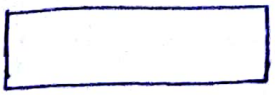
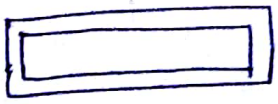
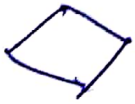
UNIT-2

ER diagram Notations:-


entity set


weak entity set


Relationship


Identifying relationship for weak entity

Primary key

one to many

many to one

one-to-one

many-to-many


Attribute


multivalue attribute


derived attribute

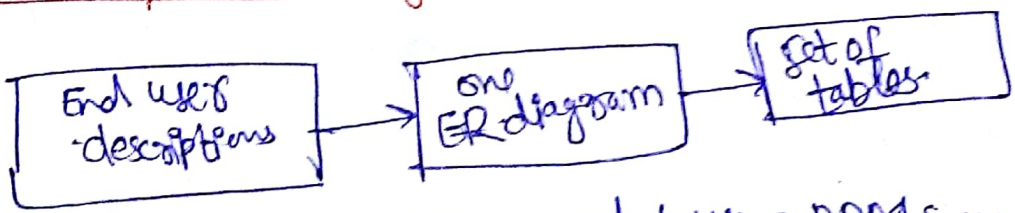

composite attribute

total participation

Partial participation

specialization or generalization

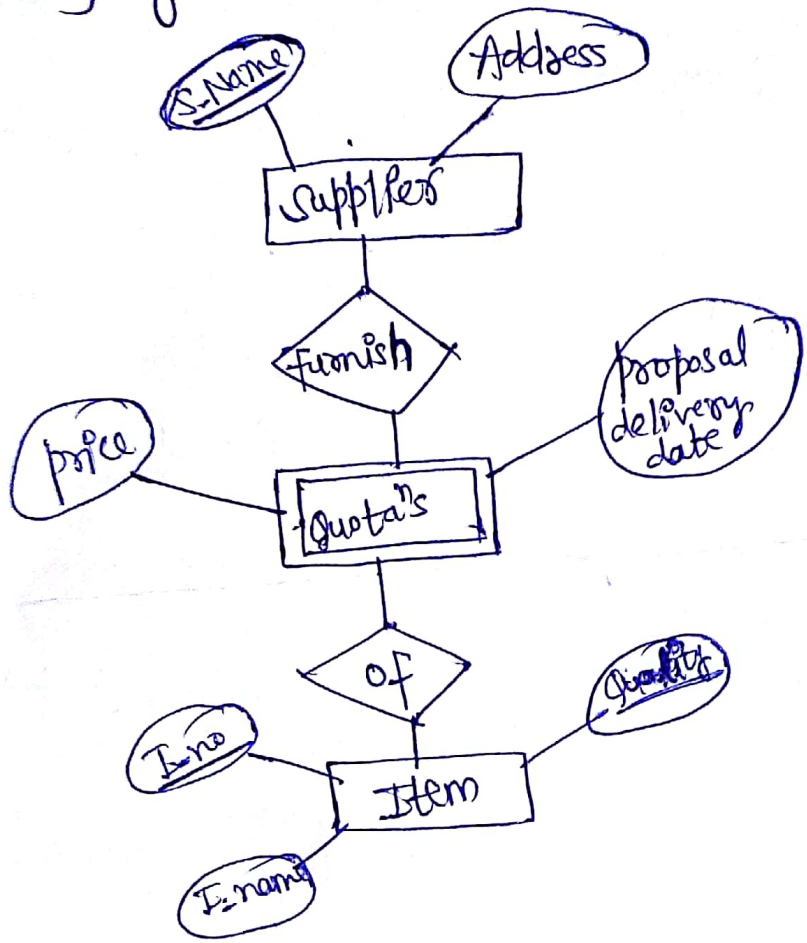
Goals of E-R diagram:-



1) used in DB design (If you want to use a DBMS, you need to be able to rep. your data in it.
2) converts the requirement of system to graphical representation, so that it can become well understandable
3) It should be simple & easy to understand with a min. of training. Therefore, the model can be used by the db designer to communicate the design to the end users

Weak entity set:-

- There are 2 types of entities namely ← dependent entities (weak entity) / regular entities (strong entity)
- It is the one whose existence depends on another entity.
- An entity set is called a weak entity set if its existence depends on the existence of primary key of strong entity.
- A weak entity set doesn't have sufficient attributes to form a primary key.



Primary key of supplier: (S_name)
" " " Item: (I_no)
" " " weak entity set Quota"s: (S_Name, I_no.) (It is formed by P.K. of 2 strong entities)

E-R diagram Construction:-

Q. Construct an ER diag for Banking system, for following descri

- Bank have customers
- customers are identified by name, cust_id, phone_no, address.
- customers can have one or more accounts. Accounts are identified b acc_no, acc type (saving, current), balance.
- customers can avail loans.

- Loans are identified by Loan_id, Loan type (car, home) & amount.
- Banks are identified by a name, code, address of main office.
- Banks have branches.
- Branches are identified by a branch_no, branch name, address
- Account & loans are related to Banks branches.

Sol^n Entities are Bank, customer, Branch, account, loan

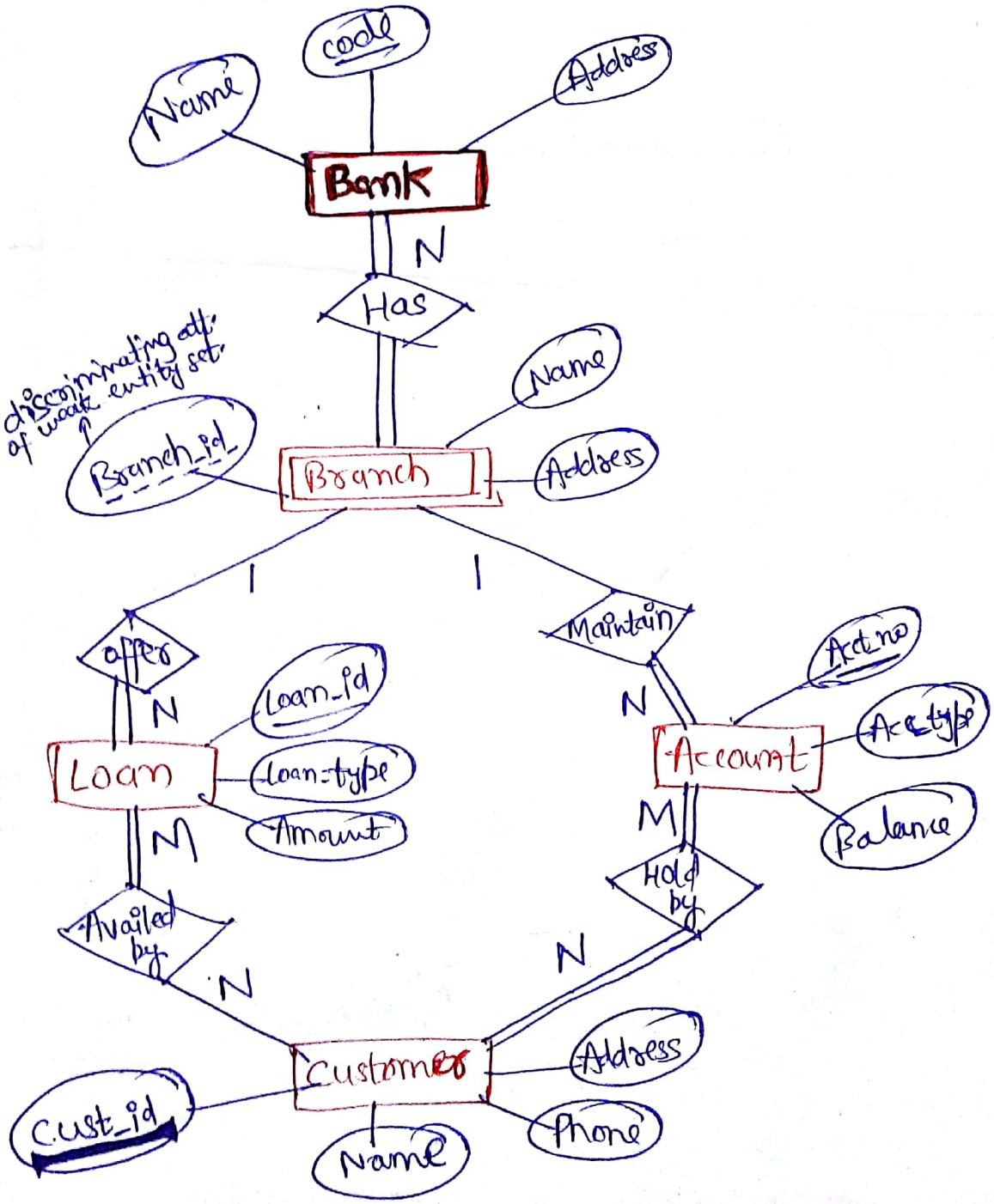
Relationships — Bank has branches → 1:N

Branch maintain accnts → 1:N

Branch offer loans → 1:N

Accnt held by customer → M:N

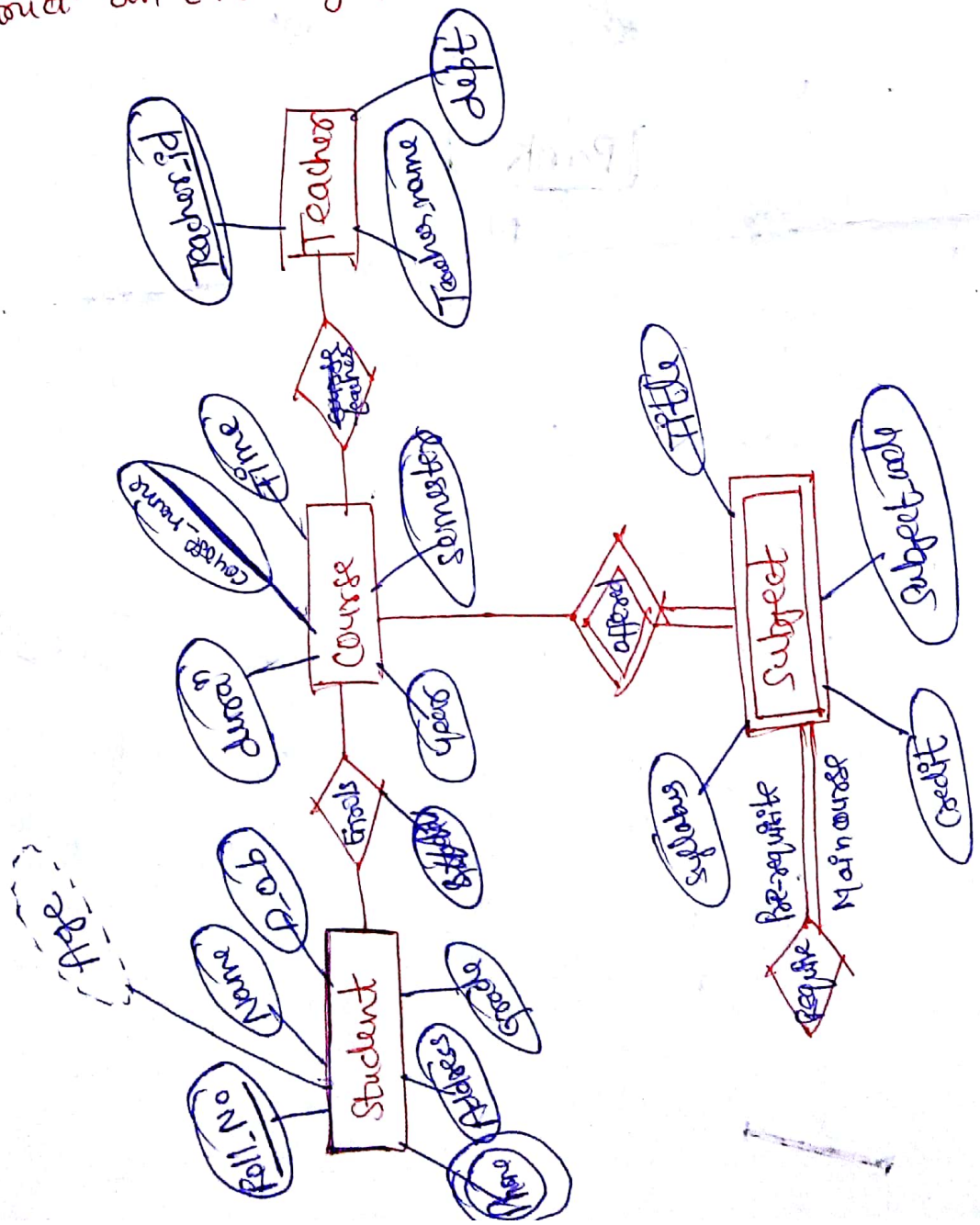
Loan availed by customer → M:N (Assume that one loan can be jointly held by many customers).



Q2. A University maintains data about following entities:

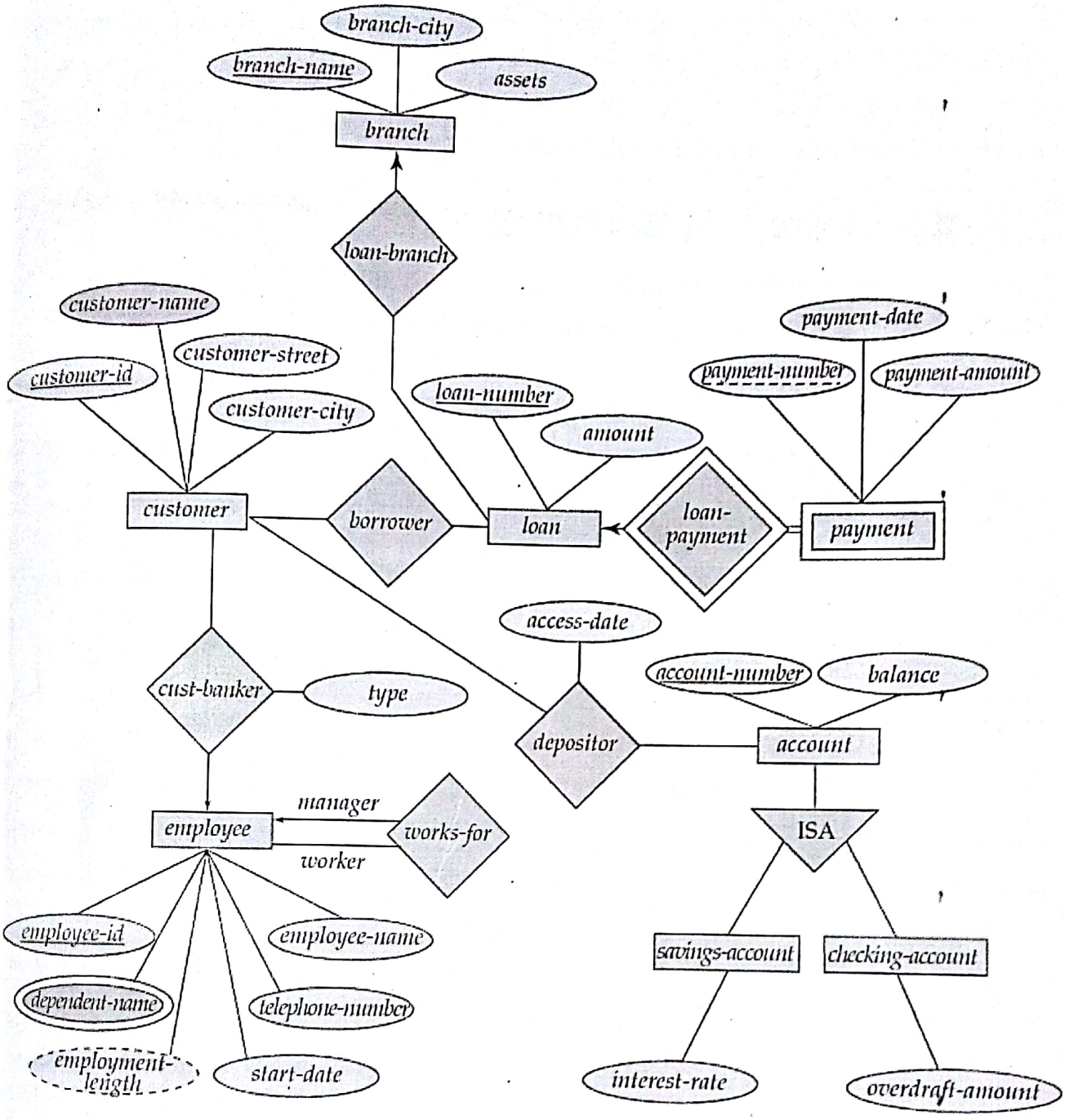
(1.) Students, including roll no, dob, address, telephone_no.
(2.) Couse (B.Tech, BCA, BBA, MBA) including duration, course_name, year, semester time.
(3.) Subject including subject_code, title, credit, syllabus, & pre-requisites.
(4.) Teacher, including teacher_id, name, dept. A teacher can teach only a specific course subject. Further the enrollment of students in courses & grades awarded to students in each student should be appropriately modelled.

Construct an ER diag. for the University.



Example of E-R Diagrams:

1. E-R diagram for a banking enterprise.



TABULAR REPRESENTATION OF ER-SCHEMAS:
OR
REDUCTION OF E-R DIAG. INTO TABLES:

An E-R diag. can be converted into tables, where name of the entity set or relationship set becomes the name of the table & its attributes becomes the columns of corresponding table.

(1) **Reduction of strong entity set into tables:** have following rules:

(a) Convert each entity type into table.

(b) Convert each single valued att. of the entity set to respective column of the table.

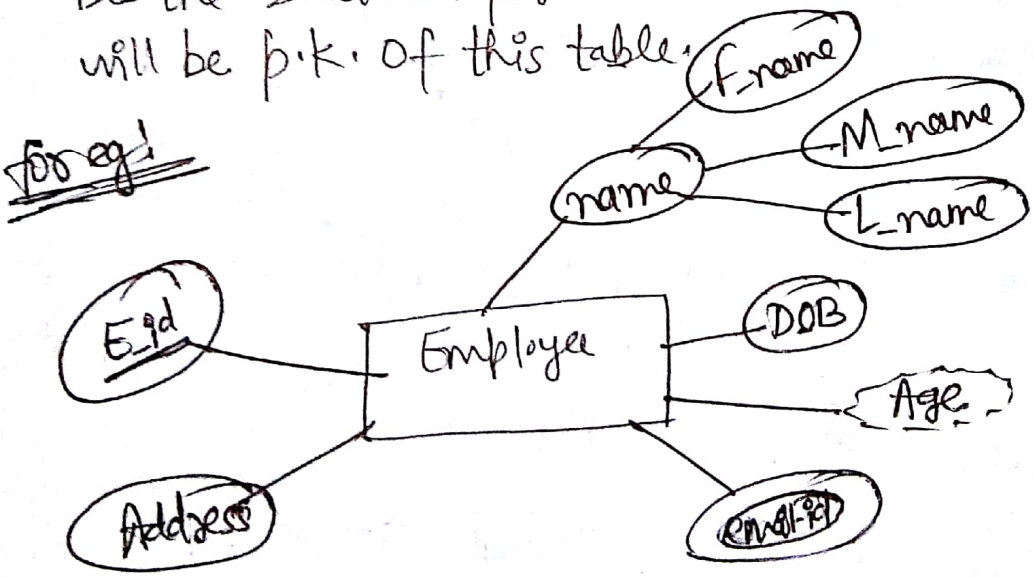
(c) Convert each composite att. by creating a separate column in the table for each comp. attribute.

(d) key att. of entity set becomes primary key of the table.

(e) Derived att. are ignored.

(f) A multivalued att. is rep. by a separate table. key attribute of the entity set & the name of multi-valued att. would be the 2 column for this new table. These 2 columns jointly will be p.k. of this table.

For eg:



The ER diag. can be converted to table using the above discussed rules:-

Employee table

E_id	F_name	M_name	Last_name	Address	DOB
1.					
2.					

Employee_Email table

E_id	Email-id
1	
2	

(2.) Reduction of weak entity set:

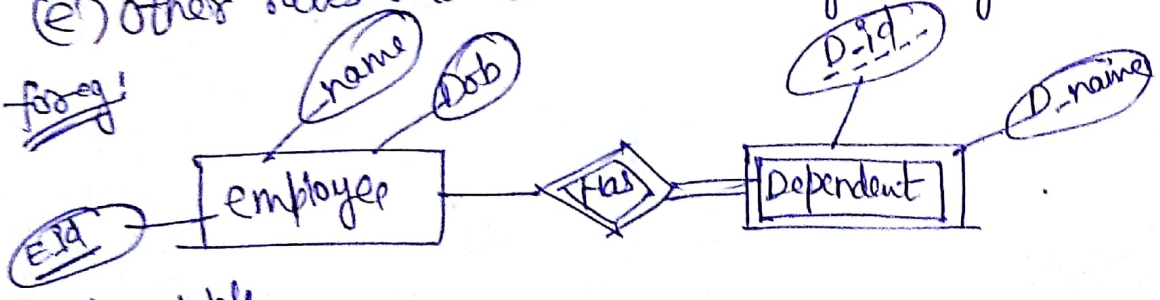
(a) Convert the weak entity set into a table of its own.

(b) Convert each attribute of weak entity set into respective column of the table.

(c) key attribute of strong entity set on which this weak entity is existence dependent will become a columns in the table it will act as foreign key in this table.

(d) Primary key of the table will be a composite key consisting of foreign key (primary key of strong entity set) + discriminator of weak entity set.

(e) Other rules are same as strong entity set.



Employee table

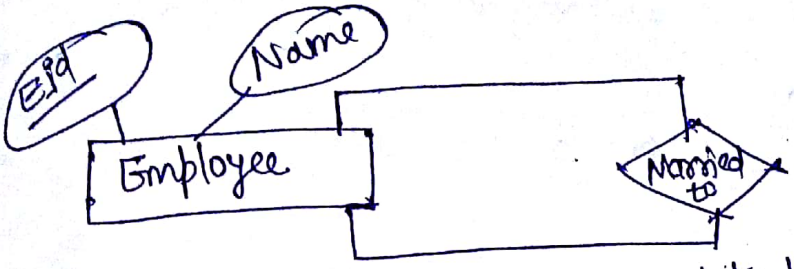
E_id	Name	DOB

Department table

E_id	D_id	D_name

(3.) Reduction of Relationship set into table :-

(a) Unary 1:1 Relationship

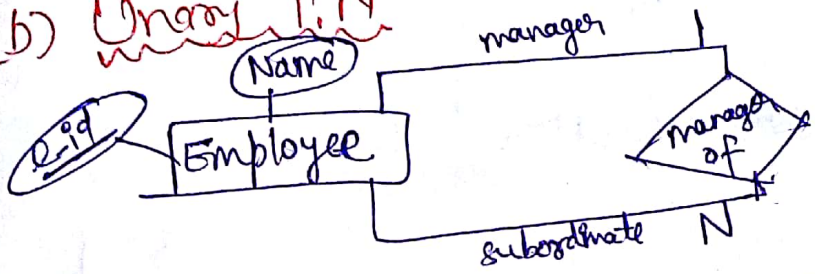


Employee

E_id	Name	Married_To_id

p.k. will become foreign key in same table.

(b) Unary 1:N

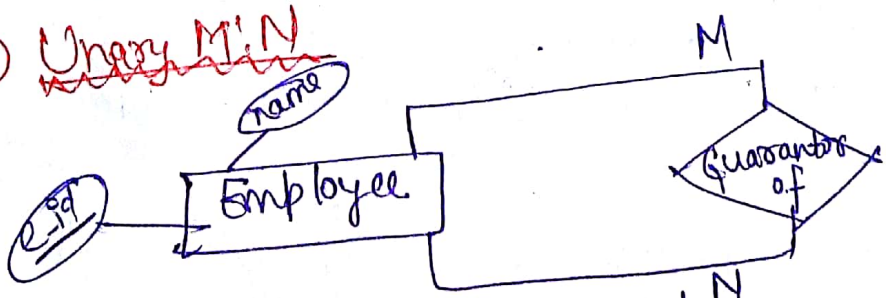


Employee

E_id	Name	Manager_id

will be same from set of E_id bcoz manager is also an employee.

(c) Unary M:N



Employee

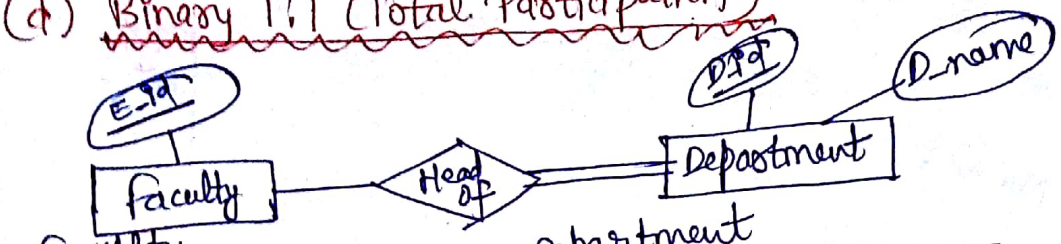
E_id	Name

Guaranty

Guarantor_E_id	Beneficiary_E_id

(d) ~~Ba~~

(d) Binary 1:1 (Total Participation)



Faculty

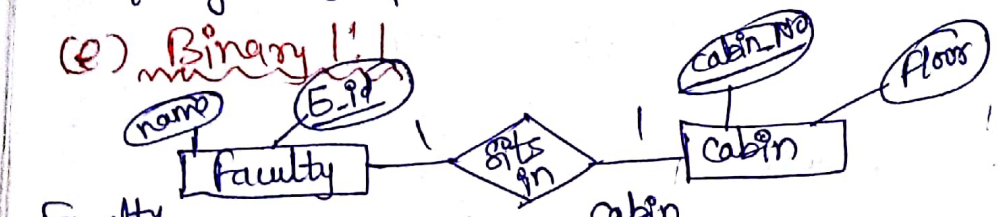
E_id	Name

Department

D_id	Name	Head_id

~~Partial~~ Primary key of partial participation will become foreign key of total participant

(e) Binary 1:1



Faculty

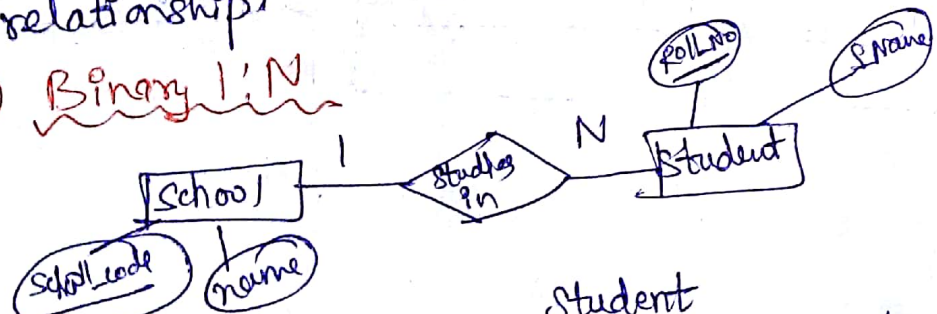
E_id	Name

Cabin

Cabin_no	Floor	E_id

Primary key of either entity set can be choosen as p.k. for the relationship.

(f) Binary 1:N



School

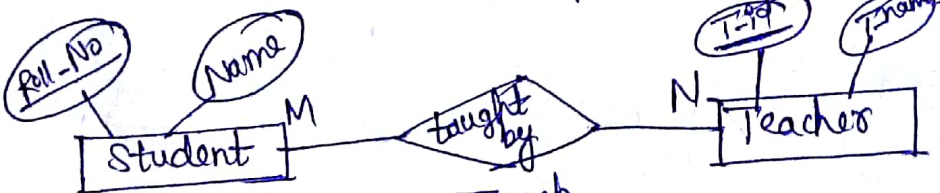
school_code	Name

Student

Roll_No.	S_Name	School_code

Primary key of table one 1 side of relationship become f.k. of table or N side.

(g) Binary M:N



student

Roll_No	Name

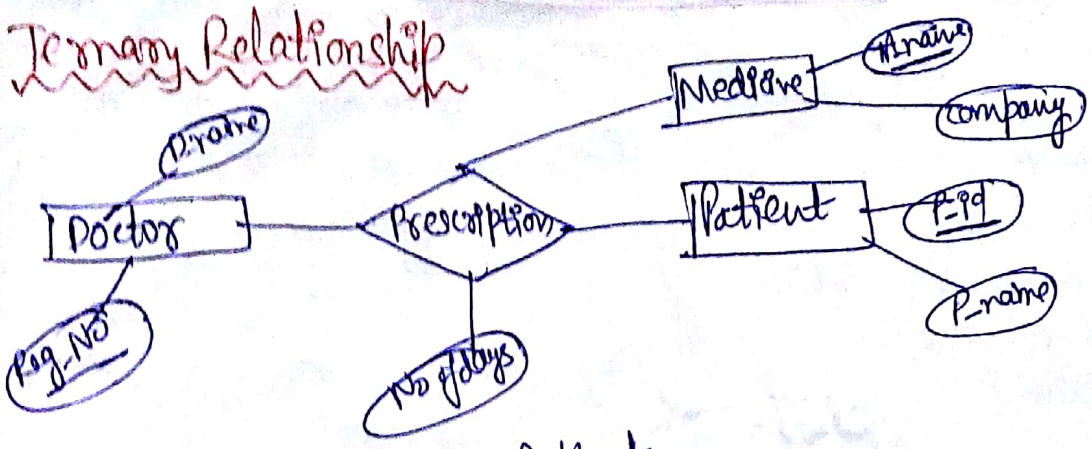
Teacher

T_id	T_Name

Teach

Roll_No	T_id

(h) Ternary Relationship



Doctor

Reg_No	D_name

Patient

P_id	P_name

Medicine

M_name	Company

Prescription

Reg_NO	M_Name	P_id	No_of_days

Q-3. Suppose you are given the following given req. for a simple db for the National Hockey League (NHL):

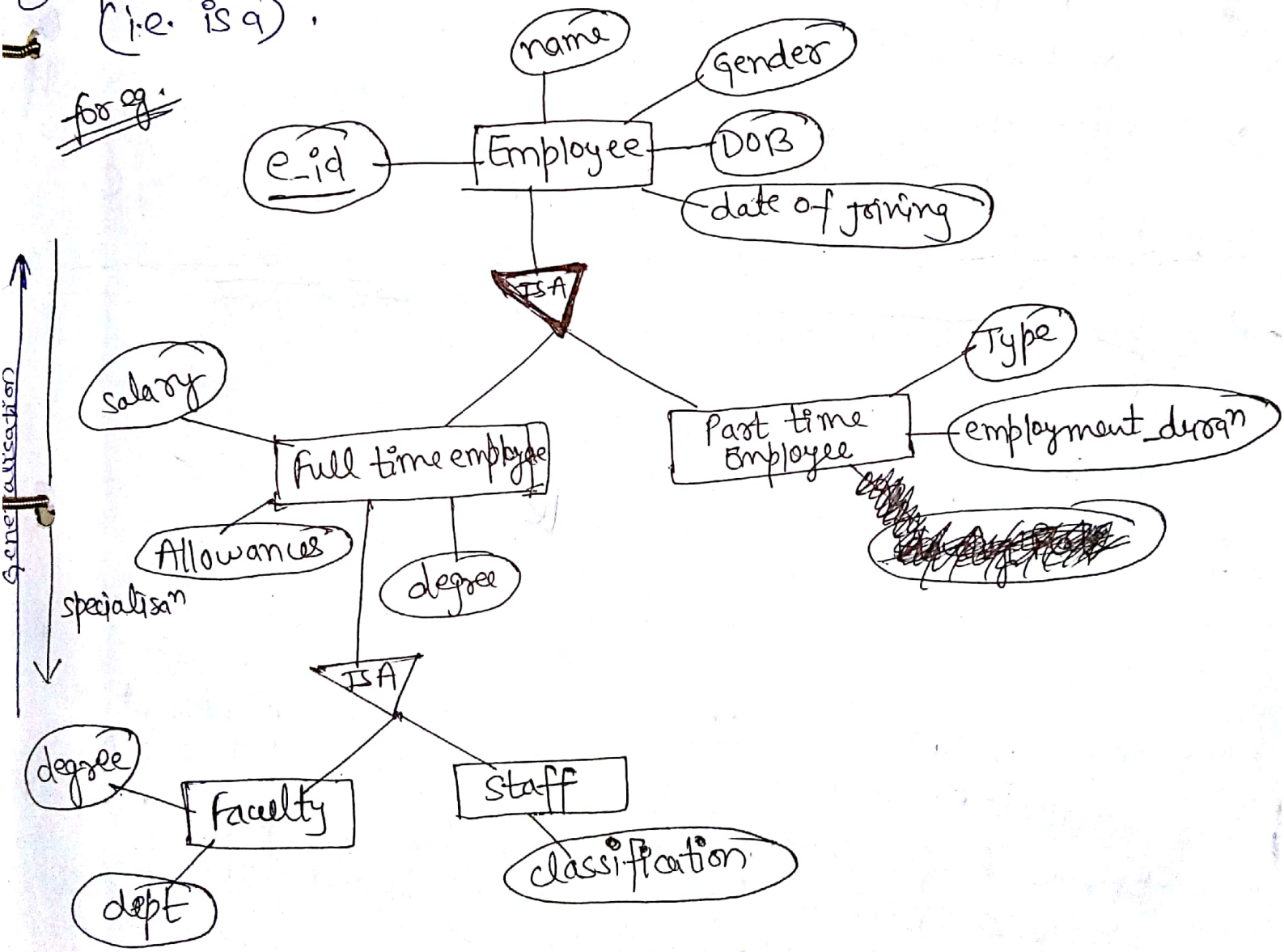
(1) the NHL has many teams
(2) each team has a name, a city, a coach, a captain, & a set of players.
(3) each player belongs to only one team.
(5) a game is played b/w 2 teams (referred to as host_team & guest_team) & has a date (such as May 11th 1999) & a score (such as 4 to 2).

Construct a clean & concise ER diag. for the NHL db.

Q1. C

→ SPECIALISATION :-

- It is the process of maximising difference b/w members of an entity by identifying their distinguishing characteristics.
- In this sub-sets of an entity set are identified ~~their disti~~ that do not share some common characteristics within an entity set.
- It is a top-down process in which super-class ~~along~~ divides into super-class.
- super-class & subclass along with unique attributes are defined next.
- specialisation is depicted by a triangle comp. labelled ISA (i.e. is a).



→ GENERALISATION :-

- It is a process of minimising diff. b/w entities by identifying their common characteristics.

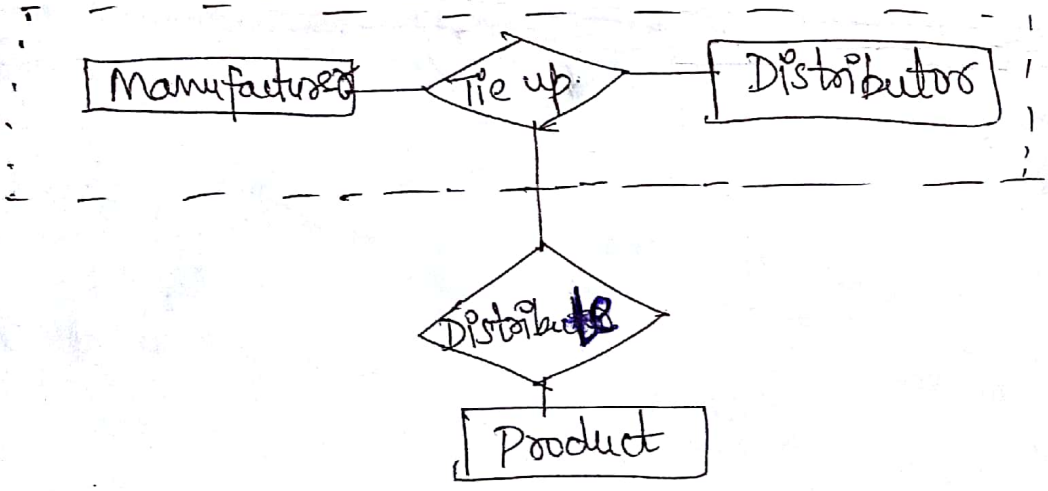
- It is a bottom-up process in which super-class is identified from sub-classes.
- It is just opp. of specialisation in which first sub-class are defined & then, super class.

→ AGGREGATION :-

- It is used when we have to model a relationship involving a relationship set. It allows us to treat a relationship set as an entity set for the purpose of participation in other relationships.
- So, to allow relationship set to establish relationship b/w another relaship set aggregation can be used.
- Therefore, aggregation is an abstraction through which relaships are treated as higher level entities, thus the relaship b/w entities A & B is treated as if it was an entity C.

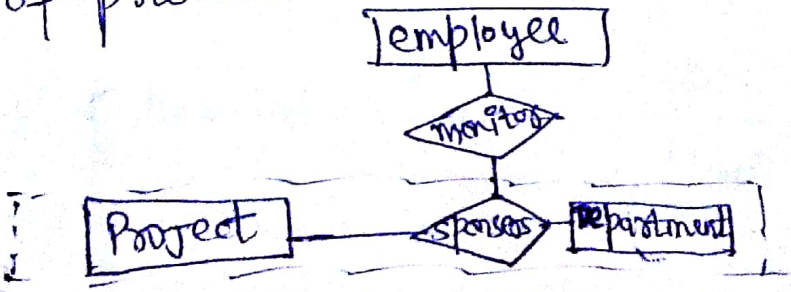
for eg:

(1.)



In above fig, There are three entities — manufacturer, distributor, & product. Manufacturer have tie-up with distributor to distribute the products. Each tie-up has specified for it the set of product which are to be distributed.

(2.)



→ AGGREGATION :- It is the process of compiling info. on an object, thereby abstracting a higher level object.
One limitation of E-R diag. model is that it cannot express relationship within relationships.
To illustrate the need for such as constructs, consider the ternary relationship work_on, b/w an employee, branch & job as shown in fig. below :-

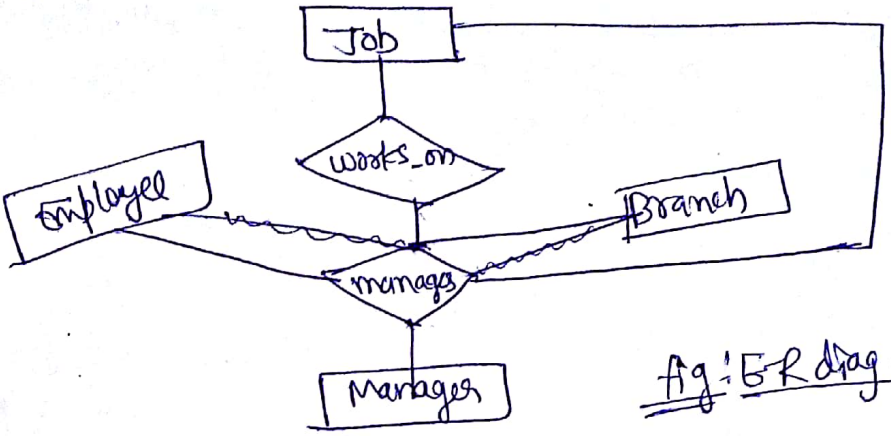


fig: E-R diag. with redundant relationship

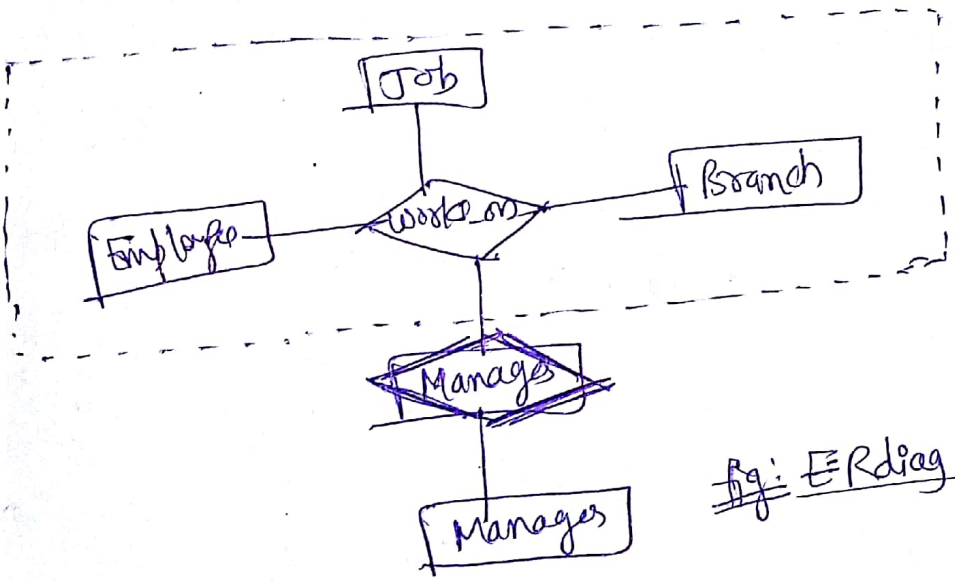


fig: E-R diag with Aggregation

① Relationship works_on relating the entity sets Employee, Branch & Job is a higher level entity set.
② Aggregation is an abstraction in which relationship sets (along with their associated entity sets) are treated as higher-level entity sets & can participate in relationships.
③ The use of aggregation — can treat the aggregate entity set as a single unit w/o concern for the details of its internal structure.

→ **KEYS!-** A key is an attribute or a set of attributes which can uniquely identify each tuple of a relation.

① **Super key:** It is an att. or set of att. whose combined value uniquely identifies a tuple within a relation.

foreg:

Employee

Emp_code	Name	Address	Dept_id

for the above relⁿ - employee, following super key can exist!

- → {Emp_code, Name, Address, Dept_id}
- → {Emp_code, Name, Address}
- → {Name, Address} (; assume no 2 employees with same name & address exist).
- → {Emp_code} etc.

② **Candidate key:** It is a minimal super key i.e. a super key which does not have any proper subset which is also a super key. A candidate key is a special case of super key,

or

A minimal set of att. that uniquely identifies each & every tuple a relation. It can be NULL.

foreg: {Emp_code} and {Name, Address} are candidate keys of a relⁿ.

{Emp_code} is a cand. key bcoz it is minimal super key that can uniquely identify any tuple of a relⁿ.

{Name, Address} is a cand. key bcoz it is a super key & its subset {Name} and {Address} are not super key & can't uniquely identify tuples within a relⁿ.

③ **Primary key:** — An attribute that uniquely identifies each & every tuple in a relⁿ. It can not be NULL.

eg: {Emp_code}

④ **Composite primary key:** — A primary key formed by the combinⁿ of one or more attributes.

foreg: {Name, Address}

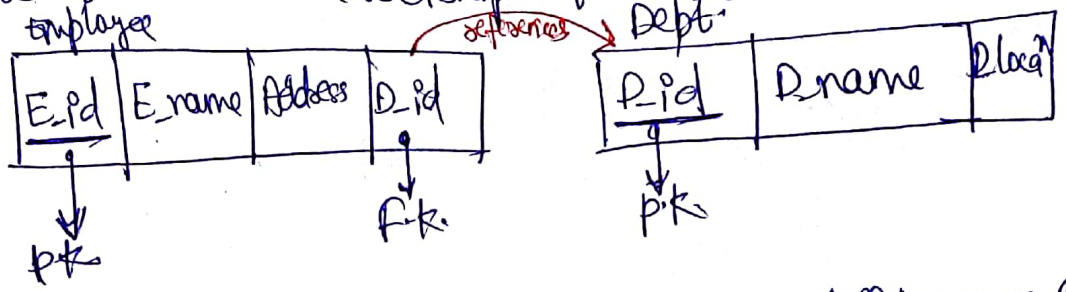
⊙ Alternate key:- ~~candidate~~ candidate keys − Primary keys = Alternate keys

⊙ Foreign key:- A set of attributes of a relⁿ whose values are required to match the values of some candidate key in same or another table.

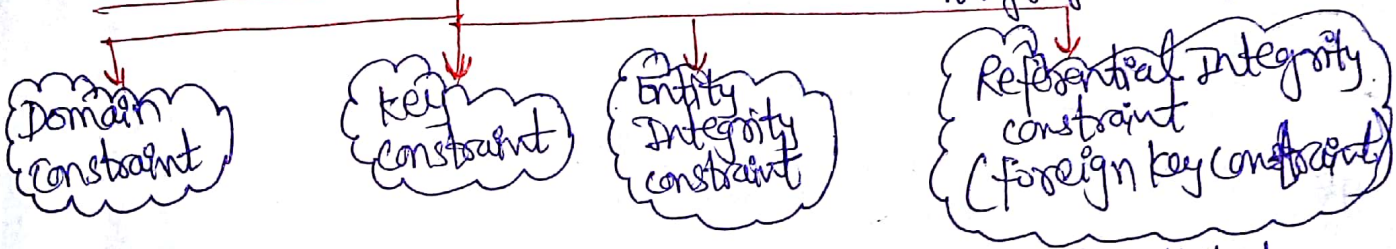
OR

When a primary key of one relation exist as an attribute in another relation then, this attribute is called a foreign key. It is used to establish a relⁿship b/w 2 tables.

foreg:



→ INTEGRITY CONSTRAINTS:- Every relⁿ has some condⁿs that must hold for it to be a valid relⁿ. These condⁿs are called integrity constraints



⊙ Domain constraints: It states that the value of each attribute must be an atomic value from the domain of that attribute.

foreg: If Name is defined to be character (char data type is char), then it can not take Integer value.

Employee

E_id	Name	D_id
1	Maya	7
2	15	3

Integers value not allowed.

⊙ Key constraint: There must be atleast one minimal set of attributes (primary key) in a relⁿ, which can identify a tuple uniquely. This minimal subset of att. is called key for a relⁿ

It has following conditions!

- In a relation with a key attribute, no 2 tuples can have identical values for key attribute.

OR

- The primary key attribute of a relation must have unique

for eg.

Employee

E_Id	E_name	D_id
1	Vinayak	5
2	Amit	2
2	Shashank	4

Not allowed.

Z(5).
X(5).
X(20).
X(5).
ZZZ.99.
X(5).
Z(5).
X(5).
Z(5).
X(71).

- **Entity Integrity constraint:** It states that the value of attribute of a primary key can not be NULL.

Employee

E_Id	Name	D_id

A relation employee in which E_Id is designated as the primary key. Now, any value in this att. cannot be NULL.

9(5).

JAL TO "Y".

- **Referential Integrity constraint:** – It states that either the foreign key can have only those values that match the p.k. of other relation or it has NULL value.

RD.
TO "Y"]
ABLE-STOCK
THAN BAL-REORDER-
ER-LINE.

Employee

E_Id	E_name	D_id
1	Angali	5
2	Vishal	2
3	Twinkle	7

D_id	D_name
5	Sales
4	Marketing
6	Accounts

Null value allowed

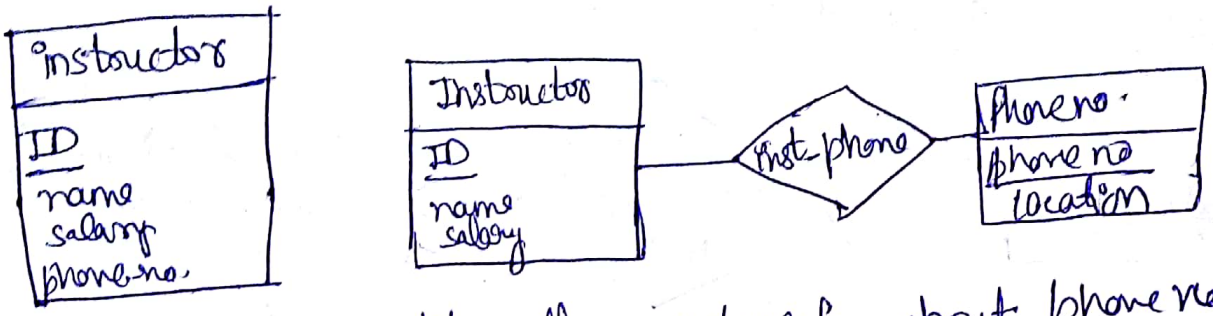
This is not allowed because D_id=2 is not present in D_id att. of the relation Dept.

ITCH.

DER-LINE.

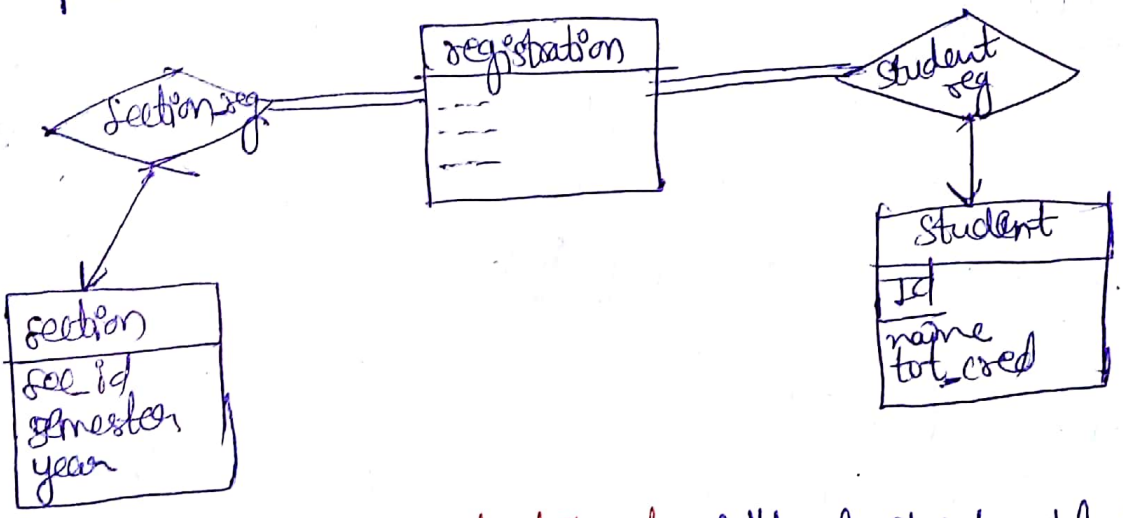
→ DESIGN ISSUES IN CHOOSING ATTRIBUTES OR ENTITY SET OR RELATIONSHIP SET :-

(1) **Use of entity sets Vs attribute:**
choice mainly depends on the structure of the enterprise being modeled, & on the semantics associated with the att. in question.



Use of phone as an entity allows extra info. about phone numbers (plus multiple phone no).

(2) **Use of entity sets Vs relationship sets:** Possible guideline is to design a relationship set to describe an action that occurs b/w entities.

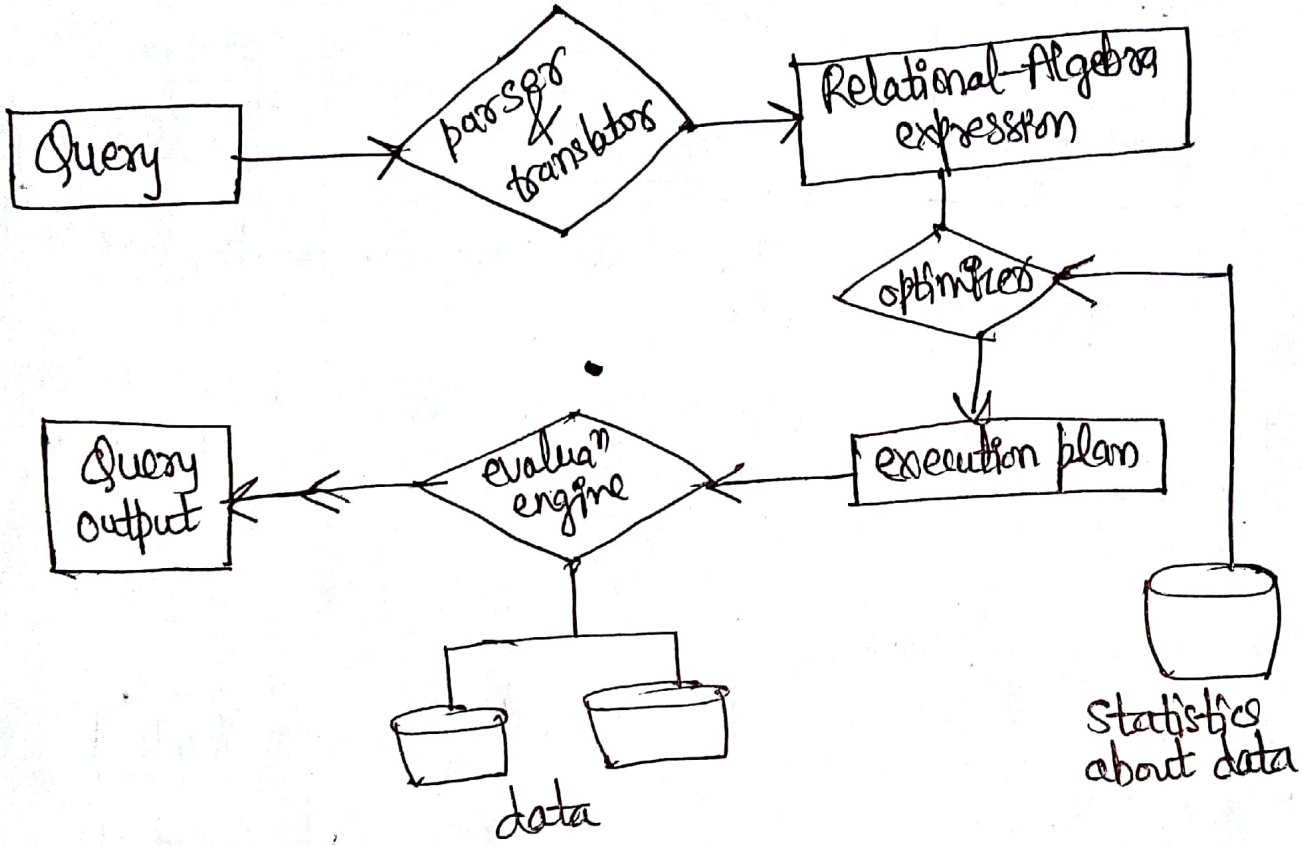


(3) **Binary versus n-ary relationship sets:** Although it is possible to replace any nonbinary (n-ary, for n>2) relationship set by a number of distinct binary relationship sets, an n-ary relationship set shows more clearly that several entities participate in a single relationship.

(4) **Placement of relationship attributes:**
Eg: attribute date as att. of advisor or as att. of student

OVERVIEW OF QUERY PROCESSING:-

① Basic steps in query processing:
- Parsing & translation
- optimization
- Evaluation



② Parsing & translation
- translate the query into its internal form. This is then translated into relational algebra.
- Parser checks syntax, verifies relations.

③ Evaluation
- The query execution engine takes a query evaluation plan, executes that plan, & returns the answers to the query.

④ ~~Optimization~~

Optimizaⁿ -

→ A relaⁿal algebra expression may have many equivalent expressions. eg: $\sigma_{salary<7500}(instructor)$ is equiv. to $\Pi_{salary}(\sigma_{salary<7500}(instructor))$

→ Each relaⁿal algebra operaⁿ can be evaluated using one of several different algos.
  ↳ correspondingly, a relaⁿal-algebra expression can be evaluated in many ways.

→ Annotated expression specifying detailed evaluaⁿ strategy is called an evaluaⁿ-plan.
  ↳ eg: can use an index on salary to find instructors with salary <7500.
  ↳ or can perform complete relaⁿ scan & discard instructors with salary ≥ 75000.

→ Query optimizaⁿ: Amongst all equivalent evaluaⁿ plans choose the one with lower cost
  → cost is estimated using statistical info from db catalog eg no of tuples in each relation, size of tuples etc
  → cost difference b/w a good & a bad way of evaluating a query can be enormous
  → Needs to estimate the cost of operaⁿs
    ↳ depends critically on statistical info. of db relaⁿs which the db must maintain
    ↳ Need to estimate statistics for intermediate results to compute cost of complex expressions.

Relational Algebra

- Procedural language
- Six basic operators
  - select: σ
  - project: ∏
  - union: ∪
  - set difference: –
  - Cartesian product: x
  - rename: ρ
- The operators take one or two relations as inputs and produce a new relation as a result.

Select Operation

- Notation: $\sigma_p(r)$
- p is called the selection predicate
- Defined as:

$$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of terms connected by: ∧ (and), ∨ (or), ¬ (not)
Each term is one of:

<attribute> *op* <attribute> or <constant>

where *op* is one of: $=, \neq, >, \geq, <, \leq$

- Example of selection:

$$\sigma_{dept\_name = "Physics"}(instructor)$$

$\sigma_{dept\_name = "physics"}(instructor)$

Select Operation – Example

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Project Operation – Example

- Relation r:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Project Operation

- Notation: $\Pi_{A_1, A_2, \ldots, A_k}(r)$

  where A_1, A_2 are attribute names and r is a relation name.

- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: To eliminate the *dept_name* attribute of *instructor*

Union Operation

- Notation: $r \cup s$
- Defined as:

  $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

- For $r \cup s$ to be valid.

  1. r, s must have the same arity (same number of attributes)
  2. The attribute domains must be compatible (example: 2nd column of r deals with the same type of values as does the 2nd column of s)

- Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

  $\Pi_{course\_id}(\sigma_{semester = \text{"Fall"} \wedge year = 2009}(section)) \cup \Pi_{course\_id}(\sigma_{semester = \text{"Spring"} \wedge year = 2010}(section))$

Union Operation – Example

- Relations r, s:

r	
A	B
α	1
α	2
β	1

s	
A	B
α	2
β	3

- $r \cup s$:

A	B
α	1
α	2
β	1
β	3

Set difference of two relations

- Relations r, s

r	
A	B
α	1
α	2
β	1

s	
A	B
α	2
β	3

- $r - s$:

A	B
α	1
β	1

Set Difference Operation

- Notation $r - s$
- Defined as

$r - s = \{t \mid t \in r \text{ and } t \notin s\}$

- Set differences must be taken between **compatible** relations
  - r and s must have the same arity
  - attribute domains of r and s must be compatible
- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$\Pi_{course\_id}(\sigma_{semester = "Fall" \wedge year = 2009}(section)) -$
$\Pi_{course\_id}(\sigma_{semester = "Spring" \wedge year = 2010}(section))$

Cartesian-Product Operation – Example

- Relations r, s:

A	B
α	1
β	2

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

- $r \times s$:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Cartesian-Product Operation

- Notation $r \times s$
- Defined as

$r \times s = \{t\, q \mid t \in r \text{ and } q \in s\}$

- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \emptyset$)
- If attributes of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$
- $r \times s$:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- $\sigma_{A=C}(r \times s)$:

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- Example:

$\rho_X(E)$

returns the expression E under the name X

- If a relational-algebra expression E has arity n, then

$\rho_{X(A_1, A_2, \ldots, A_n)}(E)$

returns the result of expression E under the name X, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

Example Query

- Find the largest salary in the university
  - Step 1: find instructor salaries that are less than some other instructor salary (i.e. not maximum)
    - using a copy of *instructor* under a new name *d*
    - $\Pi_{instructor.salary}(\sigma_{instructor.salary < d.salary}(instructor \times \rho_d(instructor)))$

OR

  - Step 2: Find the largest salary
    - $\Pi_{salary}(instructor) - \Pi_{instructor.salary}(\sigma_{instructor.salary < d.salary}(instructor \times \rho_d(instructor)))$

Example Queries

- Find the names of all instructors in the Physics department, along with the *course_id* of all courses they have taught
  - Query 1

$\Pi_{instructor.ID, course\_id}(\sigma_{dept\_name="Physics"}($
$\sigma_{instructor.ID=teaches.ID}(instructor \times teaches)))$

OR

  - Query 2

$\Pi_{instructor.ID, course\_id}(\sigma_{instructor.ID=teaches.ID}($
$\sigma_{dept\_name="Physics"}(instructor) \times$
$teaches))$

Formal Definition

- A basic expression in the relational algebra consists of either one of the following
  - A relation in the database
  - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p(E_1)$, P is a predicate on attributes in E_1
  - $\Pi_S(E_1)$, S is a list consisting of some of the attributes in E_1
  - $\rho_x(E_1)$, x is the new name for the result of E_1

Additional Operations

We define additional operations that do not add any power to the relational algebra, but that simplify common queries.

- Set intersection
- Natural join
- Assignment
- Outer join

Set-Intersection Operation – Example

- Relations r, s:

A	B
α	1
β	2
β	3

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
β	3

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r \text{ and } t \in s \}$
- Assume:
  - r, s have the *same arity*
  - attributes of r and s are compatible
- Note: $r \cap s = r - (r - s)$

Natural join: operation is same as the cp. operation it joins two relations but the diff. is that it automatically forces equality on those attributes that appears in both the relations & then also removes duplicate attributes. It is denoted by ($\bowtie$). If join attributes of two relations have same name then, they need not be specified.

Natural-Join Operation

- Notation: $r \bowtie s$
- Let r and s be relations on schemas R and S respectively. Then, $r \bowtie s$ is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from r and t_s from s.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result, where
    - t has the same value as t_r on r
    - t has the same value as t_s on s
- Example
  - $R = (A, B, C, D)$
  - $S = (E, B, D)$
  - Result schema = (A, B, C, D, E)
  - $r \bowtie s$ is defined as

~~$\Pi_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \wedge r.D = s.D} (r \times s))$~~

$$\Pi_{r.A, r.B, r.C, r.D, s.E} \left(\sigma_{r.B = s.B \wedge r.D = s.D} (r \times s) \right)$$

VIEWS:- A view is a virtual or logical table that allows to view or manipulate parts of the tables. To reduce REDUNDANT DATA to the min. possible, oracle allows the creation of an object called a VIEW.

Views, which are a type of virtual tables allows users to do the following:

- structure data in a way that users or classes of users find natural or intuitive
- Restrict access to the data in such a way that a user can see & (sometimes) modify exactly what they need & no more.
- summarize data from various tables which can be used to generate reports.

Reasons why views are created are:-

- When data security is required
- When data redundancy is to be kept to the min. while maintaining data security.

Types of views:

- **Read-only view:** → If a view is used to only look at tables i.e. allows only SELECT opera's then it is called as Read-only view.
- **Updated view:** is A view that is used to look at table data as well as INSERT, UPDATE & DELETE table data.

A view constains rows & columns, just like a ~~relo~~ real table. The fields in a view are fields from one or more real tables in the db

⊙ Syntax to **create view**: views can be created from a single table, multiple tables or another view.

```
CREATE VIEW view_name AS ~~SEEE~~
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

eg.

Customers

ID	NAME	AGE	ADD.	SALARY

```
Create view customers_view as Select name, age from
customers;
```

Now, you can query customers_view in a similar way as you query an actual table: i.e.

```
Select * from Customers_view;
```

↓ Result

Name	age

⊙ The WITH CHECK OPTION is a create view statement option. The purpose of the with check option is to ensure that all UPDATE & INSERTs satisfy the conditions in the view def. If they don't satisfy the condi's, the UPDATE or INSERT returns an error.

```
CREATE VIEW Customers_view AS
SELECT name, age
FROM customers
WHERE age IS NOT NULL
WITH CHECK OPTION;
```

⊙ UPDATING a View: a view can be update under certain conditions which are given below:—

① ~~The select clause may not~~ Views can be defined from single table.

(2.) If the user wants to INSERT records with the help of a view, then PRIMARY KEY column(s) & all the NOT NULL columns must be included in the view.

(3.) The user can UPDATE, DELETE records with the help of a view if the PRIMARY KEY column & NOT NULL column(s) are excluded from the view def.

(4.) All NOT NULL columns from the base table must be included in the view in order for the INSERT query to func^{n}.

(5.) A SELECT clause may not contain:-
- keyword DISTINCT.
- summary func^{n}s
- set func^{n}s & set operators.
- ORDER BY clause
- Multiple tables
- subqueries

(6.) The query may not contain GROUP BY or HAVING.

(7.) calculated columns may not be updated.

```
UPDATE customers_view SET age = 35 WHERE name='Ramesh';
```

(8.) INSERTING rows into a view; all rules are same as for UPDATE command.

(Here we can't insert rows in the customers_view bcoz we have not included all the NOT NULL columns in this view, otherwise you can insert rows in a view in a similar way as you insert them in a table.

(9.) DELETING Rows into a view; all rules are same as for UPDATE & INSERT commands.

```
DELETE from customers_view WHERE age=22;
```

(10.) DROPPING views: `DROP view view_name;`

eg: `DROP view customers_view;`

→ Advantages of view:-

① Security ② Query simplicity ③
③ Structural simplicity ④ Data integrity.

→ Disadvantages of View:-

① Performance ② Update restrictions