Database Management Systems (DBMS)

* A data is a known fact that can be recorded and has implicit meaning to it.
* Building block of information is data.
* Information is collection of data.

Basic Definitions:

DataBase: A collection of related data.

Data: Known facts that can be recorded and have an implicit meaning.

Mini-World: Some part of the real world about data which is stored in a database.

Database Management Systems: A software package/system to facilitate the creation and maintenance of a computerized database.

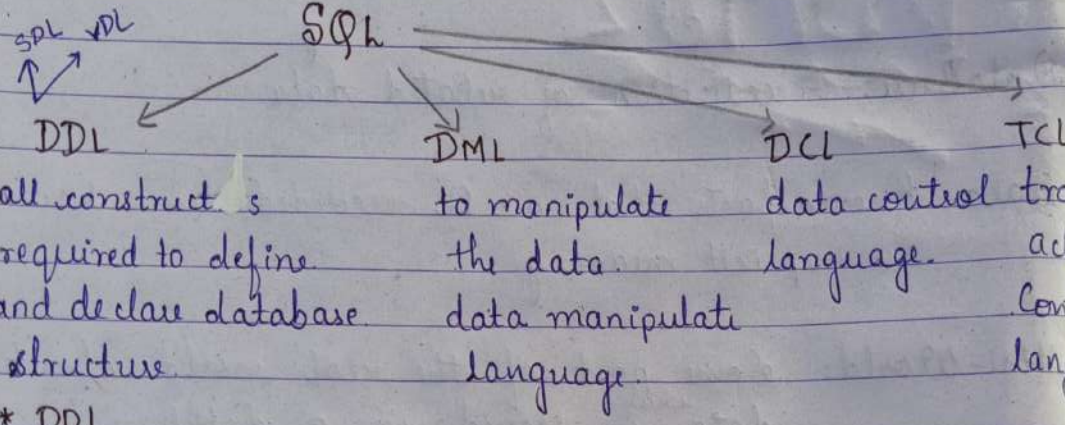
Database System: The DBMS software together with data itself. Sometime, the applications are also included.

DBMS
- Define
- Construct
- Manipulate
  - Retrieve
  - Insert
  - delete
  - Modify
- Share

DBA

21/1/18 Lab-01

* RDBMS was first proposed in 1970 by B.F. Codd.
* SQL is declarative language.
* RDMS allows to define, construct, manipulate and share the relational database.
* RDMS supports the structured Query language.



* DDL

create → create new component in database.
alter → used to make changes to created structure
drop → remove something from database (structural element).

* DML

insert → adds data to database.
delete → removes data from database.
update → used to modify existing data in database.
slect → retrival of data

*DCL

Grant → Give.
Revoke → Remove.

*TCL

commit → save the changes made by transaction.

Rollback → all changes that are made and executed can be brought back to previous state

Relational database

⇒ It is set of relations / Table.

⇒ Each row is also record in a table.

⇒ In relation or set we can't have duplicate Table can have duplicate records.

⇒ Relation : Set of tuples / record

/* ⇒ They are columns and fields */

Ordered collection of values acc to fields.

tuple: Is ordered collection of values for attribute.

Key attribute which has no duplicate value. It helps to identify unique records in table

Student (Name of table / relation)

→ fields / column / attribute

→ record

Depart No	USN	SName	DOB	Address

↓ foreign att

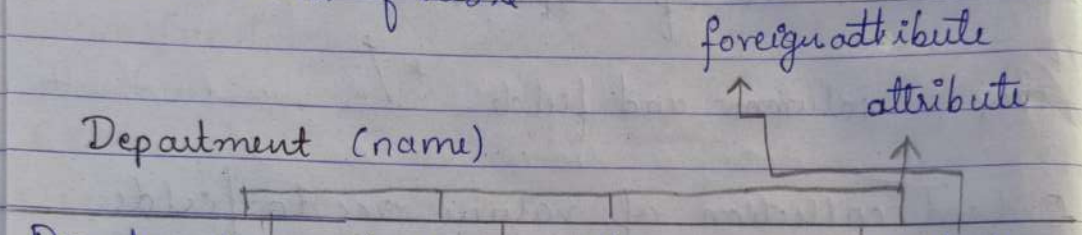
Key attributes may take null values either the value doesn't apply, value doesn't exist, value not available. So we use primary key attribute

Primary key The attribute which has no duplicate values for any records or will never have null value

If a table has primary key then such table is a relation since it will have all unique values

Records are also known as entities.

foreign key attribute: It refers to attribute which is primary key attribute in foreign attribute. This foreign attribute will build relationship with records of table



Department Number	Name	Place	Student USN
10	CSE	C-Lite	01fe18bcs211
20	Mechanical	Mech Building	01fe18bme200
30	Civil	Civil Building	01fe18bcv200

Student (name)

primary key

USN	Name	Addr	Dept. No	Emailid
01fe18bcs24	Ram	Hubli	10	r@...
01fe18me213	Sham	Dharwad	10	s@...
01fe18cv211	Sita	Davangere	20	NULL
01fe18me211	Gita	Dharwad	30	NULL
01fe18me211	Geeta	Dharwad	40 →	Error. (domain)

↳ primary key attribute

Every attribute must have its own domain and its own datatype.

Foreign key attribute a key that does not describe the entity (or record) rather describes the relation of this entity with some other entity in a foreign table.

THEORY:

Schemas versus Instances.

- Database Schema: The description of a database Includes descriptions of the database structure and the constraints that hold on the database

Three Schema Architecture diagram

Conceptual Schema has complete structure of database. The external schema is the description of structure for end users. The description of how the data is stored in the physical memory (what memory locations etc), file format size format etc is stored in internal schema. It is low level description. Conceptual or external is external description (logical data)

* The process of transforming requests and results between levels are called mappings

Program - data independency.

* Ability to change schema at one level without affecting the other schemas → data independency.

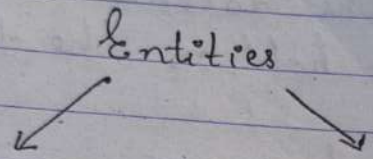
Types:-

1) Logical data independency:- It is the ability to change the conceptual schema without having to change the external schema. It exists b/w external schema and conceptual schema.

2) Physical data independency:- It is the ability to change the physical data without having to change the conceptual data. It exists b/w internal schema or physical schema and conceptual schema

Meta-data is in the catalogue.

E-R Model or Entity-Relationship model:
Entity are objects or independent objects that exist in miniworld and whose data is stored.



logical entities Physical entities.

Entity type
Entities are described by what are known as attributes for entities OVAL CAPS.
Attribute is a property that describes a particular entity
Entity type is a group of entities that have the same attribute

CAPS IN RECTANGULAR.

Relationship

The association of an entity of entity type with an entity of another entity type is called relationship instance.

Ex:- Faculty works for department

works for → relationship.

The association is known as relationship.

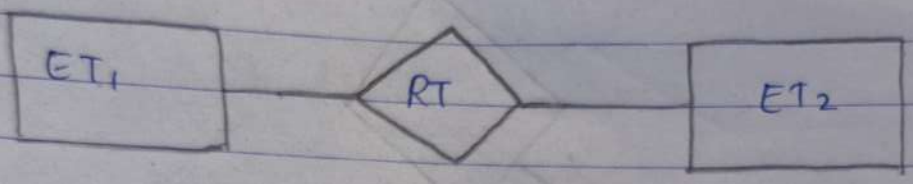
Collection or set of relationship instances having the same meaning is called relationship type.

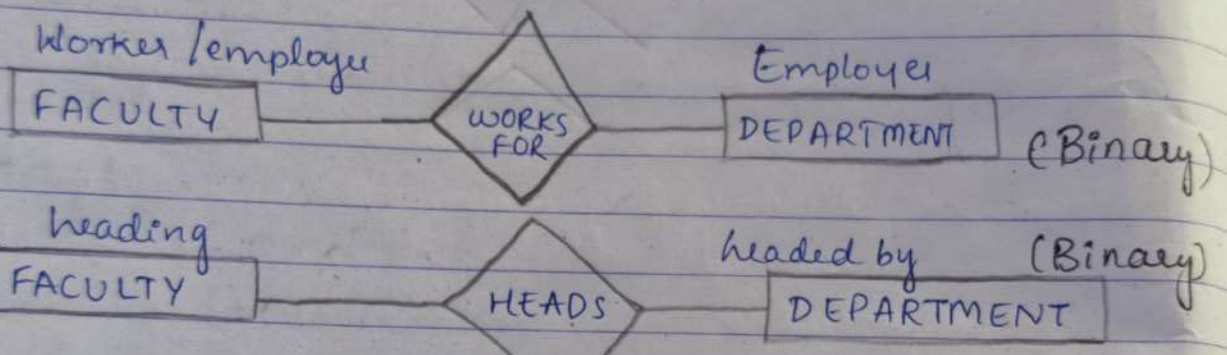
Ex:- Mgk works for CS (Relationship instance)

There can be multiple associations
Ex:- HOD who is always one of the faculty head the department

Faculty works for department.
Faculty heads the department

Relationship set. 
It exists between the entity type
CAPS

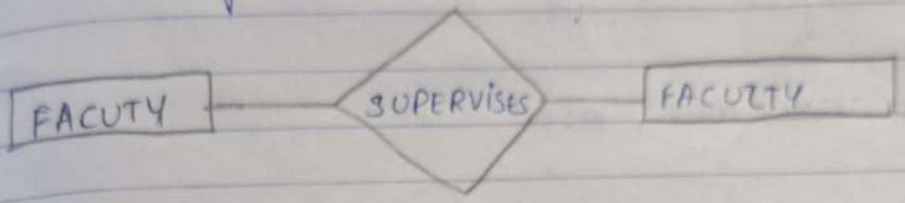




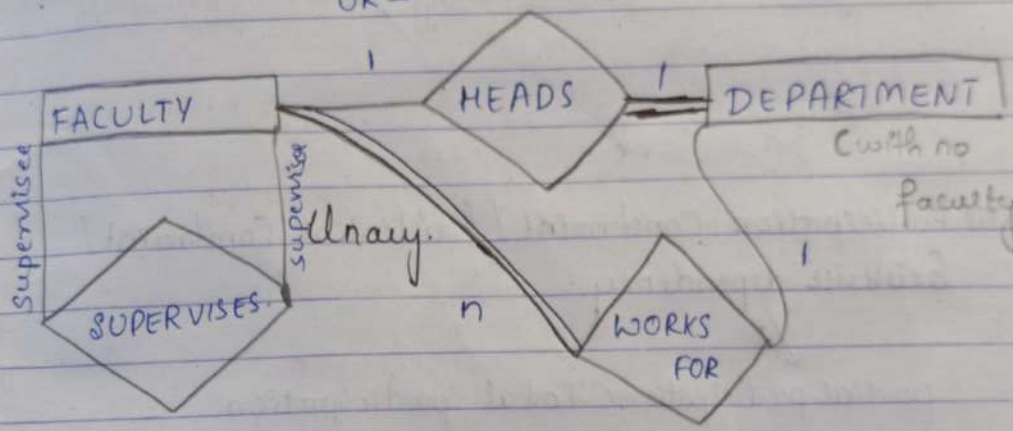
Unary: Two entities of same entity type are related to each other.
Binary: Ex: Faculty heads dept.
Ternary: Three entity types involved.
n-ary:

Unary: Entity types participating are same
-OR-
Recursive relationships

Ex: Junior faculty is supervised by senior faculty
-OR-
Senior faculty supervises the junior faculty



-OR-



Ternary:- Supplies

Supplier supplies part to project

3 entities of 3 entity type

Role names for unary is must

f_1 → supervision → f_2

supervisee

supervisor

Structural Constraints

They are the restrictions imposed on relationship type

(i) Cardinality Ratio.

(ii) Participation Constraint

(i) Cardinality Ratio

Cardinality represents the total no. of relationship instances that an entity from one side can have with an entity of other type

1:1, 1:N, N:1, M:N

(ii) Participation Constraint / Existential Constraint / Existence dependency.

Partial participation Total participation.

It represents how the entities from both side are participating in the relationship type

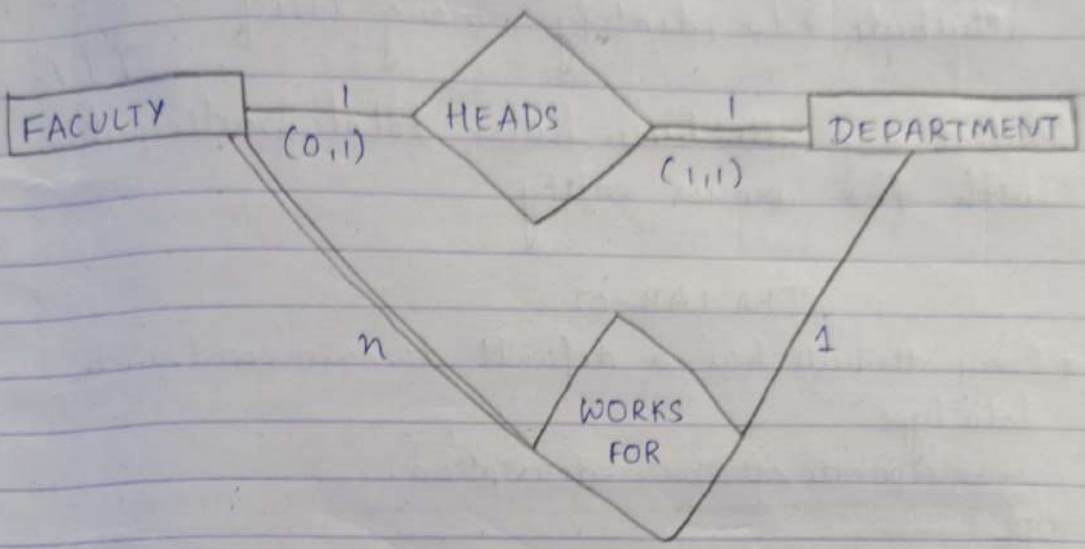
If all entities of the entity type are not participating in the relationship type then partial participation "—" represents partial participation

If all entities of the entity type participate in the relationship type then it is called total participation "=" represents total participation

Ex: Every department entity will have an instance of

that relation type.

Cardinality Ratio is represented using minmax notation.



Ex:- (1,6)

WEAK Entity type:-

Their data is stored about such entities because of the existence of some other entity in the miniworld and whose physical presence in the miniworld doesn't exist

Ex:- Parents information about student.

weak entity is identified by regular type.

regular entity / Owner/strong entity

It is an entity which does not have key

 → weak relationship type (Relation b/w weak entity type & it's identifying entity type)

attribute of its own So the weak entities can't be uniquely identified.

Partial Key attribute is formed using the key attribute of owner entity and a unique attribute to identify among itself

Partial Key attributes have total participation with the owner entity

DBA LAB-02

* Every attribute has a default domain constraints, data type.

, → separate attribute description

DDL

(i) Create

To create new components.

Syntax: create table <tab_name>

(Att_name1 data_type const2 const...,
Att_name2 data_type const2 const...,
⋮
Att_namen " " "
);

Some constraints are imposed on more than one attribute

Every table /relations will have only one attribute but may be formed by multiple attribute.

Constraints are condition that must hold on all valid relationship instances

```
create table <table_name>
( Att_name1  data_type,
  Att_name2  data_type,
      :
      :
  Att_namen  datatype,
      cost1,
      cost2,
        :
        :
      costn
);
```

Datatypes in SQL. i) Numeric datatypes
int, float, integer, double, number, number(10,5)
↙ ↓
total no of digits in real number; no of significant digits after .

ii) Non-Numeric Data types.
char, char(10) // array of
↳ fixed length character

varchar, varchar(10) → 1 byte for character; memory reserved during compilation time.
↳ variable length. ↳ size of array

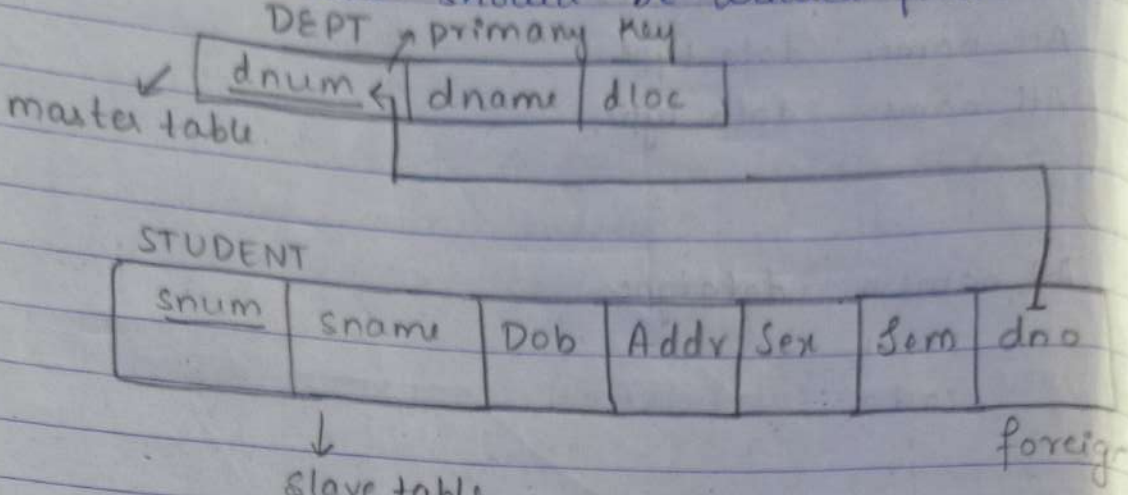
varchar2(n), varchar2
⇓
2 bytes to store a character
→ dynamic allocation.
// when values are put.

date, time.

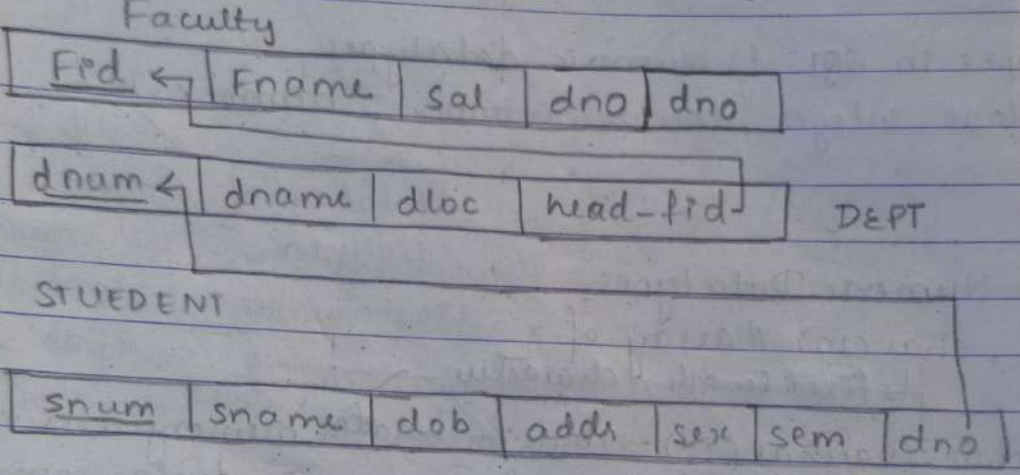
Table with foreign key ↔ Slave / Child / Referencing

The table with primary key att pointed by foreign key from other table → Master Table / Parent Table / Referenced Table

The master table should be created first



Confuse:-



// When both tables have foreign key attribute then just create table then add constraint

varchar2 → takes 2 bytes for a character
varchar → takes one byte.
char → static allocation - fixed size.
varchar → dynamic allocation.

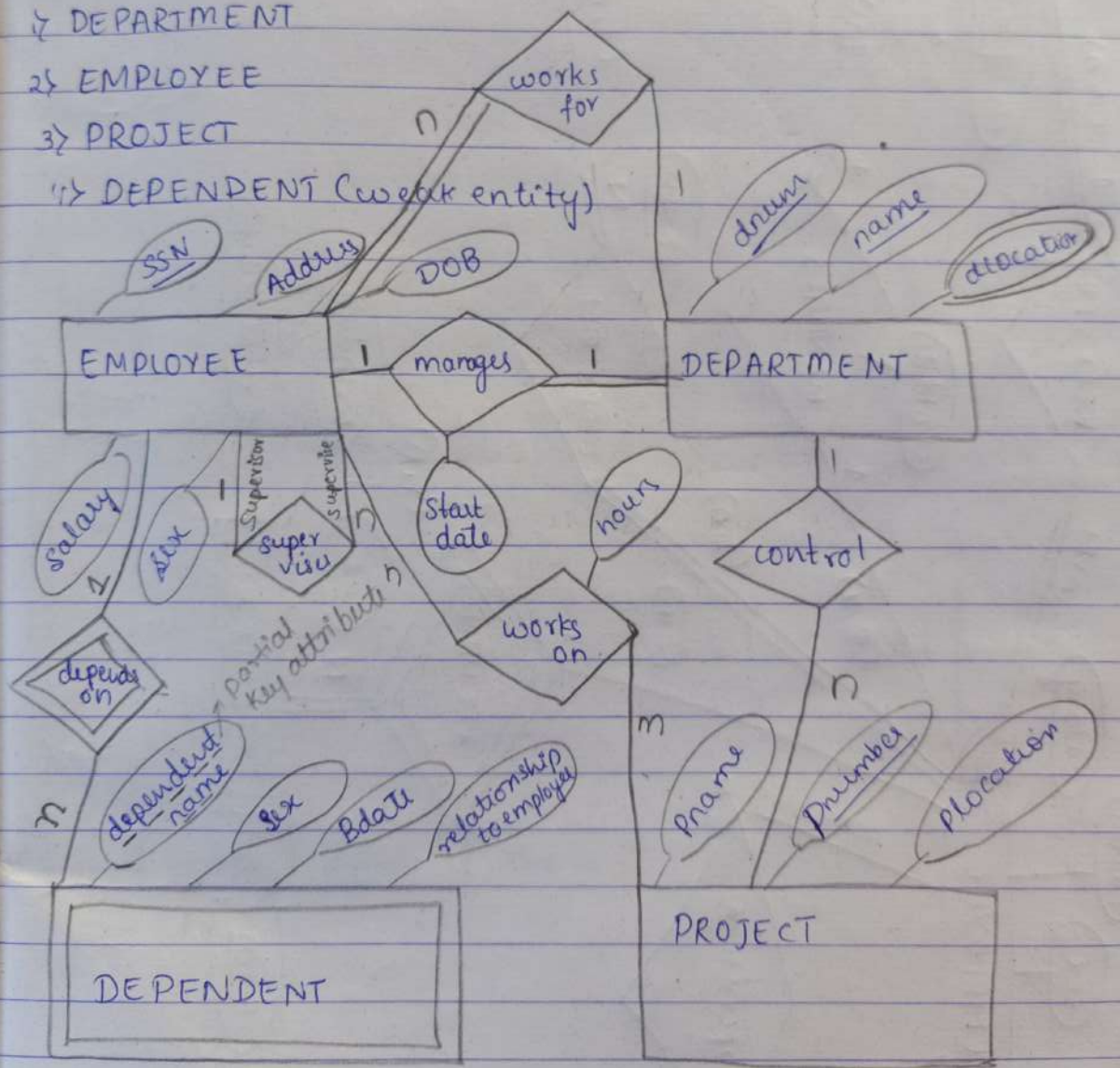
desc - describe table name.

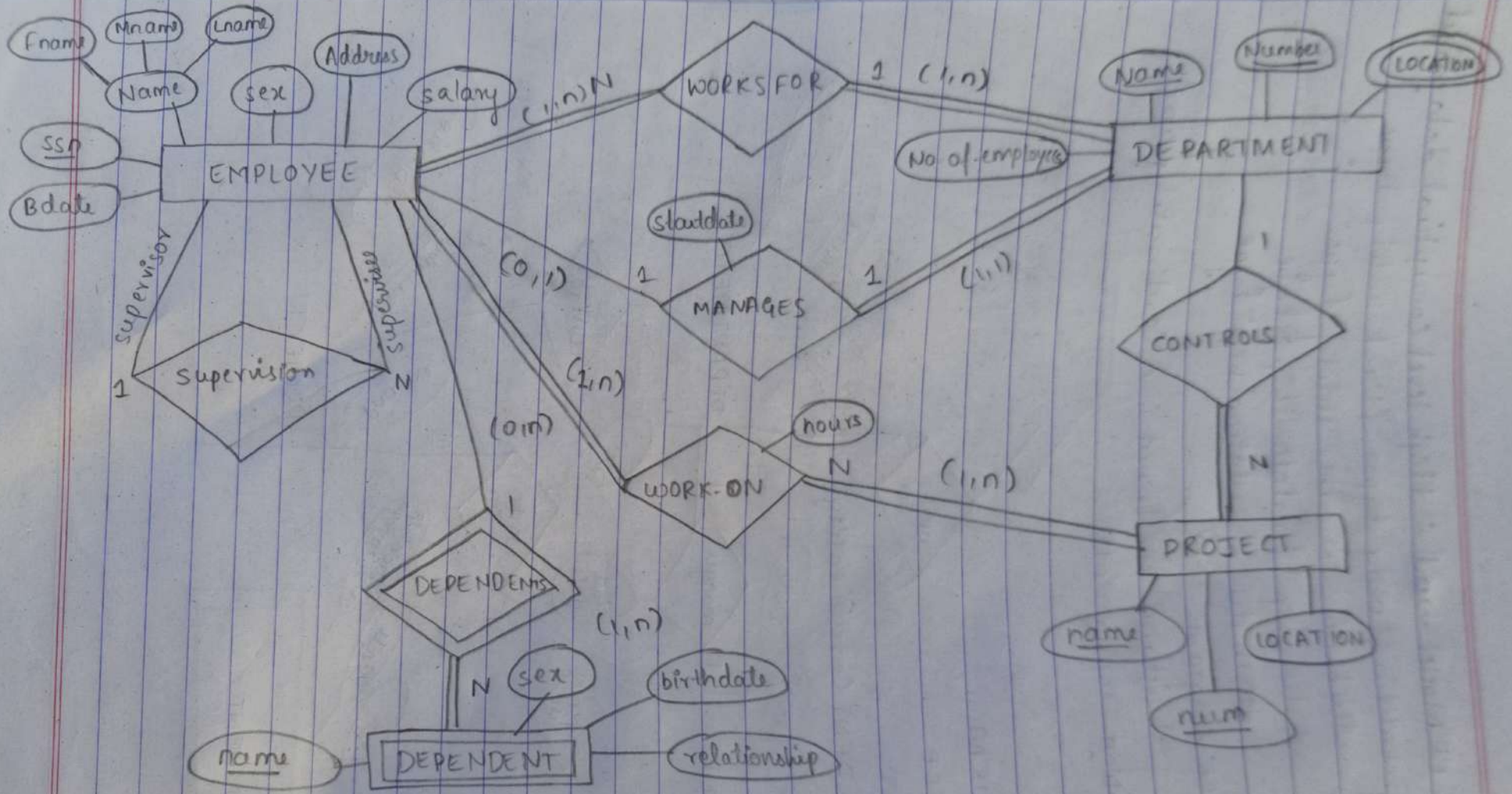
Steps for ER Model - : High level conceptual data model

1) Identify Entities / Entity types / Weak entity type
2) Identify attributes for the entity types
3) Identify relationship / relationship type
4) Identify attributes of relationship type
5) Draw ER diagram
6) Identify structural constraint (i.e Cardinality ratio and participation constraint)

Entities

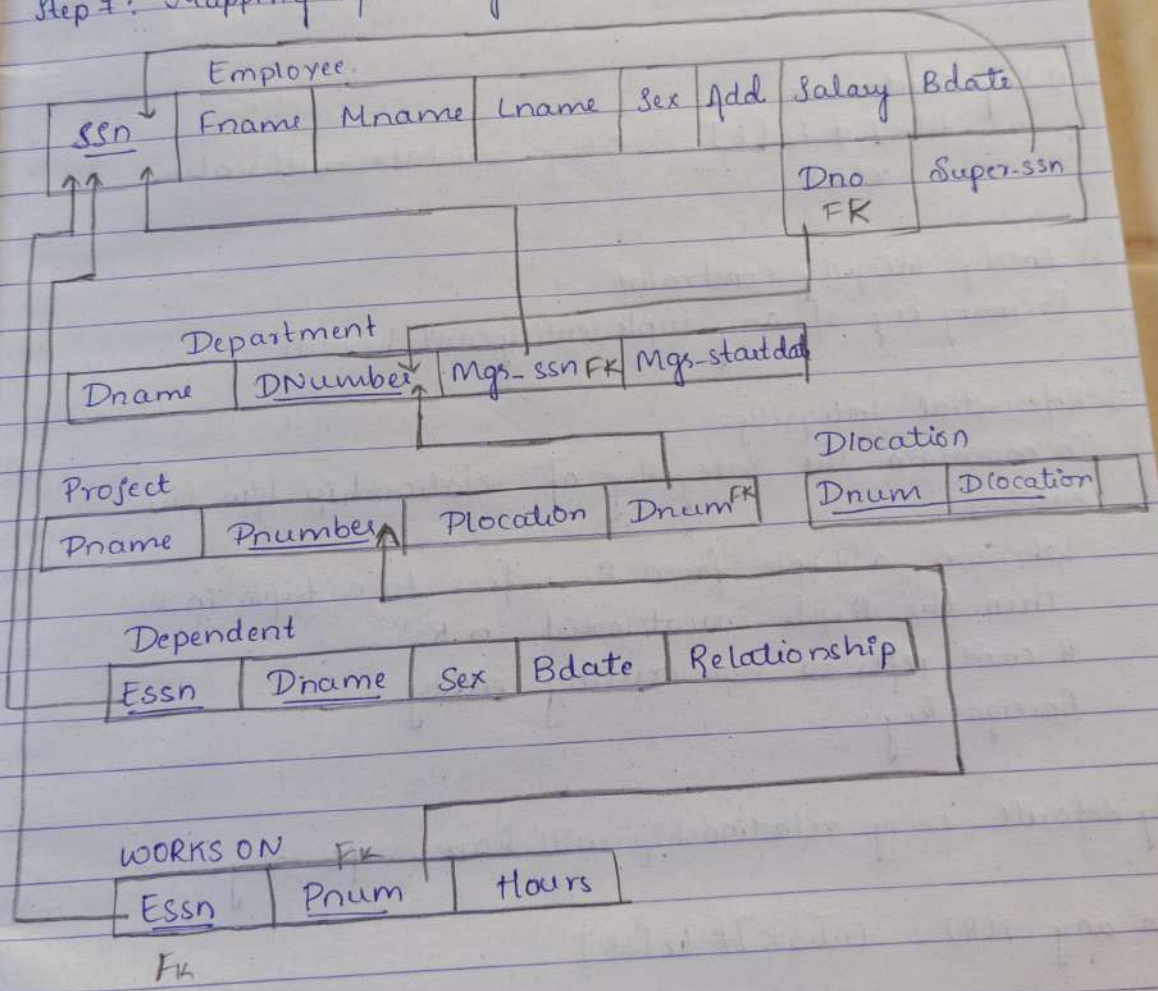
1) DEPARTMENT
2) EMPLOYEE
3) PROJECT
4) DEPENDENT (weak entity)




Fname
Minit
Lname
Name
sex
Address
salary
SSN
Bdate
EMPLOYEE
(1,n) N
WORKS FOR
1
(1,n)
Name
Number
LOCATION
No of employees
DEPARTMENT
supervisor
supervisee
1
Supervision
N
(0,1)
1
StartDate
MANAGES
1
(1,1)
1
CONTROLS
(1,n)
(0,n)
hours
WORK-ON
N
(1,n)
N
PROJECT
1
DEPENDENTS
(1,n)
name
LOCATION
num
N
sex
birthdate
name
DEPENDENT
relationship

Characteristics of relationship(*)

Step 1: Mapping of regular entity types
Step 2: Mapping of weak entity types
Step 3: Mapping of binary 1:1 Relationship types.
Step 4: Mapping of binary 1:N relationship types
Step 5: Mapping of binary M:N relationship types
Step 6: Mapping of multivalued attribute
Step 7: Mapping of N-ary relationship types.



Relational Integrity Constraints.

1) Key Constraints

$t_1[K] \neq t_2[K]$ // key can take null value

$r(R)$.

2) Entity Intigrity constraint

Primary key of any tuple/entity can't be null

3) Referential integrity:-

To maintain the integrity of relationship b/w the entities.

"Whenever a tuple from R_1 refers to a tuple in R_2 then the tuple must exist in R_2"

It can't refer to non existing entity

Foreign key

By default every relationship will have super key.

In any $r(R)$ $t_1[SK] \neq t_2[SK]$

* Retrieve no. of dependents depending on employees of admin department?

* for each department retrieve name, number & no. of its location.

* for each department display its dname, min sal, max salary.

* for each project no. of employees working for that project along with its name.

* for each dept whose avg salary of its employees is greater than 16K retrieve dept name and avg salary

for each - group by
cond'n - having
reason

* for each dept in which less than 2 employees are working retrive dept name and no of employees working

THEORY

RELATIONAL ALGEBRA

* **Unary Operators:-** They operate on only one relation
δ → selection operator
Π → projection operator
ρ → rename operator.

δ - used to select the required tuples from the relation.

$\sigma_{<condition>}^{R}$

#NOTE: ← assignment operator

Result of any relational algebra operator is a relation

$R' \leftarrow \sigma_{GPA>3.5}^{STUDENT}$

Student	Name	GPA	SSN	Home-Phone	Add	Off ph	Age
	Dick Davidson	3.53	422-11-2320	null	---		
	Charles Cooper	3.93	409-22-1100	376-9821	--		

* Π - projection operator (duplicate eliminator)

Vertical bisection of relation R

Syntax:-

$R' \leftarrow \Pi^{R}_{A1, A2, ----, An}$

Ex: EMP

Eno	Ename	Salary
1	x	10000
2	y	10000
3	w	5000
4	z	5000

$R' \leftarrow \Pi^{EMP}_{salary}$.

If projection list is not containing key attribute then we may loose the tuples ∵ of the duplicate values ∵ the return value is also a relation set so duplicates are eliminated

Output: R'

Salary
10000
5000

→ Since this also a relation the duplicates are also deleted.

NOTE: We can't have only projection so selection should be followed by projection by cascading the operators.

Query:- Retrieve age and GPA of all students.

$$R' \leftarrow \Pi^{STUDENT}_{age, GPA}$$

$$R' \leftarrow \Pi_{A_1, A_2 - -} \left(\delta^{R}_{<end>}\right)$$

Query: Retrieve SSN, fname and salary of all employee who work for dept no 5

$$R_1 \leftarrow \delta^{EMPLOYEE}_{dno=5}$$

$$R' \leftarrow \Pi^{R_1}_{SSN, fname, salary}$$

-OR-

$$R \leftarrow \Pi_{SSN, fname, salary}\left(\delta^{EMPLOYEE}_{dno=5}\right)$$

Query: Retrieve name of all female employee

$$R \leftarrow \Pi_{fname, lname}\left(\delta^{EMPLOYEE}_{sex='F'}\right)$$

ρ - rename operator

ρ - rename attribute, name of relation and both name of relation and attribute

Query: Retrieve fname, SSN and dept Name of the employee.

Binary Operators: they operate on two relations

$\times$: Cross product

$\bowtie$: Join operator

$-$: Set difference -OR- Symmetric difference

$\cup$: Union

$\cap$: Intersection

X

EMP

Eno	Ename	Dno
1	X	10
2	Y	20
3	Z	10

DEPT

Dnum	dname
10	CS
20	EC

* X - Cartesian product

To combine the two tables. (m+n attributes in result)

Eno	Ename	Dno	Dnum	dname

If m is no. of tuples in t_1 and n is the no. of tuples in t_2 then the resultant of this relation/table over cross product is $m*n$.

If m is no. of att in t_1 and n is no. of att in t_2. After cross product the resultant table has m+n values/attributes

Eno	Ename	Dno	Dnum	dname
1	X	10	10	CS
1	X	10	20	EC
2	Y	20	10	CS
2	Y	20	20	EC
3	Z	10	10	CS
3	Z	10	20	EC

Disadvantage:- Many spurious tuples are generated.

Query Result: $R_1 \leftarrow$ Employee $\times$ Department

$R_2 \leftarrow \sigma_{dno=dnum}(R_1)$

$R_3 \leftarrow \Pi_{fname, SSN, Dname}(R_2)$

$R_3 \leftarrow (\Pi_{fname, SSN, Dname}(\sigma_{dno=dnum}(Employee \times Department)))$

Rename Operator:-

$\rho_S(B_1, B_2, B_3 \ldots B_n)(R)$ → both attribute and relationship

$\rho_S(R)$ → Relation rename

$\rho(B_1, B_2 \ldots B_n)(R)$ → Attribute only renamed

Set Operations:-

Since relations are sets we can apply set operations. But there is some condition to apply the set

* iff operand relations are union compatible

↳ Two relations are said to be union compatible iff degree of R_1 = degree of R_2.

↳ Degree of relation = No of attributes in that relation.

* Domain of attributes (corresponding attributes) must be same

domain of $R_1(A_i)$ = domain of $R_2(B_i)$

i.e data types of corresponding attributes must be same.

For the set operations $\cup, \cap, -$ to be applied the operand relations must follow the above rules.

Query Examples:-

FACULTY

Fid	Fname	Dno
1	Raj	10
2	Radha	20
3	Sham	10
4	Shama	20

INSTRUCTOR

Iid	Iname	Dno
1	Raj	10
5	Ram	20
6	Rani	10
4	Shama	20

The above relations are union compatible.

Default: If the new relation is not named then the resultant relation has the name of first operand relation and name of first relation attribute

Q: Retrieve details of faculty who are working either as faculty or instructor.

FACULTY ∪ INSTRUCTOR

R	Fid	Fname	Dno
	1	Raj	10
	2	Radha	20
	3	Sham	10
	4	Shama	20
	5	Ram	20
	6	Rani	10

Q: Retrieve the info of faculty working both as faculty and instructor.

$R \leftarrow FACULTY \cap INSTRUCTOR$

R	Fid	Fname	Dno
	1	Raj	10
	4	Shama	20

Q: Retrieve info of all faculty working exclusively as faculty

$R \leftarrow FACULTY - INSTRUCTOR$

R	Fid	Fname	Dno
	2	Radha	20
	3	Sham	10

Q: Retrieve names of the employee who are either working for dept no 5 or working for dept no 4

Source: ppt

$R_1 \leftarrow \delta^{EMPLOYEE}_{dno=5}$

$R_2 \leftarrow \delta^{EMPLOYEE}_{dno=4}$

$R_3 \leftarrow \Pi^{R_1}_{fname}$

$R_4 \leftarrow \Pi^{R_2}_{fname}$

$R_5 \leftarrow R_3 \cup R_4$

-OR-

$R \leftarrow \Pi_{fname}(\delta^{EMPLOYEE}_{dno=4}) \cup \Pi_{fname}(\delta^{EMPLOYEE}_{dno=5})$

Q: To retrieve the social security number of all employees work in department 5 or directly supervise an employee who works for department 5.

$$R1 \leftarrow \Pi_{SSN}\left(\sigma^{EMPLOYEE}_{dno=5}\right)$$

$$R_2(ssn) \leftarrow \Pi_{superssn}\left(\sigma^{EMLOYEE}_{dno=5}\right)$$

$$R \leftarrow R_1 \cup R_2$$

// Retrieve names of those employees.

$$R3 \leftarrow R \times EMLOYEE$$

$$R_4 \leftarrow \sigma^{R3}_{r.ssn=ssn.(employee)}$$

$$R_5 \leftarrow \Pi^{R_4.}_{fname}$$

Aggregate functions:-

Fundions that operate on the group of data /values and produce the result which is aggregate to all values. These functions operate only on the values of the table.

$\mathcal{F}$ → script f functions

$\mathcal{F}$ functionname/attributelist, functionname2 Attlist

[func-name-att, funname att]

Q. Retrieve total salary, min salary and average salary of the employees.

R ←

$\mathcal{F}$ MIN salary, max salary, sum salary, avg salary (Employee)

MINIMUM_SAL	MAXIMUM_SAL	SUM_SAL	AVG_SAL
2500	5500		

Q2: Retrieve total and avg salary paid to department no 5 employee

$R \leftarrow \sigma_{dno=5}(EMPLOYEE)$

RESULT ← $\mathcal{F}_{SUM\ salary,\ avg\ salary}(R)$

Q2.1: Department Wise.

$R \leftarrow {}_{DNO}\mathcal{F}_{SUM\ salary,\ AVERAGE\ salary}(EMLOYEE)$

group by

Dno	Sum(salary)	Average(salary)

Q3: Retrieve department wise total no. of projects controlled by each department

$R \leftarrow {}_{Dnum}\mathcal{F}_{COUNT\ PNUMBER}(PROJECT)$

Dnum	Count
5	3
4	2
1	1

Q4: For each dept retrieve department name and no. of projects each department is controlling

$R1 \leftarrow$ PROJECT X DEPARTMENT

$R_2 \leftarrow \delta_{DNUM = DNUMBER}^{R1}$

$Result \leftarrow Dname\ \mathcal{F}_{count\ PNUMBER}^{R_2}$

Dname	Count_Pnumber
Research	3
Administration	2
Headquarter	1

$Result \leftarrow \underbrace{\delta_{count\_Pnumber > 1}^{RESULT}}_{having}$

Qs For each department retrieve dept.name no. of location it has

— rename attribute /alias/ dot —

$R_1 \leftarrow$ (DEPARTMENT X DEPT_LOCATION)

$R_2 \leftarrow \delta^{R_1}$ (DEPARTMENT.DNUMBER = DEPT_LOCATION.DNUMBER)

$Result \leftarrow Dname\ \mathcal{F}_{count\ DLOCATION}^{R_2}$

with department number

$Result \leftarrow Dname\ Dnumber\ \mathcal{F}_{count\ DLOCATION}^{R_2}$

Q6. For each employee retrieve his SSN his name and no. of dependents that he has.

$R_1 \leftarrow$ (EMPLOYEE $\times$ DEPENDENT)

$R_2 \leftarrow \sigma_{ESSN = SSN}^{R_1}$

Result $\leftarrow$ ESSN, fname $\mathcal{F}_{count}^{R_2}$ DEPENDENT_NAME

DIVISION OPERATOR:-

* The relation operands must be division compatible.

(i) $R(z) \div S(x)$

↓ set of att in R ↓ set of att in S

$T(y) \leftarrow$

$x \subset z$ (S should be proper subset of z)

i.e $X \subset Z$

Set of att in S are subset of att in z.

$y = z - x$

Qo. Retrieve names of all those employee who work on all the project in which John Smith is working

$John\text{-}Smith \leftarrow \sigma_{fname='John' \text{ and } lname='Smith'}^{EMPLOYEE}$

$Smith\_SSN \leftarrow \prod_{SSN}^{John\text{-}Smith}$

$Smith\_proj \leftarrow \sigma_{SSN = Essn}(works\_on \times Smith\_SSN)$

$Smith\text{-}PNOS \leftarrow \prod_{PNO}^{Smith\_Proj}$

$first_{(SSN, PNO)} \leftarrow \prod_{PNO, ESSN}^{works\_on}$ // we need ESSN // order only

$Result \leftarrow first \div Smith\text{-}PNOS$

$R \leftarrow \Pi_{fname, lname}$ (Result $\times$ EMPLOYEE)
Result.SSN = EMPLOYEE.SSN

* Join Operator

$$R \leftarrow R_1 \bowtie_{<condition>} R$$

It is a cross product followed by selection of tuples that satisfies the condition.

If the condition has relational operators —
θ-join.
↓
any relational operator.

If join operation has only "=" operator -OR- where θ = "=" operator. Then it is called equijoin.

* Natural join.

EMP

Eno	Ename	Salary	Dno
1	Ram	10K	10
2	Sita	5K	10
3	Shaw	6K	20
4	Shama	5K	20
5	Raghu	4K	30

$\bowtie_{dno=dnum \text{ and } 5K<salary}$

DEPT

Dnum	Dname
10	CS
20	EC
30	Mech.

EMP_DEPT

Eno	Ename	salary	Dno	Dnum	Dname
1	Ram	10K	10	10	CS
3	Sham	6K	20	20	EC.

Demerits of natural join:

Normal - join will have records from table with foreign key.

And the presence of reddendent attribute, Space inefficient.

Natural Join:-

So we go for natural-join → it is a equijoin

$R \leftarrow R_1 * R_2$

Conditions to perform natural join:

To perform join operation:-

the operand relation must have atleast one attribute with same name

default join condition is the equality b/w the attributes with same name in both the relation

Advantage:

One of the reddent attribute is eliminated in result

R ← EMP * DEPT.

All natural joins are equi-joins

All equijoins are not natural join

R	Eno	Ename	Salary	Dnum	Dname.
	1	Ram	10K	10	CS
	2	Sita	5K	10	CS
	3	Sham	6K	20	EC
	4	Shama	5K	20	EC
	5	Raghu	4K	30	Mech

Qo: Retrieve name of the employee and name of the project on which he is working

$R_1 \leftarrow$ WORKSON $\bowtie_{Essn = SSN}$ EMPLOYEE

$R_2 \leftarrow R_1 \bowtie_{pno = pnumber}$ PROJECT

$Result \leftarrow \prod_{Fname, Pname} R_2$

If tuples don't have relationship are lost in result.
dept no of Raghu = NULL
then Raghu record goes missing

Eno	Ename	Salary	Dno
Raghu	Raghu	4K	null

Eno	Ename	Salary	Dno	Dnum	Dnaw.
1	Ram	10K	10	10	CS
2	Sita	5K	10	10	CS
3	Sham	6K	20	20	EC
4	shama	5K	20	20	EC

↳ Raghu is missing

Outer Join:-
It also retains the tuples with no relation based on condition.

Left outer join: ⟕

It will retain all the tuples on LHS even though they don't have matching tuples in RHS
The records of RHS is padded with NULL
We loose tuples from RHS.
R:

Eno	Ename	Salary	Dno	Dnum	Dname
1	Ram	10K	10	10	CS
2	Sita	5K	10	20	CS
3	Sham	6K	20	20	EC
4	Shama	5K	20	20	EC
5	Raghu	4K	null	null	null

Right outer join: ⟖

It will retain all tuples on RHS even though they don't have matching tuples in LHS.
The records of LHS is padded with NULL
We loose tuples from LHS.

Eno	Ename	Salary	Dno	Dnum	Dname
1	Ram	10K	10	10	CS
2	Sita	5K	10	10	CS
3	Sham	6K	20	20	EC
4	Shama	5K	20	20	EC
5	null	null	null	30	mech

Full Outer Join: ⋈
Retains tuples of both relations

Eno	Ename	Salary	Dno	Dnum	Dname
1	Ram	10K	10	10	CS
2	Sita	5K	10	10	CS
3	Shaw	6K	20	20	EC
4	Shama	5K	20	20	EC
Null	null	null	null	30	Mech
5	Raghu	4K	null	null	null

Set operations
→ UNION
→ MINUS
→ INTERSECT

Nested Queries / Sub Queries
SQL Query is the set of clauses
Whenever a SQL query is placed within the clause of the outer query then the inner query is called subqueries or nested queries.

```
Select a1, a2
from t1, t2
where <condition>
Group by a3, a4....
having <conditions>
order by a1 asc, a2 desc;
```

Qo. Retrieve names of all employees whose salary is more than the average salary of all the employees

```
select fname
from employee
where salary > ( select avg(salary)
                 from employee;
               );
```

// whenever the nested query is self-sufficient or is independent on outer query for processing it Then the inner query is executed first and the answer is placed // Now the inner query is dataindependent and is executed only once.

// Inner tables can access the attribute of outer query table This type of nesting is correlated nested query There is dependency of data. The inner query dependends on the outer query data The execution begins from outer query

For every record in the final table the inner query runs once

Q1. Retrieve names of all the employees whose salary is more than the average salary of his own department employee

Correlated nested query

```
select fname
from employee e1
where e1.salary > (select avg(salary)
                   from employee e2
                   where e1.dno =
                         e2.dno
                  );
```

In, Exists, Not;

v_1 In (v_1, v_2, v_3 ..)

(v_1, v_2) In ((v1, v2), (v3, v4) ...)

EXISTS (...)
↳ Unary // True or false

Both in and exists can be prefixed with not

Q. Retrieve names of all the employee who has dependents.

```
select *
from employee
where SSN in (select ESSN
              from dependent
             );
```

Q3. Retrieve first name of all employee who has dependent with same name as his name

```
select fname
from employee
where EXISTS ( select *
               from dependent
               where SSN = ESSN
               and fname = dependent_name
             );
```