4) Discuss the Entity Integrity and referential integrity constraints. Why each considered important.

15CS53 Jan/Feb
18CS53 Jan/Feb 2021

⇒ Entity integrity constraint :-

The Entity Integrity constraint states that, "in the relations, the value of primary key can not be null". The NULL represents a value for an attribute that is currently unknown or is not applicable for this tuple. The NULL's are always to deal with incomplete or exceptional data.

The primary key value helps in uniquely identifying every row in the table. Thus, if the users of the database want to retrieve any row from the table or perform any action on that table, they must know the value of the key for that row. Hence, it is neccessary that the primary key should not have the NULL value.

For example,

Roll-No	Name	Marks
001	AAA	88
002	BBB	83
003	CCC	97
	DDD	69

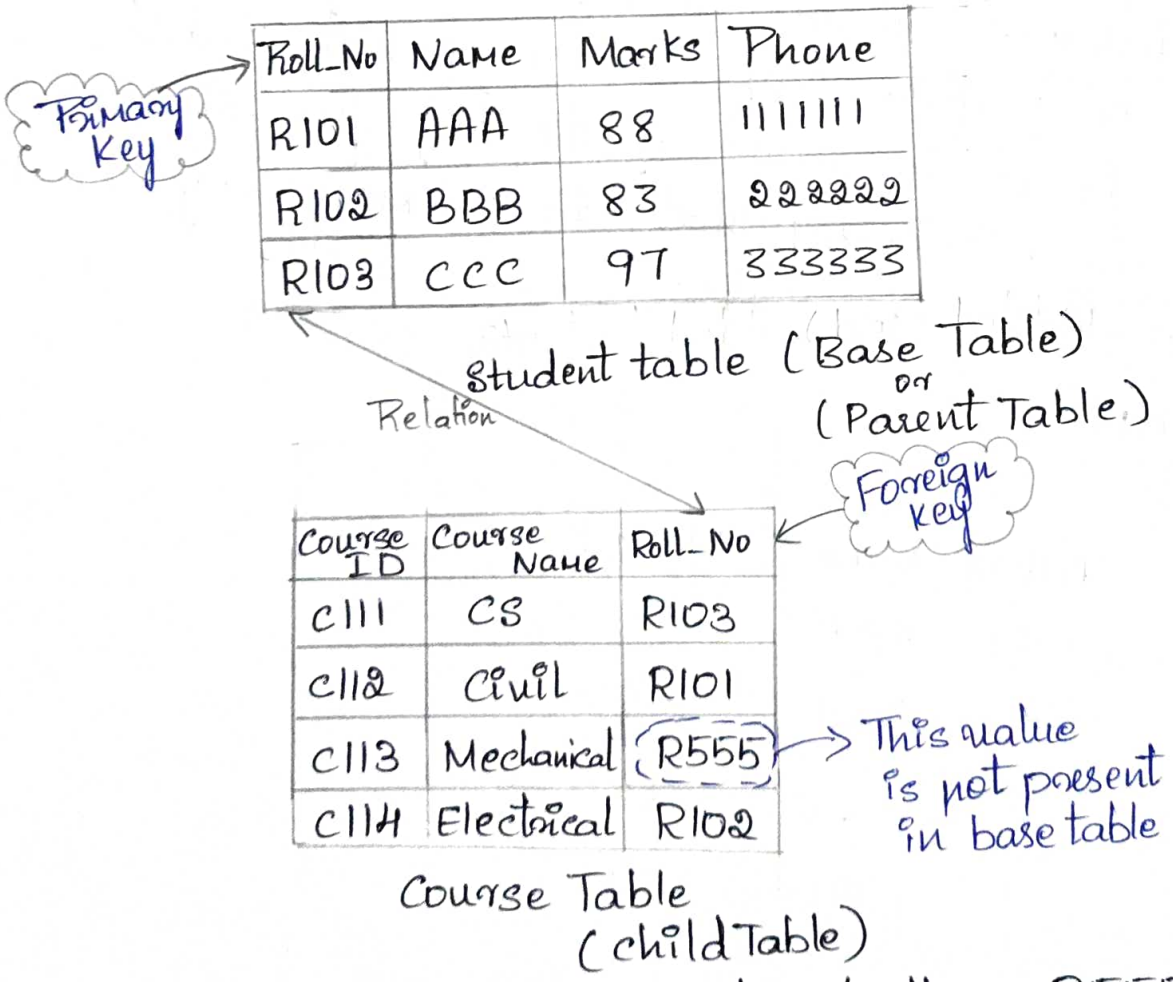
NULL is not allowed

Referential Integrity Constraint :-

In relationships, data is linked between two or more tables. This is achieved by having the foreign key (in the associated table) reference a primary key value (in base table). Because of this we need to ensure that data on both the sides of the relationship remain intact.

The referential integrity constraint states that, "whenever a foreign key value is used, it must reference a valid, existing primary key in the base table."

For example, consider two tables



In above relation, the registration/roll_no R555 is not existing, still if it is present in course table, then we say that it is violating referential integrity constraint

5) How the aggregate function and grouping are specifie -d in relational model? Explain 18CS53 July/Aug 2021

⇒ The aggregate functions are the functions where values of multiple rows are grouped together as input on certain criteria to form a single value. For example, finding average or finding total no of employee, such operations need to use aggregate function.

Various aggregate functions are,

⊛ count() ⊛ sum() ⊛ avg() ⊛ Min() ⊛ Max()

The aggregate function is denoted by using the operator $\mathcal{F}$. It is pronounced as script F.

Syntax of using aggregate function is,

$$\langle \text{grouping attributes} \rangle \mathcal{F} \langle \text{function list} \rangle (R)$$

where, <grouping attributes> is a list of attributes of the relation R and <function list> is a list of (<function> and <attribute>). In this case, <function> denotes any of the aggregate function and <attribute> denote the attribute of relation R.

Consider, a relation Student

Roll_No	Name	Marks
1	AAA	40
2	BBB	55
3	CCC	70
4	DDD	55
5	EEE	NULL

1) count() :- It will return total number of records.

$\mathcal{I}_{\text{COUNT ROLL-NO}}$ (STUDENT)

⇒ will return 5

2) SUM() :- It will return the SUM of the values present in the attribute. For example,

$\mathcal{I}_{\text{SUM marks}}$ (STUDENT)

⇒ will return 220

3) Avg() :- It will return average value. For example,

$\mathcal{I}_{\text{AVERAGE marks}}$ (STUDENT)

⇒ will return 44

4) Maximum :- It will return maximum value present in particular attribute.

For example, $\mathcal{I}_{\text{MAXIMUM marks}}$ (STUDENT)

⇒ will return 70

5) Minimum :- It will return minimum value present in particular attribute.

For example, $\mathcal{I}_{\text{MINIMUM marks}}$ (STUDENT)

⇒ will return 40

6) Define the following, ⊛ Relation State ⊛ Domain ⊛ Relation Schema ⊛ Arity 17CS53 AUG/Sept 2020

⇒ **Relation State** :-

A relation state 'r' of a relation schema R(A1, A2, ..., An) also denoted by r(R), is the set of n tuples. The relation state represents a relation with all the tuples at any particular point in time.

⊛ **Domain** :-

The domain is the datatype describing the type of values that can appear in each column.
For example: S_id Integer, Name varchar(10)

⊛ **Relation Schema** :-

A relation schema R, denoted by R(A1, A2, A3, ..., An) comprises a relation name R and a list of attributes. It is used to describe relation.
For ex:- Student with attributes id, name, city

⊛ **Arity** :-

The degree (or Arity) of a relation is the number of attributes of its relation schema. (or) it refers to no of columns in a table.
Ex:- Here, table has 3 columns, so its arity is 3.

Student

Sid	Name	City
1	A	XYZ
2	B	DEF

⇒ Example for relation Schema,
Student (id, Name, City)

7) Define the following with an examples

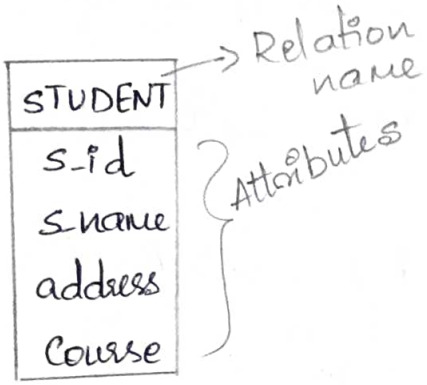
⊛ Key ⊛ Super Key ⊛ candidate Key ⊛ Primary key

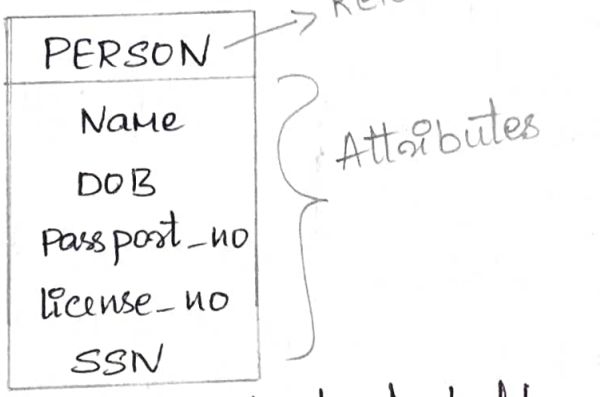
⊛ Foreign Key

18CS53 July/Aug 2021
17CS53 Dec/Jan 2020

⇒ ⊛ **Key** :- A key refers to an attribute or set of attributes that help to uniquely identify a tuple (or row) in a relation (table). It is also used to establish and identify relationships between tables.

Example :- Consider 2 tables,





s_id is used as a key in student table because it is unique for each student. In the PERSON table, passport_no, license_no, SSN are keys since they are unique for each person.

⊛ **Super key** :- A superkey is a set of these keys that identify a row or tuple uniquely. The word super denotes superiority of a key. Thus, a superkey is a superset of candidate key.

Example :- Consider an employee table

{SSN, Name} is super key

{Name, age} can't be super key

SSN	Name	Age
101	John	40
102	Mary	37
103	Cary	45
104	Riddle	55

Any set of attributes that includes SSN for example {SSN, Name, Age} or {SSN, Name} is a superkey. So, all these attributes in a table that is capable, of identifying other attributes of a table in a unique way are all super keys.

⊛ Candidate Key :-

In a relation schema, more than one attribute may identify a tuple uniquely. In that case, except for the primary key, the remaining attributes are called as candidate keys.

Example :- consider an employee table

E_id	E_name	License_no	e-mail
101	John	TVP-347	john@gmail.com
102	Mary	432-TFY	mary@gmail.com
103	Cary	RSK-629	cary@gmail.com

→ candidate keys

According to this example, E_id is best for primary key. The rest of the attributes like license_no and e-mail are considered as candidate key.

⊛ Primary Key :-

Primary Key is a key whose values are used to identify a tuple uniquely. The values of that attribute should be unique and not null. We use the convention that the attributes that form primary key are underlined.

Example :- Consider a student table

S_id	Name	Age	Phone_no
111	Hari	15	9876543210
222	Giri	16	8987536981
333	Raj	15	8871893521

Here, S_id is the primary key.

⊛ Foreign Key :-

Foreign keys are the column of the table used to refer to the primary key of another table.

Example :- Consider 2 tables department and employee

Department

Dept_id	D-name
1	Sales
2	Finance
3	Tech

↗ Primary key

Employee

E_id	E_name	Dept_id
101	Hari	1
102	Giri	2
103	Rishi	2
104	Rekha	3

↳ Foreign key

The table containing primary key is called parent table and the table containing foreign key is called child table.

8) Explain unary relational operators along with their syntax and Example. 15CS53 July/Aug

The unary operations are of two types

1. Selection operation
2. projection operation
3. Rename operation

1. Selection operation :-

SELECT operation is used to select a subset of the tuples from a relation that satisfy a selection condition. It is a filter that keeps only those tuples that satisfy a qualifying condition - those satisfying the condition are selected while others are discarded

Syntax :- The Syntax is

$\sigma_{predicate}(relation)$

where σ represents the select operation. The predicate denotes some logic using which the data from the relation (table) is selected.

- for Example - Consider the relation Student as follows :

Sid	Sname	age	gender
1	Ram	21	Male
2	Shyam	18	male
3	Seeta	16	Female
4	Geeta	23	Female

Query :- Fetch student with age more than 18
we can write it in relational algebra as

$\sigma_{age > 18}$ (Student)

the output will be -

Sname
Ram
Geeta

we can also specify conditions using and, or operation

$\sigma_{age > 18 \text{ and } gender = 'male'}$ (Student)

Sname
Ram

2. Projection:

* project operation is used to project only a certain set of attributes of a relation. That means if you want to see only the names all of the students in the student table, then you can use project operation
* Thus to display particular column from the relation, the projection operator is - used.

* It will only project or show the columns or attributes asked for, and will also removed duplicate data from the columns

* Syntax :

$\pi_{C1, C2 \ldots}(r)$

where C1, C2 etc. are attribute names (column names).

* for Example - Consider the student table given in Fig

Query : Display the name and age all the students

This can be written in relational algebra as

$\pi_{Sname, age}(Student)$

Above statement will show us only the Name and Age columns for all the rows of data in Student table.

Sname	age
Ram	21
Shyam	18
Seeta	16
Geeta	23

3. Rename operation

This operation is used to rename the output relation for any query operation which returns like select, project etc. Or to simply rename a relation (table). The operator ρ(rho) is used for renaming.

The rename operator is ρ. The general Rename operation can be Expressed by any of the following forms:

* $S(B_1, B_2 \ldots B_n)(R)$ is a renamed relations S based on R with column names $B_1, B_2 \ldots B_n$
* $S[R]$ is a renamed relation S based on R (which does not Specify column names).
* $(B_1, B_2 \ldots B_n)[R]$ is a renamed relation with column names $B_1, B_2 \ldots B_n$ which does not Specify a new relation name.

Syntax: ρ(RelationNew, RelationOld)

for Example: If you want to create a relation Student_names with Sid and Sname from student, it can be done using rename operator as:

$$\rho(\text{Student\_names}, (\pi_{\text{Sid\_name}}(\text{Student}))$$

14] Explain the steps in mapping from ER to relational schema. Discuss each step with example. 15CS53 July/Aug

OR 18CS53 Jan/Feb 2021, 17CS53 Aug/Sept 2020

Give the ER to relational mapping algorithm. Discuss each step with example. 15CS53 Jan/Feb, 17CS53 Dec/Jan 2020

OR 17CS53 Jan/Feb 2021

Summarize / Enumerate / Explain the steps involved in converting ER constructs to relational schema.

Basic steps to convert the Basic ER model to relational Database schema.

step 1: mapping of regular entity types

step 2: mapping of weak entity type

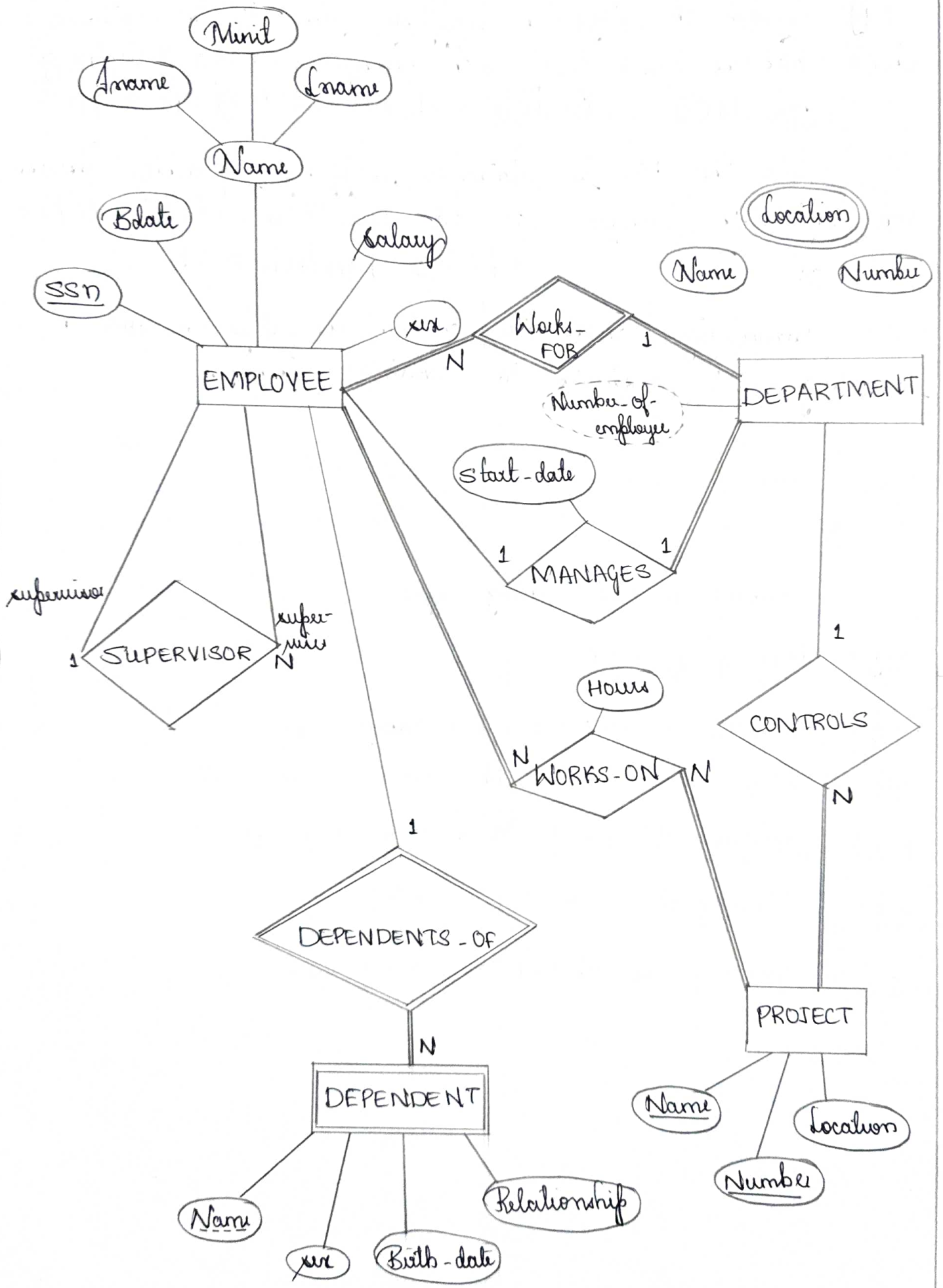
step 3: mapping of binary 1:1 relationship types

step 4: mapping of binary 1:N relationship types

step 5: mapping of binary M:N relationship type

step 6: Mapping of multivalued attribute

step 7: Mapping of N-ary Relationship types



E-R Diagram of company database

Step 1 : Mapping of Regular Entity Types

1] For each entity type, create a relation R that includes a simple attribute E

2] Include only simple component attribute

3] Choose one of the key attributes of E as primary key for R.

4] If the chosen key for E is a composite, then the set of simple attribute that form it will together as a primary key of R.

5] If multiple key were identified for E during conceptual design, the information during the attribute that form each additional key is kept in order to specify secondary key of relation R.

6] In this example - company database we have relation EMPLOYEE, DEPARTMENT, PROJECT

7] We choose ssn, Dnumber and Pnumber as primary key for relation EMPLOYEE, DEPARTMENT and PROJECT

8] The relation that are created from mapping of entity types called entity relation because each tuple represents an entity instance.

EMPLOYEE

Fname	Minit	Lname	ssn	Bdate	Address	sex	salary

DEPARTMENT

Dname	Dnumber

PROJECT

Pname	Pnumber	Plocation

Step 2 : Mapping of weak Entity Types

1] For each weak entity type, create a relation R and include all simple attributes of R.

2] Include primary key attribute as a foreign key attribute of R.

3] In our example, we create a relation DEPENDENT to correspond to weak entity type DEPENDENT

4] Include primary key of an of EMPLOYEE relation - which corresponding to owner entity type - as a foreign key attribute in DEPENDENT ; we rename it to Essn.

5] The primary key of DEPENDENT relation is the combination [Essn, Dependent_name] because DEPENDENT has Dependent_name as partial key.

6] It is common to choose the propogate (CASCADE) option for the relational triggered action on foreign key in relation corresponding to weak entity type, since a weak entity is dependent on owner entity

7] This can be used for both ON UPDATE and ON DELETE

DEPENDENT

ESSN	Dependent_name	Sex	Bdate	Relationship

Step 3 : Mapping of Binary 1:1 Relationship Types

1] For each binary 1:1 relationship type R in ER schema, identify the relations S and T that corresponds to entity types participating in R.

2] There are three possible approach

* Foreign key approach
* merged relationship approach
* cross reference or relationship relation approach

1] Foreign-key approach

1] Choose one of the relation - S, say - and include as a foreign key in S the primary key of T.

2] It is better to choose an entity type with total participation in R in the role of S.

3] Include all simple attributes of 1:1 relationship type by choosing relationship type R as attribute of S.

4] Here we map 1:1 relationship type by choosing the participating entity type DEPARTMENT to serve in role of S because its participation in MANAGES relation type is total

5] We include the primary key of EMPLOYEE relation as foreign key in DEPARTMENT relation and rename it Mgr-ssn.

6] We also include simplest attribute start-date of MANAGES relationship type in DEPARTMENT relationship and rename it Mgr-start-date.

2] Merged relation approach

1] merge the two entity types and relationship into a single relation

2] This is possible when both participation are total, as this would indicate that the two tables will have the same number of tuples at all times

3] Cross-reference or relationship relation approach.

1] set up a third relation R for purpose of cross-referencing the primary-keys of two relations S and T representing the entity types

2] required for binary M:N relationship

3] The relation R is called a relationship relation, because each

tuple in R represents a relationship instance that relates one tuple from S with one tuple from S

4] The relation will include the primary key attributes of S and T as foreign key to S and T.

5] The primary key of R will be one of the two foreign keys, and other foreign key will be unique key of R.

6] The drawback is having an extra relation, and requiring an extra join operation when combining related tuples from tables.

Step 4 : Mapping of Binary 1:N Relationship Types

1] For each regular binary 1:N relationship type R, identify the relation S that representing participating entity type at N-side of relationship

2] Include as foreign key in S the primary key of relation T that represent the other entity type participating in R.

3] Include any simple attribute of the 1:N relationship type as attribute of S.

4] In our example, we now map the 1:N relationship types Works-FOR, CONTROLS, SUPERVISION.

5] For WORKS-FOR we include primary key Dnumber of DEPARTMENT relation as foreign key in EMPLOYEE relation itself - and call it ~~super-ssn~~.

6] The SUPERVISION we include primary key of EMPLOYEE relation as foreign key in EMPLOYEE relation itself - because the relation is recursive - and call it Super-ssn.

7] The CONTROLS relation is mapped to foreign key attribute Dnum of PROJECT, with reference the primary Dnumber of DEPARTMENT

Step 5: Mapping of Binary M:N relationship Types.

1] For each binary M:N relationship type
   * create a new relation R
   * Include primary key of participating entity type as foreign key attribute in R
   * Include any simple attributes of M:N relationship type

2] In our example, we map M:N relationship type WORKS-ON by creating the relation WORKS-ON. We include primary key of PROJECT and EMPLOYEE relation is foreign key in WORKS-ON and rename them Pno and Eno, respectively.

3] We also include an attribute Hours in WORKS-ON to represent the Hours attribute of relationship type.

4] The primary key of the WORKS-ON relation is combination of foreign key attribute (Eno, Pno)

WORKS-ON

Eno	Pno	Hours

6] The propogate (CASCADE) option for the referential triggered action should be specified on foreign key in the relation corresponding to relation R, since each relationship instance has an existence dependency on each entity it relates. This can be used for both ON UPDATE and ON DELETE

Step 6: Mapping of multivalued attribute

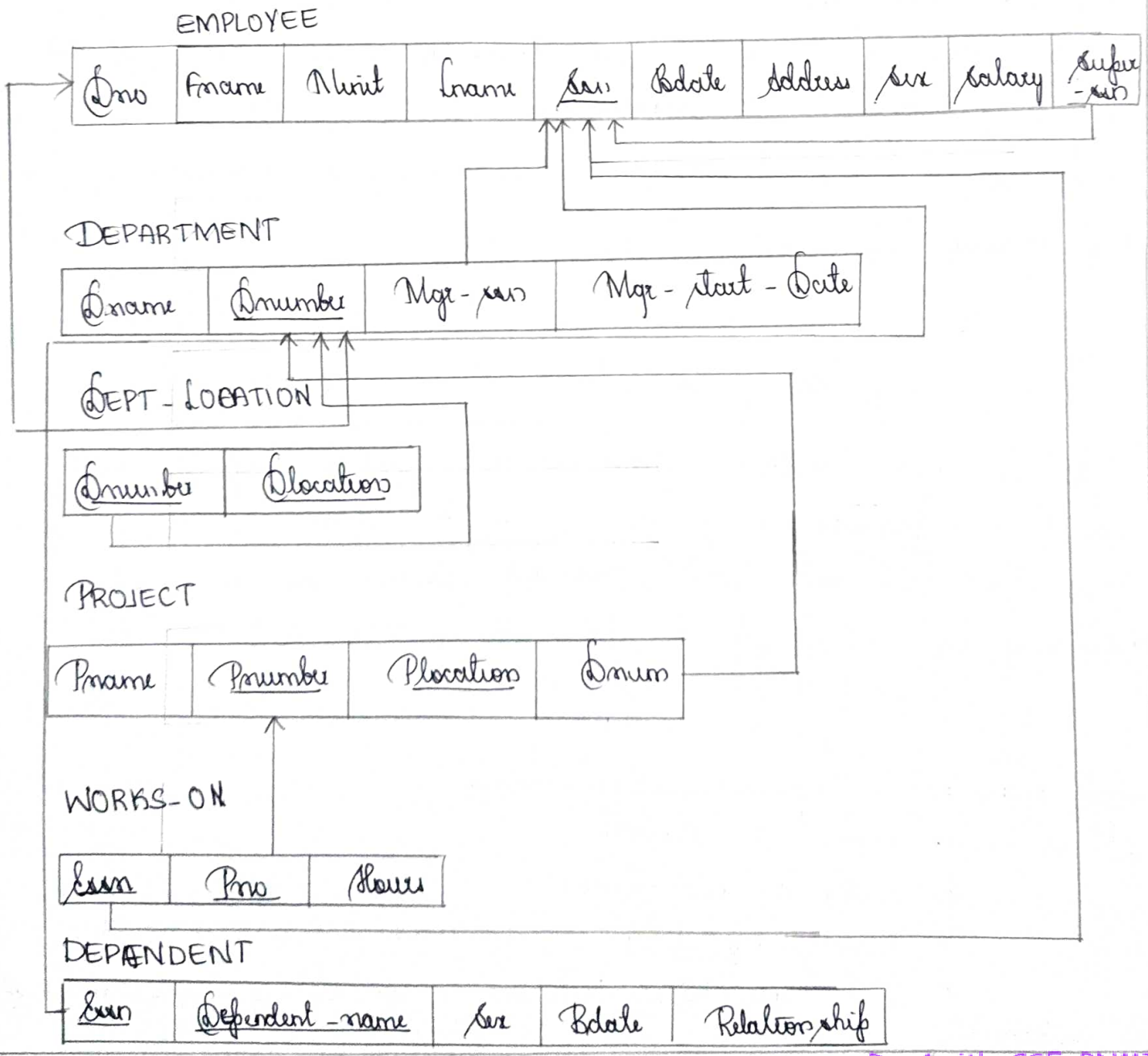
1] For each multivalued attribute
   * create a new relation
   * Primary key of R is the combination of A and K
   * If a multivalued attribute is composite, include its simple component.

2] In our example, we create a relation DEPT - LOCATIONS.

3] The attribute Dlocation represents the multivalued attribute LOCATIONS of DEPARTMENT, while Dnumber as foreign key represents the primary key of DEPARTMENT relation

4] The primary key of DEPT - LOCATIONS for each location department has.

5] The propogate (CASCADE) option for referential triggered action should be specified on foreign key in relation R corresponding to multivalued attribute for both ON UPDATE and ON DELETE

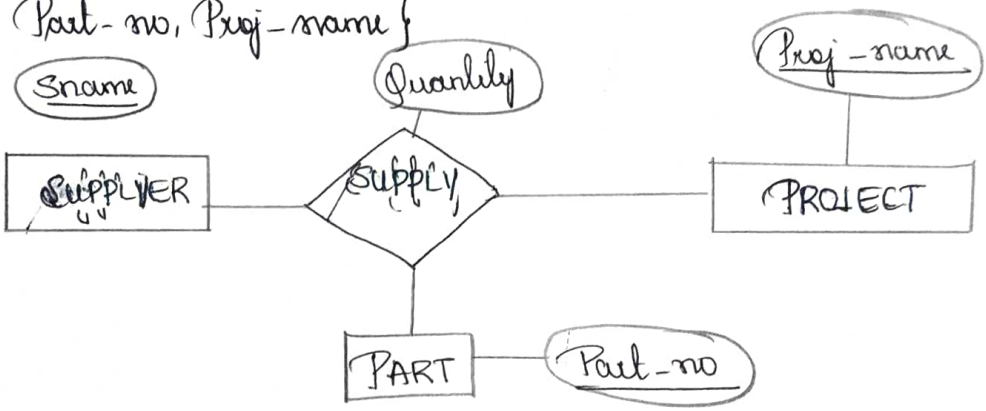


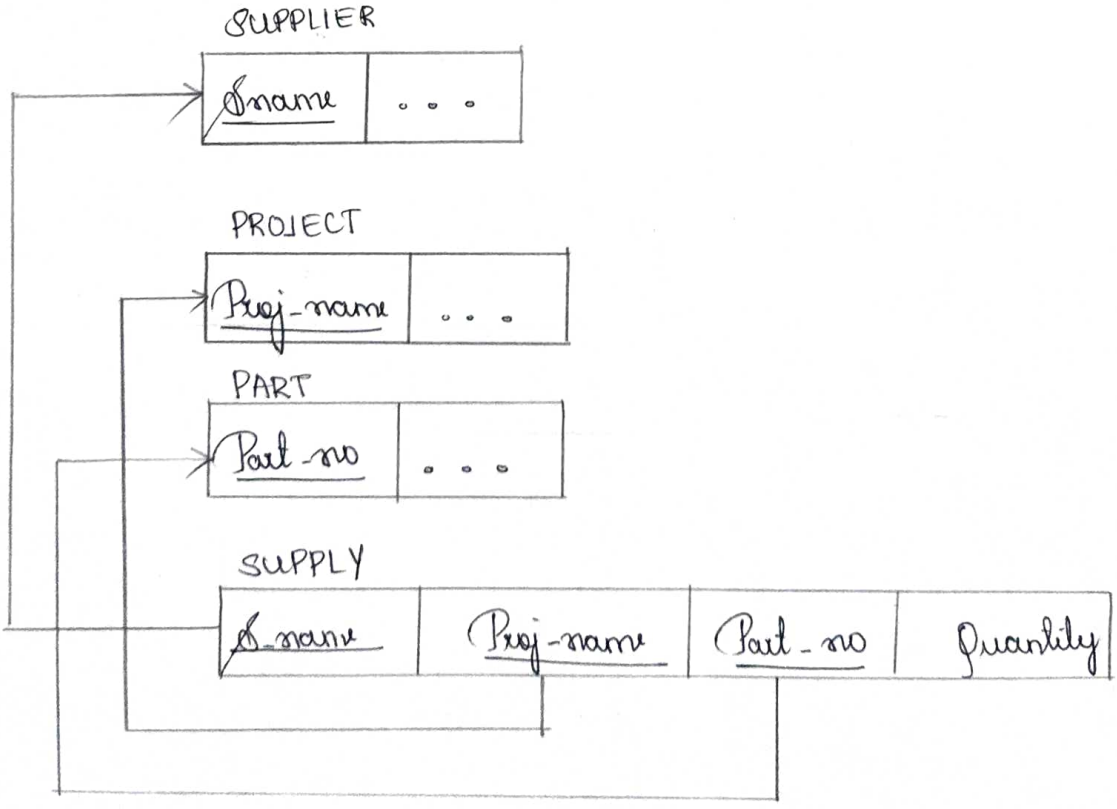
step 7 : Mapping of N-ary Relationship Types

1] For each n-ary relationship type R
   * create a new relationship S to represent R.
   * Include primary key of participating entity type as foreign keys
   * Include any simple attribute

2] The primary key of S is usually combination of all foreign keys that refrence the relation representing the participating entity

3] For eg; consider relation type SUPPLY. This can be mapped to the relation SUPPLY whose primary key is combination of three foreign key { Sname, Part-no, Proj-name }





17) Explain with example, the basic constraints that can be specified when a database table is created in SQL.
17CS53 DEC/JAN 2020.

⇒ Constraints are the conditions that can be enforced on the attributes of a relation.

There are various types of constraint supported in sql. They are,

*) NOT NULL
*) Unique constraint
*) Primary key constraint
*) Foreign key constraint
*) Default constraint
*) check constraint
*) Domain constraint

*) NOT NULL :-

NOT NULL constraint makes sure that a column doesnot hold NULL value. When we don't provide value for a particular column while inserting a record into table, it takes NULL value by default. So, by specifying NOTNULL constraint, we make sure that particular column cannot have NULL value.

Example :-
```
CREATE TABLE STUDENT(
    ROLL_NO  INTEGER  NOTNULL,
    NAME     VARCHAR(20),
    AGE      INTEGER,
    PRIMARY KEY (ROLL_NO));

INSERT INTO STUDENT VALUES (NULL, 'DHRUV');
```
⇒ column 'ROLL_NO' cannot be NULL - error

*) **Unique** :-

UNIQUE constraint is used to prevent same values. In the employee table for example you might want to prevent two or more employees from having an identical designation. Then, ith that case, we must use unique constraint.

Example :-
```
CREATE TABLE EMPLOYEE(
    EID INTEGER NOTNULL,
    NAME VARCHAR(20) NOTNULL,
    DESIGNATION VARCHAR(20) NOTNULL UNIQUE,
    PRIMARY KEY (EID));

INSERT INTO EMPLOYEE VALUES (111,'Rashmi','Manager);
INSERT INTO EMPLOYEE VALUES (222,'Riya','Manager');
```
⇒ when you try to again insert with same designation you will get an error as "duplicate entry 'MANAGER' for key 'DESIGNATION'.

*) **Primary key constraint** :-

The primary key constraint is defined to uniquely identify the records from the table. The primary key must contain unique values. Hence, primary key should be chosen carefully.

For example,
```
CREATE TABLE PERSON(
    AADHAR_NO  INTEGER,
    NAME       VARCHAR(20),
    CITY       VARCHAR(20),
    PRIMARY KEY (AADHAR_NO));

INSERT INTO PERSON VALUES (3765,'Raksha','DVG');
INSERT INTO PERSON VALUES (3765,'Rohini','BLG');
```
error will be thrown :- duplicate entry '3765' for key 'Primary'

*) **Foreign Key constraint** :-

Foreign key for one table, is actually a primary key of another table. This constraint is used to enforce referential integrity constraint.

Example :- Consider 2 tables.

Employee

E_id	Name	Age
1	Rajesh	30
2	Sharman	23
3	Tushar	20

Department

D_ID	D_Name	EID
1	Accounts	3
2	Production	3
3	Sales	2
4	Purchase	1 ..

The E_id column in employee table is primary key and E_id in department table is a foreign key. The foreign key constraint is used to prevent actions that would destroy links between tables.
The department table can be created as,

```
CREATE TABLE DEPT (
D_ID INTEGER,
D_NAME VARCHAR(20),
E_ID INTEGER,
PRIMARY KEY (D_ID),
FOREIGN KEY (EID) REFERENCES EMPLOYEE (E_ID));
```

*) **Default constraint** :-

The default constraint provides a default value to a column when there is no value provided while inserting a record into table.

```
CREATE TABLE STUDENT(
 Roll_no    INTEGER  NOTNULL, PRIMARY KEY,
 NAME       VARCHAR(20),
 AGE        INTEGER,
 EXAM_FEE   INTEGER  DEFAULT 1000);

INSERT INTO STUDENT(ROLL_NO, NAME, AGE) VALUES
(111, 'MEGHA', 18);
```

O/P :-

ROLL_NO	NAME	AGE	EXAM_FEE
111	MEGHA	18	1000

↳ default value

*) **check constraint** :-

This constraint is used for specifying range of values for a particular column of a table. When this constraint is being set on column, it ensures that specified column must have value falling in specified range.

Example :-
```
CREATE TABLE ORDERS(
    OID INTEGER PRIMARY KEY,
    AMT INTEGER CHECK (AMT>0));

INSERT INTO ORDERS VALUES (55, 1000);
INSERT INTO ORDERS VALUES (73, -50);
```

When we give amt < 0 then it will return error, as check constraint 'ORDERS_chk_1' is violated.

*) **Domain constraint** :-

They are used defined columns that help the user to enter value according to datatype. And if it encounters a wrong value it gives message to user that the column is not fulfilled properly.

Domain Constraint = datatype (int/character/time/string etc) + Constraints (NOT NULL, Unique, default, primary Key, Foreign Key)

20) Given, the Schema

Passenger (pid, pname, pgender, pcity)
Agency (aid, aname, acity)
Flight (fid, fdate, time, src, dest)
Booking (pid, aid, fid, fdate)

i) Get the complete details of all flights to new Delhi

$$\sigma_{dest='NewDelhi'}(Flight)$$

ii) Find only the flight numbers for passanger with pid 123 for flights to chennai before 06/11/2020

$$\pi_{f\_id}\left[\sigma_{P\_id=123 \wedge dest='chennai' \wedge fdate < date('6/11/2020')}(Flight \bowtie Booking)\right]$$

iii) Find the passanger names for those who do not have any bookings in any flights.

$$\pi_{P\_name}\left(\sigma_{f\_id='NULL'}(Passenger \bowtie Booking)\right)$$

iv) Get the details of flights that are scheduled on both dates 01/12/2020 and 02/12/2020 at 16:00 hrs

$$\sigma_{fdate=date('1/12/2020') \wedge time=16:00hrs}(Flight) \cap \sigma_{fdate=date('2/12/2020') \wedge time=16:00hrs}(Flight)$$

v) Find the details of Male Passengers who are associated with jet agency.

$$\pi_{Pname,\ Pid,\ Pgender,\ Pcity}\left(\sigma_{Pgender='Male' \wedge aname='jet'}\ (Agency \bowtie Booking) \bowtie Passenger\right)$$

---

21) Consider, the following SAILORS Database

Sailors (Sid, Sname, rating, age)
Boats (bid, bname, color)
Reserves (sid, bid, day)

i) Find names of sailors who have reserved green boat

$$\pi_{sname}\left(\sigma_{color='green'}\ (Sailors \bowtie Reserves) \bowtie Boats\right)$$

ii) Find the names of sailors who have reserved all boats.

$$\pi_{sname}\left(\pi_{sid,bid}(Reserves) \div \pi_{bid}(Boats) \bowtie Sailors\right)$$

iii) Find names of sailors who have reserved, boat 103

$$\pi_{sname}\left(\sigma_{bid=103}\ (Sailors \bowtie Reserves)\right)$$

iv) Retrive Sailors names who have reserved red and green

$$\pi_{sname}\left(\sigma_{color='green' \wedge color='red'}\left(Sailors \bowtie Reserves\right) \bowtie Boats\right)$$

v) Retrive the color of boat reserved by Raj

$$\pi_{color}\left(\sigma_{Pname='Raj'}\left(Sailors \bowtie Reserves\right) \bowtie Boats\right)$$

vi) Retrive the SIDs of sailors with age > 20, who have not reserved a red boat.

$$\pi_{sid}\left(\sigma_{age>20} Sailors\right) - \pi_{sid}\left(\sigma_{color='red'} Boats \bowtie Reserves\right)$$

vii) Retrive the sailor name with age > 20 years and reserved black boat.

$$\pi_{sname}\left(\sigma_{age>20 \wedge color='black'}\left(Sailors \bowtie Boats\right) \bowtie Reserves\right)$$

viii) Retrive sailor names who have reserved green boat on Monday

$$\pi_{sname}\left(\sigma_{color='green' \wedge day='Monday'}\left(Boats \bowtie Reserves\right) \bowtie Sailors\right)$$

ix) Retrive the number of boats which are not reserved.

$$\mathcal{F}_{COUNT(Bid)}\left(\pi_{Bid}(Boats) - \pi_{Bid}(Reserves)\right)$$

x) Retrive the sailors names who is the oldest sailor with rating 10.

$$\pi_{sname}\left(\sigma_{\mathcal{F}_{MAX(age)} \wedge rating=10}(Sailors)\right)$$

xi) Find names of sailors who have reserved a red boat.

$$\pi_{sname}\left(\sigma_{color='red'}(Sailors \bowtie Reserves) \bowtie Boats\right)$$

xii) Find names of sailors who have reserved a red or green boat.

$$\pi_{sname}\left(\sigma_{color='red' \vee color='green'}(Sailors \bowtie Reserves) \bowtie Boats\right)$$

22] Discuss Division operation. Find the quotient for the following: A / B1, A/B2 and A/B3 where A, B1, B2 and B3 are

17CS53 Jan/Feb 2021

SNo	PNo
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4

A =

B1 =

PNo
P2

B2 =

PNo
P2
P4

B3 =

PNo
P1
P2
P4

Sol:

The Division operator is used when we have to evaluate queries which contain the key word 'ALL'. It is denoted by A/B where A and B are instances of relation.

For example: Find all the customer having accounts in all the branches. For that consider two tables customer and Accounts.

Customer

Name	Branch
A	Pune
B	Mumbai
A	Mumbai
C	Pune

Account

Branch
Pune
Mumbai

Now A/B will give us

Name
A

Here we check all branches from Account table against all the names from customer table. We can then find that only customer A has all the accounts in all the branches.

Formal Definition of Division Operation:

The operation A/B is defined as the set of all x values (in the form of unary tuples) such that for every y value in (a tuple of) B, there is a tuple $<x, y>$ in A.

Solution for the given question:

① A/B1

SNo
S1
S2
S3
S4

② A/B2

SNo
S1
S4

③ A/B3

SNo
S1

SQL Queries

① For the following relations for a book club:

Members (Member-id , Name , Designation , Age)

Books (Book id , Book-Title , Book-Author , Book-Publisher, Book price)

Reserves (Member-id , Book-id , Date)

write the SQL queries:

i) Find the names of members who are professors older than 45 years

```
SELECT  NAME
FROM    MEMBERS
WHERE   DESIGNATION = 'PROFESSOR'
AND     AGE > 45 ;
```

ii) Find Id's of member who have ~~not~~ reserved books that cost more than Rs. 500

```
SELECT MEMBER_ID
FROM BOOKS B, RESERVES R
WHERE B.BOOK_ID = R.BOOK_ID
AND BOOK-PRICE > 500

OR
SELECT MEMBER_ID
FROM BOOKS NATURAL JOIN RESERVES
WHERE   BOOK-PRICE > 500
```