# Module - 2

1. List and Explain characteristics of Relation.

OR

Describe any two characteristics of relations with suitable example for each.

OR

Discuss characteristics of relation that make them different from ordinary tables.

17CS53 Jan/Feb 2021
17CS53 July/Aug 2021
-11- Aug/Sept 2020

There are mainly four characteristics of Relation

① Ordering of Tuples in a Relation.
② Ordering of values within a Tuple and an Alternative defination of a relation.
③ Values and Null in the tuples.
④ Interpretation (Meaning of a relation)

## ① Ordering of Tuples in a Relation

A relation is defined as a set of tuples. Mathematically, elements of a set have no order among them. Hence, tuples in a relation do not have any particular order. Tuple ordering is not a part of a relation definition because a relation attempts to represent facts as a logical or abstract level. Many tuple orders can be specified on the same relation.

For example:
Tuples in Student relation can be ordered based on roll no, name, age and some other attribute.

② Ordering of values within a Tuple and Alternative Definition of a Relation

The ordering of values (attributes) in relation schema is important. However at a logical level, the order of attributes and their values is not that important as long as the correspondence between attributes and values is maintained.

An alternative definition can be given such that the ordering of tuples in a relation is not necessary.
For example: Consider the following schema -

| RollNo | Name | Marks | Phone |
|---|---|---|---|
| 001 | AAA | 88 | 1111111111 |
| 002 | BBB | 83 | 22222222 |
| 003 | CCC | 98 | 33333333 |

$t = <(RollNo, 003), (Name, CCC), (Marks, 98), (Phone, 333333)>$

$t = <(Name, CCC), (RollNo, 003), (Phone, 333333), (Marks, 98)>$

These two tuples are same as the corresponding attribute and value pair is maintained in each tuple.

## ③ Values and Null in Tuple

Each value within a tuple is atomic. That means it is not divisible into components within the framework of the basic relational model. Hence, composite and multivalued attributes are not allowed. This model is also called the 'flat relational model'.

Nulls are those values which are used to represent the values of attributes that may be unknown or may not apply to a tuple. The null is a special value.

For example: some STUDENT tuples have Null for their office phones because they do not have an office.

## ④ Interpretation of Relation

The relation schema can be interpreted as a declaration or as a type of assertion. For example the schema of the STUDENT relation as given asserts that student entity has Roll no, Name, Marks and Phone number. Each tuple in the relation can be interpreted as a fact or a particular instance of the assertion.

For example: The first tuple in the Student relation is a student whose Roll no is 001, name is AAA, marks are 88 and phone number is 11111111.

2) List Set theory operations used in relation data model. Explain any two with examples.

17 CS 53 Aug/sept 2021

Various set operations are - union, Intersection and set difference.

① Union

- This operation is used to fetch data from two relations (tables) or temporary relation (results of another operation).
- For this operation to work, the relations (tables) specified should have same number of attributes (columns) and same attribute domain. Also the duplicate tuples automatically eliminate from the result.

Syntax: $A \cup B$ where A & B are relations.

For example: If there are two tables student and book as follows.

Student

| Sid | Sname | age |
|---|---|---|
| 1 | Ram | 21 |
| 2 | Shyam | 18 |
| 3 | Seeta | 16 |

Book

| Isbn | bname | Author |
|---|---|---|
| 005 | DBMS | XYZ |
| 006 | OS | PQR |
| 007 | DAA | ABC |

Query: We want to display both student name & book names from both the tables then

$$\Pi_{sname}(Student) \cup \Pi_{bname}(Book)$$

## ② Intersection:

- This operation is used to fetch data from both tables which is common in both the tables.
- Syntax : $A \cap B$ where A & B are relations.
- Example: Consider two tables - Student and Worker

Student

| Name | Branch |
|---|---|
| AAA | Computer |
| BBB | Mechanical |
| DDD | Electrical |

Worker

| Name | Salary |
|---|---|
| XXX | 3000 |
| AAA | 2000 |
| DDD | 2500 |

- Query: If we want to find out the names of the students who are working in a company then

$$\Pi_{name}(Student) \cap \Pi_{name}(Worker)$$

| Name |
|---|
| AAA |
| DDD |

## ③ Set-Difference:

- The result of set difference operation is tuples, which are present in one relation but are not in the second relation.
- Syntax : A - B

- For Example : Consider two relations Full_Time_Employee and Part_Time_Employee if we want to find out all the employee working for Fulltime, then the set difference operator is used

$$\Pi_{EmpName}(Full\_Time\_Employee) - \Pi_{EmpName}(Part\_Time\_Employee)$$

3) Briefly discuss the different types of Update Operations on relational database. Show an example of a violation of the referential integrity in each of the update operations. 17CS53 July/Aug 2021

- The data manipulation operations are - Insert, Delete and Update (or Modify).
- The Insert operation inserts new data in the database delete operation deletes some data from the database and update operations makes some changes in the data present in the database. Whenever these operations are applied on the database, the integrity constraints specified on the relational database must not be violated.

① Insert Operation

Consider the following tables

Student

| RegNo | Name |
|---|---|
| R101 | AAA |
| R102 | BBB |
| R103 | CCC |

Course

| CId | Cname | RegNo |
|---|---|---|
| C11 | CS | R103 |
| C12 | Civil | R101 |
| C13 | IS | R102 |

If we perform

Insert on Course => Insert (C14, EEE, R105)

then this operation will be rejected as it violates referential integrity constraint. If the value of any foreign key in it refers to a tuple that does not exist in the referenced relation, then the violation takes place. Here R105 does not exist in parent table student. Hence if insertion violates one or more integrity constraints then the default option is to reject the insert operation.

## ② Delete operation:

Student

| RegNo | Roll No | Name | Marks |
|---|---|---|---|
| R101 | 001 | AAA | 88 |
| R102 | 002 | BBB | 83 |
| R103 | 003 | CCC | 98 |
| R104 | 004 | DDD | 67 |

Course

| C_ID | C_Name | RegNo |
|---|---|---|
| C111 | Computer | R103 |
| C112 | Electrical | R101 |
| C113 | Civil | R101 |
| C114 | Mechanical | R102 |

If we perform

Delete on student table where RegNo = R101

This violates the referential integrity constraint. Because there are two tuples in Courses table that may be affected if we delete the RegNo = R101 (assuming that we have to delete only one student record)

The delete operation can violate only referential integrity.
There are several options available if a deletion operation violates referential integrity.

i) One option is called restrict - it rejects the delete operation.

ii) Second option is set null or set default. This option either sets the value to NULL or changes the reference to valid tuples.

iii) Third option is to cascade or propagate the deletion by deleting tuples that reference the tuple that is being deleted.

## ③ Update :

This operation makes changes in the values of one or more attributes of tuples.

For example - Consider two tables Student & Course

Student

| RegNO | RollNo | Name | Marks |
|---|---|---|---|
| R101 | 001 | AAA | 93 |
| R102 | 002 | BBB | 95 |
| R103 | 003 | CCC | 87 |
| R104 | 004 | DDD | 63 |

Course

| C_ID | C_Name | Reg No |
|---|---|---|
| C111 | Computer | R103 |
| C112 | Civil | R101 |
| C113 | Electrical | R104 |
| C114 | IS | R102 |

If we apply

Update on RegNo of Course with C_ID = C112 to R555

The result of above operation is rejected as it

violates referential integrity. The R555 registration number is not in existence in the parent table Student.

Updating an attribute that is neither part of a primary key nor of a foreign key usually causes no problems.

The DBMS need only check to confirm that the new value is of the correct datatype and domain.

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 |

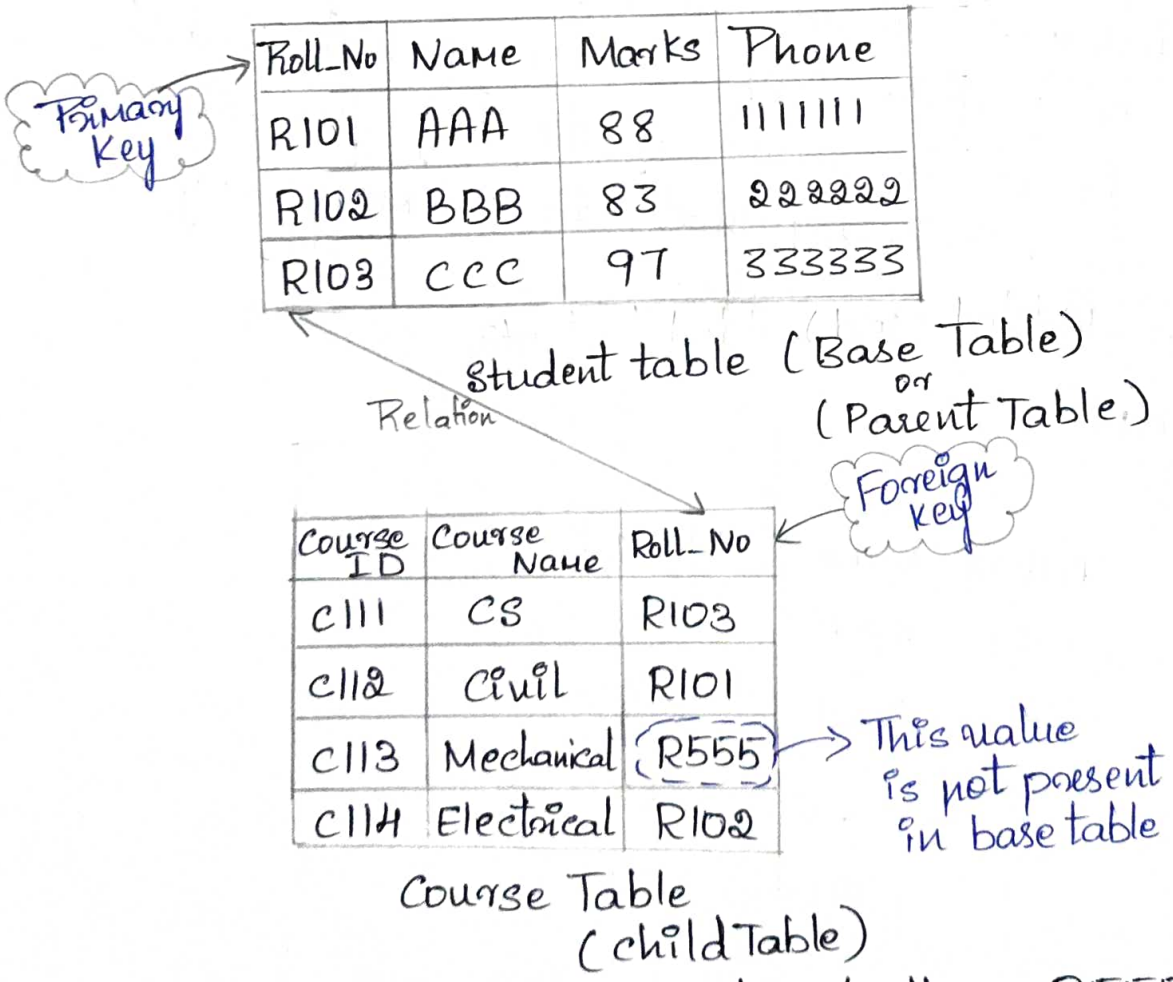
NULL is not allowed

## 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

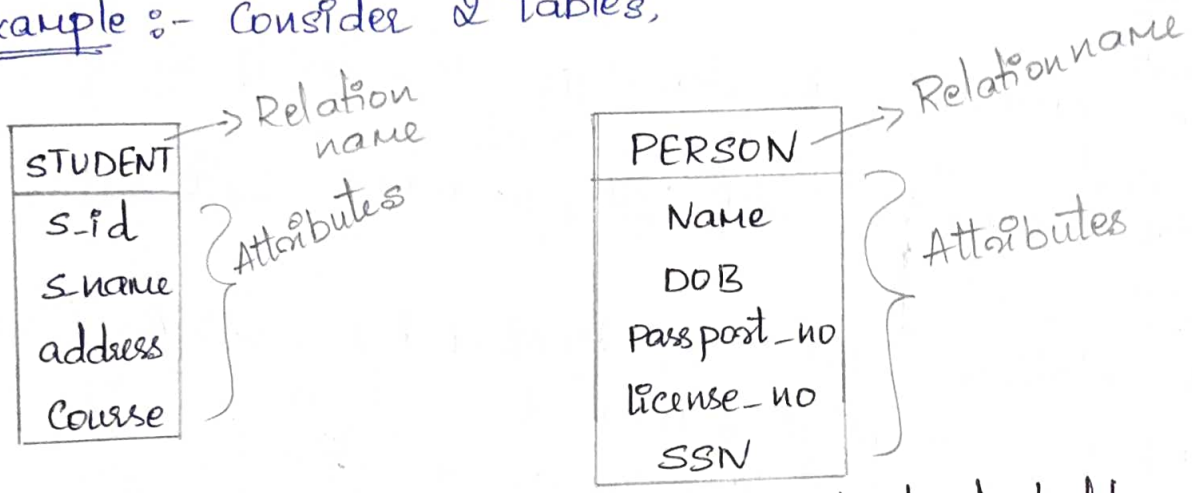
⊛ Key ⊛ Super Key ⊛ candidate Key ⊛ Primary key

⊛ Foreign Key

18CS53 July/Aug 2021
17CS53 Dec/Jun 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 $\sigma$ 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 $\rho$(rho) is used for renaming.

The rename operator is $\rho$. 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: $\rho$(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:

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

9) Discuss equijoin and natural join with suitable example using relational algebra notation.

OR 18CS53 Jan/Feb 2021

10) Explain with example left Outer join and right Outer join. 15CS 53 July/Aug

OR

11) Explain the various inner join operations in relational algebra with examples. 18CS 53 July/Aug 2021

OR

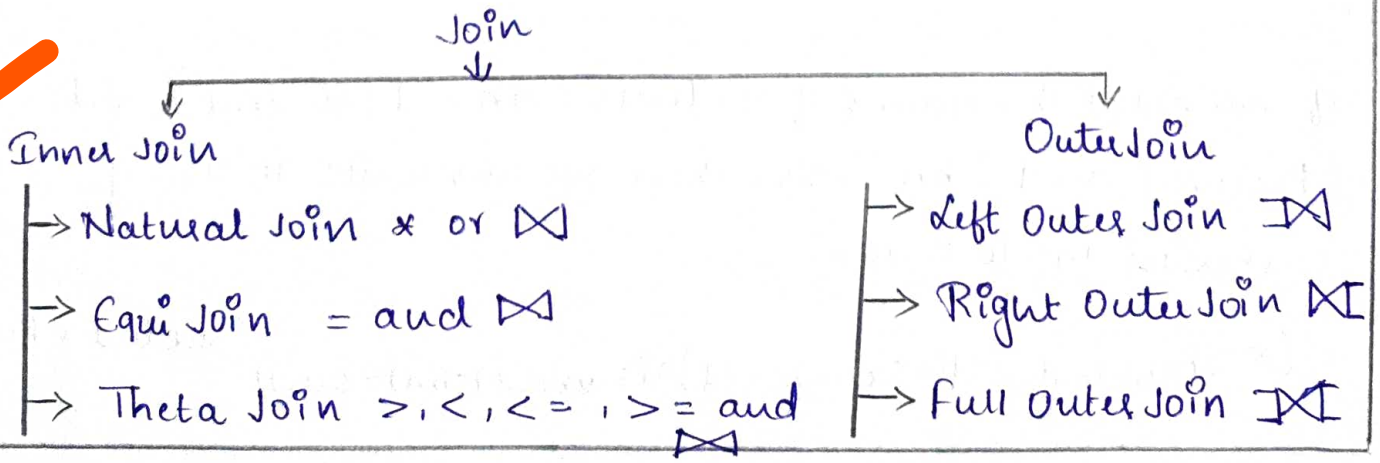
12) Discuss the various types of JOIN operations with an example. Why is THETA join required? 17CS53 Aug/Sept 2020

## The Join Operation

The Join Operation is used to combine information from two or more relations.

Formally join can be defined as a cross-product followed by selections and projections, Joins arise much more frequently in practice than plain cross-products. The Join operator is used as ⋈.

There are two types of Join Operations - Inner Join and Outer Join.



## 1. The Inner Join Operation

Inner Join is used to return rows from both tables which satisfy the given condition. It is the most widely used Join operation and can be considered as a default Join-type. There are three types of inner Joins used in relational algebra.

i) **Theta Join** : This is an operation in which information from two tables is combined using some condition and this condition is specified along with the Join operator.

$$A \bowtie_c B = \sigma_c (A \times B)$$

Thus $\bowtie$ is defined to be a cross-product followed by a selection. Note that the condition c can refer to attributes of both A and B. The condition C can be specified using $<, <=, >=, >$ or = operators.

Student

| Sid | Sname | age |
|---|---|---|
| 1 | Ram | 21 |
| 2 | Shyam | 18 |
| 3 | Seeta | 16 |
| 4 | Geeta | 23 |

Reserve

| Sid | Isbn | day |
|---|---|---|
| 1 | 005 | 07-07-18 |
| 2 | 005 | 03-03-17 |
| 3 | 007 | 08-11-16 |

If we want the names of students with sid (Student) = sid (Reserve) and isbn = 005, then we can write it using cartesian product as -

$$(\sigma_{((Student.sid = Reserve.sid) \wedge (Reserve.(Isbn) = 005))} (Student \times Reserve))$$

Here there are two conditions as

i) (Student.sid = Reserve.sid) and ii) (Reserve.isbn = 005) which are joined by ∧ Operator.

Now we can use $\bowtie_c$ instead of above statement and write it as -

(Student $\bowtie$ (Student.sid = Reserve.sid) ∧ (Reserve.(Isbn) = 005) Reserved))

The result will be -

| sid | Sname | age | isbn | day |
|---|---|---|---|---|
| 1 | Ram | 21 | 005 | 07-07-18 |
| 2 | Shyam | 18 | 005 | 03-03-18 |

ii) Equijoin : This is a kind of join in which there is equality condition between two attributes (columns) of relations (tables). For example - If there are two table Book and Reserve table and we want to find the book which is reserved by the student having isbn 005 and name of the book is 'DBMS' then :

Book

| isbn | bname | Author |
|---|---|---|
| 005 | DBMS | XYZ |
| 006 | OS | PQR |
| 007 | DAA | ABC |

Reserve

| sid | isbn | day |
|---|---|---|
| 1 | 005 | 07-07-18 |
| 2 | 005 | 03-03-17 |
| 3 | 007 | 08-11-16 |

$(\sigma_{bname = 'DBMS'} (Book \bowtie_{(book.isbn = Reserve.isbn)} Reserve))$

Then we get

| isbn | bname | Author | sid | day |
|---|---|---|---|---|
| 005 | DBMS | XYZ | 1 | 07-07-18 |
| 005 | DBMS | XYZ | 2 | 03-03-18 |

iii) **Natural Join** : When there are common columns and we have to equate these common columns then we use natural join. The symbol for natural join is simply $\bowtie$ without any condition or sometimes * is used. For example, consider two tables -

Book

| isbn | bname | Author |
|---|---|---|
| 005 | DBMS | XYZ |
| 006 | OS | PQR |
| 007 | DAA | ABC |

Reserve

| sid | isbn | day |
|---|---|---|
| 1 | 005 | 07-07-18 |
| 2 | 007 | 03-03-17 |
| 3 | 006 | 08-11-16 |

Now if we want to list the books that are reserved, then that means we want to match Books.isbn. Hence it will be simply

$$Books \bowtie Reserve$$

OR

$$Books * Reserve$$

The result of above two tables on natural join will be

| isbn | bname | Author | Sid | day |
|---|---|---|---|---|
| 005 | DBMS | XYZ | 1 | 07-07-18 |
| 006 | OS | PQR | 3 | 08-11-16 |
| 007 | DAA | ABC | 2 | 03-03-17 |

## ② Outer Join Operation

An Outer Join doesn't require each record in the two join tables to have a matching record. In this type of join, the table retains each record even if no other matching record exists.

There are three types of outer joins - Left Outer Join,
Right Outer Join and
Full Outer Join.

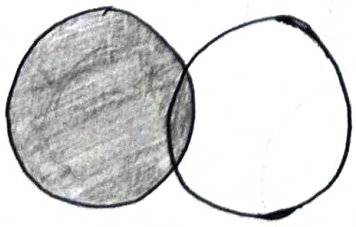
### i) Left Outer Join :

→ This is a type of join in which all the records from left table are returned and the matched records from the right table gets returned.

→ The result in NULL from the right side, if there is no match.

→ The symbol used for left outer join is ⟕

→ This can be graphically represented as follows.



All rows from left table and
only matching rows from right table.

→ For example - Consider two tables Student and Course as follows -

Student

| Roll No | Name |
|---|---|
| 1 | AAA |
| 2 | BBB |
| 3 | CCC |

Course

| Roll No | Course Name |
|---|---|
| 1 | Computer |
| 3 | Electrical |
| 5 | Civil |
| 6 | Mechanical |

The result of left Outer join will be

| Roll No | Name | Course Name |
|---|---|---|
| 1 | AAA | Computer |
| 2 | BBB | NULL |
| 3 | CCC | Electrical |

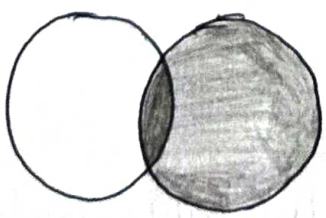
### ii) Right Outer join

→ This is a type of join in which all the records from right table are returned and the matched records from the left table gets returned.

→ The result in NULL from the left side, if there is no match.

→ The symbol used for right Outer join is ⟖

→ This can be graphically represented as follows.



All rows from right table and only matching rows from left table

• For example - Consider two tables Student and Course as follows-

Student

| RollNo | Name |
|---|---|
| 1 | AAA |
| 2 | BBB |
| 3 | CCC |

Course

| RollNo | Course Name |
|---|---|
| 1 | Computer |
| 3 | Electrical |
| 5 | Civil |
| 6 | Mechanical |

The result of right Outer Join will be

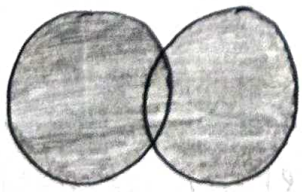
| RollNo | Course Name | Name |
|---|---|---|
| 1 | Computer | AAA |
| 3 | Electrical | CCC |
| 5 | civil | NULL |
| 6 | Mechanical | NULL |

### iii) Full Outer Join

→ In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.

→ It is denoted by ⟗

→ Graphically it can be represented as



All rows from both tables

For example -

-> Consider two tables Student and Course as follows -

Student

| RollNo | Name |
|---|---|
| 1 | AAA |
| 2 | BBB |
| 3 | CCC |

Course

| RollNo | Course Name |
|---|---|
| 1 | Computer |
| 3 | Electrical |
| 5 | Civil |
| 6 | Mechanical |

The result of full outer join will be

| RollNo | Name | Course Name |
|---|---|---|
| 1 | AAA | Computer. |
| 2 | BBB | NULL |
| 3 | CCC | Electrical |
| 5 | NULL | Civil |
| 6 | NULL | Mechanical |

The Symbols used are -

| | |
|---|---|
| A LEFT OUTER JOIN B ON $x=y$ | $A ⟕_{x=y} B$ |
| A RIGHT OUTER JOIN B ON $x=y$ | $A ⟖_{x=y} B$ |
| A FULL OUTER JOIN B ON $x=y$ | $A ⟗_{x=y} B$ |

(13) Discuss the following relational algebra operation. Illustrate with an example for i) Join ii) Difference iii) select iv) union 15CS53 Jan/Feb

Refer Question no. (2) union & set difference and (8) for select and (9) for Join

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

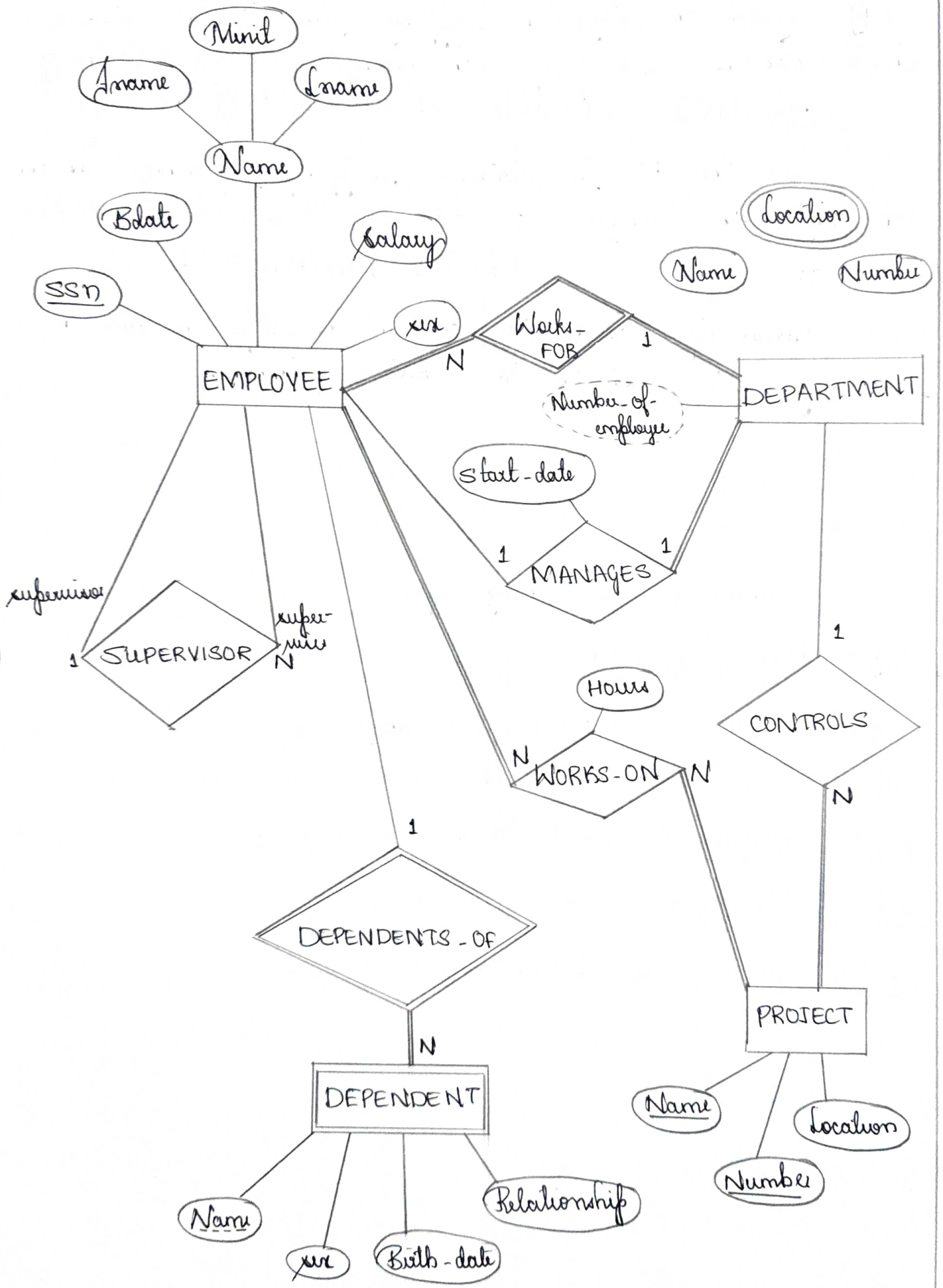
step 3: mapping of binary 1:1 relationship types

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

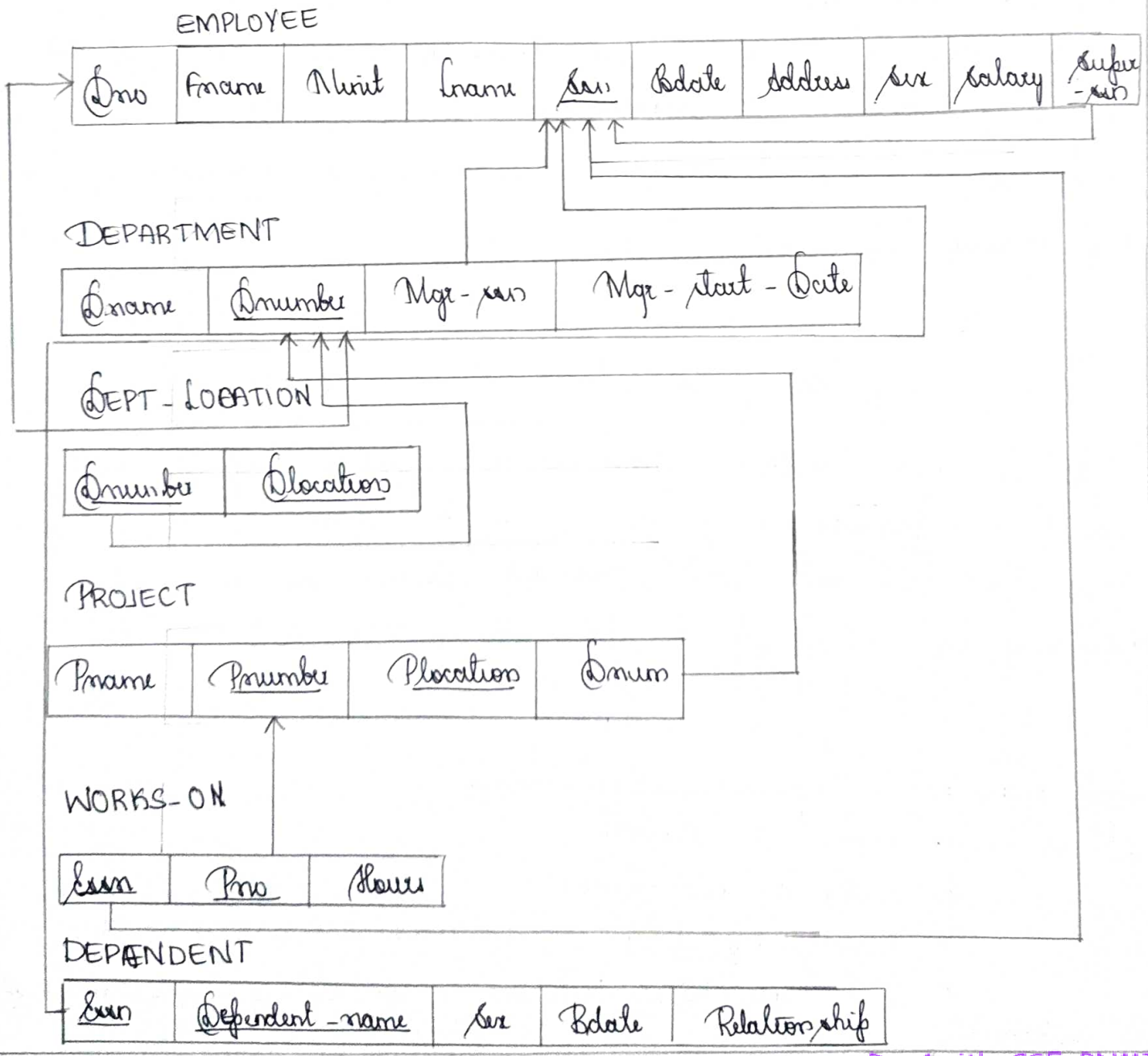
1] For each multivalued attribute
   * create a new relation
   * Primary key of R is the combination of A and K
   * If a multivalued attribute is composite, include its simple component.

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

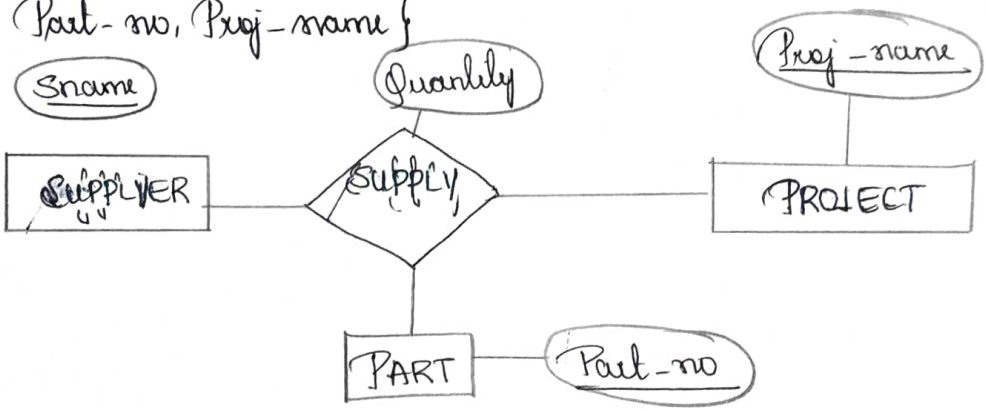


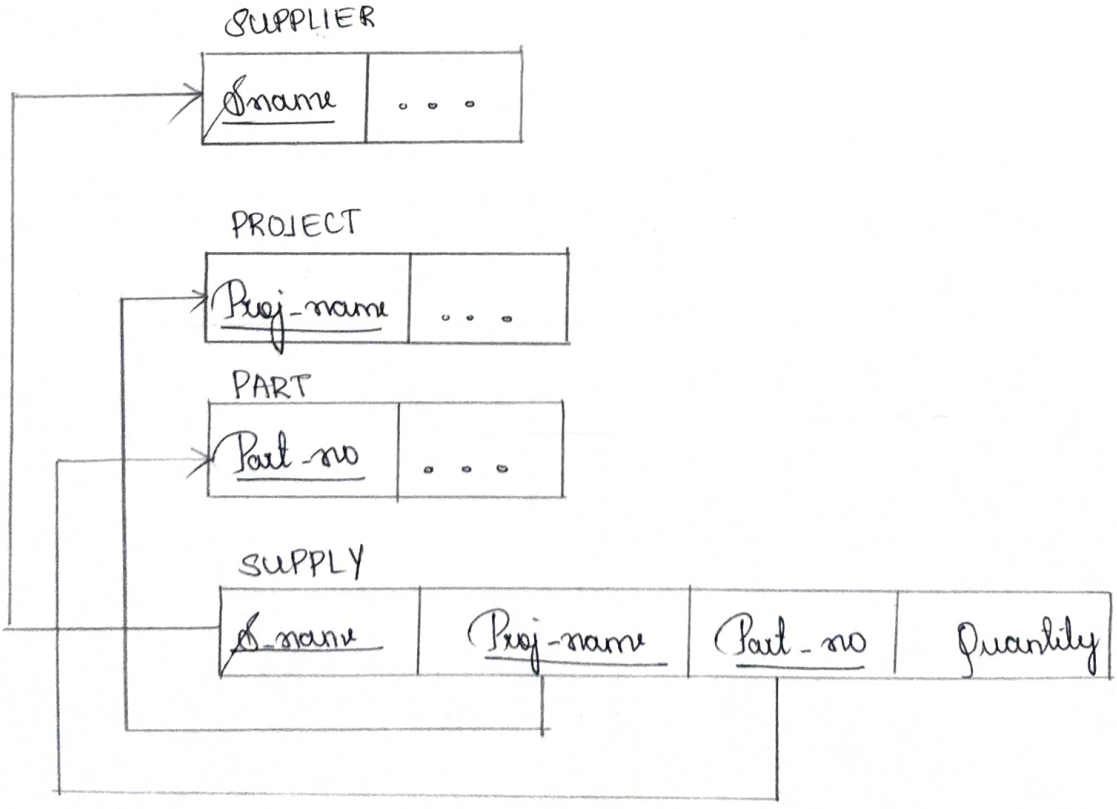
## step 7 : Mapping of N-ary Relationship Types

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 }



15) Explain the basic data types available for attributes in SQL.

17CS53 Jan/Feb 2021

Some of the basic data types in SQL are

① Numeric data types

- This includes integer numbers of various size (Integer or INT, and SMALLINT, BIGINT)
- Floating-point (real) numbers of various precision (FLOAT or REAL, and DOUBLE PRECISION)
- Formatted number can be declared by using DECIMAL(i,j) - or DEC(i,j) or NUMERIC(i,j)

  where i - precision, total number of decimal digits
  j - scale, number of digits after the decimal point.

② Character-string data types

- fixed length - CHAR(n) or CHARACTER(n), where n is the number of characters.
- varying length - VARCHAR(n) or CHAR VARYING(n) or CHARACTER VARYING(n), where n is the maximum number of characters.
- When specifying a literal string value, it is placed between single quotation marks (apostrophes), and it is case sensitive.
- For fixed length strings, a shorter string value is padded with blank characters to the right.
- For example, if value 'smith' is for an attribute of type CHAR(10), it is padded with five blank characters to become 'smith ' if needed.

- Padded blanks are generally ignored when strings are compared.
- Another variable-length string data type called CLOB or CHARACTER LARGE OBJECT is also available to specify columns that have large text values, such as documents.
- The CLOB maximum length can be specified in kilobytes (K), megabytes (M), or gigabytes [Gr].
- For Example: CLOB(20M) specifies a maximum length of 20 megabytes.

### ③ Bit string data types

- They are either of fixed length n - BIT(n) or varying length - BIT VARYING(n), where n is the maximum number of bits.
- The default for n, the length of a character string or bit string is 1.
- Literal bit strings are placed between single quotes but preceded by a B to distinguish them from character strings: Eg: B'10101'
- Another variable-length bitstring data types called BINARY LARGE OBJECT or BLOB is also available to specify columns that have large binary values, such as images.
- The maximum length of a BLOB can be specified in kilobits (K) megabits (M), or gigabits [Gr].
- For example: BLOB(30Gr) specifies a maximum length of 30 gigabits.

④ A Boolean data type

- This has the traditional values of TRUE or FALSE. In SQL, because of the presence of NULL values, a three-valued logic is used, so a third possible value for a Boolean datatype is UNKNOWN.

⑤ The DATE data type

- It has at least ten positions and its components are YEAR, MONTH and DAY in the form YYYY-MM-DD.

⑥ The time data type

- It has at least eight positions, with the components HOUR, MINUTE, and SECOND in the form HH:MM:SS. Only valid dates & times should be allowed by the SQL implementation.

⑦ TIME with TIME ZONE

This includes an additional six positions from specifying the displacement from the standard Universal time zone, which is in the range of +13:00 to -12:59 in units of HOURS:MINUTES. If WITH TIME ZONE is not included, the default is the local time zone for the SQL session.

## Additional Data types.

① Time stamp data type (TIMESTAMP)

It includes the DATE and TIME fields, plus a minimum of six position for decimal fractions of seconds and an optional WITH TIME ZONE qualifier.

## (2) INTERVAL Datatype

This specifies an Interval - a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp. Intervals are qualified to be either YEAR/MONTH Intervals or DAY/TIME Intervals.

16) Explain the following SQL command : CREATE, INSERT, UPDATE, SELECT. Give syntax and atleast one example for each

17CS53 July/August 2021

## 1] CREATE

* A database can be considered as a container for tables and a table is a grid with row and column to hold data.

* Individual statement are called query and to execute the query we have to execute various task such as creation of tables, injection of data into tables, deletion and so on.

step 1 : We create a database using following SQL statement

syntax: CREATE DATABASE database-name;

eg: CREATE DATABASE STUDENT;

step 2 : Table can be created as :

```
CREATE TABLE tablename (
        col1-name datatype,
        col2-name datatype,
        :
        coln-name datatype
);
```

eg:

```
CREATE DATABASE ST (
        NAME    VARCHAR(20),
        USN     VARCHAR(20),
        MOBILE  INTEGER
);
```

ST

| NAME | USN | MOBILE |
|---|---|---|

## 2] INSERT

we can insert one or more tuples in the relation.

syntax is:

```
INSERT INTO table-name (column1, column2, column3...)
VALUES (value1, value2, value3);
```

eg: 
```
INSERT INTO ( ST (NAME, USN, MOBILE)
VALUES ('PALLAVI', '4GM20IS033', 9920117891);
```

OR

```
INSER INTO ST VALUES ('PALLAVI', '4GM20IS033', 9920117891);
```

syntax for inserting for specific value insertion

```
INSERT INTO ST (NAME, USN)
VALUES ('SONIA', '4GM20IS069');
```

ST

| NAME | USN | PHONE |
|---|---|---|
| PALLAVI | 4GM20IS033 | 9920117891 |
| SONIA | 4GM20IS069 | NULL |

## 3] UPDATE

For modifying the existing record of a table, update query is used.

syntax:

```
UPDATE tablename
SET col1 = value1, col2 = value2 ....
WHERE condition;
```

Details

| AadharNo | Name | Address | City |
|---|---|---|---|
| 111 | SUNIL | M.G.Road | Pune |
| 222 | SURAJ | Shivaji road | Mumbai |
| 333 | SOUMYA | Viman Nagar | Delhi |

By using update query we can update value:

```
UPDATE Details
SET city = 'Chennai'
WHERE AadharNo = 333;
```

| AadharNo | Name | Address | City |
|---|---|---|---|
| 111 | SUNIL | M.G.Road | Pune |
| 222 | SURAJ | Shivaji Road | Mumbai |
| 333 | SOUMYA | Viman Nagar | Chennai |

4] SELECT

* Select statement is used to fetch data from database
* The results return the data in form of table
* Result table are called resultset

SELECT <attribute list>
FROM <table list>
WHERE <condition>;

Here
* <attribute list> is a list of attribute whose values are to be retrieved by query
* <table list> is a list of relation name required to process query
* <condition> is a conditional (Boolean) expression that identify the tuples to be retrieved by query.

consider table
Details

| Aadhar No | Name | City |
|---|---|---|
| 111 | AAA | Pune |
| 222 | BBB | Banglore |
| 333 | CCC | Mumbai |

SELECT AadharNo, Name, City FROM Details WHERE city = 'Pune'

| Aadhar No | Name | City |
|---|---|---|
| 111 | AAA | Pune. |

* Select clause of SQL specify the attribute whose value are to be retrieved are called projection attribute
* WHERE clause specify Boolean condition that must be true for any retrieved tuple, which is known as selection condition
* If we want to select all records present we make use of *

eg 2 :

```
SELECT * FROM Details;
```

| Aadhar No | Name | City |
|---|---|---|
| 111 | AAA | Pune |
| 222 | BBB | Banglore |
| 333 | CCC | Mumbai |
| 444 | DDD | Mumbai |

eg 3 :

```
SELECT * FROM Details WHERE city = 'Mumbai';
```

| Aadhar No | Name | City |
|---|---|---|
| 333 | CCC | Mumbai |
| 444 | DDD | Mumbai |

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)

18) Describe six clauses in syntax of SQL retrieval query with example. Which of six are required and which are optional.

-15CS53
July / Aug

OR

Describe the six clauses in syntax of SQL retrieval query

18CS53
July / Aug 2021

1) **Group By:**

* GROUP BY clause is a SQL command that is used to group rows that have same values.
* GROUP BY clause is used in the SELECT statement
* optionally it is used in conjunction with aggregate function
* The query that contain the GROUP BY clause are grouped query.
* The query returns a single row for every grouped item

Syntax:

```
SELECT  column_name(s)
FROM    table_name
WHERE   condition
 GROUP  BY  column_name(s);
```

eg:

| Sid | Sname | marks | city |
|---|---|---|---|
| 1 | AA | 60 | Pune |
| 2 | BB | 70 | Mumbai |
| 3 | CC | 90 | Pune |

Query:

Find the total marks of each student in city

```
SELECT SUM(marks), city
FROM Student
GROUP BY city;
```

result will be:

| SUM(marks) | City |
|---|---|
| 70 | Mumbai |
| 150 | Pune |

a) Having

* HAVING filters records that works on summarize GROUP BY results.
* HAVING applies to summarize group records, WHERE applies to individual records.
* only the groups that meet Having condition will be returned.
* Having requires that a GROUP BY clause present.
* WHERE and HAVING can be in same query.

Syntax:

```
SELECT column-name
FROM table-name
WHERE condition
GROUP BY column-name
HAVING condition;
```

Example: consider the following student table

| sid | sname | marks | city |
|---|---|---|---|
| 1 | AA | 60 | Pune |
| 2 | BB | 70 | Mumbai |
| 3 | CC | 90 | Pune |
| 4 | DD | 55 | Mumbai |
| 5 | EE | 84 | Chennai |

Query:

Find the total marks of each student in city named 'Pune' and 'Mumbai' only

```
SELECT  SUM(marks), city
FROM    student
GROUP BY city
HAVING city IN('Pune', 'Mumbai');
```

result will be

| SUM(marks) | city |
|---|---|
| 60 | Pune |
| 70 | Mumbai |

## 3] Order by

* In a table if the records are arranged in the increasing order of some column then it is ascending order.
* In a table if the records are arranged in the decreasing order of some column then it is called descending order.
* For getting the sorted order in the table we use ORDER BY command.

Syntax:

```
SELECT col1, col2, col3 .... coln
FROM table-name
ORDER BY col1, col2 .... ASC/DESC;
```

ASC : ascending order display

DESC : descending order display

consider example:

Person-details

| Aadhar_No | Name | City |
|---|---|---|
| 111 | AAA | Pune |
| 222 | BBB | Pune |
| 333 | CCC | Delhi |

```
SELECT *
FROM person_details
ORDER BY Aadhar_No DESC;
```

above query will result in :

person_details

| Aadhar_No | Name | City |
|---|---|---|
| 333 | CCC | Delhi |
| 222 | BBB | Pune |
| 111 | AAA | Pune. |

4] FROM clause

FROM is used to specify which table to select or delete data from.

syntax: SELECT attribute 1, attribute 2 .... attribute n
FROM tablename;

consider the following table

Customer_detail

| AadharNo | Name | City |
|---|---|---|
| 111 | AAA | Pune |
| 222 | BBB | Mumbai |
| 333 | CCC | Delhi |

1) SELECT * FROM Customer_detail
WHERE city = 'Pune';

| Aadhar No | Name | City |
|---|---|---|
| 111 | AAA | Pune |

2) DELETE FROM Customer_detail
WHERE Name = 'BBB';

SELECT * FROM customer_detail;

| AadharNo | Name | City |
|---|---|---|
| 111 | AAA | Pune |
| 333 | CCC | Delhi |

5] SELECT clause

* Select is used to fetch data from database
* The result return the data in form of table

Syntax:

```
SELECT <attribute list>
FROM <table list>
WHERE <condition>
```

consider the following table

| Aadhar No | Name | City |
|---|---|---|
| 101 | AAA | Pune |
| 111 | BBB | Pune |
| 121 | CCC | Banglore |
| 131 | DDD | Delhi |

consider the following query

```
SELECT AadharNo, FirstName and City
FROM Details
WHERE city = 'Pune';
```

| AadharNo | Name | City |
|---|---|---|
| 101 | AAA | Pune |
| 111 | BBB | Pune |

## 6] WHERE clause

* WHERE command specify the Boolean condition that must be true for any retrieved tuple, which is known as specified condition

* WHERE command is used to specify some condition Based on this condition data present in table can be displayed, updated or deleted

Syntax:

```
SELECT col1, col2 .... coln
FROM table-name
WHERE condition;
```

consider the following table

| Aadhar No | FName | LName | City |
|---|---|---|---|
| 111 | Sunil | Choudary | Pune |
| 222 | Sahil | Shukla | Delhi |
| 333 | Sonia | Das | Mumbai |

Query:

```
SELECT Aadhar No
FROM Details
WHERE city = 'Pune';
```

| Aadhar No | City |
|---|---|
| 111 | Pune |

Out of these six clauses SELECT and From are required and WHERE, HAVING, GROUP BY, ORDER BY are optional.

19) Explain the following in SQL 15CS53 Jan/Feb

i) Unspecified where clause and use of Asterisk
ii) Explicit and Nulls
iii) Renaming attributes and joined tables

## i) Unspecified where clause and use of Asterisk

[ Unspecified where clause means that the where clause is missing in the SQL Query

A missing WHERE clause indicates no condition on a tuple selection. Hence all tuples of a relation specified in the FROM clause qualify and are selected for the query result.

If more than one relation is specified in the FROM clause and there is NO WHERE clause, then the CROSS PRODUCT - all possible tuple combinations of these relations is selected.

For Example: The below Queries select all EmployeeSsn and all combinations of Employee SSn & department Dname in the database

① 
```
SELECT   Ssn
FROM     Employee;
```

②
```
SELECT   Ssn, Dname
FROM     Employee, Department;
```

The 2nd Query result in cross product of two relations Employee and department.

## Use of Asterisk (*)

To retrieve all the attributes values of the selected tuples, we do not have to list the attribute names explicitly in SQL, we just specify an asterisk (*), which stands for all the attributes.

The * can be prefixed by the relation.

For example:

① 
```
SELECT  *
FROM    EMPLOYEE
WHERE   Dno = 5;
```

②
```
SELECT  *
FROM    EMPLOYEE, DEPARTMENT
```

In 1st example, it retrieves all the attribute values of any EMPLOYEE who works in DEPARTMENT number 5.

In 2nd example, it retrieves all the attributes of Employee and Department ie specifies the cross Product of Employee and Department relations.

## Explicit Sets And Nulls in SQL

It is also possible to use an explicit (enumerated) set of values in the WHERE clause rather than a nested query.

For example: Retrieve SSNs of employees who work on Project no 1,2,3

```
SELECT DISTINCT ESSN
FROM   WORKS_ON
WHERE  PNO in (1,2,3)
```

The above example retrive the social security numbers of all employees who work on project number 1, 2, or 3

## NULLS

SQL allows queries that check if a value is NULL (missing or undefined or not applicable) SQL uses IS or IS NOT to compare NULLS because it considers each NULL value distinct from other NULL values, so equality comparison is not approximate.

For Example: Retrieve ~~all~~ the names of all employees who do not have supervisors.

```
SELECT  ENAME
FROM    EMPLOYEE
WHERE   SUPERSSN IS NULL;
```

Note: If a join condition is specified, tuples with NULL values for the join attributes are not included in the result.

## Renaming attributes and joined Tables

The SQL 'AS' is used to assign temporarily a new name to a table column or table (relation) itself. One reson to rename a relation is to replace a long relation name with a shortned version that is more convenient to use elsewhere in the query.

For example: "Find the names of student and isbn of book who reserved the books".

Student

| Sid | sname |
|---|---|
| 1 | Ram |
| 2 | Seeta |
| 3 | geeta. |

Reserve

| Sid | isbn |
|---|---|
| 1 | 005 |
| 2 | 007 |

```
SELECT S.sname, R.isbn
FROM   STUDENT as S, RESERVE as R
WHERE  S.sid = R.sid
```

In this case we could shorten the names of tables student and Reserve as S and R respectively.

Another reason to rename a relation is a case where we wish to compare tuples in the same relation. We then need to take the cartesian product of a relation with itself.

For example: select student Id who has enrolled in atleast 2 cources.

STUDY

| S_ID | C_ID | Year |
|---|---|---|
| S1 | C1 | 2013 |
| S2 | C2 | 2014 |
| S1 | C2 | 2014 |

```
SELECT S_ID
FROM   STUDY as T1, STUDY as T2
WHERE  T1.S_ID = T2.S_ID
AND    T1.C_ID <> T2.C_ID;
```

Here STUDY is renamed as both T1 and T2.

## Join Tables

The SQL join clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

**For example:**

Consider two tables for using the joins in SQL. Here cid is common column in the following tables.

STUDENT

| Sid | Cid |
|---|---|
| 1 | 101 |
| 2 | 102 |
| 3 | 103 |
| 4 | 104 |

RESERVE

| cid. | Cname |
|---|---|
| 101 | Ram |
| 102 | Hari |
| 101 | Giri |

To join (combine) the above two tables we can write the query as shown below.

```
SELECT   *
FROM     STUDENTS JOIN RESERVE R
ON       S.Cid = R.cid. ;
```

OR

```
SELECT *
FROM   STUDENT NATURAL JOIN RESERVE;
```

For more detailed answer refer question no. (9) but write SQL examples as shown above for each of the types of JOIN.

LEARN AND PRACTICE AS MUCH AS QUERIES IN RELATIONAL ALGEBRA AND SQL..

Refer All the possible sources and practice / Execute. Do not limit to one material / book / website

20) Consider 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

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

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

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

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)

∩

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

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

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

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

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

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

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

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

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

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
```

ii) List the titles of books reserved by professors

```
SELECT BOOK-TITLE FROM MEMBERS M, BOOKS B,
RESERVES R WHERE M.Member_ID = R.Member_ID
AND M.DESIGNATION = 'PROFESSOR'
AND B.BOOK_ID = R.BOOK_ID.

OR

SELECT BOOK-TITLE
FROM((MEMBERS NATURAL JOIN RESERVES)
NATURAL JOIN BOOKS)
WHERE Designation = 'PROFESSOR'
```

iii) Find the authors and titles of Books reserved on 27-May-2017.

```
SELECT BOOK_AUTHOR, BOOK_TITLE
FROM BOOKS NATURAL JOIN RESERVES
WHERE DATE = '27-May-2017'
```

v) Find the names of members who have reserved all books.

```
SELECT M.NAME
FROM MEMBERS M, RESERVES R
WHERE M.MEMBERID = R.MEMBERID

OR

SELECT NAME
FROM MEMBERS NATURAL JOIN RESERVES;
```

② Write SQL query for the following database schema

EMPLOYEE (EMPLOYEE_NAME, STREET, CITY)
WORKS (EMPLOYEE_NAME, COMPANY_NAME, SALARY)
COMPANY (COMPANY_NAME, CITY)
MANAGERS (EMPLOYEE_NAME, MANAGER_NAME)

18CS53 Jan/Feb 2021

i) List the name, street, city of all employee who work for First Bank Corporation and earn more than 10000

```
SELECT  E.EMPLOYEE_NAME, E.STREET, E.CITY
FROM  EMPLOYEE E, WORKS W
WHERE  E.EMPLOYEE_NAME = W.EMPLOYEE_NAME
AND COMPANY_NAME = "FIRST BANK CORPORATION"
AND SALARY > 10000;
```

(or)

```
SELECT  EMPLOYEE_NAME, STREET, CITY
FROM  EMPLOYEE NATURAL JOIN  WORKS
WHERE  COMPANY_NAME = "FIRST BANK CORPORATION"
AND  SALARY > 10000;
```

ii) Find names of all employees in the database who do not work for first bank corporation. Assume that all people work for exactly one company.

```
SELECT  EMPLOYEE_NAME
FROM  WORKS
WHERE  COMPANY_NAME <> "FIRST BANK CORPORATION";
```

iii) Find the names of all employees in the database who earn more that every employee of "Small Bank Corporation". Assume that all people work for atmost one company.

```
SELECT EMPLOYEE_NAME
FROM WORKS
WHERE SALARY = (SELECT MAX(SALARY) FROM WORKS
WHERE COMPANY_NAME = "SMALL BANK CORPORATION");
```

iv) Find the name of company that has smallest payroll.

```
SELECT COMPANY_NAME
FROM WORKS
WHERE SALARY = (select MIN (SALARY));
```

v) Find names of all employees in database who live in same cities and in same streets as do their managers.

```
SELECT E.EMPLOYEE_NAME
FROM EMPLOYEE E, EMPLOYEE M, MANAGES R
WHERE E.EMPLOYEE_NAME = R.EMPLOYEE_NAME AND
M.EMPLOYEE_NAME = R.MANAGER_NAME AND
E.CITY = M.CITY AND E.STREET = M.STREET;
```

③ Write SQL queries for the following relational schema:

CUSTOMER (CID, CNAME, EMAIL, ADDR., PHONE)
ITEM (ITEM_NO, ITEM_NAME, PRICE, BRAND)
SALES (CID, ITEM_NO, #ITEMS, AMOUNT, SALE_DATE)
SUPPLIER (SID, SNAME, SPHONE, SADDR)
SUPPLY (SID, ITEM NO, SUPPLY_DATE, QTY)

18CS53 Jan/Feb 2021

i) List the items purchased by customer Prashant.

```
SELECT ITEM_NAME
FROM ITEM I, CUSTOMER C, SALES S
WHERE C.CID = S.CID AND I.ITEM_NO = S.ITEM_NO
AND C.CNAME = 'PRASHANT';
```

ii) Retrive items supplied by all suppliers starting from 1st Jan 2019 to 30th Jan 2019.

```
SELECT ITEM_NAME
FROM ITEM, SUPPLIER, SUPPLY
WHERE ITEM.ITEM_NO = SUPPLY.ITEM_NO AND
SUPPLY.SID = SUPPLIER.SID AND
SUPPLY.SUPPLY_DATE='1 JAN 2019' to '30 JAN 2019'
```

iii) Get the details of customers whose total purchase of items worth more than 5000 rupees.

```
SELECT CUSTOMER
FROM CUSTOMER C, SALES S
WHERE
C.CID = S.CID AND S.AMOUNT > 5000;
```

iv) List Total Sales amount, total items, average sale amount of all items.

```
SELECT
SUM(AMOUNT), COUNT(ITEM), AVG(AMOUNT)
FROM SALES
WHERE SALES.ITEM_NO = ITEM.ITEM_NO ;
```

v) Display customers who have not purchased by items.

```
SELECT CID, CNAME, Email, ADDR, PHONE
FROM CUSTOMER
WHERE CID
NOT IN (SELECT CID FROM SALES);
```