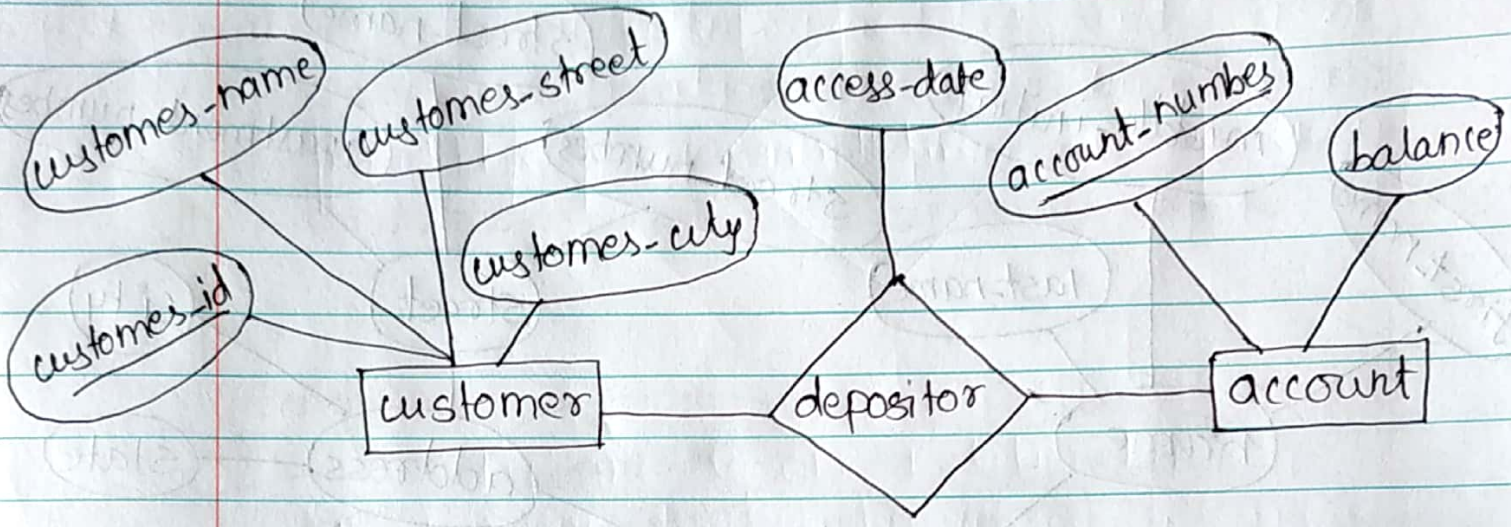
Example:



E-R Diagram With Composite, Multivalued, and Derived Attributes.



Relationship sets with Attributes.



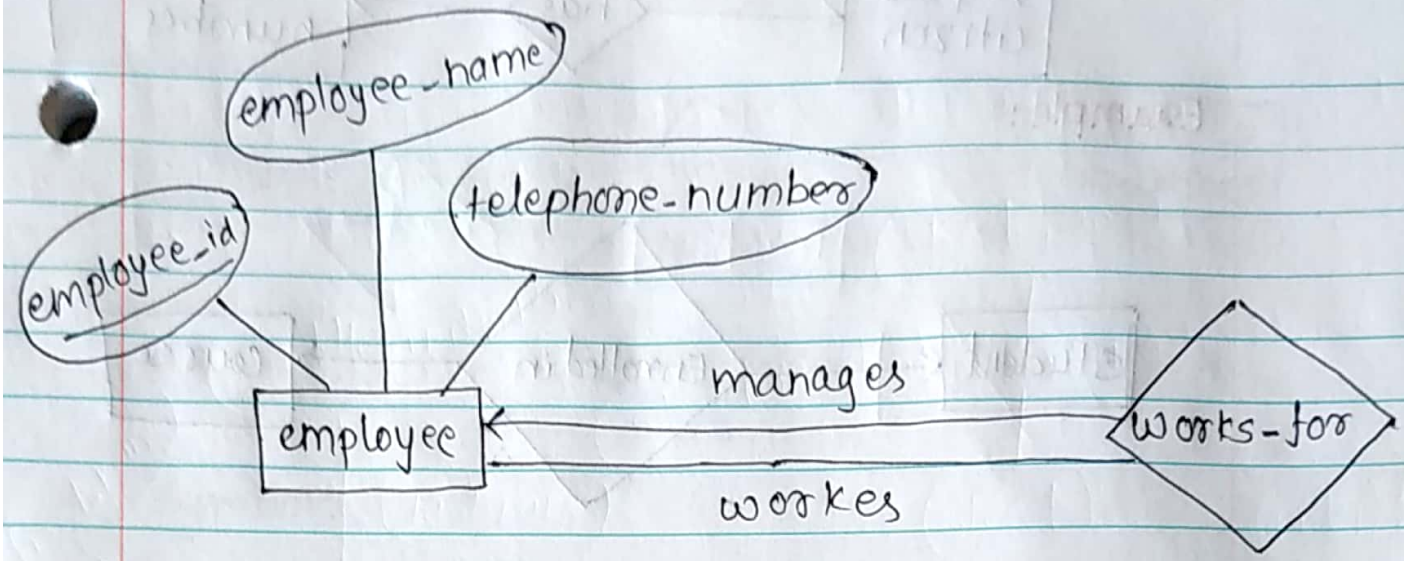
Roles.

↳ Entity sets of a relationship need not be distinct

↳ The labels "manager" and "workes" are called roles; they specify how employee entities intesact via the works-for relationship set.

↳ Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.

↳ Role labels are optional, and are used to clarify semantics of the relationship.

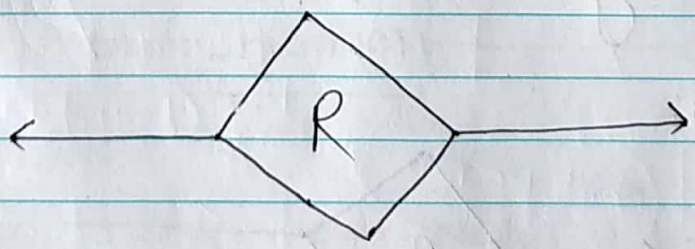


Cardinality Constraints :

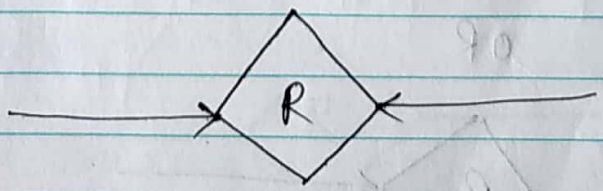
We express cardinality constraints by drawing either by directed line (→), signifying "one" or an undirected line (—), signifying "many", between the relationship set and the entity set.

One-to-one relationship.

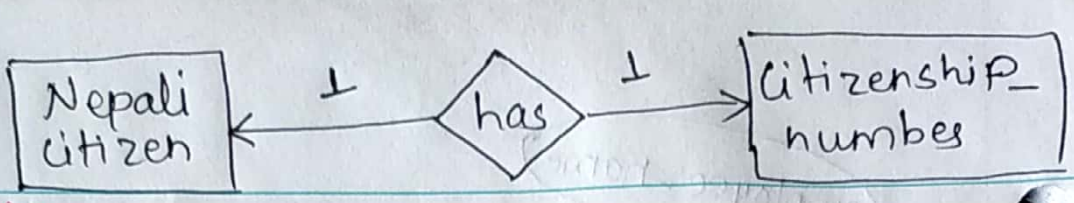
symbols:



OR



cardinality Ratio = 1:1



Eaample:

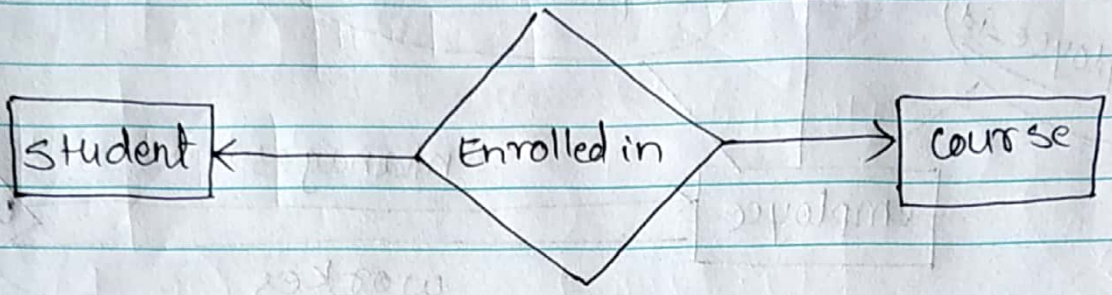


Fig: one to one Relationship.

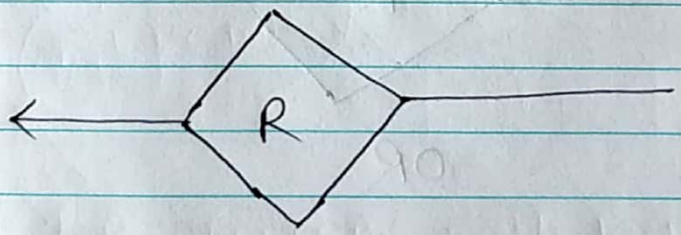
Here,

↳ one student can enroll in at most one course

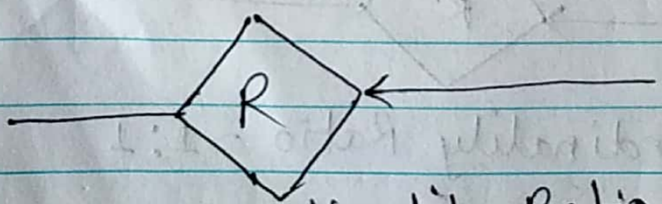
↳ one course can be enrolled by at most one student.

One to Many Relationship.

Symbols:



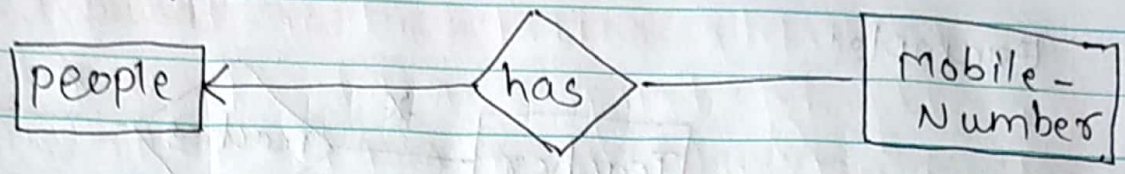
OR



Cardinality Ratio = 1 : n

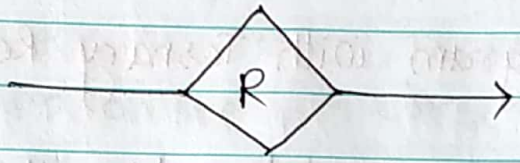
many → *

Example:

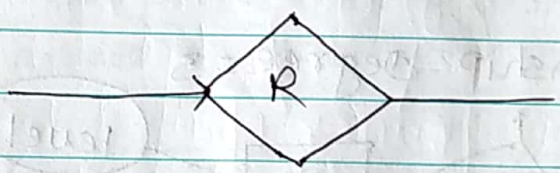


Many to one Relationship

Symbols:

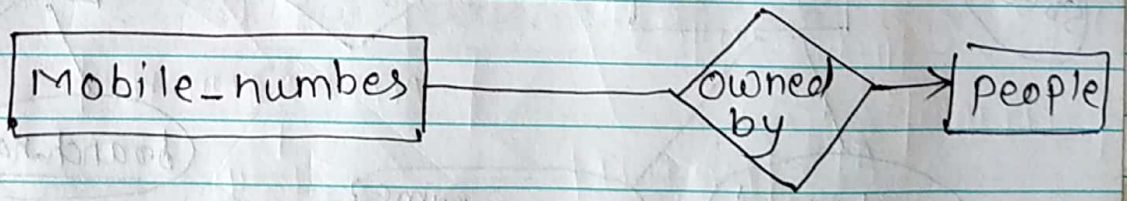


OR



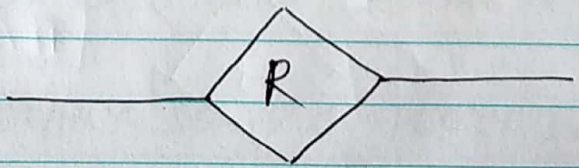
Cardinality Ratio = m:1

Example:



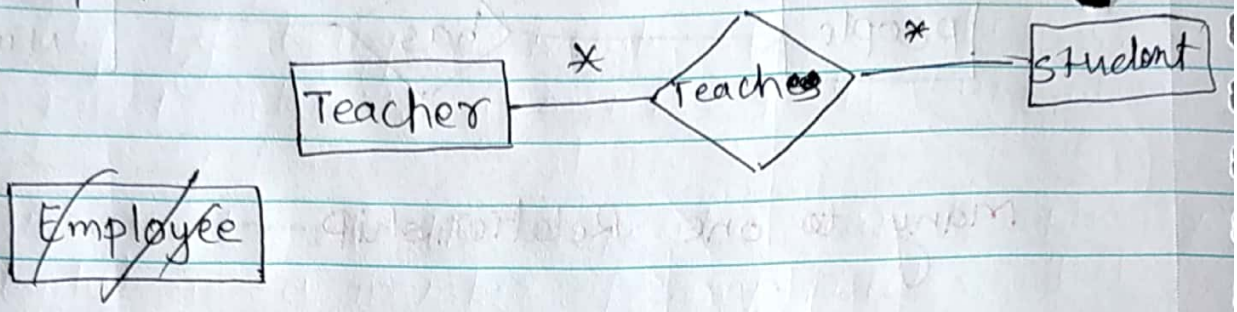
Many to many Relationship

Symbols



Cardinality Ration = m:n

Example:



Employee

E-R Diagram with Tesnary Relationship.

In Tesnary relationship three different Entities taken part in a Relationship.

Relationship Degree = 3

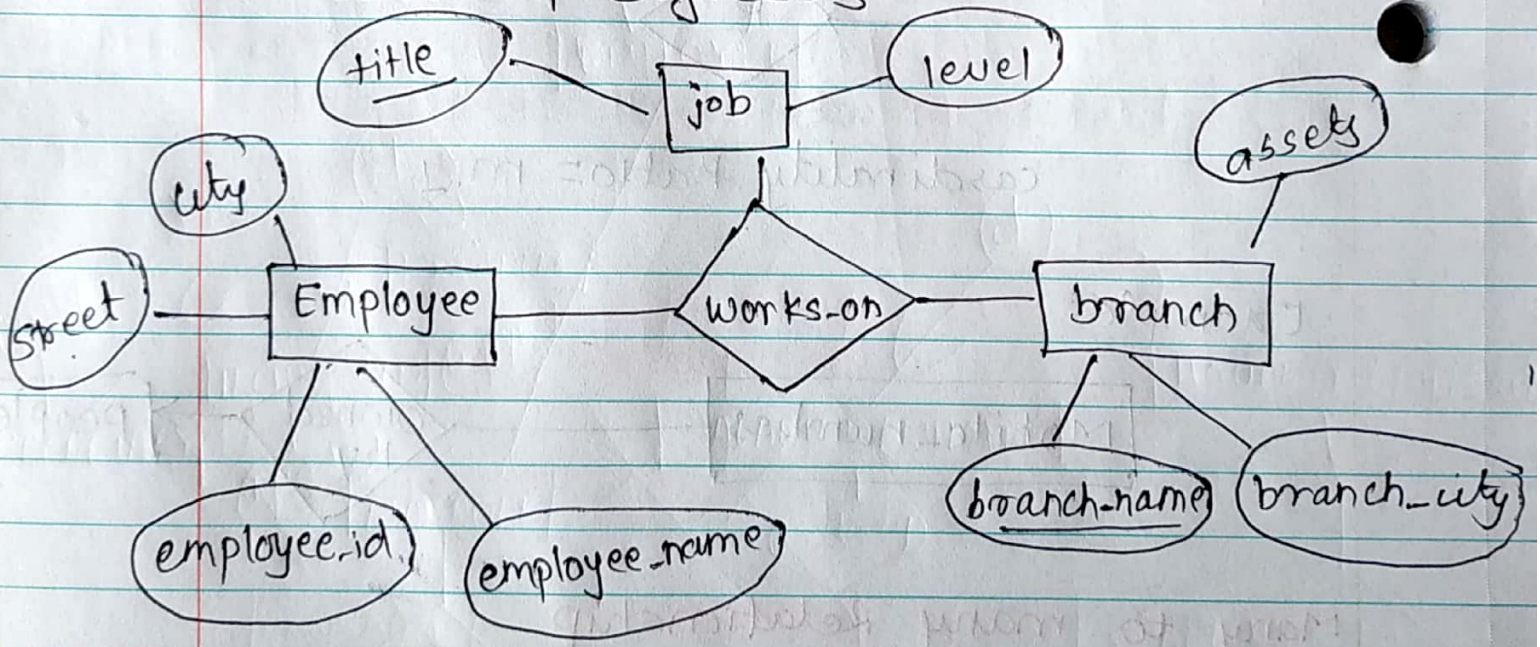


Fig: ER Diagram with Tesnary Relationship.

Subclasses and Superclasses.

In many cases an entity type has numerous subgroupings of its entities that are meaningful and need to be represented explicitly because of their significance to the database application.

Example: EMPLOYEE may be further grouped into:

SECRETARY, ENGINEER, TECHNICIAN, ...

Based on the EMPLOYEE's job.

MANAGER

EMPLOYEE who are managers.

SALARIED_EMPLOYEE, HOURLY_EMPLOYEE

Based on the EMPLOYEE's method of pay

We call each of these subgroupings a subclass of the EMPLOYEE entity type, and the EMPLOYEE entity type is called the superclass for each of these subclasses.

These are called superclass/subclass (as well as simply class/subclass):

EMPLOYEE / SECRETARY

EMPLOYEE / TECHNICIAN

EMPLOYEE / MANAGER

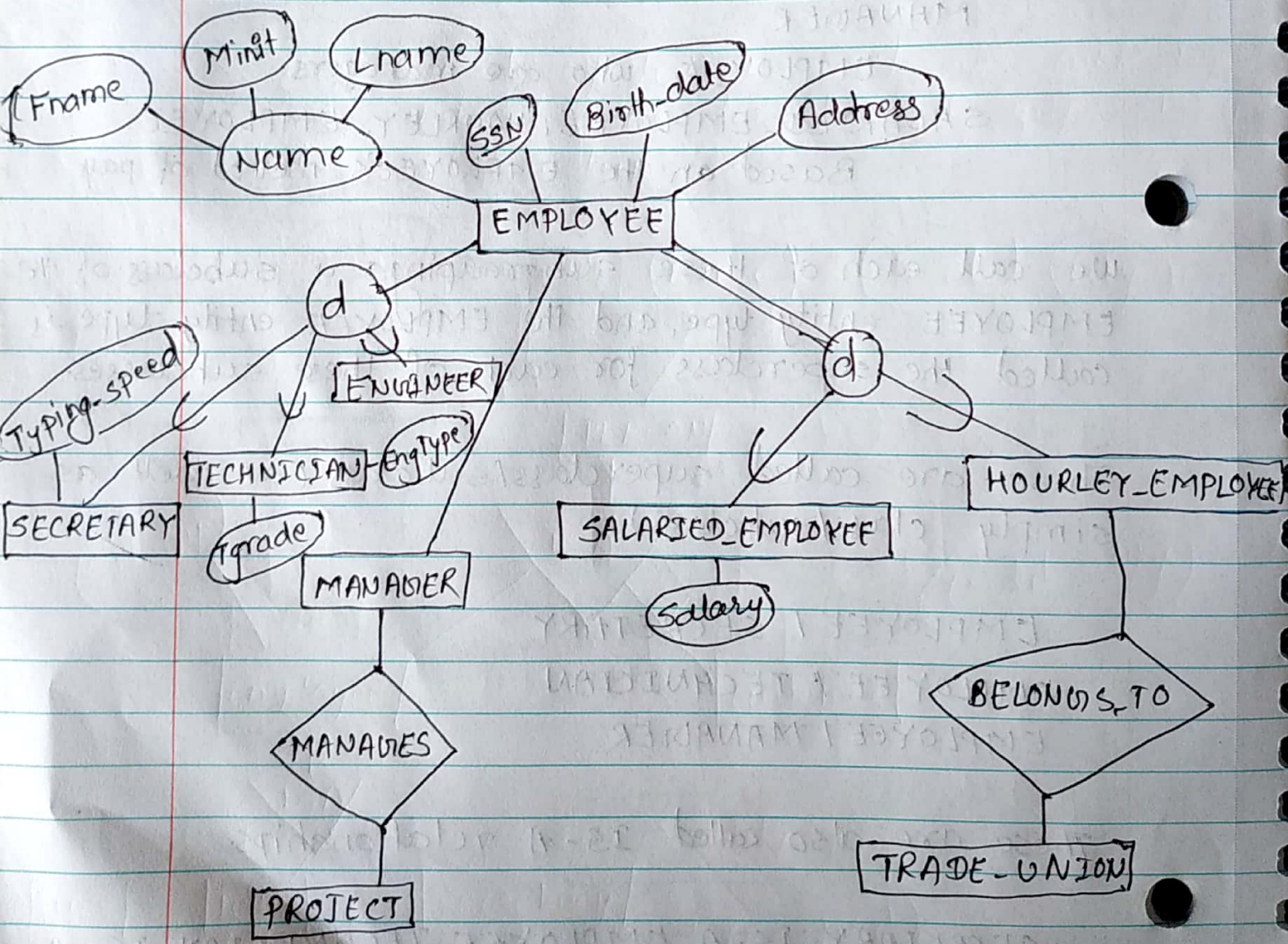
These are also called IS-A relationships.

SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, ...

↳ An Entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass

↳ A member of the superclass can be optionally included as a member of any number of its subclasses.

Subclasses and Superclasses Example.



EER diagram to represent subclasses & specialization.

Three specializations of EMPLOYEE:

{ SECRETARY, TECHNICIAN, ENGINEER }
{ MANAGER }
{ HOURLY_EMPLOYEE, SALARIED_EMPLOYEE }

Specialization.

↳ Specialization is the process of defining a set of subclasses of an entity type.

↳ This is a Top-down design process in which we designate subgroupings within an entity set that are distinctive from other entities in the set.

↳ These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to higher level entity set.

↳ The set of subclasses is based upon ~~the~~ some distinguishing characteristics of the entities in the superclass.

Example: { SECRETARY, ENGINEER, TECHNICIAN } is a specialization of EMPLOYEE based upon job type.

Depicted by a triangle - a lower level entity set inherits all the attributes and relationship

pasticipation of the highes level entity set to which it is linked.

↳ It may have seveal specializations of the same supesclass.

Example: Anothes specialization of EMPLOYEE based on method of pay is

{SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}

↳ The subset symbol on each line connecting to ∈ indicates the direction of supesclass/subclass relationship.

Specialization example.

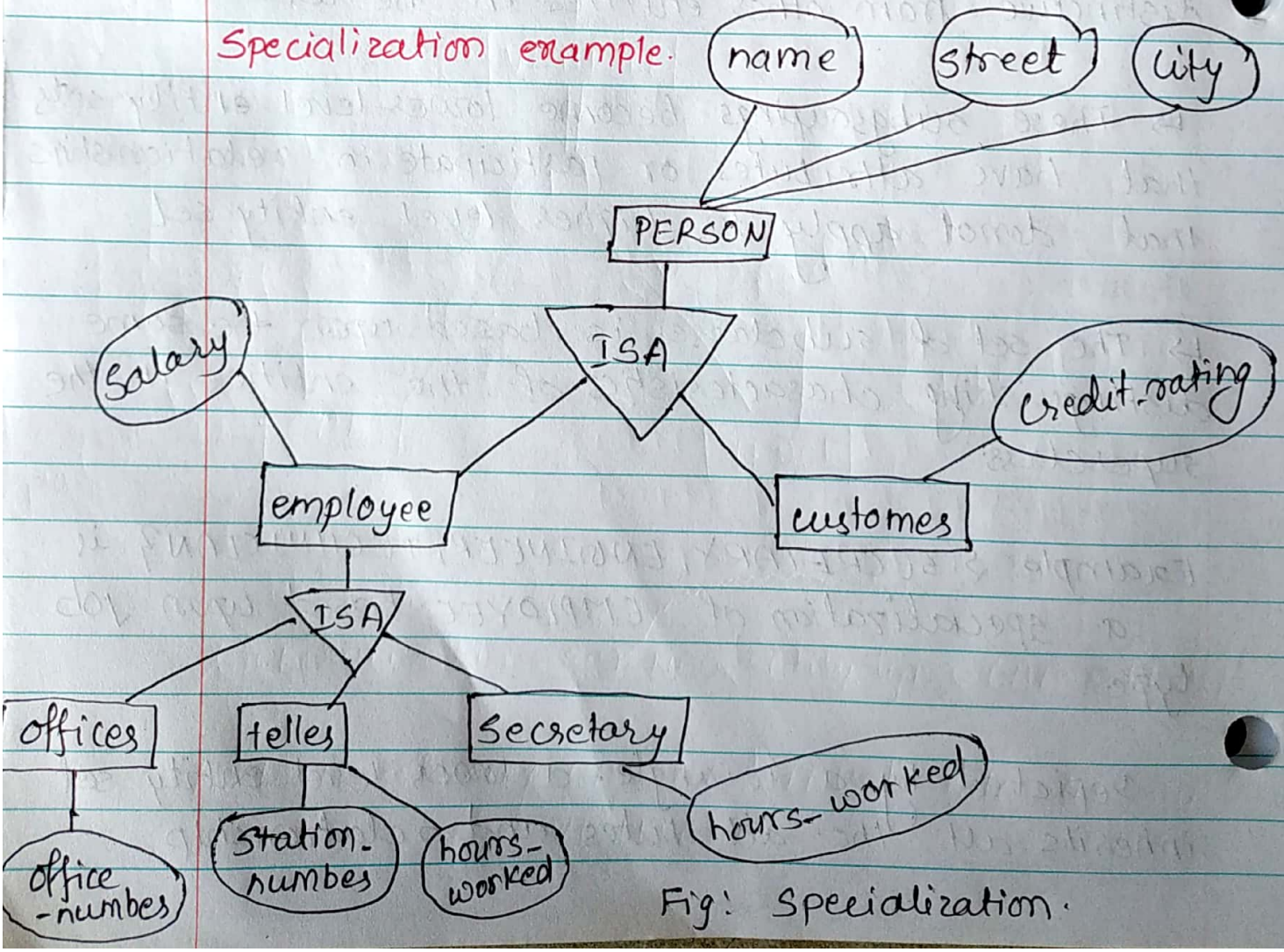


Fig: Specialization.

Generalization.

↳ A bottom up design process - combine a number of entity sets that share the same features into a higher-level entity set.

↳ Specialization and generalization are simple inversions of each other; they are represented in an ER diagram in the same way.

↳ The terms specialization and generalization are used interchangeably.

↳ Generalization is the reverse of the specialization process.

↳ several classes with common features are generalized into a superclass; original classes become its subclasses.

Example:

CAR, TRUCK generalized into VEHICLE.

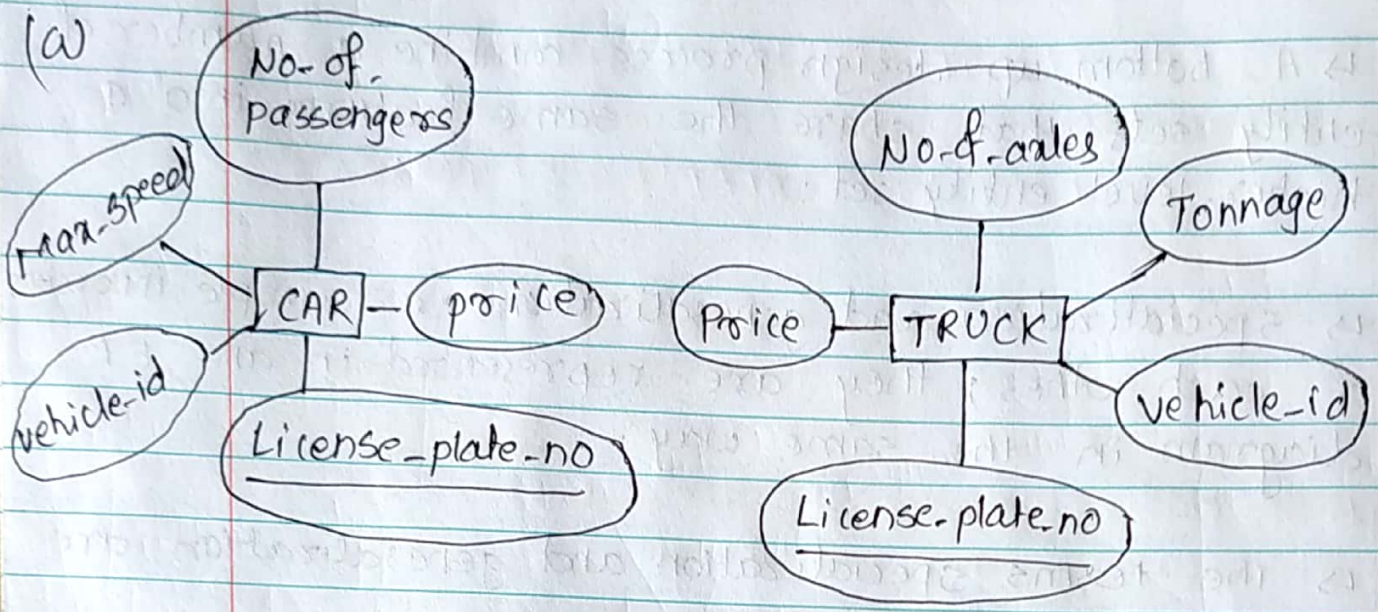
both CAR, TRUCK become subclasses of the superclass VEHICLE

We can view {CAR, TRUCK} as a specialization of VEHICLE.

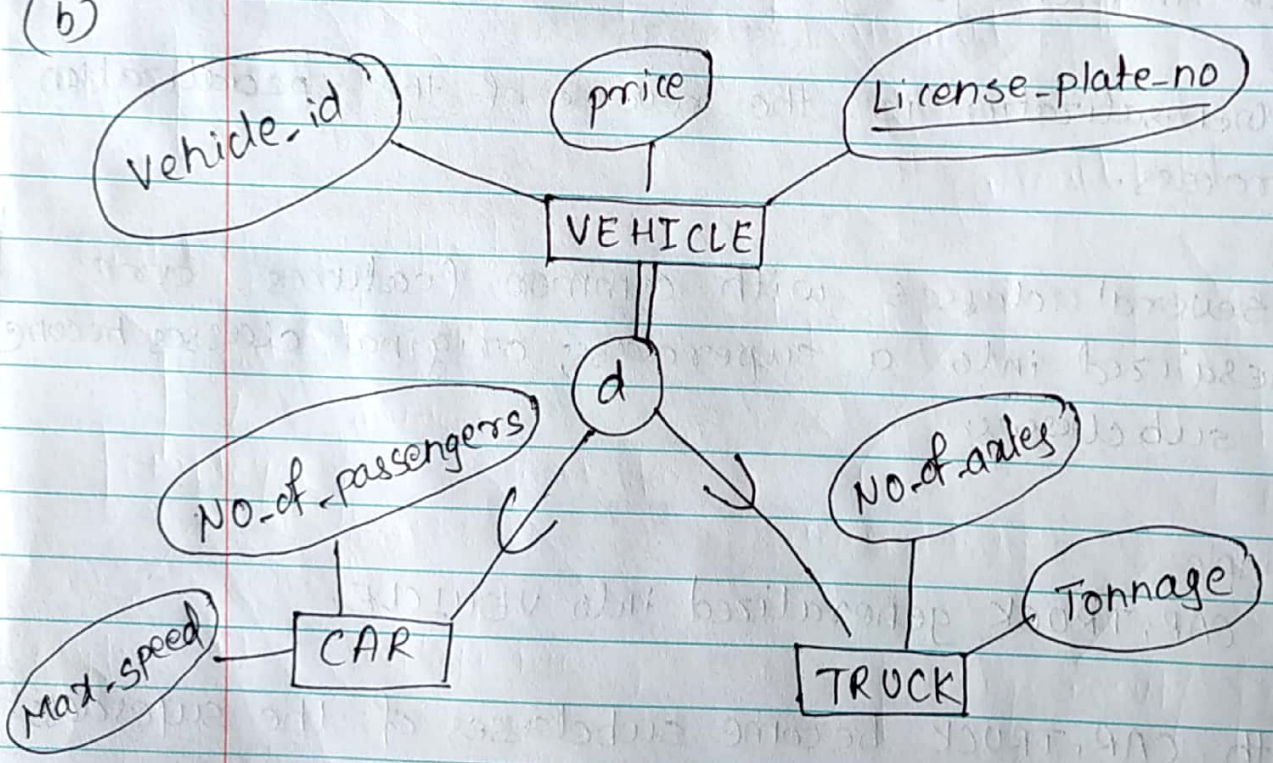
Alternatively, we can view VEHICLE as a generali-

zation of CAR & TRUCK.

(a)



(b)



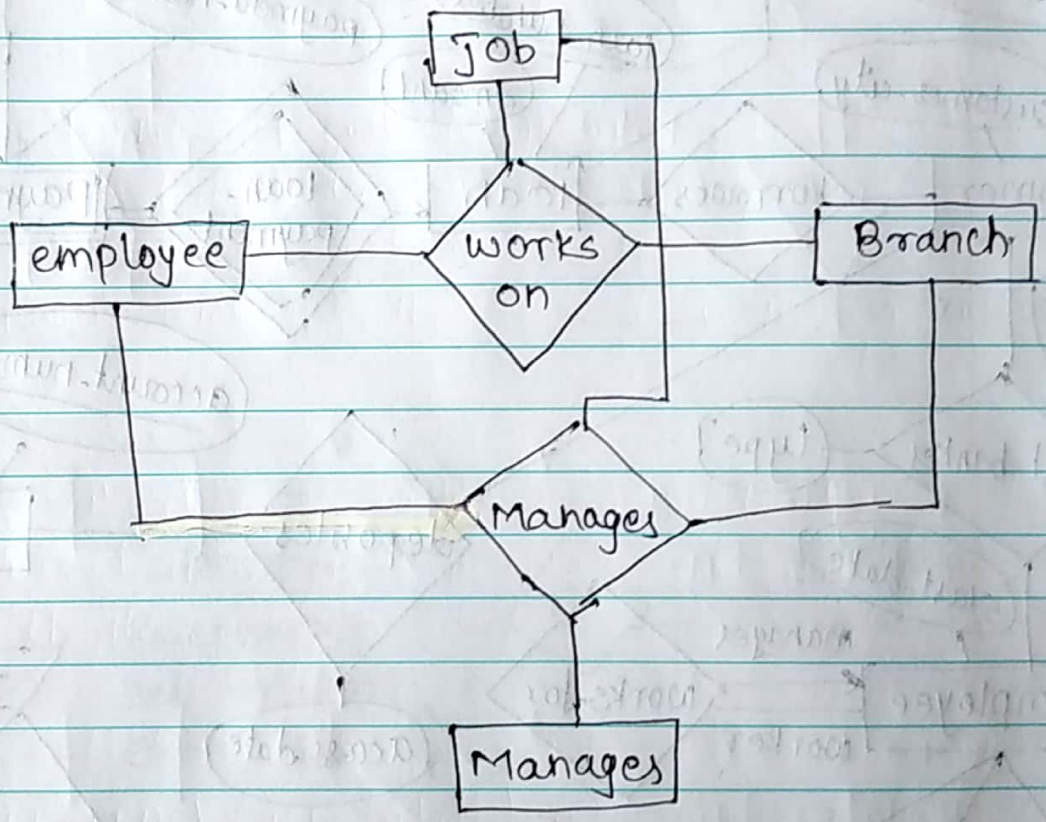
Generalization. (a) Two entity types, CAR & TRUCK.

b. Generalizing CAR and TRUCK into the superclass VEHICLE.

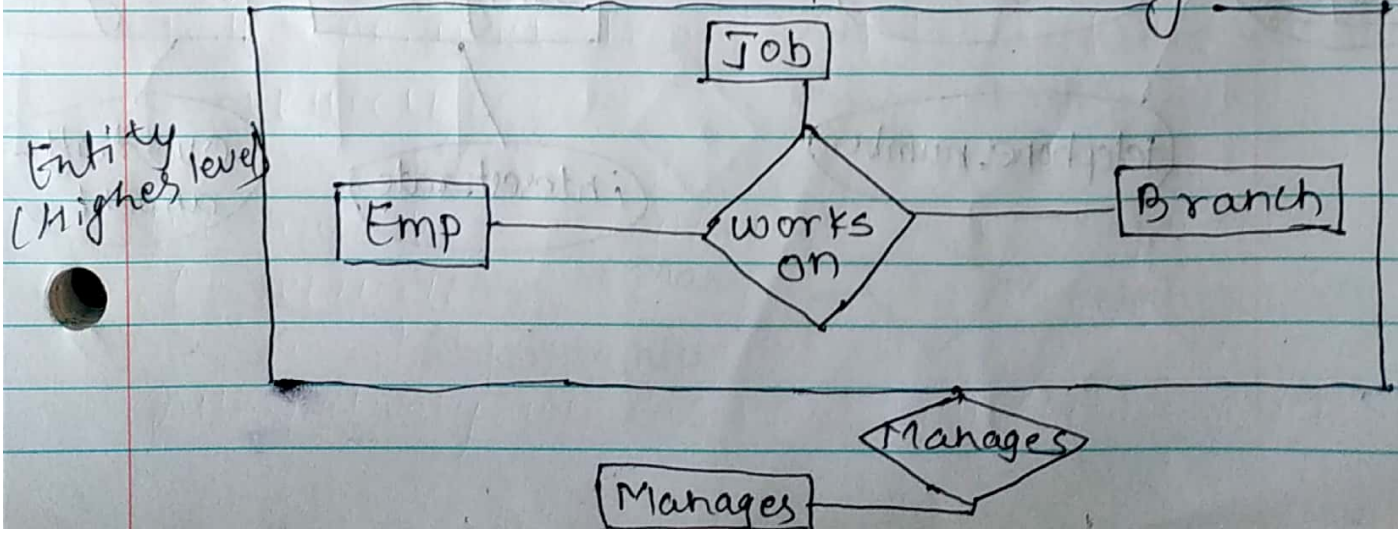
Aggregation :

↳ Aggregation is a abstraction through which we can represent relationships as highes level entities.

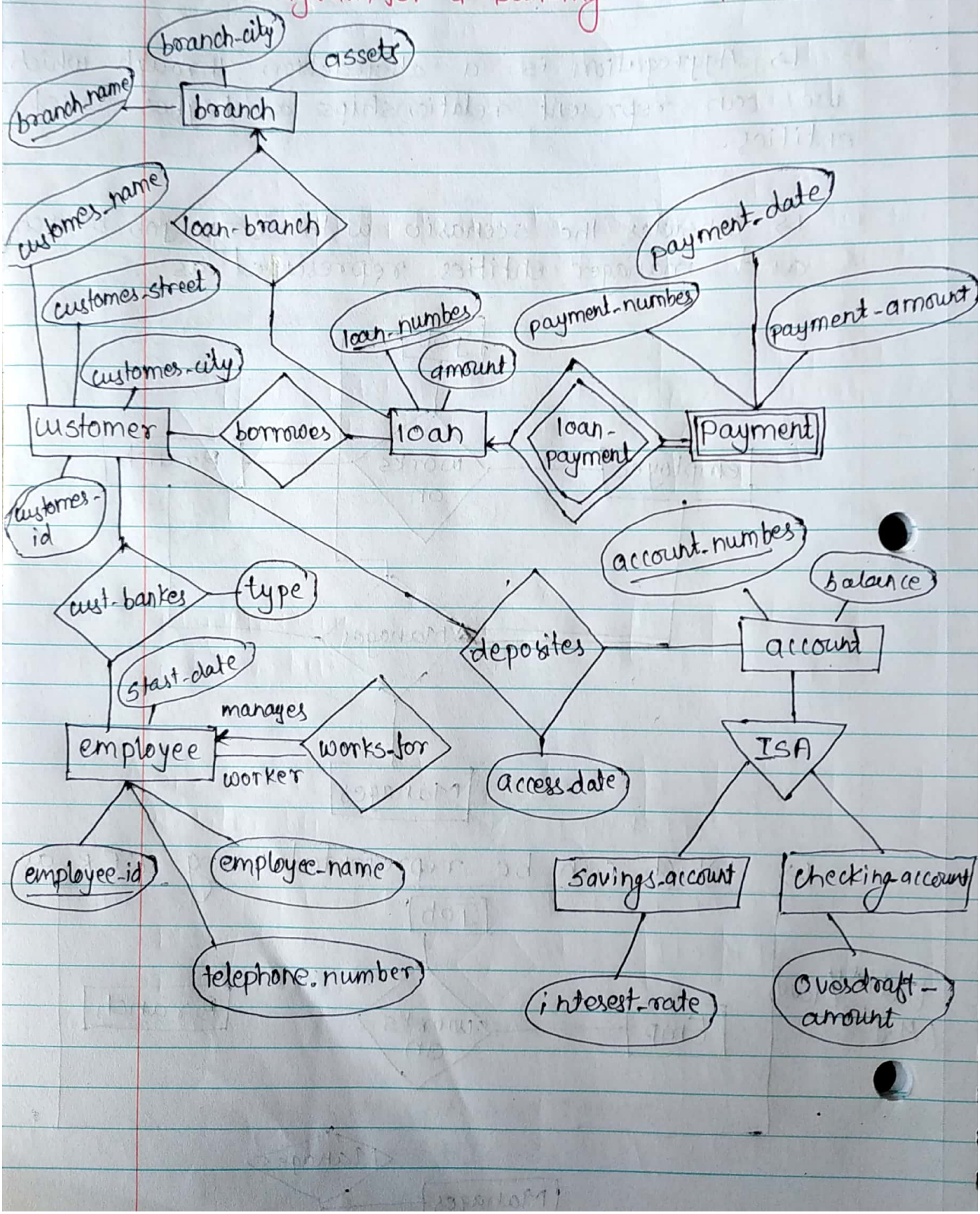
↳ Consides the scenasio with emp, job, branch and manager entities represented as



↳ Which can be represented using EER as

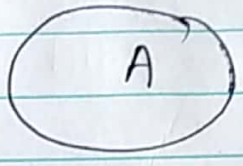


E-R Diagram for a Banking Enterprise.



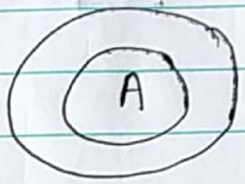
Summary of symbols Used in E-R Notation.

Entity set

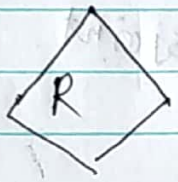


Attribute

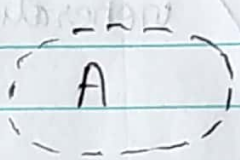
Weak Entity set



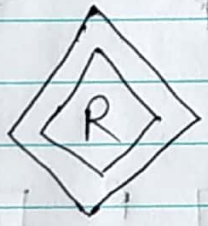
Multivalued Attribute.



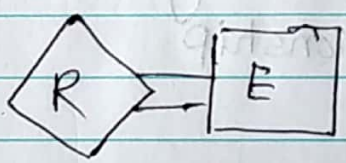
Relationship set



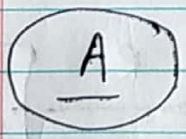
Derived Attribute



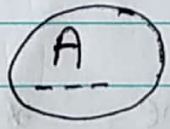
Identifying Relationship set for weak Entity set



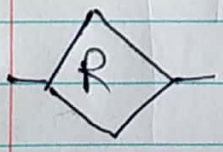
Total participation of Entity set in Relationship



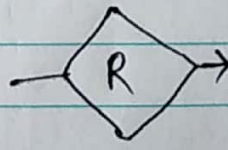
Primary key



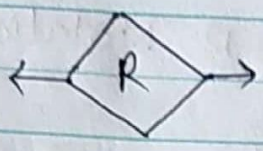
Discsiminating attribute of weak entity set.



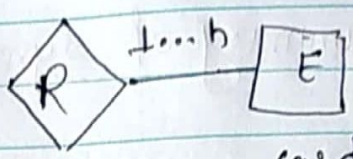
Many to Many Relationship



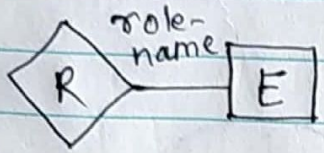
Many to one Relationship



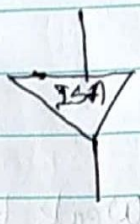
One to one Relationship



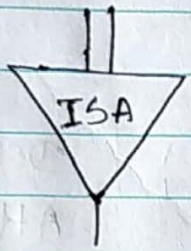
cardinality Limits



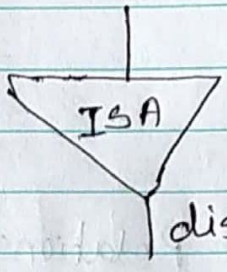
Role indicator



ISA (Specialization or Generalization)

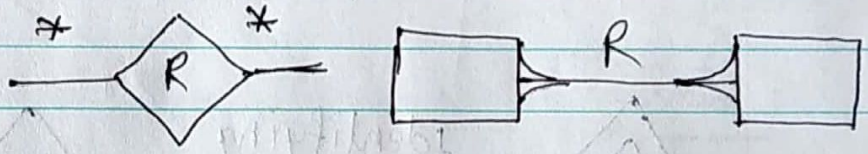


Total Generalization

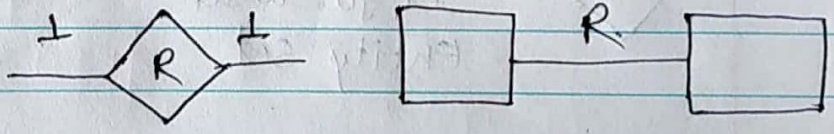


Disjoint Generalization

Many to Many Relationship



One to one Relationship



Many to One Relationship