UNIT-II

Introduction to Relational Model

1) Integrity constraints over Relations
2) Querying Relational data
3) Logical database design
   - ER to Relational
4) Relational Algebra and calculus
   4.1) Preliminaries
   4.2) Relational Algebra
   4.3) Relational calculus
   4.4) Expressive power of Algebra and calculus.

Relational Model

↳ Data Models define how the logical structure of a database is modeled.

Data Models

- Relational Model
- Network Model
- Hierarchical Model
- . . . .

Relational Model → Both data and Relationships are represented as Relations (tables)

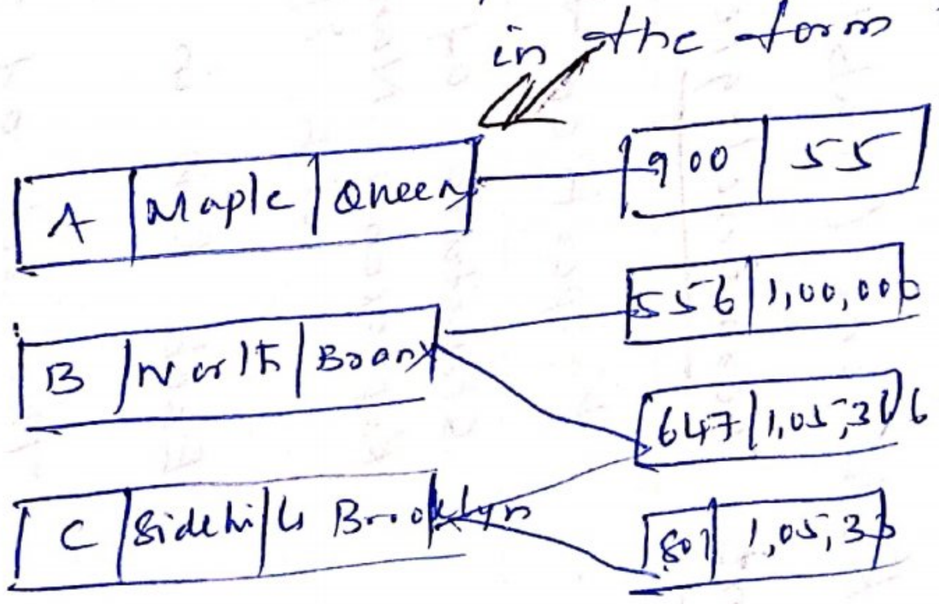
name	street	city	number
A	Maple	Queens	900
B	North	Bronx	556
B	North	Bronx	647
Hedges	Sidehill	Brooklyn	801
C	"	"	647

number	Balance
900	55
556	1,00,000
647	1,05,316
801	1,05,33

Network Model → Data is represented as records

~~Links~~ RLs are represented as links

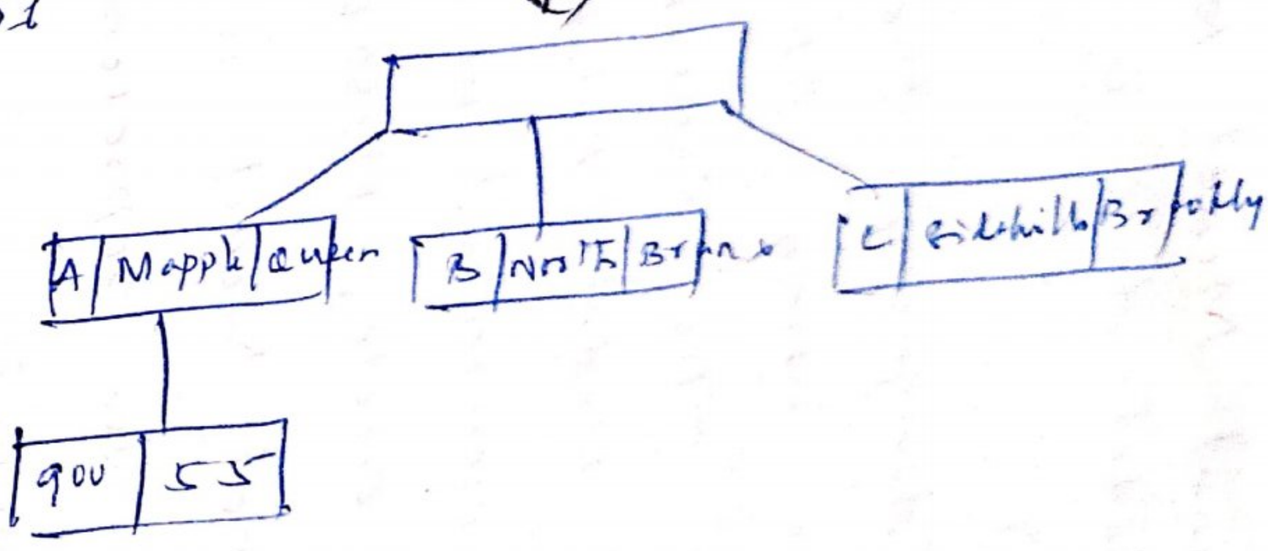
organization of data is in the form of graphs



Hierarchical Model → Data is represented as Records

RLs " " " Links

org of data is in the form of tree.



↳ Relational Model was proposed by E.F. Codd to Model data in the form of Relations (tables)

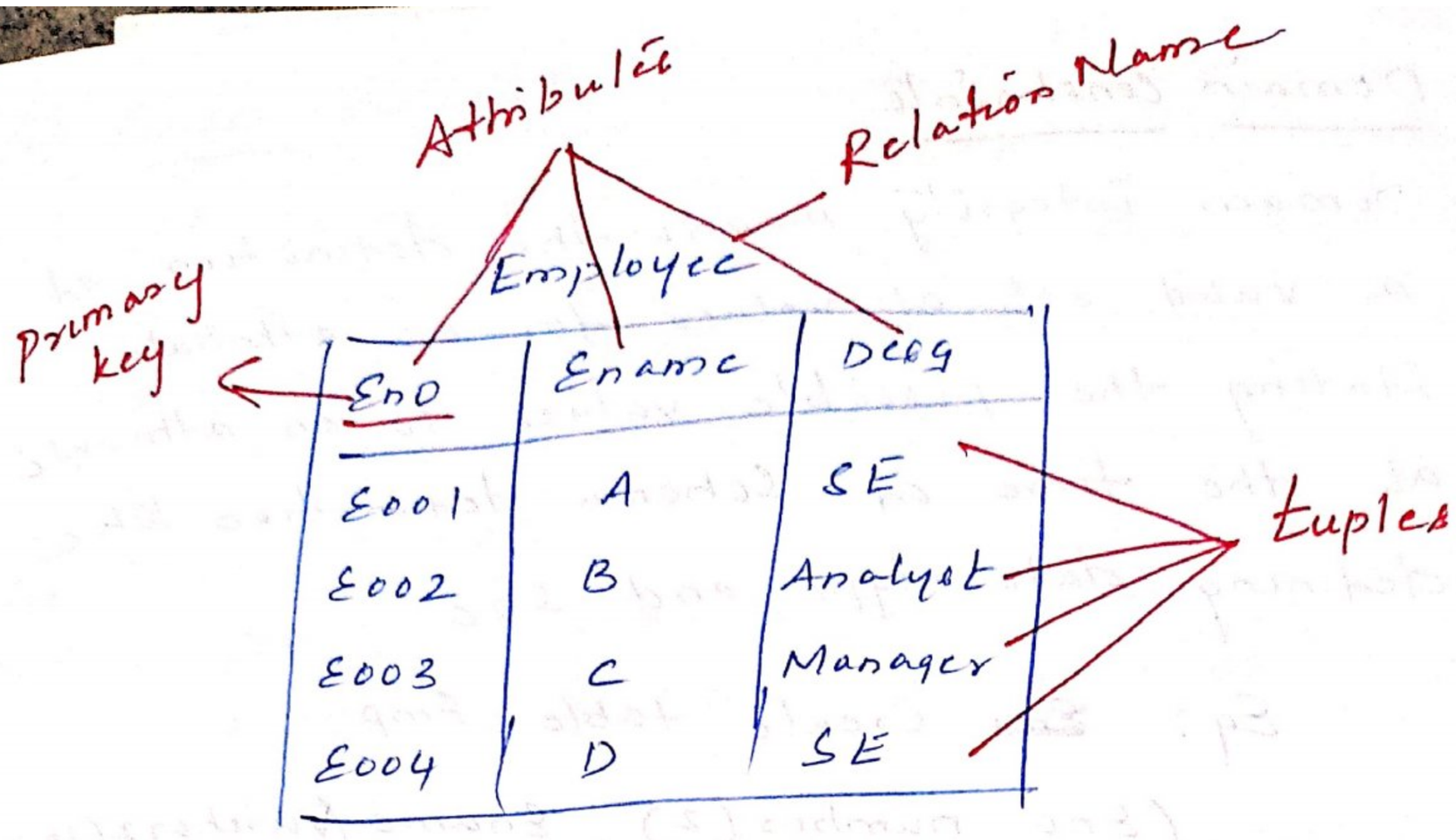
↳ After designing the Conceptual Model of database using ER diagram, we need to convert the conceptual Model into the Logical Model (Relational Model) which can be implemented using any RDBMS language like oracle, MySQL Etc--.

↳ Relational Model is the popular used model for representing data and its relationships.

↳ In Relational Model, both data and Relationships are represented in the form of Relations (tables).

Important Terminologies

Attribute :	Properties that define a Relation
Relation Schema	Structure of a Relation Represents Name of the Relation with its attributes
Tuple	Each Row in a Relation
Relation instance	The set of tuples of a Relation at a particular instance of time It can change whenever there is insertion, del & up in the
Cardinality	Number of tuples in a Relation
Degree	Number of attributes in a Relation.
Domain	Possible values of an attribute (or) valid set of values for an attribute.



Employee (Eno, Ename, Dept) — Relation schema

Degree of Employee Relation = 4 (# of tuples)

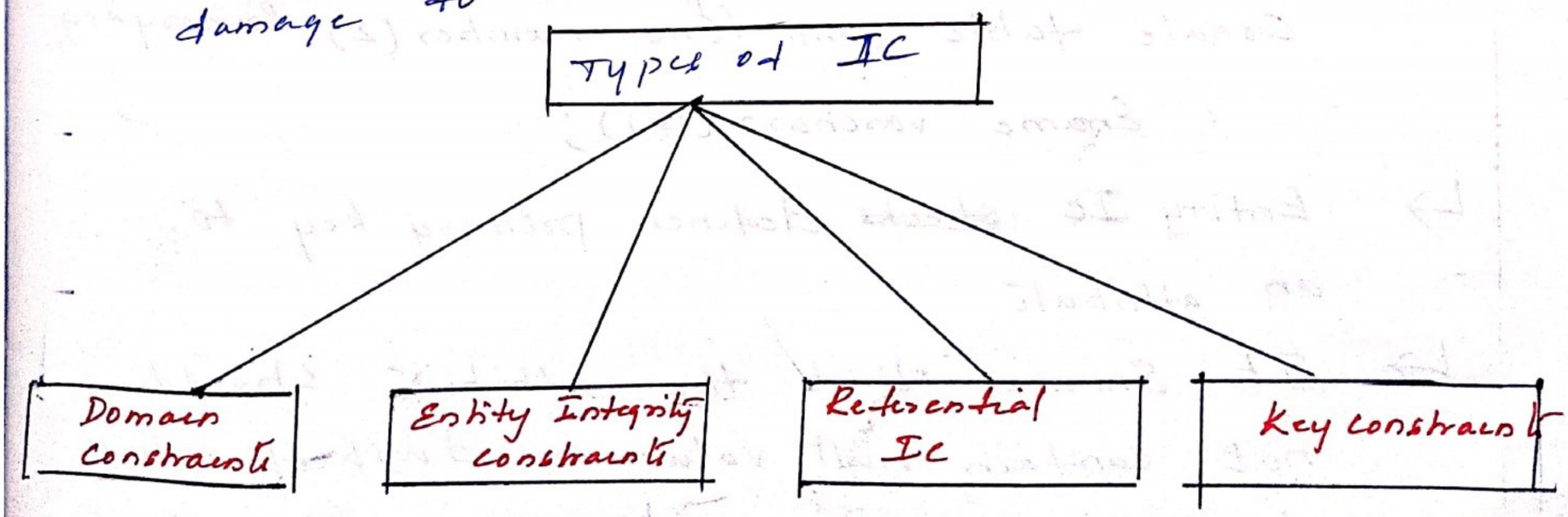
Cardinality " " = 3 (# of attributes)

1) Integrity constraints over Relation

↳ Integrity constraints are set of rules (or) conditions to be satisfied by the relations in the database

↳ Integrity constraints Ensures data consistency when data insertion and updations are performed

↳ IC is used to guard against accidental damage to the db.



① Domain Constraint

↳ Domain integrity means the definition of a valid set of values for an attribute

↳ Stating the possible values to an attribute at the time of schema definition like defining data type and size

Eg: Create table Emp

(Eno number(2), Ename Nvarchar2(10))

Domain constraint of Eno

Domain constraint of Ename

Domain constraint of Eno → all possible 1 digit and 2 digit numbers.

↳ if we try to insert a 3 digit number in Eno attribute, means we are violating it's domain constraint.

② Entity IC

↳ States that 'primary key value can't be null.

↳ For instance,

Entity IC

Create table Emp (Eno number(2) Primary key,

Ename varchar2(15));

↳ Entity Ic ~~checks~~ defines primary key to an attribute

↳ It ensures that the attribute should not contain null value or duplicate value because the primary key value is used to identify individual rows in relation.

③ Referential IC

↳ It states that if a foreign key exists in a relation then the foreign key attribute values should be derived from its primary key attribute or it can be NULL value.

↳ The Rules are

✓ you can't delete a record from a primary table if matching records exists in a Related table.

✓ you can't change a primary key value in the primary table if that record has related records

✓ you can't enter a value in the foreign key field of the related table that doesn't exist in the primary key of the primary table.

✓ However, you can enter a null value in the foreign key, specifying that the records are unrelated.

④ Key constraints

↳ Keys are attributes or set of attributes through which the tuples are identified.

↳ key attributes are identified from candidate keys, which in turn is defined from super keys.

(3) Logical database design

↳ After designing the conceptual Model of database using ER diagram, then we need to convert the conceptual Model into logical Model i.e Relational Model.

↳ Steps to convert ER Model to Relational Model (ER diagram) to Logical Model (Relational)

(1) Mapping of Regular Entity sets
(2) Mapping of Weak Entity set
(3) Mapping of Binary 1:1 Relationship types
(4) Mapping of Binary 1:N Relationship types
(5) Mapping of Binary M:N Relationship types
(6) Mapping of Multivalued attributes
(7) Mapping of N-ary Relationship types.
(8) Mapping EER Model to Relations.

(1) Mapping of Regular Entity sets

↳ For Each Regular (or) strong Entity set E in the ER diagram, create a Relation R that includes all the simple attributes of E.

↳ Choose one of the key attribute of E as the primary key for R

↳ if the choosen key of E is Composite, the set of simple attributes of the composite attribute form the primary key of R..

For Example, the ER diagram had 3 strong Entity sets like EMPLOYEE, DEPARTMENT and PROJECT. So Create three Relations as EMPLOYEE, DEPARTMENT & PROJECT

Make the key attributes of EMPLOYEE, DEPARTMENT and PROJECT as the primary key of the Relations EMPLOYEE, DEPARTMENT and PROJECT

EMPLOYEE

Step 1 Result

fname	Minit	Lname	SSN	Bdate	Address	gender	Salary

DEPARTMENT

Dnumber	Dname

PROJECT

Pnumber	Pname	Plocation

(2) Mapping of weak Entity set

↳ For each weak Entity set W in the ER diagram with owner Entity set E, create a Relation R and includes all simple attributes of W as attributes of R.

↳ Include primary key attribute of owner Entity set as foreign key of R

↳ The primary key of weak Entity set is combination of

primary key of owner Entity set + partial key of Weak ES

EMPLOYEE

fname	Minit	Lname	SSN	Bdate	Address	gender	Salary

DEPARTMENT

Dnumber	Dname

PROJECT

Pnumber	Pname	Plocation

DEPENDANT

Essn	Dependent_name	gender	Bdate	Relationship

③ Mapping of Binary 1:1 Relationships

↳ For each binary 1:1 Relationship set R, identify Entity sets S and T that participates in 1:1 Relationship set R.

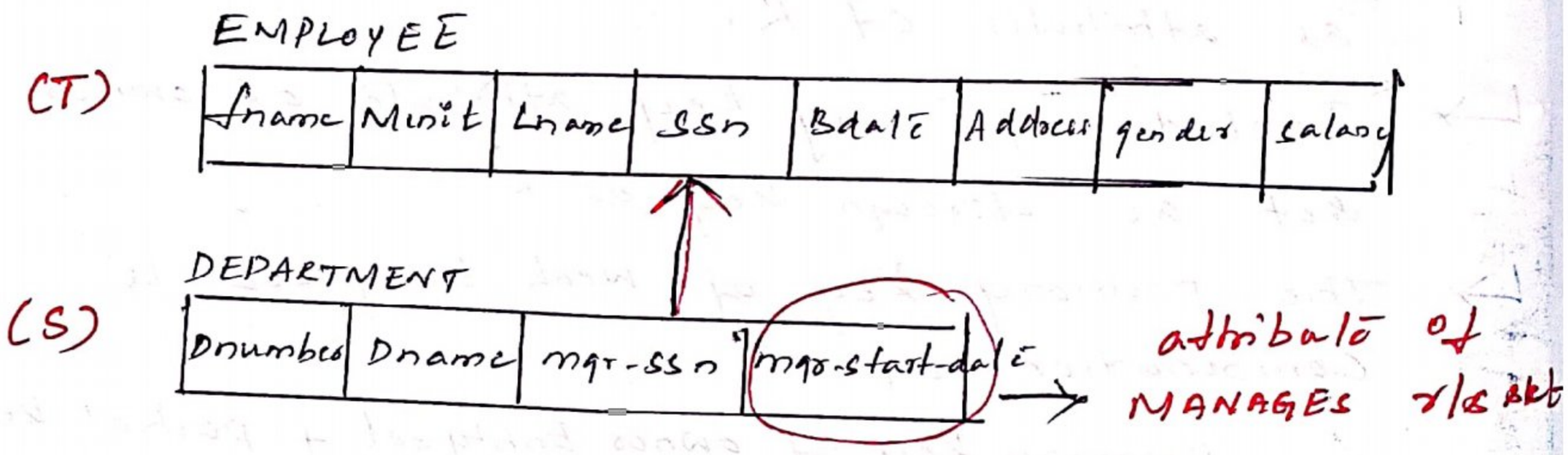
↳ Choose the total participation Entity set as S.

↳ Make primary key of T as foreign key of S

For Eg: MANAGES is an 1:1 Relationship set between EMPLOYEE and DEPARTMENT.

Since the participation of DEPARTMENT Entity set with MANAGES R/s is total consider it as S and EMPLOYEE Entity set as T

So make primary key of EMPLOYEE (SSN) as foreign key of DEPARTMENT (mgr-SSN)



↳ Include attributes of R/s set as attributes of S

④ Mapping of Binary 1:N (N:1) Relationships

↳ For Each binary 1:N or N:1 Relationship set R, identify N-side Entity set as S. and 1-side Entity set as T

↳ Make primary key of T as foreign key of S.

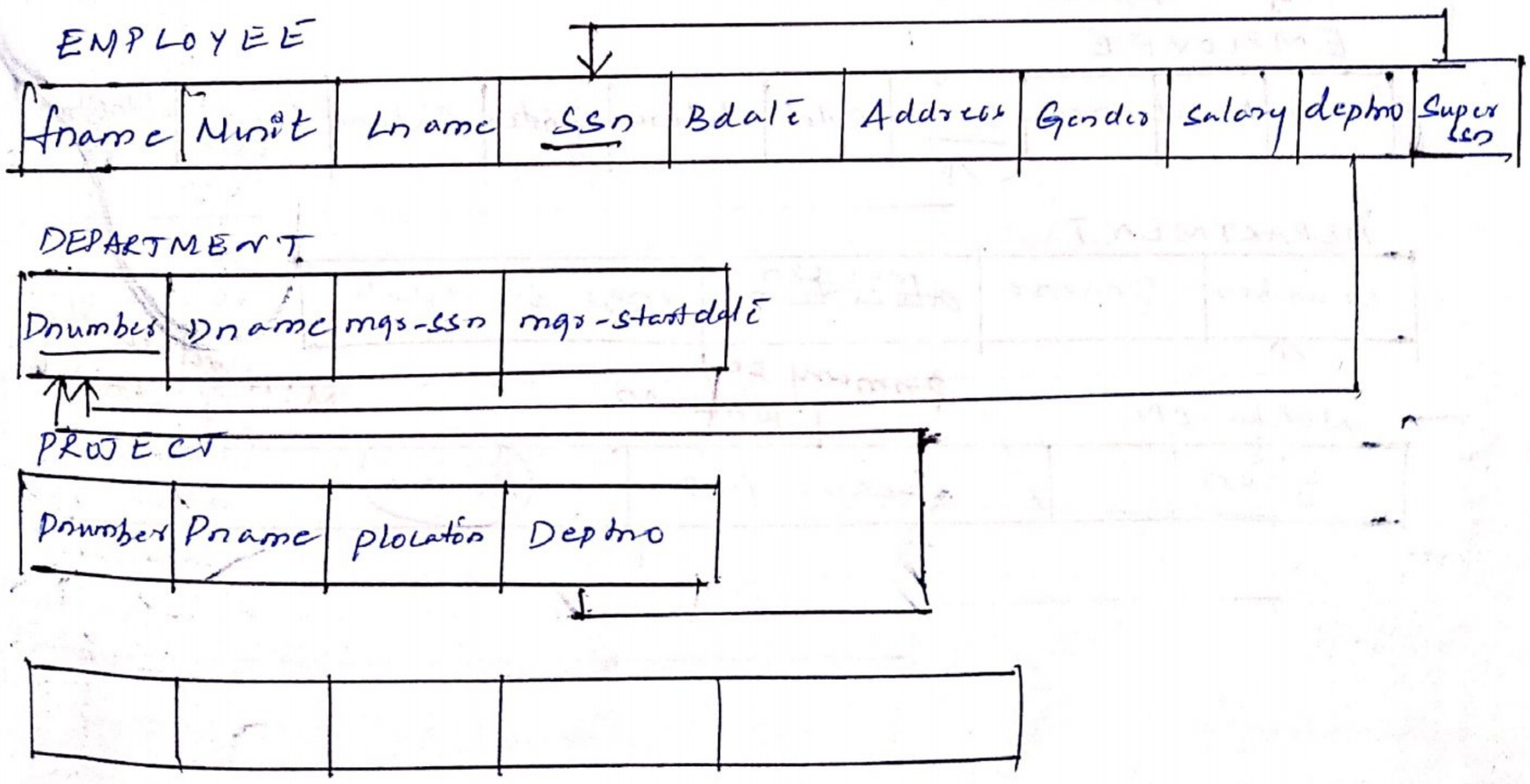
↳ Include any simple attributes of Relationship set R as attributes of S.

↳ In our Example, we have 3 1:N Relationships like WORKS_FOR, CONTROLS and SUPERVISION

↳ In WORKS_FOR Relationship set
EMPLOYEE is in N-side, so consider it as S
and DEPARTMENT is in 1-side, so consider it as T
So, make primary key of DEPARTMENT (DNUMBER) as foreign key of EMPLOYEE (dept no)

↳ In CONTROLS Relationship set
PROJECT is in N-side, so consider it as S and
DEPARTMENT is in 1-side, so consider it as T
So, include primary key of DEPARTMENT (DNUMBER) as foreign key of PROJECT (dept no)

↳ In SUPERVISION relationship set,
EMPLOYEE is in N-side, so consider it as S and
EMPLOYEE is in 1-side, so consider it as T
So, include the primary key of EMPLOYEE (ssn) as foreign key of EMPLOYEE and call it as Super-ssn



Step 5 : Mapping of Binary M:N Relationship set

↳ For Each Binary M:N Relationship set R, create a new Relation called A and

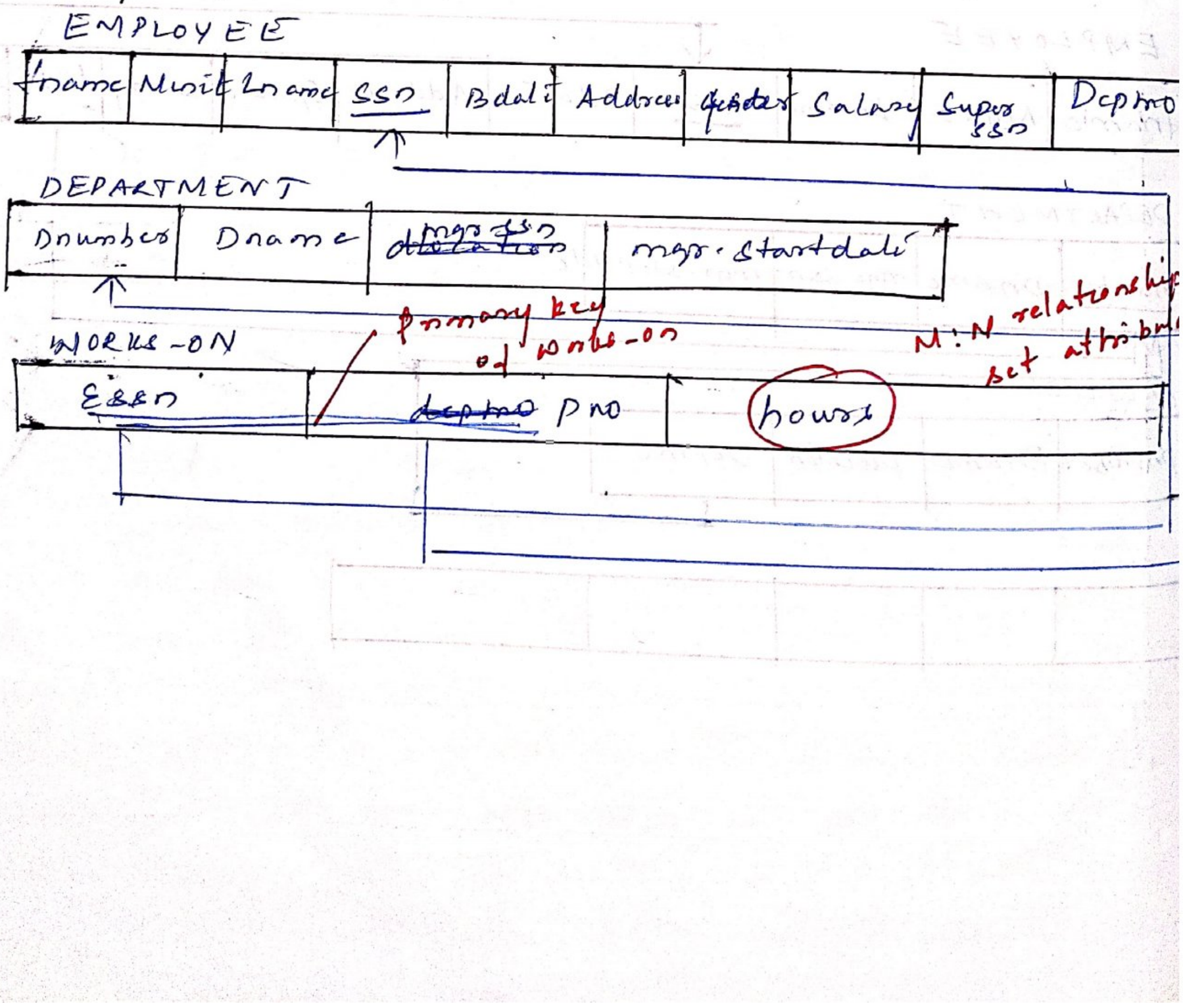
↳ include the primary keys of the Entity sets participating in R as foreign key of the new Relation A

↳ include attributes of M:N Relationship set R as attribute of A.

For Eg, In our Example WORKS-ON is an M:N Relationship set.

So create a Relation called WORKS-ON and make primary keys of both participating Entity sets (EMPLOYEE and PROJECT as foreign keys of WORKS-ON Relation

Include the Simple attributes of M:N Relationship set as attributes of WORKS-ON



Step 6 : Mapping of Multivalued attribute of an Entityset E

↳ For Each Multivalued attribute MA, Create a new Relation R

↳ Include the multivalued attribute as an attribute of R. and

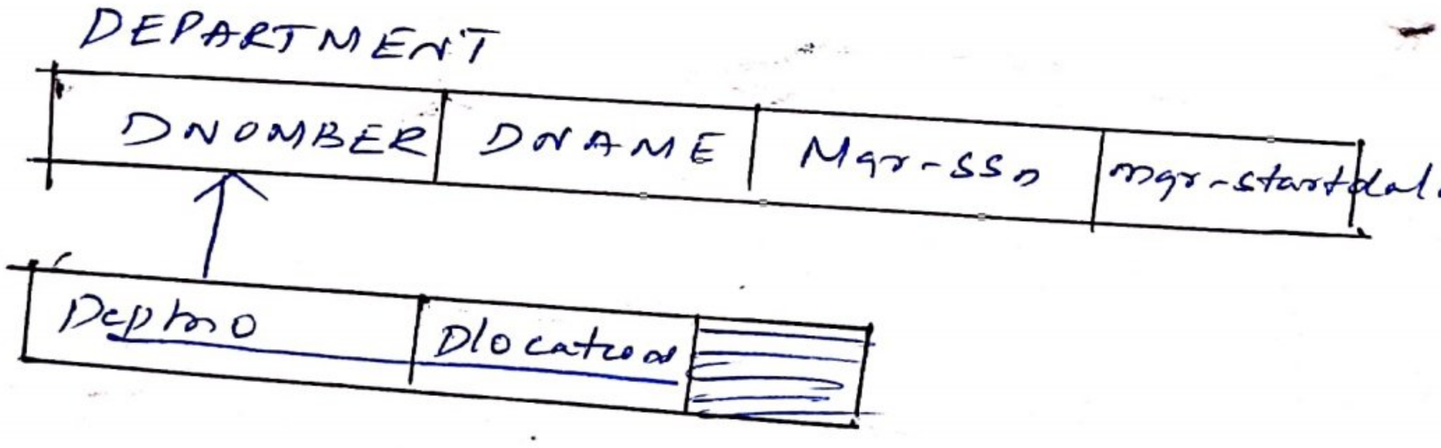
↳ Include the primary key of the Entityset E (that contains Multivalued attribute) as an attribute of R.

↳ The primary key of R is primary key of Entityset E and Multivalued attribute.

↳ DLocations is an Multivalued attribute of the Entity set DEPARTMENT so Create a Relation Called DEPT_LOCATIONS and

Include the Multivalued attribute (DLOCATION) as an attribute of DEPT_LOCATIONS and

Include primary key of DEPARTMENT as a foreign key of the new Relation

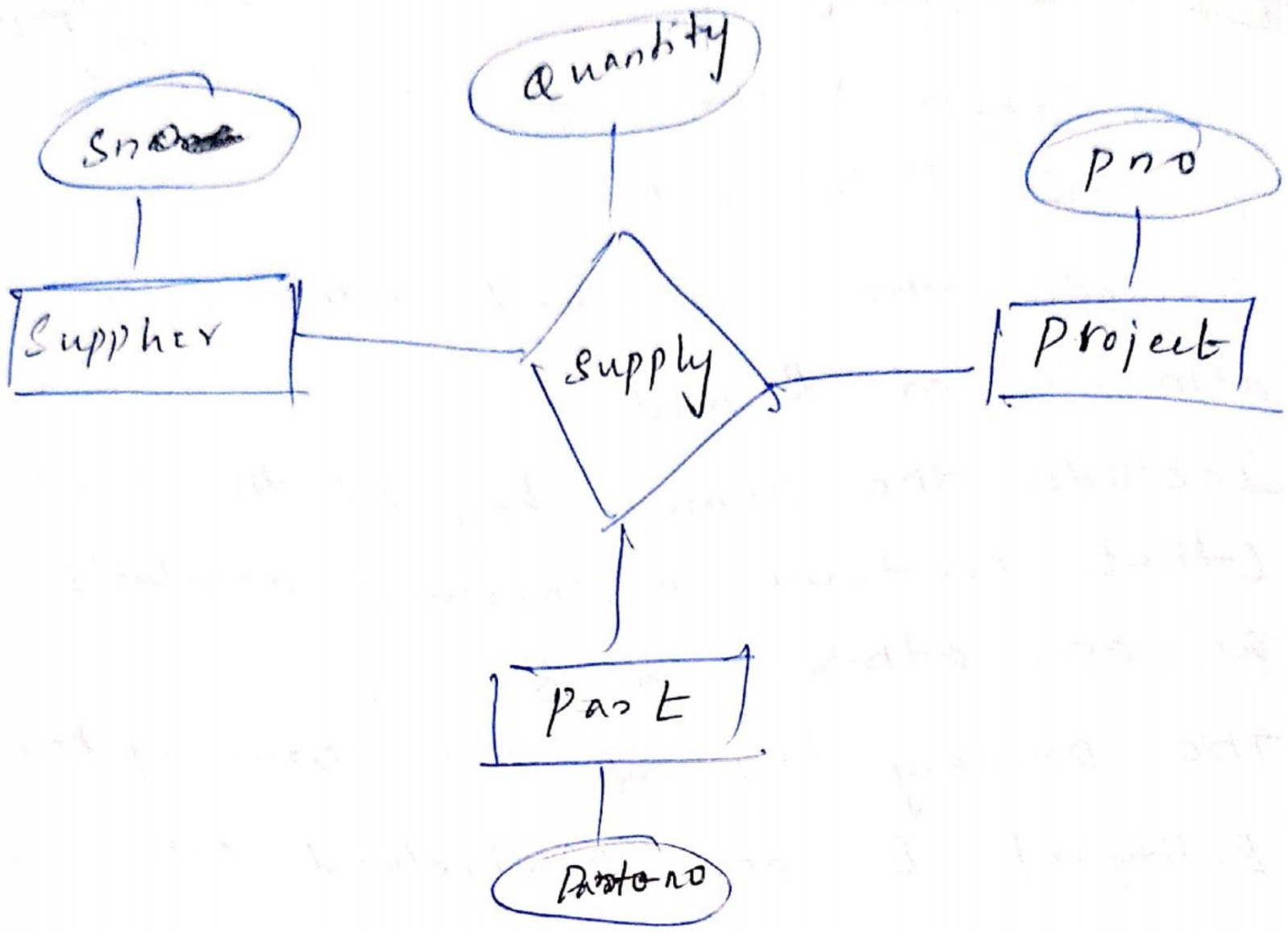


Step 7 : Mapping of N-ary Relationship set

↳ For Each n-ary relationship set R, where $n > 2$, Create a new Relation NR to represent the Relationship set R.

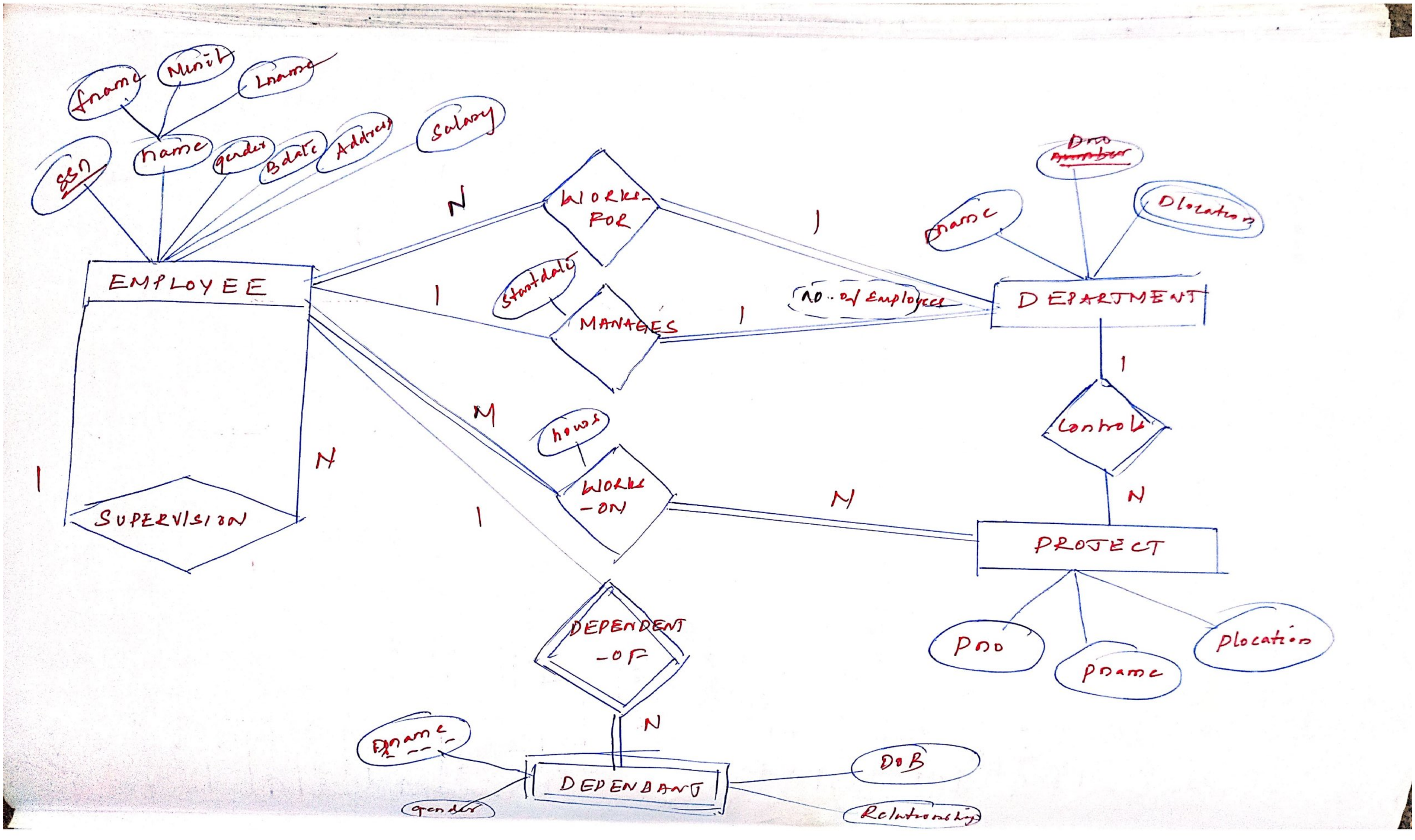
↳ Include the primary keys of all Entity sets participating in the n-ary relationship set as a foreign keys of new Relation NR.

↳ Include the Simple attributes of n-ary Relationship set as attributes of new Relation NR.



SUPPLY

Sno	part_no	pno	quantity


fname
Minit
Lname
name
SSN
gender
Bdate
Address
Salary
EMPLOYEE
WORKS-FOR
N
1
Startdate
MANAGES
1
1
DEPARTMENT
Dno
Number
Dname
Dlocation
No. of Employee
1
Controls
N
N
PROJECT
Pno
Pname
Plocation
WORKS-ON
N
1
hours
SUPERVISION
1
N
DEPENDENT-OF
N
Ename
DEPENDANT
gender
DoB
Relationship

EMPLOYEE

SSN	fname	minit	Lname	gender	Bdate	address	salary	Dno	Super-ssn

DEPARTMENT

Dno	Dname	mgr-ssn	mgr-startdate

PROJECT

Pno	pname	plocation	dno

DEPENDENT

Essn	DPname	gender	DOB	Relationship

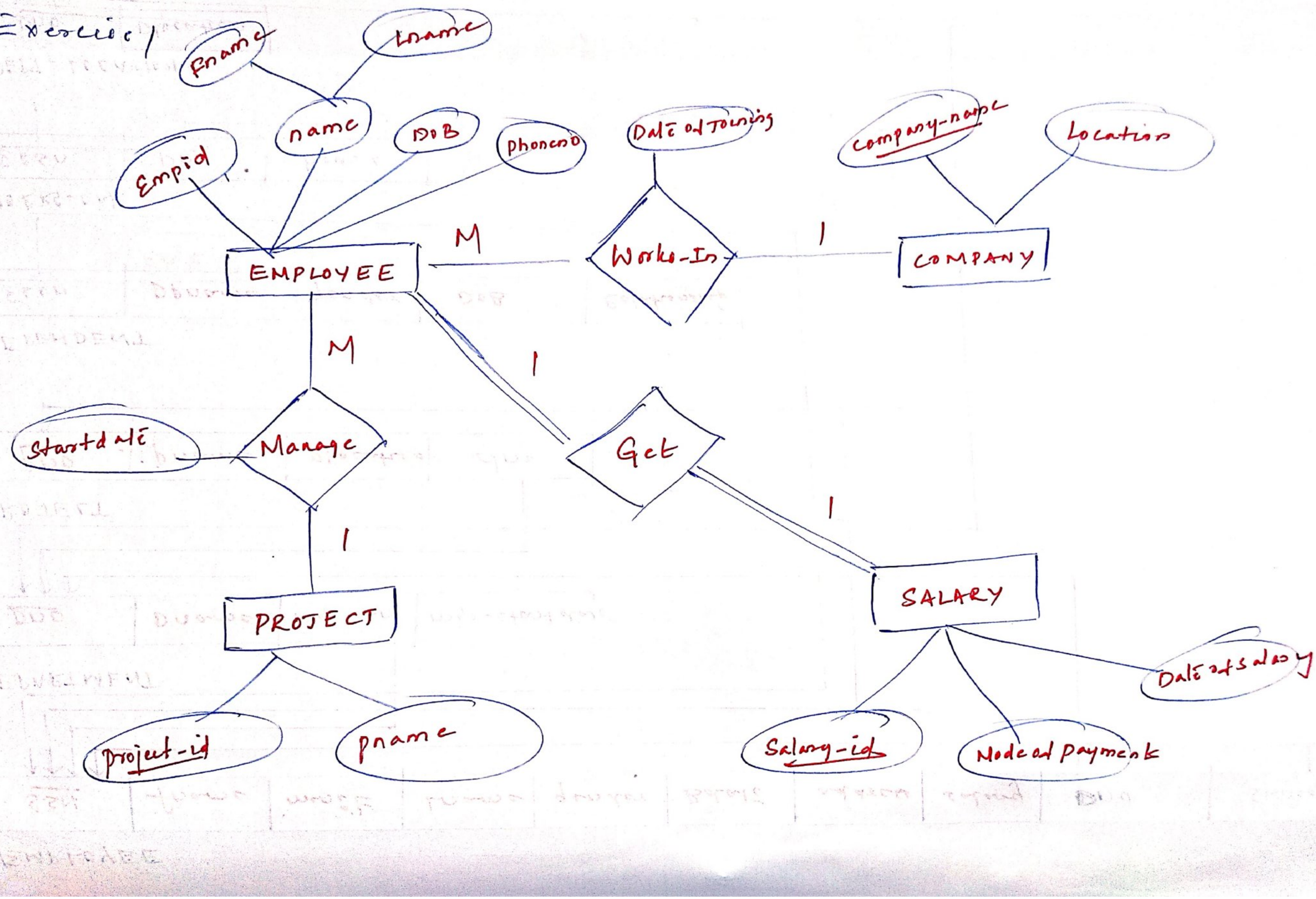
WORKS-ON

Essn	Pno	hours

DEPT_LOCATIONS

Dno	Dlocation

Exercise 1


Fname
Lname
name
Empid
DOB
Phoneno
Date of Joining
company-name
Location
EMPLOYEE
M
Works-In
1
COMPANY
M
1
Startdate
Manage
Get
1
1
PROJECT
SALARY
Project-id
pname
Salary-id
Medical payment
Date of salary

EMPLOYEE

Empid	fname	Lname	DOB	phoneno	project-id	company-name	Date of Joining	start date

PROJECT

Projectid	pname

SALARY

Salary-id	MOP	DOS	Emp-id

COMPANY

Company-name	Location

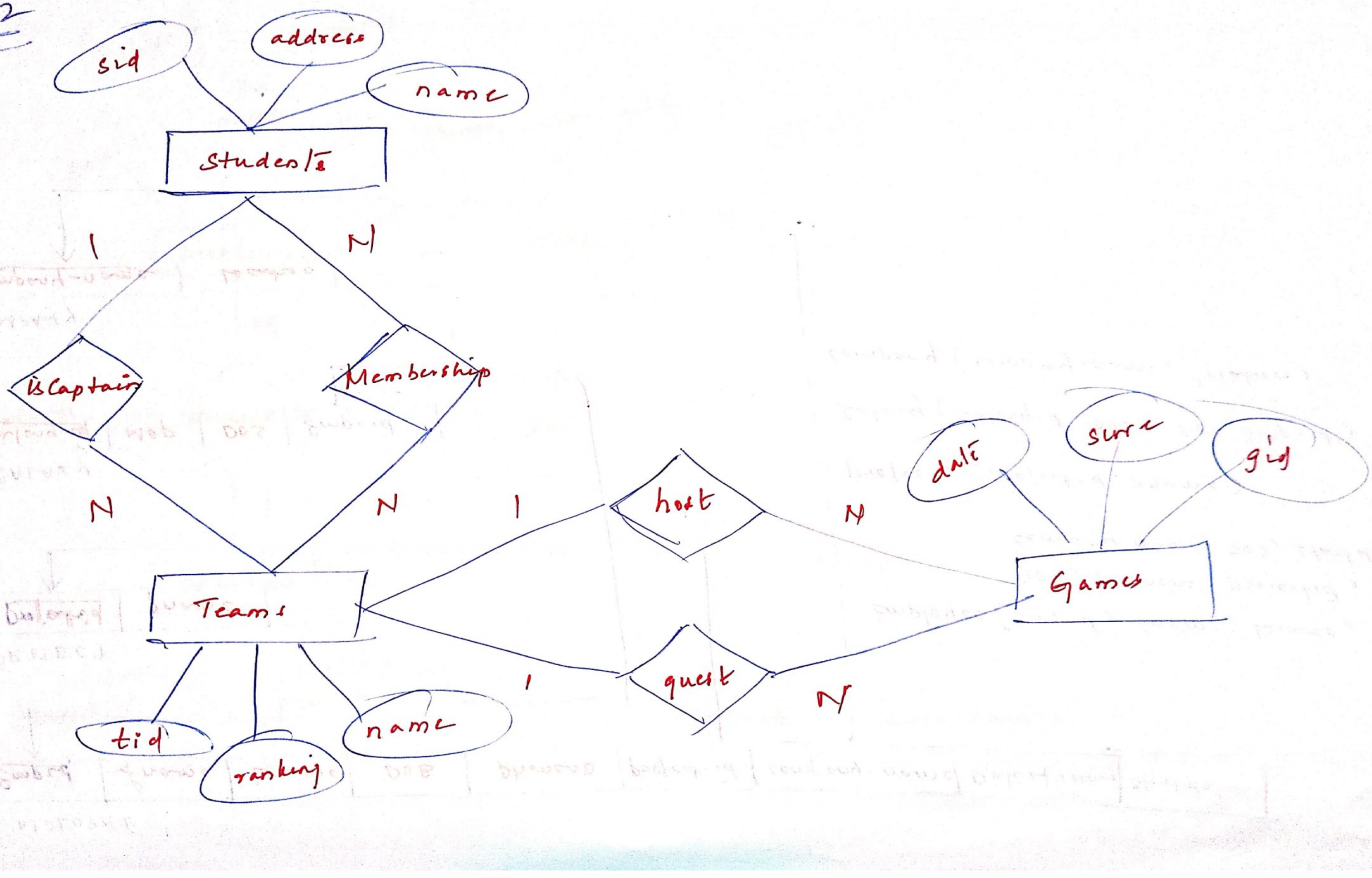
Employee (Empid, fname, Lname,
DOB, phoneno, projectid,
Company-name, DOJ, start date)

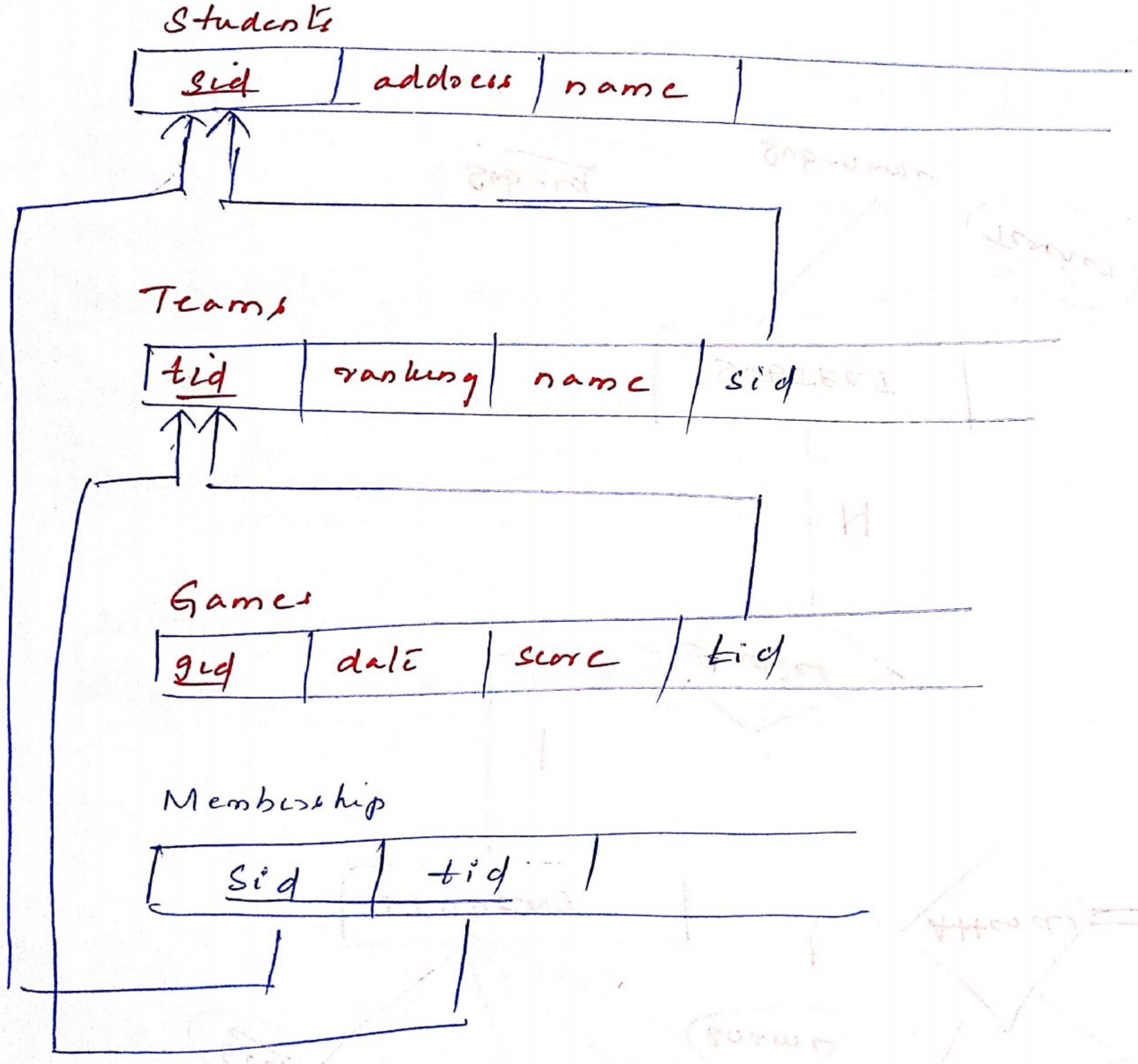
project (projectid, pname)

Salary (salaryid, MOP, DOS, Emp-id)

Company (company-name, location)

Ex No: 2


sid
address
name
Student
1
M
isCaptain
Membership
N
N
1
host
N
date
score
gid
Games
Teams
1
quest
N
tid
ranking
name



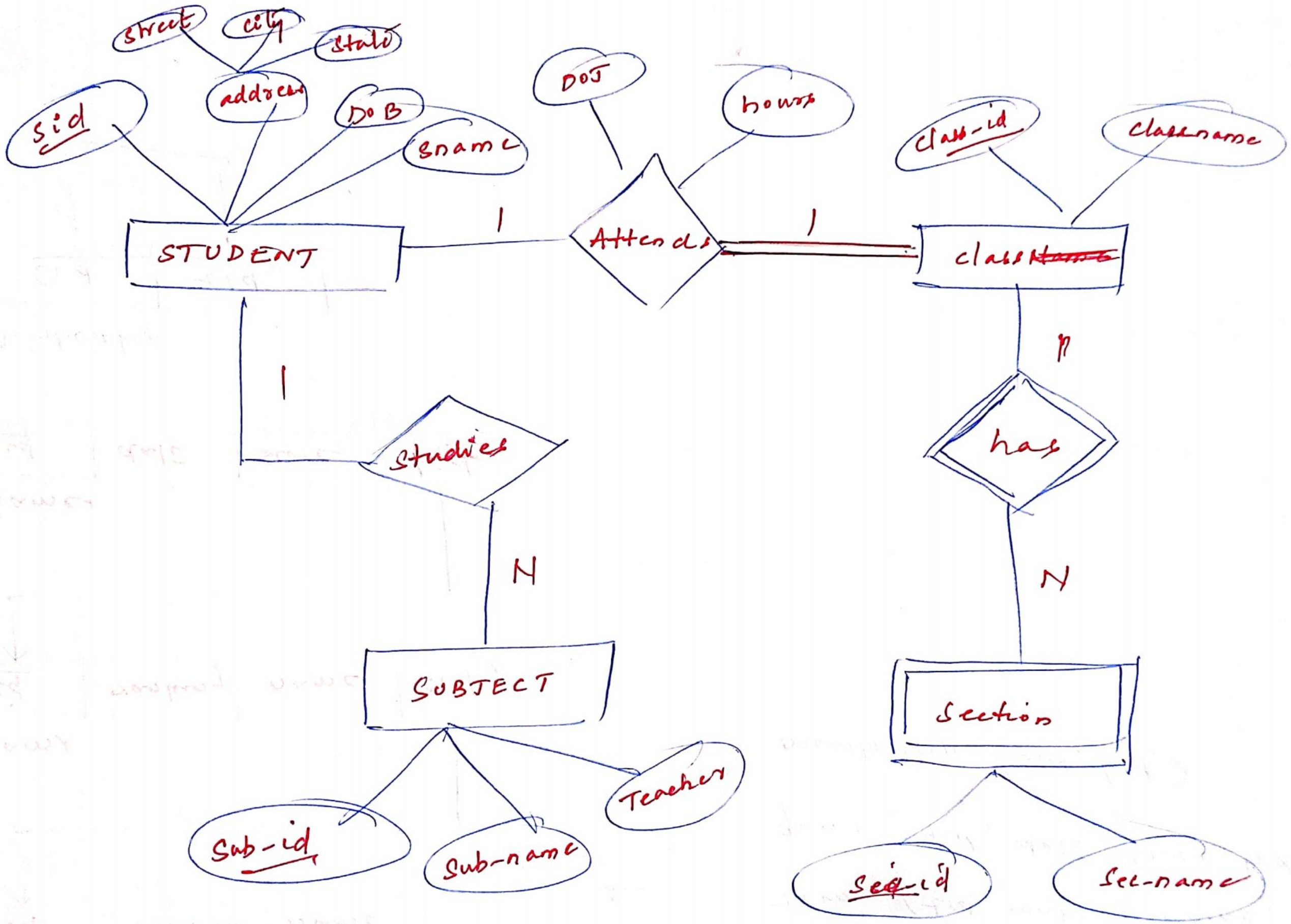
Students (sid, address, name)

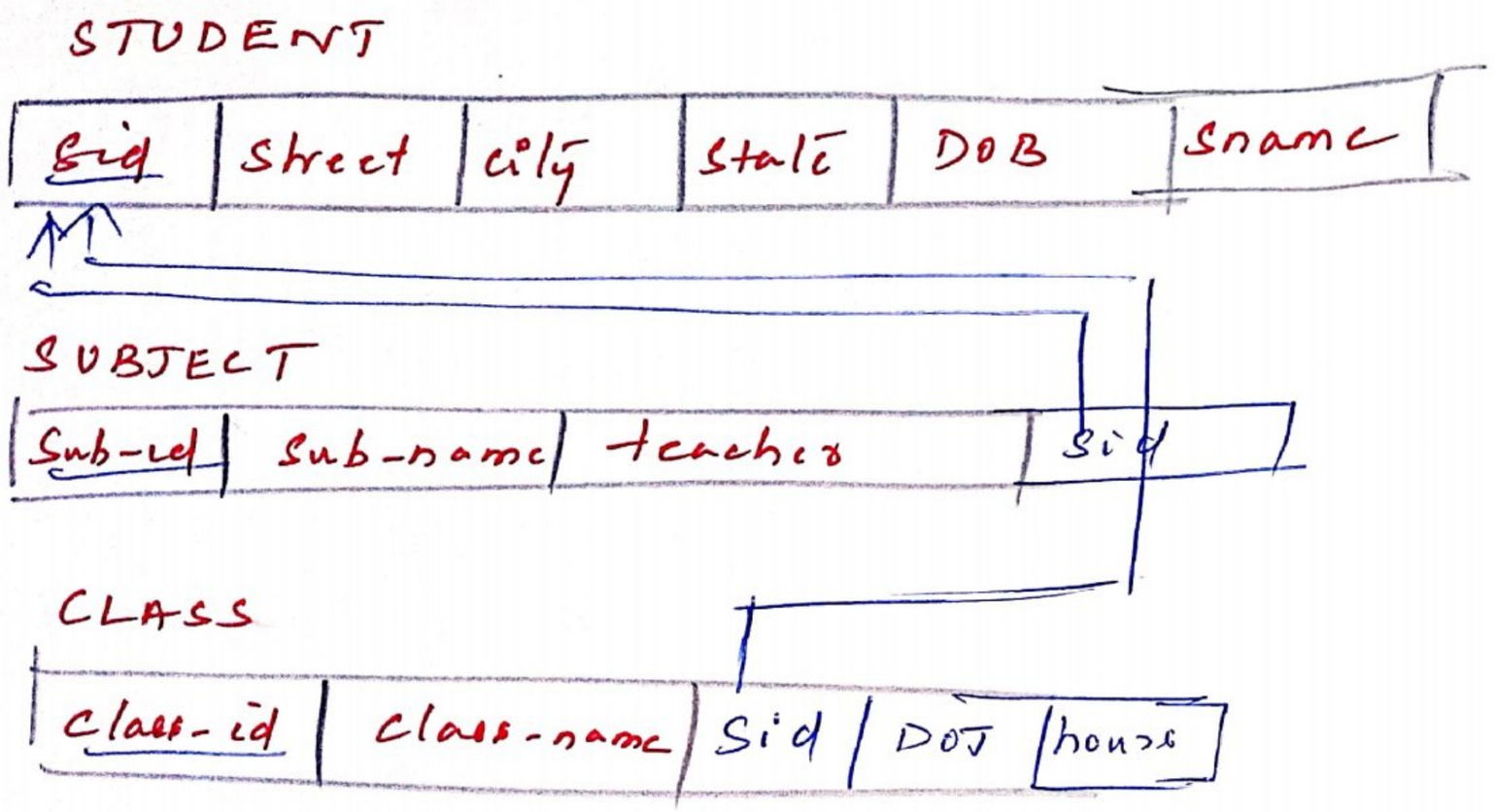
Teams (tid, ranking, name, sid)

games (gid, date, score, tid)

membership (sid, tid)

Ex No :3


street
city
state
address
sid
DoB
sname
DOJ
hours
Attends
STUDENT
1
1
class-id
classname
classHours
1
N
Studies
has
N
SUBJECT
Section
Teacher
Sub-id
Sub-name
Sec-id
Sec-name



Student (Sid, Sname, DOB, street, city, state)

Subject (Sub-id, Sub-name, teacher, Sid)

class (class-id, class-name, Sid, DOJ, hours)

Section (class-id, Sec-id, Sec-name)

SECTION

class-id	Sec-id	Sec-name

Relational Algebra

↳ Relational Algebra is a Query language

↳ Query language is a Language in which user requests information from the database

SQL is a Query Language.

↳ Two types of Query Languages are

(1) Procedural Query Language
(2) Non Procedural Query Language.

(1) Procedural Query Language

↳ In Procedural Query Language, user instructs the system to perform a series of operations to produce the desired results.

↳ user tells what data to be retrieved from database and how to retrieve it

(2) Non Procedural or Declarative Query Language

↳ In Non Procedural query language, user instructs the system to produce the desired results without telling the step by step process.

↳ user tells what data to be retrieved from db but doesn't tell how to retrieve it

Example: if we ask our daughter or son to bring a cup of tea then it comes under Non procedural Query lang.
if we ask them to get a cup of tea and also tell them step by step procedure to make a tea, then it comes under procedural Query language

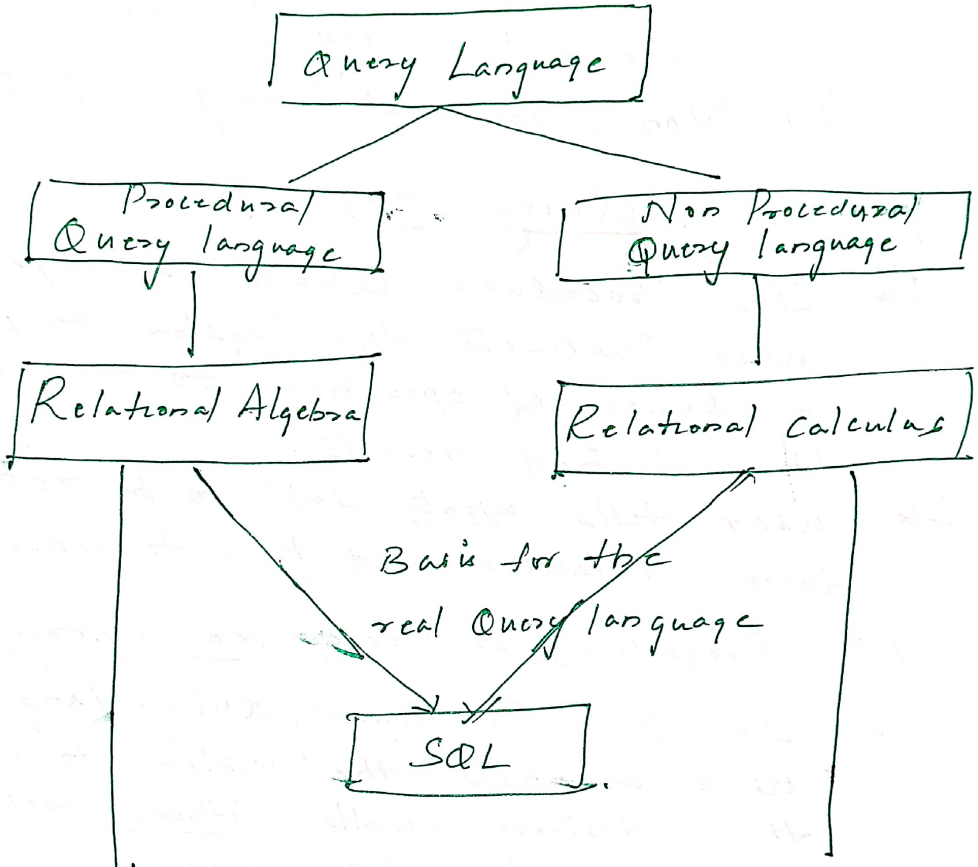
↳ Two pure or basic Query languages are

① Relational Algebra (Procedural Query lang)

② Relational Calculus (Non Procedural ")

2.1) Tuple Relational Calculus

2.2) Domain Relational calculus



Query lang where we specify what data to be retrieved and how to retrieve it

Query lang where we specify what data to be retrieved and not how to retrieve it.

Relational Algebra

↳ It is a procedural Query language, in which the user should instruct the system a series of operations to get the desired result.

↳ Here, the user must tell *what* *data* to be retrieved and *how* to retrieve it.

↳ It takes a *Relation* as an *input* and generates a *relation* as an *output*

↳ It uses *operators* to perform queries

↳ An operator can be *unary* or *Binary*

Basic operations	Additional operations
1. Selection (σ) (operator / operation)	1. Natural Join ($\bowtie$)
2. projection (Π)	2. Left, Right, Full outer Join
3. union ($\cup$)	3. set Intersection ($\cap$)
4. set Difference ($-$)	4. Division ($\div$)
5. Cartesian product ($\times$)	5. Assignment ($\leftarrow$)
6. rename (ρ)	
σ, Π, ρ are unary operators because they operate on one Relation	$\bowtie$, $\cap$, $\div$ and $\leftarrow$ are Binary operators because they operate on two Relations.
$\cup$, $-$, $\times$ are binary operators because they operate on two Relations.	

Select operator (σ)

↳ Select operator (σ) is a unary operator in Relational algebra that performs a selection operation.

↳ It selects tuples that satisfy the given condition from a relation.

↳ It is denoted by Sigma (σ)

↳ Notation

$$\sigma_{p}(r) \quad (\text{or}) \quad \sigma_{(\text{condition})}(\text{RelationName})$$

↳ WHERE clause of a SQL command corresponds to relational select operator (σ)

↳ Let us take the three - Relations sailors, Reserves and boats.

Sailors (sid: integer, Sname: string, rating: integer, age: real);

Boats (bid: integer, bname: string, color: string)

Reserves (sid: integer, bid: integer, day: date)

Sailors

Sid	Sname	Rating	Age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35
64	Horatio	7	35
71	zorba	10	16
74	Horatio	9	40
85	Art	3	25.5
95	bob	3	63.5

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	clipper	green
104	Marine	red

Reserves

sid	bid	day
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-8
22	104	1998-10-7
31	102	1998-11-10
31	103	1998-11-6
31	104	1998-11-12
64	101	1998-9-5
64	102	1998-9-8
74	103	1998-9-8

Ex Eg: To select sailors with rating 7, we write in Relational Algebra as

$$\sigma_{rating=7}(Sailors) \quad \text{— RA}$$

O/P

Schema is that same as that of I/P Relation

sid	sname	rating	age
22	Dustin	7	45
64	Horatio	7	35

SQL Equivalent query is

SQL

```
Select * from sailors where rating = 7
```

In Selection operation, schema of Resulting relation is identical to schema of input relation

① Display sailors with age > 35

$$\sigma_{age>35}(Sailors)$$ — Relational Algebra Query

```
Select * from Sailors where age>35;
```
— SQL query

As we said the where clause in SQL command corresponds to relational Algebra Select operator.

② Display all red boats

$$\sigma_{color='red'}(boats)$$ — RA

```
Select * from boats where color='red';
```
— SQL

③ Display ~~sailors~~ boats reserved for sailors with id 22

$$\sigma_{sid=22}(reserves)$$ — RA

```
Select * from reserves where sid=22;
```
— SQL

④ Display reservation details of boats with ~~name~~ id ~~clipper~~ 104

$$\sigma_{bid=104}(reserves)$$

```
Select * from reserves where bid = 104;
```

(5) Display Sailors with rating 7 and age > 35

$$\sigma_{(rating = 7) \wedge (age > 35)}(Sailors)$$

```
Select * from sailors where rating = 7 and
                            age > 35
```

$$\sigma_{(rating = 7 \wedge age > 35)}(Sailors)$$

(6) Display boats that are reserved by Sailor id 64 and 74

$$\sigma_{Sid = 64 \vee Sid = 74}(reserved)$$

```
Select * from reserved where Sid = 64 or
                             Sid = 74;
```

(7) Display details of boats reserved with id 103 and 104

$$\sigma_{bid = 103 \vee bid = 104}(reserved)$$

Project operator (π)

↳ π is a unary operator in relational algebra that performs a projection operation

↳ It projects (or displays) particular columns of a Relation.

↳ Notation

$$\pi_{A1, A2, \ldots, An}(r) \text{ or } \pi_{Attribute\text{-}List}(RelationName)$$

↳ Duplicate Rows are automatically [illegible] from Result.

(1) Display only sailor names

Π_{sname} (sailors)

Equivalent SQL Query is

```
select sname from sailors;
```

Note: The SQL SELECT command corresponds to Relational Project(Π)

(2) Display sailors ages from sailors relation

Π_{age} (sailors)

O/P

age
45
33
55.5
25.5
35
16
40
63.5

Duplicate rows (ages) are automatically removed from result.

(3) Display all sailors ~~with~~ names with their ratings

$\Pi_{sname, ratings}$ (sailors)

O/P

Sname	ratings
Dustin	7
Brutus	1
- -	- -
- -	- -
bob	3

Equivalent SQL Query

```
Select sname, ratings
from sailors;
```

(4) Display sailor names with ratings whose rating is >= 7

$$\Pi_{sname, rating}\left(\sigma_{rating >= 7}(sailors)\right)$$

In SQL,

```
Select sname, rating from sailors
where rating >= 7
```

O/p

sname	rating
Dustin	7
Lubber	8
Andy	8
Rusty	10
Horatio	7
Zorba	10
Horatio	9

(5) Display boatnames with its colors

$$\Pi_{bname, color}(boats)$$

(6) Display boatnames with its colors whose color is red

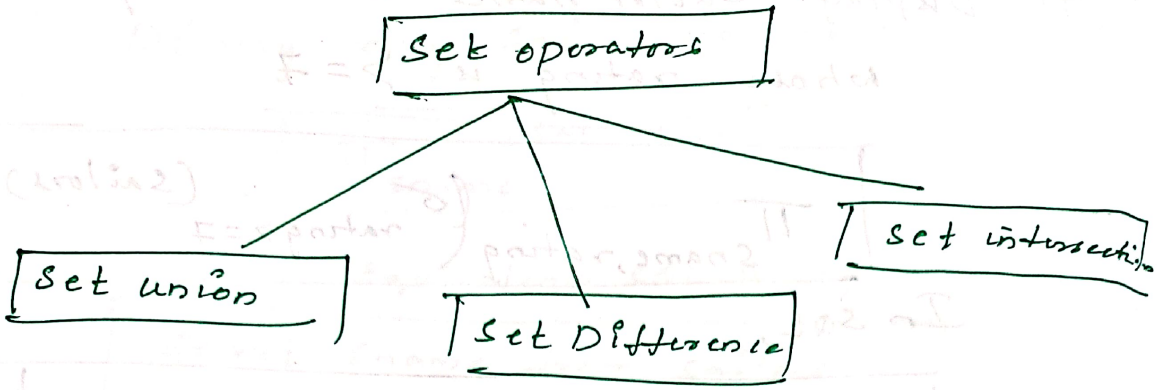
$$\Pi_{bname, color}\left(\sigma_{color = "red"}(boats)\right)$$

In SQL

```
Select bname, color from boats
where color = "red";
```

O/p

bname	color
Interlake	red
Marine	red



↳ union, difference and intersection are called set operators

↳ They are binary operators as they take two input Relations.

↳ TO use set operators on two Relations
- The two Relations must be compatible.

↳ Two Relations are compatible if

(1) Both the Relations must have same number of attributes

(2) Corresponding attribute must have same domain (or type)

↳ Duplicate tuples are automatically Eliminated.

Set union or union operator (∪)

↳ Suppose R and S are two Relations. The union operation Selects all the tuples that are in either R or S or in both R & S.

↳ It Eliminates the duplicate tuples

↳ To apply union, difference and intersection operator, the following conditions must hold

① Both the Relations R and S must have same number of attributes

② The attributes of R and S must occur in the same order

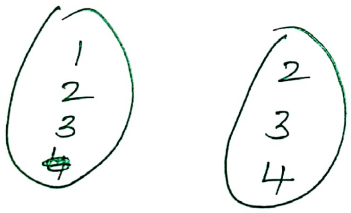
Set difference

↳ Suppose R and S are two Relations. The set difference operation selects all the tuples that are present in first Relation R but not in second Relation S.

Set intersection

↳ Suppose R and S are two Relations. The set intersection operation selects all the tuples that are in both relations R and S.

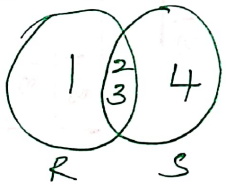
Example :



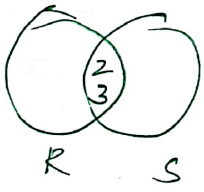
R S

$S - R = \{4\}$

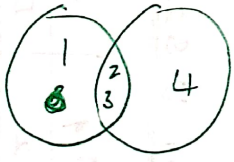
$R \cup S = \{1,2,3,4\}$



$R \cap S = \{2,3\}$



$R - S = \{1\}$



Sid	Sname	Rating	age
22	Dustin	7	45
31	Lubber	8	55
58	Rusty	10	35

S1 – instance of Sailors

Sid	Sname	Rating	age
28	Yuppy	9	35
31	Lubber	8	55
44	guppy	5	35
58	Rusty	10	35

S2 – instance of Sailors.

S1 ∪ S2

Sid	Sname	Rating	age
22	Dustin	7	45
28	Yuppy	9	35
31	Lubber	8	55
44	guppy	5	35
58	Rusty	10	35

S1 ∩ S2

Sid	Sname	Rating	age
31	Lubber	8	55
58	Rusty	10	35

S1 – S2

Sid	Sname	Rating	age
22	Dustin	7	45

S2 – S1

Sid	Sname	Rating	age
28	Yuppy	9	35
44	guppy	5	35

$\Pi_{sname}(S1) \cup \Pi_{sname}(S2)$

sname
Dustin
Yuppy
Lubber
guppy
Rusty

$\Pi_{sname}(S1) \cap \Pi_{sname}(S2)$

sname
~~Dustin~~ Lubber
Rusty

$\Pi_{sname}(S1) - \Pi_{sname}(S2)$

sname
Dustin

$\Pi_{sname}(S2) - \Pi_{sname}(S1)$

sname
yuppy
guppy

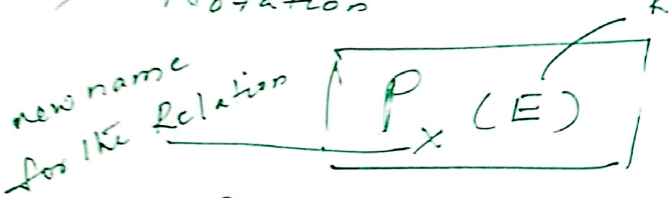
Rename operator

↳ The Results of Relational Algebra operators are also Relations but without any name.

↳ The Rename operator is used to rename the output of a Relation

↳ We may want to save the results of a relational algebra Expression as a relation so that we can use it later.

↳ Notation

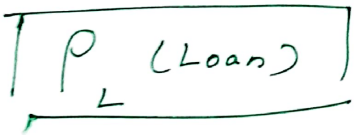


where

ρ is the Rename operator

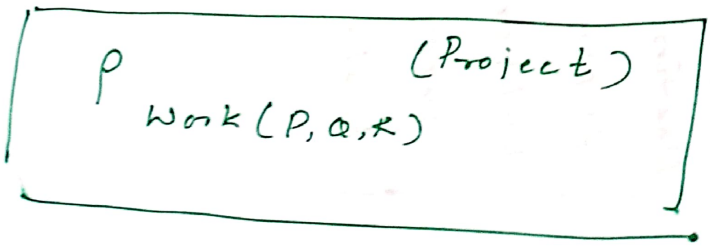
E is the Result of sequence of operations which is saved with the name X

↳ Query to rename a Relation name Loan to L



— Notation 1 Example

↳ Query to rename the Relation name Project to work and its attributes to P, Q, R



— Notation 2 Example

Three Notations are there

(1) $\rho_x(E)$ [To Rename only Relation Name]

To Rename Relation name E as X

(2) $\rho_{X(A1, A2, .., An)}(E)$ [To Rename both Relation & Attribute names]

To Rename Relation name E as X and attributes of X will be A1, A2, .., An

(3) $\rho_{(A1, A2, .., An)}(E)$ [To Rename only attribute names of the Relation]

To Rename the attribute names of the Relation E

↳ Query to rename attribute names of the relation Student as a, b (sno, sname)

$$\rho_{(a,b)}(Student)$$

Rename operator

↳ The Results of Relational Algebra operations are also relations but without any name.

↳ The Rename operator is used to rename the output of a Relation.

↳ Three Notation of Rename operator

① $\rho_x(E)$ → To Rename only Relation Name alone

ρ - Rename operator

E - Existing Relation Name

X - New Relation Name.

Eg: $\rho_L(Loan)$

Loan → Renamed → L

Lid	Cname
L001	Smith
L002	Jones

Lid	cname
L001	Smith
L002	Jones

② $\rho_{X(A1, A2, \ldots, An)}(E)$ → To Rename both Relation name and attribute names of existing Relation

Eg: $\rho_{L(Loanid, cust\_name)}(Loan)$

Loan

Lid	Cname
L001	Smith
L002	Jones

Both Relation & attribute names are renamed → L

Loanid	cust_name
L001	Smith
L002	Jones

(3) $\rho_{(A1, A2, \ldots, An)}(E)$ → To Rename only attribute names of the Existing Relation

Eg: $\rho_{(a,b)}(Loan)$

Loan

Lid	cname
L009	smith
L002	Jones

attributes alone renamed →

Loan

a	b
L009	smith
L002	Jones

Cartesian Product (or) Cross product

↳ Cartesian Product combines information of two different relations into one.

↳ Generally, a cartesian product is never a meaningful operation when it is performed alone. However, it becomes meaningful when it is followed by select operator.

↳ Notation

$R1 \times R2$

Example

Sid	Sname	Rating	age
22	Dustin	7	45
31	Lubber	8	55
58	Rusty	10	35

S1 - instance of Sailors

Sid	bid	day
22	101	10/10/96
58	103	11/12/96

R1 - instance of reserves

Sid	Sname	Rating	age	Sid	bid	day
22	Dustin	7	45	22	101	10/10/96
22	Dustin	7	45	58	103	11/12/96
31	Lubber	8	55	22	101	10/10/96
31	Lubber	8	55	58	103	11/12/96
58	Rusty	10	35	22	101	10/10/96
58	Rusty	10	35	58	103	11/12/96

→ S1 × R1

↳ if Relation R1 has m attributes and Relation R2 has n attributes, then the resultant Relation will have m+n attributes

↳ In our Example S1 × R1, S1 has 4 attributes and R1 has 3 attributes, so the resultant relation has 4+3 = 7 attributes.

↳ if Relation R1 has n1 tuples and Relation R2 has n2 tuples, then the resultant relation will have n1×n2 tuples

in our Example S1×R1, S1 has 3 tuples and R1 has 2 tuples, so the resultant relation has 3×2 = 6 tuples

↳ As we said the resultant relation of Cartesian product is not a meaningful one.

To make it meaningful, we need to apply select operator

$$\sigma_{S1.sid = R1.sid}(S1 \times R1)$$

→ This will give sailors who have reserved boats

o/p

sid	sname	Rating	age	sid	bid	day
22	Dustin	7	45	22	101	10/10/96
58	Rusty	10	35	58	103	11/12/96

$$\Pi_{sname}\left(\sigma_{S1.sid = R1.sid}(S1 \times R1)\right)$$

→ Sailor names alone who have reserved boats

o/p

sname
Dustin
Rusty

Division operator

↳ Division operator is suited to queries that include the keyword "all" or "every".

For eg,

1) Find the person who has account in all the banks of a particular city
2) Find sailors who have reserved all boats
3) Find students who have registered for Every course.

↳ Notation

$$A \div B \quad \text{or} \quad A/B$$

↳ Division operator can be applied if and only if attributes of B is proper subset of attributes of A.

↳ The Relation returned by division operator will have

attributes = (All attributes of A) − (all Attributes of B)

↳ The Relation returned by division operator will return those tuples from relation A which are associated to Every B's tuple

Eq:

A

x	y
a	1
b	2
a	2
b	4

÷

B

y
1
2

⇒

A ÷ B

x
a

No. of att
A − B : (x, y) − x
Sx − y = 1

Every tuple of B is associated with only a
So, the resultant relation contains a

Eg 2:

A

SNO	PNO
S4	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4

B1

PNO
P2

B2

PNO
P2
P4

B3

PNO
P1
P2
P4

A | B1

SNO
S1
S2
S3
S4

↓

S1, S2, S3, S4 are associated with P2

A | B2

SNO
S1
S4

↓

only S1 & S4 are associated with bolts P2 & P4

A | B3

SNO
S1

↓

only S1 is associated with all tuples of B2.

TO find all customers who have an account at all branches located 'brooklyn' city

R

branch-name
Brighton
Downtown

↓

branches in brooklyn city

S

Customer name	branch name
Hayes	Perryridge
Johnson	Downtown
Johnson	Brighton
Jones	Brighton
Lindsay	Redwood
Smith	Mianus
Turner	Round Hill

S ÷ R

Customer name
Johnson

↓

customers having account at different branches

grey tuple of R is associated with only Johnson

JOIN operator

↳ Join is a combination of a cartesian product followed by a select operator

join = cartesian product + selection

↳ A Join operation pairs two tuples from different relations, if and only if a given join condition is satisfied.

↳ Notation

$A \bowtie_c B$ where

A and B are relations

$\bowtie$ is a join operator &

c is the join condition.

↳ $A \bowtie_c B \equiv \sigma_c (A \times B)$

Types of Joins

- Inner Join
  - ↳ Theta Join
  - ↳ Equi Join
  - ↳ Natural Join
- Outer Join
  - ↳ Left outer Join
  - ↳ Right outer Join
  - ↳ Full outer Join

Inner Join

↳ Contains only those tuples that satisfy the matching condition

(1) Theta (Θ) Join

↳ $A \bowtie_c B$

↳ all types of comparison operators can be used in the condition c

(<, >, <=, >=, =, ≠)

(2) ~~Equi Join~~

↳ Example:

S1

Sid	Sname	Rating	age
22	dustin	7	45
31	Lubber	8	55
58	rusty	10	35

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

$S1 \bowtie_{S1.sid < R1.sid} R1$

o/p

S1.sid	Sname	rating	age	R1.sid	bid	day
22	dustin	7	45	58	103	11/12/96
31	lubber	8	55	58	103	11/12/19

(2) Equi Join

↳ special case of theta Join

↳ uses only Equality (=) comparison operator

(or)

↳ Equi Join is a special case of theta Join where condition contains Equalities (=)

↳ Notation

$A \bowtie_c B$

Condition c should contain only Equal to operator

For the same above Example

$$S1 \bowtie_{s.sid = r.sid} r1$$

↳ sailors who have reserved boats.

O/P

s1.sid	sname	rating	age	r1.sid	bid	day
22	dustin	7	45	22	101	10/10/96
58	rusty	10	35	58	103	11/12/96

(3) Natural Join

↳ Natural Join can only be performed if there is atleast one common attribute that exists between two relations.

In addition, the attributes must have the same name and domain.

↳ Natural Join does not use any comparison operator

↳ It is same as equi join which occurs implicitly by comparing all the common attributes in both relation, but difference is that in Natural Join the common attributes appears only once.

↳ The Result of Natural Join is that set of all combinations of tuples in two relations A and B that are equal on their common attribute name.

↳ Notation

$$A \bowtie B$$

↳ Natural Join = Cartesian Product + selection + projection.

For the same Relations

$S1 \bowtie R1$

gives same o/p as Equi join Except the common attribute appears once

Sid	Sname	rating	age	bid	day
22	Dustin	7	45	101	10/10/96
58	Rusty	10	35	103	11/12/96.

Inner Join

- Theta Join → $A \bowtie_c B$ → where c is condition which can have any comparision operator like <, >, <=, >=, =, ≠
  - Eg: $\sigma_{S1.sid < R1.sid}(S1 \times R1)$
  - operator used is less than
- Equi Join → $A \bowtie_c B$ → c is a condition which should have only one operator called Equal to
  - $\sigma_{S1.sid = R1.sid}(S1 \times R1)$
  - operator used is Equal to
- Natural Join → $A \bowtie B$ → no need to specify the condition. returns matched tuples based on common attribute
  - $R1 \bowtie S1$
  - no need to specify the condition

OUTER JOIN

↳ contains matching tuples that satisfy the matching condition, along with same or all tuples that do not satisfy the matching condition

↳

Outer Join

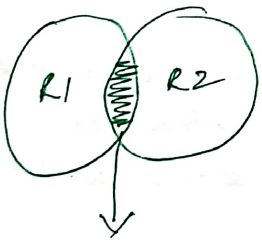
- Left outer Join
  - ↓ Left relation tuples will always be in result whether the value is matched or not
- Right outer Join
  - ↓ Right relation tuples will always be in result whether the value is matched or not
- Full outer Join
  - ↓ Tuples from both relations are present in result, whether the value is matched or not

Left outer Join (R1 ⟕ R2)

↳ Records that have matching in both R1 & R2

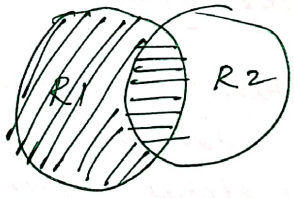
↳ and Records in left Relation that have no matching in Right relation R2

R1 ⋈ R2



Natural Join

R1 ⟕ R2



Left outer Join

Courses

cid	course
100	Database
101	Mechanics
102	Electronics

faculty

cid	fname
100	Rohan
102	Sara
104	Jiya

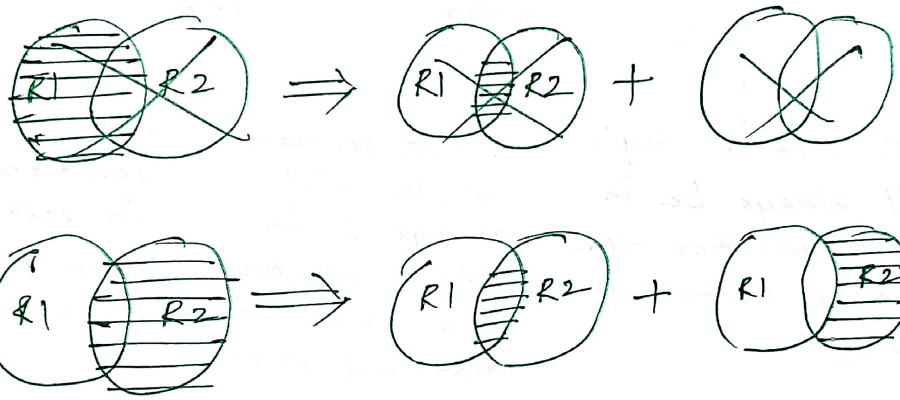
courses ⟕ faculty

cid	course	fname
100	Database	Rohan
101	Mechanics	NULL
102	Electronics	Sara

Right outer Join

↳ returns all the rows that having matching in both relations R1 and R2.

In addition to that it returns all the rows of Right relation that have no records match in left relation



Eg: courses ⟖ faculty

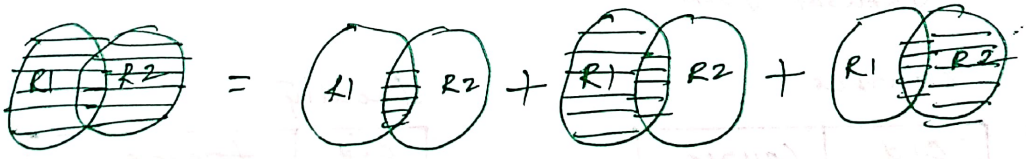
cid	course	fname
100	Database	Rohan
102	Electronics	Sara
104	NULL	Jiya

→ Matching tuples

→ Extra from Right relation

Full outer Join

↳ returns all matching tuples from R1 and R2 in addition to that if any extra or unmatched tuples from both R1 & R2 will be returned



Eg: courses ⟗ faculty

cid	course	fname
100	database	Rohan
102	Electronics	Sara
101	Mechanics	NULL
104	NULL	Jiya

→ Matched tuples

→ Extra from left Relation

→ Extra from Right relation

Left outer join (R1 ⟕ R2)

Returns
- all Matching tuples from R1 & R2
- and also R1 tuples that does not have matching tuples in R2

Eg: courses (c1)

cid	courses
100	Database
101	Electronics
102	Electronics

faculty (f1)

cid	fname
100	Rohan
102	Sara
104	jiya

R1 ⟕ f1

cid	courses	fname
100	Database	Rohan
102	Electronics	Sara
101	Mechanics	NULL

↓ Extra

Right outer join (R1 ⟖ R2)

Returns
- all Matching tuples from R1 & R2
- and also R2 tuples that does not have matching tuples in R1

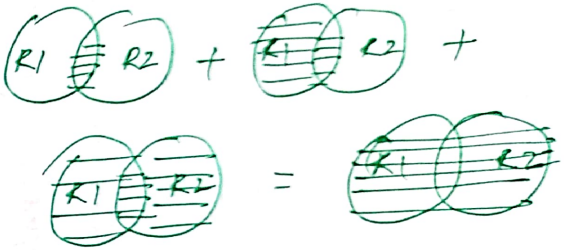
c1 ⟖ f1

cid	courses	fname
100	Database	Rohan
102	Electronics	Sara
104	NULL	jiya

↓ Extra

FULL OUTER JOIN R1 ⟗ R2

Returns
- all Matching tuples from R1 and R2
- and also R1 tuples that have no matching tuples in R2
- and also R2 tuples that have no matching tuples in R1



c1 ⟗ f1

cid	courses	fname
100	Database	Rohan
102	Electronics	Sara
101	Mechanics	NULL
104	NULL	jiya

↓ Extra

Relational Calculus

↳ Relational calculus is a non-procedural or declarative language.

↳ Relational calculus tells what to do but never explains how to do.

↳ When applied to database, it comes in 2 flavors.

(1) Tuple Relational Calculus (TRC)
- Proposed by codd in the year 1972
- Works on tuples.

(2) Domain Relational Calculus (DRC)
- proposed by Lacroix and pirotte in 1977
- Works on domain of attribute

↳ Calculus has variables, constants, comparision operator, logical connectives and quantifiers.

↳ Tuple Relational calculus is a non-procedural query language.

↳ TRC is used for selecting the tuples in a relation that satisfy the given condition

↳ A Query in TRC is Expressed as

$\{t \mid P(t)\}$ where

- t denotes resulting tuple
- P denotes predicate used to fetch tuple t

↳ Query returns set of all tuples t such that predicate P is true for t.

↳ The predicate or the atomic formula can be one of the following

- ↳ $R \in rel$ — (1)
- ↳ $R.a\ op\ s.b$ — (2)
- ↳ $R.a\ op$ constant — (3)
- ↳ $\neg p$, $p \wedge q$, $p \vee q$, or $p \Rightarrow q$ — (4)
- ↳ $\exists R\,(P(R))$, where R is a tuple variable — (1)
- ↳ $\forall R\,(P(R))$, where R is a tuple variable — (1)

Examples

(1) To display sailors with rating 7

$$\{ s \mid s \in sailors \wedge s.rating = 7 \}$$

of (1) form of (3) form.

(2) Find sailors whose rating is 7 and age above 35

$$\{ t \mid t \in sailors \wedge t.rating = 7 \wedge t.age > 35 \}$$

(3) Find Red color boats

$$\{ b \mid b \in boats \wedge b.color = \text{"red"} \}$$

(4) Find boats that are reserved with id 103

$$\{ r \mid r \in Reserves \wedge r.bid = 103 \}$$

(5) Find sailor names whose rating is above 7

$$\{ t \mid \exists s \in sailors\ (s.rating > 7 \wedge t.name = s.sname) \}$$

(6) Find names and ages of sailors whose age is Exactly 35

$$\{ t \mid \exists s \in sailors \ (s.age = 35 \wedge t.name = s.name \wedge t.age = s.age) \}$$

(7) Find sid and day of boats with id 102

$$\{ t \mid \exists r \in Reserves \ (r.bid = 102 \wedge t.sid = r.sid \wedge t.day = s.day) \}$$

(8) Find boatnames and colors of boat id 102

$$\{ t \mid \exists b \in Boats \ (b.bid = 102 \wedge t.bname = b.bname \wedge t.color = b.color) \}$$

(9) Find sailors who have reserved boats

$$\{ t \mid s \in sailors, r \in Reserves \ (s.sid = r.sid) \}$$

(10) Find sailor names who have reserved boats

$$\{ t \mid \exists s \in sailors, \exists r \in Reserves \ (s.sid = r.sid \wedge t.sname = s.sname) \}$$

(11) Find sailor names who have reserved boat with id 103

$$\{ t \mid \exists s \in sailors, \exists r \in Reserves \ (r.bid = 103 \wedge s.sid = r.sid \wedge t.sname = s.sname) \}$$

(12) Find Sailors who have reserved Red boats

$$\{ t \mid \exists s \in Sailors, \exists r \in Reserves, \exists b \in Boats\ (b.color = "red" \wedge b.bid = r.bid \wedge r.sid = s.sid) \}$$

(13) Find Sailor Names who have Reserved green boats

$$\{ t \mid \exists s \in Sailors, \exists r \in Reserves, \exists b \in Boats\ (b.color = "green" \wedge b.bid = r.bid \wedge r.sid = s.sid \wedge t.sname = s.sname) \}$$

(14) Find sailor names who have Reserved Interlake boats

(15) Find Sailor Names who have reserved Boats on 11/12/98.

Domain Relational calculus

↳ A Domain Variable is a variable that ranges over the values in the domain of some attributes.

For Eg: if the domain or datatype of an attribute is integer then the domain variable of that attributes ranges over the Integers.

For Eg in SQL if we declare Sno number(4) then the domain of the attribute Sno is any 4 digit number.

if S is a domain variable of Sno then S ranges over its domain (i.e any 4 digit number)

↳ A DRC, the Records are filtered based on the domains.

↳ DRC uses list of attributes to be selected from Relation based on the condition

↳ DRC is same as TRC but differs by selecting the attributes rather than tuples.

↳ In DRC, the queries are represented as

$$\{ a_1, a_2, a_3, \ldots, a_n \mid P(a_1, a_2, \ldots, a_n) \}$$

$a_1, a_2, a_3, \ldots, a_n$: represents Domain variables

$P(a_1, a_2, \ldots, a_n)$: represents predicate

↳ The Result of the query is set of all tuples ~~for~~ which the formula Evaluates to true.

↳ The predicate or Atomic formula in DRC is one of the following

(1) $(x_1, x_2, x_3, \ldots, x_n) \in Rel$

(2) x op y

(3) x op constant

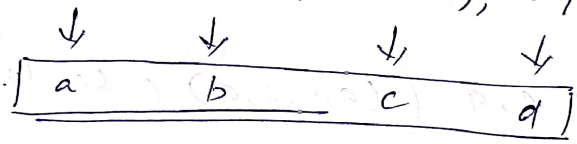
(4) $\neg P$, $P \wedge Q$, $P \vee Q$ or $P \Rightarrow Q$

(5) $\exists x (P(x))$

(6) $\forall x (P(x))$

Examples

Sailors (sid, sname, rating, age)



→ domain variables (whose values ranges over domain)

(1) Find sailors whose rating is above 7

~~$\{(a,b,c,d) | (a,b,c,d)$~~

$\{ (a,b,c,d) \mid (a,b,c,d) \in \text{sailors} \wedge c > 7 \}$

(2) Find red color boats

Boats (bid, bname, color)
x y z

$\{ (x,y,z) \mid (x,y,z) \in \text{boats} \wedge z = \text{"red"} \}$

(3) Find boat names which are green

or

Find boat names whose color is green

Boats (bid, bname, color)
x y z

$\{ y \mid \exists x, z ((x, y, z) \in \text{boats} \wedge z = \text{"green"}) \}$

(4) Find sailor names whose age is above 35

Sailors (sid, sname, rating, age)
a b c d

$\{ b \mid \exists a, c, d ((a, b, c, d) \in \text{sailors} \wedge d > 35) \}$

(5) Find sailors who have reserved boats

Sailors (sid, sname, rating, age)
a b c d

Reserves (bid, bid, day)
m n o

$\{(a, b, c, d, n, o) \mid (a, b, c, d) \in \text{sailors} \wedge (m, n, o) \in \text{Reserves} \wedge a = m \}$

(6) Find sailor names who have Reserved boats

$\{ b \mid \exists a, c, d (a, b, c, d) \in \text{Sailors} \wedge (m, n, o) \in \text{Reserves} \wedge a = m \}$

Expressive Power of Algebra and Calculus

↳ Relational Algebra and Relational calculus are two formal query languages.

↳ The question is "Are they Equivalent in power"

↳ if the Query in RA ~~can~~ is able to Express in RC and a query in RC is able to Express in RA.

Then they all Equivalent in power.

↳ But in Reality

Every Query that can be Expressed in Relational Algebra also be Expressed in Relational calculus but

Every Query that is Expressed in R.C can not be Expressed in RA because in RC, there is a possibility of writing unsafe queries

Example of unsafe Query

$\{ s \mid \neg (s \in sailors) \}$.

It asks for all tuples s such that s is not in sailors. The set of such s tuples is obviously infinite (or)

if a query returns infinite tuples then it is unsafe Query.

possibility of unsafe queries are more in Relational calculus.