Relational Algebra

Basic concepts

- Domain: There are set of permitted values for every attribute, called its domain.
  - Exp- Domain of roll number{10,11,23,56,78}
  - Domain of branch{CSE,IT,ME,ECE}
- Tuple: Each row in a relation is called tuple.
- Relation: Collection of homogeneous tuples.
- Degree or Arity: Number of attributes in relation R.
- Cardinality: Number of tuples in relation R.

- Keys:
- Compatibility of Relations: Relations R and S are said to be compatible if
  - Both have same number of attributes.(same arity)
  - And domain on ith attribute of R must be same as of ith attribute of relation S.

Integrity Rules

- Rule 01
- Entity integrity
  - If an attribute A of relation R is selected as primary key then it cannot accept null values.

Student

Roll-No	Name	Class	Semester
100	XYZ	CSE	THIRD
-	ABC	ECE	FIFTH
102	-	ME	SEVENTH
-	ABC	ECE	FIFTH

- Rule 02
- Referential Integrity

Department

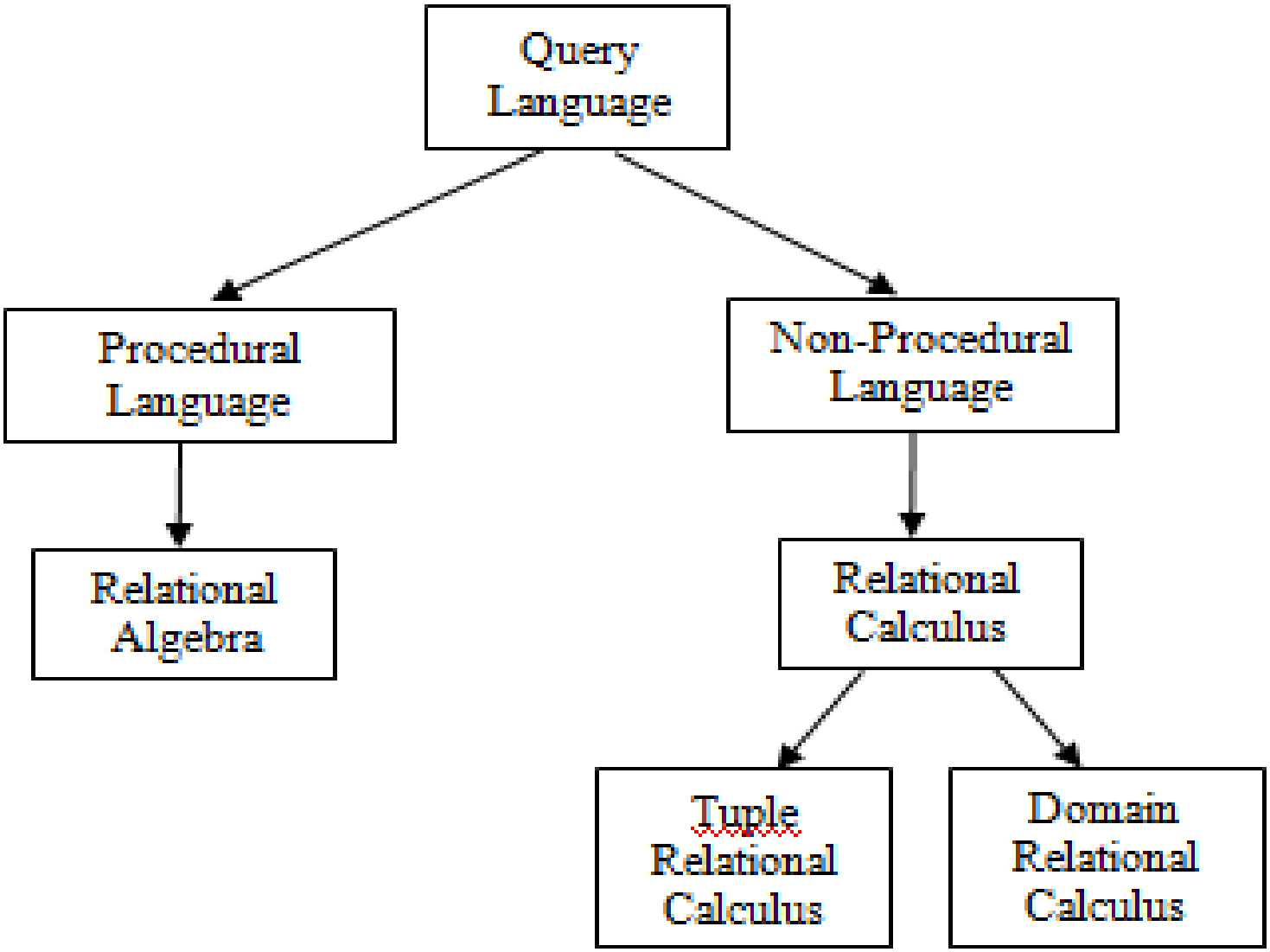
Deptt-No	Name
01	CSE
02	IT
03	ECE
04	ME

Course

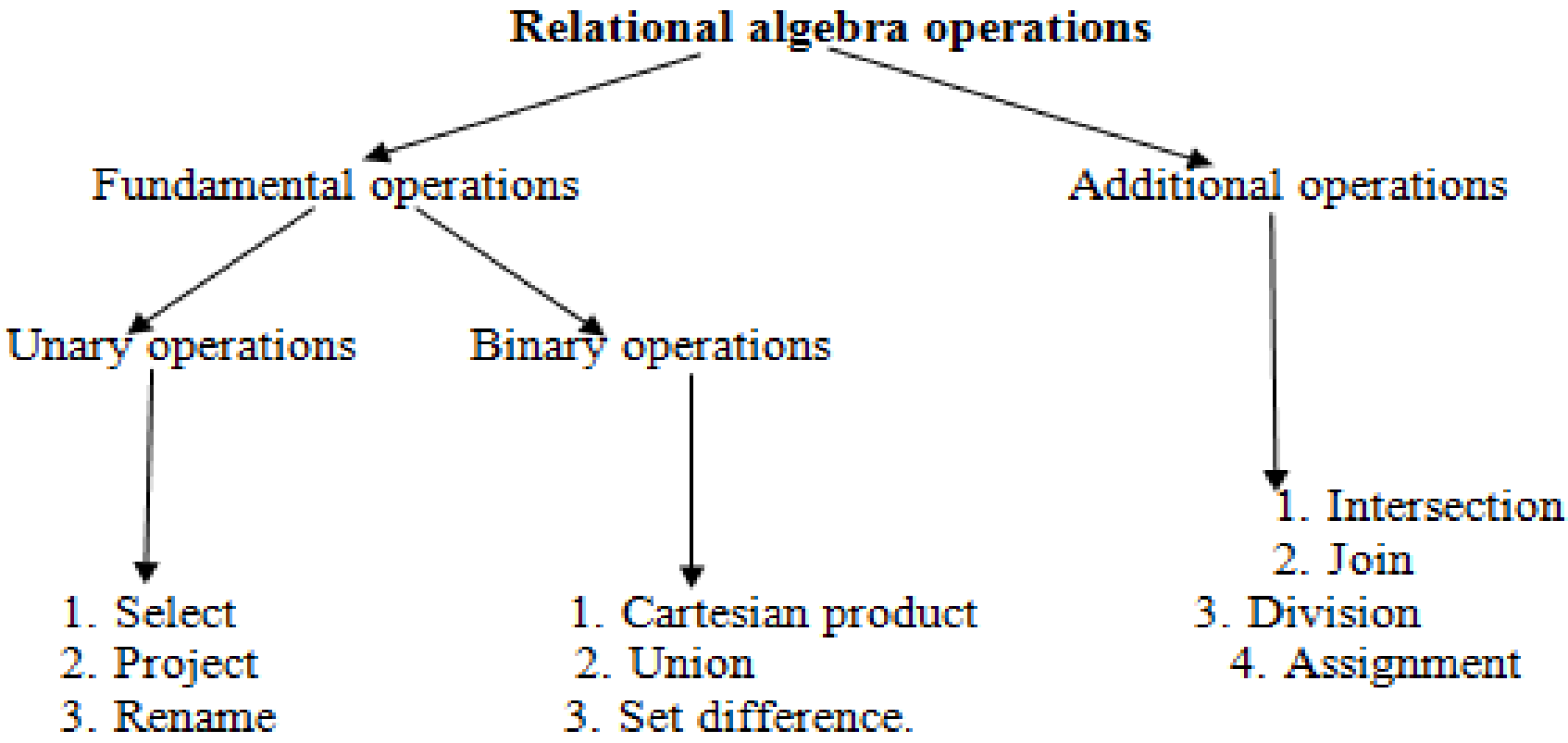
Course-No	Title	Deptt-No
CS-305	DBMS	01
CS-311	CPI	01
ME-309	NM	04
IT-303	WP	02
EC-307	MP	03

Query Language

- Interface between the user and the data base.
- A query Is a statement written in query language for retrieval od data from database.


Query Language
Procedural Language
Non-Procedural Language
Relational Algebra
Relational Calculus
Tuple Relational Calculus
Domain Relational Calculus

INTRODUCTION



Select

- **Select**
- Select operation selects the rows that satisfy a given predicate (condition).
- The Greek letter *sigma* is used to denote the select operation.
- **Syntax:**

Syntax:

$$\sigma_{\text{Predicate}}(\text{Relation})$$

Student

RollNo	Name	Class	Semester	Age	City
100	Agrima	CSE	First	17	Qadian
101	Jessica	ECE	Third	18	Noida
102	Surbhi	IT	Fifth	19	Jalandhar
103	Mehak	EEE	Third	17	Jalandhar
104	Vikram	ECE	Fifth	19	Gurdaspur
105	Vicky	CSE	Seventh	20	Qadian

- Find the students from the student relation who belong to qadian CITY.

Query:

$$\sigma_{\text{city} = \text{"qadian"}} (\text{Student})$$

Result relation

RollNo	Name	Class	Semester	Age	City
100	Agrima	CSE	First	17	Qadian
105	Vicky	CSE	Seventh	20	Qadian

2. Find all the students with age greater than or equal to 19.

Query:

$$\sigma_{age \geq 19}(\text{Student})$$

Result relation

RollNo	Name	Class	Semester	Age	City
102	Surbhi	IT	Fifth	19	Jalandhar
104	Vikram	ECE	Fifth	19	Gurdaspur
105	Vicky	CSE	Seventh	20	Qadian

Project

- Use to select attributes from the relation

Syntax:

$$\pi_{\text{attribute-list}}(\text{Relation})$$

Queries:

1. Show all the cities in the student relation.

Query:

$$\pi_{\text{city}} (\text{student})$$

Result relation

City
Qadian
Noida
Jalandhar
Gurdaspur

2. Show the roll numbers and names of all the students.

Query:

$$\pi_{\text{rollno, name}} (\text{student})$$

Result relation

RollNo	Name
100	Agrima
101	Jessica
102	Surbhi
103	Mehk
104	Vikram
105	Vicky

Rename

- Used for following
- 1. When we want to change the name of any existing relation
- 2. When we want to give a name to a new relation which is obtained as a result of any relational algebra expression

-

$$\rho_{s}(R)$$

$$\rho_{s}(E)$$

- **Example-2:** Query to rename the attributes Name, Age of table Department to A,B.

$\rho_{(A, B)}$ (Department)

- **Example-3:** Query to rename the table name Project to Pro and its attributes to P, Q, R..

$\rho_{Pro(P, Q, R)}$ (Project)

- **Example-4:** Query to rename the first attribute of the table Student with attributes A, B, C to P.

$\rho_{(P, B, C)}$ (Student)

Cartesian Product

- Cartesian product is denoted by cross (×) and it combines the information from any two relations.
- Cartesian product of two relations *R* and *S* is denoted as *R*×*S*, which result in a new relation that contains all the possible combinations of tuples in *R* and *S*.

- Degree or arity (P) = arity(R) + arity(S)

- Cardinality (P) = cardinality(R) * cardinality(S)

Employee

Emp-id	Name
1	Raj
2	Nav
3	Harish
4	Pardeep

Project

Proj-id	Proj-Name
P1	Java
P2	dbms

Cartesian product of these two relations is given as:

$$R = Employee \times Project$$

R = Employee × Project

Emp-id	Name	Proj-id	Proj-Name
1	Raj	P1	Java
1	Raj	P2	dbms
2	Nav	P1	Java
2	Nav	P2	dbms
3	Harish	P1	Java
3	Harish	P2	dbms
4	Pardeep	P1	Java
4	Pardeep	P2	dbms

Union

- the union of two sets results in a new set that contains all the elements belonging to both the sets but does not include the duplicate elements

For example, consider the expression:

$$S = \{A, B, C, D\} \cup \{C, D, E, F\}$$

$$\Rightarrow \quad S = \{A, B, C, D, E, F\}$$

Depositor

Acc-no	Cust-name
A-101	Tej
A-102	Arun
A-103	Sunil

Borrower

Loan-no	Cust-name
L-400	Sunil
L-401	Vicky

Queries:

1. Show the names of customers having an account or loan or both.

Query:

$$\Pi_{cust-name}(\text{depositor}) \cup \Pi_{cust-name}(\text{borrower})$$

Result relation

Cust-name
Tej
Arun
Sunil
Vicky

Set Difference

- the difference of two sets results in a new set that contains all the elements of first set which are not present in the second set.

For example:

$$S = \{A, B, C, D\} - \{C, D, E, F\}$$
$$S = \{A, B\}$$

Queries:

1. Show all the customers who have taken loan but do not have any account.

Query:

$$\pi_{cust\text{-}name}(borrower) - \pi_{cust\text{-}name}(depositor)$$

Result relation

Cust-name
Vicky

Intersection

- Intersection of two sets results in a new set which contains the common elements from both the sets.

For example:

$$S = \{A, B, C, D\} \cap \{C, D, E, F\}$$
$$S = \{C, D\}$$

Example: Show the name of customers who have an account in a bank and also they have take a loan.

Query:

$$\pi_{cust-name}(\text{depositor}) \cap \pi_{cust-name}(\text{borrower})$$

Result relation

Cust-name
Sunil

Natural Join

- *Join* operation allows us to combine certain selections and cartesian product into one operation.

Department

Dept-id	Name
10	CSE
11	IT
12	ECE
13	ME

Hod

Dept-id	Hod-name
10	ABC
11	XYZ
12	MNO
13	PQR

Department ⋈ Hod

Dept-id	Name	Hod-name
10	CSE	ABC
11	IT	XYZ
12	ECE	MNO
13	ME	PQR

- **Example:** Show all the customers who have an account and also have taken a loan.

Query:

$$\Pi_{cust\text{-}name}(depositor) \cap \Pi_{cust\text{-}name}(borrower)$$

Query:

$\Pi_{\text{cust-name}}$ (depositor ⋈ borrower)

Depositor ⋈ Borrower

Acc-no	Cust-name	Loan-no
A-103	Sunil	L-400

Result relation

Cust-name
Sunil

Outer Join

- *Outer-join* operation is an extension of natural join which deals with missing information.

Customer relation

Name	Street	City
Smith	Main	Banglore
Arun	M.G.road	Pune
Ajay	G.B.road	Bombay

Account relation

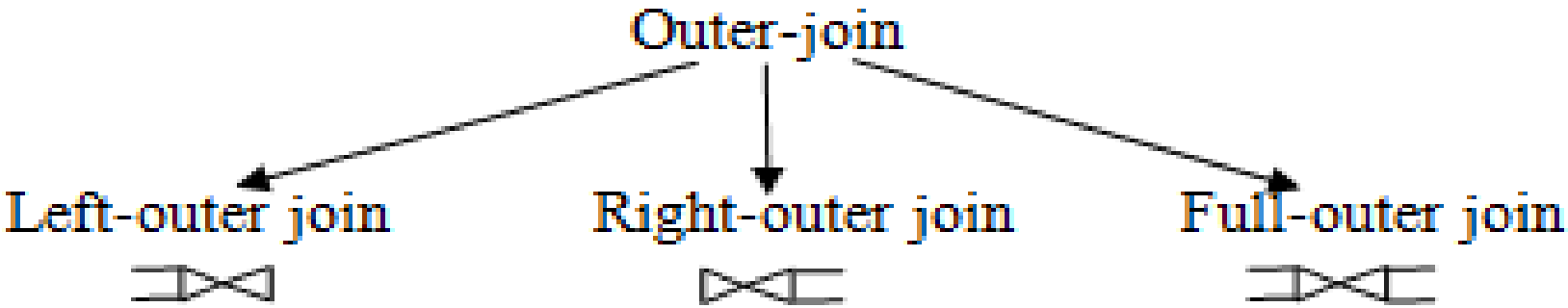
Name	Account	Branch
Smith	A-101	Banglore
Parveen	A-102	Jalandhar
Arun	A-103	Pune

Customer ⋈ Account

The resultant relation will be as follows:

Result relation

Name	Street	City	Account	Branch
Smith	Main	Banglore	A-101	Banglore
Arun	M.G.road	Pune	A-103	Pune


Outer-join
Left-outer join
Right-outer join
Full-outer join

1. Customer ⟕ Account

Result relation

Name	Street	City	Account	Branch
Smith	Main	Banglore	A-101	Banglore
Arun	M.G.road	Pune	A-103	Pune
Ajay	G.B.road	Bombay	------	------

2. Customer ⟕ Account

Result relation

Name	Street	City	Account	Branch
Smith	Main	Banglore	A-101	Banglore
Arun	M.G.road	Pune	A-103	Pune
Parveen	------	------	A-102	Jalandhar

3. Customer ⋈ Account

Result relation

Name	Street	City	Account	Branch
Smith	Main	Banglore	A-101	Banglore
Arun	M.G.Road	Pune	A-103	Pune
Ajay	G.B.Road	Bombay	------	------
Parveen	------	------	A-102	Jalandhar

Division

- The *division* operation is suited to the queries that include the phrase "*for all*".

is denoted by $\div$. Let R and S be two relations and $S \subseteq R$,

R

A	B
M1	N1
M1	N2
M2	N1
M3	N1
M4	N2
M5	N1
M5	N2

S

B
N1

$$P = R \div S$$

A
M1
M2
M3
M5

S

B
N1
N2

then,

P = R ÷ S

A
M1
M5

- **Assignment**

Assignment operation works similar to the assignment in any programming language. It is denoted by "←".

Syntax:

```
Relation variable ← Expression
```

The expression on the right side is evaluated and its result is assigned to the variable on the left side of ←.