UNIT-2
Relational Algebra and Calculus

Syllabus:

- Basics of relational model,
- ER diagrams to relational design,
- **Relational algebra**:
- Simple operations and extended operations,
- writing relational algebra expressions for queries
- Introduction to tuple **relational calculus** and writing basic queries using tuple calculus

RELATIONAL MODEL

Introduction:

- The Relational model proposed by EF codd in 1970.
- In this model, data is in the form of tables with rowsand columns.
- This model represents how data can be stored in a Relational Database(RDB).
- The relations are implemented by using different DBMS software like Mysql,DB2,Oracle 10g,Oracle 11g etc.

Characteristics of Relation:

- Data represented in Relational database model must be in the form of rows and columns.
- All values are scalar i.e, row or column position in the relation there is only one value.
- No two rows are identical.
- Data in a relational database specify tablename,column name.

Components of Relational Model:

a.Table: In relational model data saved in the form of rows and columns where rows represent records and column represent Attributes.

Ex: Student

Sid	**Name**	**age**	**marks**
1	a	20	75
2	b	21	90
3	c	20	95
4	d	21	80
5	e	20	85

b.Tuple: It is a row of table which contains singlerecord of a table(relation).
Ex: from above table5 e 20 85

c.Attribute: It is columns of a table and also called as“fields”
Ex: from above table sid,name,age,marks

d.Relational schema: It describe the name of relationand its attributes.
Ex: from above table Student(sid,name,age,marks)

Characteristics of relation:

- **Values and NULLs in the Tuples –**
  - Each value in a tuple is an **atomic value.**
  - composite and multi-valued attributes are not allowed.
  - NULL value is used to represent
    - The values of attributes that may be unknown.
    - The values of attributes may not apply to a tuple.

Characteristics of relation:

- ❑ **Interpretation (Meaning) of a Relation –**
  - ❑ The relation schema can be interpreted as a declarationor a type of assertion.

e.Realtional instance: It describe finite set of tuples in a relation at particular instance of time.

➢ It can be changed whenever there is insertion,deletion and updation etc.

Ex: In above table
Student with 5 tuples

f.Degree of relation: It describe no.of attributes inarelation and determine its degree.
Ex: from above table degree is 4

g.Cardinality ratio: It describe no of rows or recordsin a relation.
Ex: from above table cardinality is 5.

➢ Relational Query Language:-

- ❑ A query language is a language in which a userrequests information from the database.

- ❑ Higher level than that of a standard programminglanguage.

- ❑ Provides fundamental techniques to extract thedata from database.

These are two types

1. Procedural languages
2. Non – procedural languages

1. Procedural languages:-

The user instructs the system to perform a sequence of operations on the database to compute the desired result.

Example:- Example: Relational algebra.

2. Non – procedural languages:-

The user describes the desired information without giving a specific procedure for obtaining that information

Example: Relational calculus.

RELATIONAL ALGEBRA:

➔ The ***relational algebra*** consists of a set of operations that take one or two relations as input and produce a new relation as their result.

➔ It is a procedural language which takes relation as an output.

➔ It use operators to perform queries that can be unary or binary.

➔ **Basic operations**

1. Selection (σ) – Selecting rows

2. Projection (Π) – Selecting attributes.

Operations in Relational Algebra:

(1)SIMPLE OPERATIONS:

(a)SET OPERATIONS:

There are 4 different set operations in SQL.

(i) UNION:

first table

Sid	name
500	sanju
501	suppu

second table

Sid	name
501	suppu
502	bujji

- ➔ It is used to combine result of 2 (or) more select statement.
- ➔ However,It will eliminate(excluding) the duplicates rows from the table.
- ➔ The no.of columns and **datatype& size& column name** must be same inboth the tables.

Syntax: select<columnslist>from<tablename1>UNION select <columnslist>from<tablename2>

Ex: select *from first union select *from second;

Output:-

Sid	Name
500	Sanju
502	Bujji

(ii) UNION ALL:

It is also combine the result of two (or) more select statements.

Ex: select *from first union all select *from second;

Sid	name
500	Sanju
501	suppu
501	Suppu
502	Bujji

(iii) <u>INTERSECTION:</u>

- ➔ It is used to combine both two select statements but it return records or rows which are common from both select statements.
- ➔ The no.of columns and datatypes must be same in both tables.

Ex:

select *from first intersection select *from second;

Sid	Name
501	Suppu

(iv) <u>MINUS:(or) set difference (or)Except:-</u>

- ➔ It also combines result of two select statements and return only those in final result which belongs to the first set of result.

Ex: select *from first minus select *from second;

Output:-

Sid	Name
500	Sanju

(b) PROJECTION: It is used to project required columns/attributes from a relation.

SYNTAX:

$$\pi_{A_1,A_2-A_n}(r)$$

Where A1,A2,..,An are Attributes.

And r is a relation.

Sid	Name	age	Marks
1	A	20	90
2	B	21	92
3	C	19	87
4	D	21	100

Ex: select sid, sname from student;

Sid	name
1	a
2	b
3	c
4	d

(c) SELECTION:

➔ It is used to take tuples from the relation which satisfies given condition.

SYNTAX: $\sigma_p(r)$

where r= relation and
p= propositional logic with operators like
>,<,>=,<=,==,!=.....

Ex:

Select * from student;

sid	name	age	marks
2	b	21	95
4	d	21	96

(d) RENAME: It is used to rename the relation.

Syntax: p(relation new,relation old)

(2)EXTENDED OPERATORS:

→ The operators which are derived from the simple operators are called “**extended operation**”.

➔ Join ,division are the extended operators.

(a) JOINS:

➔ An SQL join is used for combine the columns from one or more tables by using values common to both the tables.

➔ It is used to fetch data from two or more tables.

→ **NOTE:** Minimum required condition for joining table is “**n-1**”

where ‘n’ is no.of tables

➔ A table can also join itself is known as “self join” and denoted with “ ”

TYPES OF JOINS:

(1)Cross join (or) Cartesian product

(2) Inner join(or) Equi join

(3)Outer join

(a) left outer join

(b) right outer join

(c)Full outer join

first table

Sid	name
501	ram
502	raj
503	rocky

second table

Sid	address
501	vijayawada
502	hyderabad
504	guntur

(1) Cross join (or) Cartesian product:

➔ It is a type of join which returns the cartesian product of rows from both the tables.

➔ It will return a table which consist of combine the rows of first table with each row of second table

Ex:

select *from first cross join second;

OUTPUT:

Sid	Name	Sid	Address
501	Ram	501	Vijayawada
502	Raj	502	Hyderabad
503	Rocky	504	Guntur
501	Ram	501	Vijayawada
502	raj	502	Hyderabad
503	rocky	504	Guntur
501	ram	501	Vijayawada
502	raj	502	Hyderabad
503	rocky	504	Guntur

(2)INNER JOIN (or) EQUI-JOIN(or)natural join:

➔ In this join result is based on matched data as per the equality condition specified in query.

Ex: select *from first, second where first.sid=second.sid;

OUTPUT:

Sid	name	Sid	Address
501	Ram	501	vijayawada
502	Raj	502	hyderabad

NATURAL JOIN:

➔ It is a type of inner join which is based on the columns having same name and same data type present in both the tables to be joined.

Ex: select *from first natural join second;

Sid	name	Sid	Address
501	ram	501	vijayawada
502	raj	502	hyderabad

(3)OUTER JOIN: It is based on the matched and unmatched data.It is further divided into 3 types.

(i)Left Outer join:

It return a result with matched data of 2 tables then remaining rows of the left table and null for the right table column.

Ex:

select *from first left outer join second on first.sid=second.sid;

sid	name	Sid	Address
501	ram	501	Vijayawada
502	raj	502	Hyderabad
503	rocky	NULL	NULL

(ii)Right Outer join:

- ➔ It return a result with matched data of 2 tables then remaining rows of the right table and null for the left table column.

Ex:

select *from first right outer join second on

first.sid=second.sid;

OUTPUT:

Sid	name	Sid	Address
501	ram	501	Vijayawada
502	raj	502	Hyderabad
NULL	NULL	504	Guntur

(iii) Full Outer join:

It return a result with matched data of 2 tables then remaining rows of both left table and then right table.

Ex:

select *from first full outer join second on first.sid=second.sid;

Sid	name	sid	Address
501	ram	501	Vijayawada
502	raj	502	Hyderabad
503	rocky	NULL	NULL
NULL	NULL	504	Guntur

(b) DIVISION(for all,at all,every):

- ➔ It is denoted as "/"
- ➔ It division operation is performed with the the following rules:

(1) All the attributes of B are proper subset of 'A'.

(A,B are two relations).

(2) All the attributes of A- All the attributes of B.

(3) It will return tuple from relation A which are associated to every 'B' tuple.

Ex: Find sid of student who have enrolled in every course ?

Enroll table(A)

Sid	cid
s1	c1
s2	c1
s1	c2
s3	c2

Course table(B)

cid
c1
c2

Ans:(i) B is subset to A.

(ii) all attributes of B-all attributes of A= (sid,cid)-cid=sid.

(iii) resultant relation is:

sid
s1

➢ writing relational algebra expressions for queries:-

Basic operations:

- Selection.
- Projection.
- Union.
- Difference.
- Cartesian product.
- Rename

Derived operations:-

Interaction

Join

Division

Example: STUDENT

Select all the student details who has the GRADE ‘C’

ID	NAME	CONTACT	CGPA	GRADE
1234	RADHA	9000000456	8.2	B
1235	SYAM	8000000987	7.8	C
1236	RAM	7090909023	9.4	X
1237	SHARMA	6784567888	8.9	A

1238	SURABHI	7056078091	7.2	C
1239	SAKHI	9034567890	8.1	B

$\sigma_{GRADE='C'}$ (STUDENT)

S e l e c t a l l t h es t u de n t de t a i l sw h o h a s t h eGRADE ‘C’

Select all the student names and their grades.

Example: STUDENT

:

ID	NAME	CONTACT	CGPA	GRADE
1234	RADHA	9000000456	8.2	B
1235	SYAM	8000000987	7.8	C
1236	RAM	7090909023	9.4	X
1237	SHARMA	6784567888	8.9	A
1238	SURABHI	7056078091	7.2	C
1239	SAKHI	9034567890	8.1	B

Resultant relation:

Select all the student names andtheir grades.

$\Pi_{NAME,GRADE}$ (STUDENT)

NAME	GRADE
RADHA	B
SYAM	C

RAM	X
SHARMA	A
SURABHI	C
SAKHI	B

Example: STUDENT

ID	NAME	CONTACT	CGPA	GRADE
1234	RADHA	9000000456	8.2	B
1235	SYAM	8000000987	7.8	C
1236	RAM	7090909023	9.4	X
1237	SHARMA	6784567888	8.9	A
1238	SURABHI	7056078091	7.2	C
1239	SAKHI	9034567890	8.1	B

Select the student names and contact numbers whose grade is “C”.

Step 1: Select the tuples who has GRADE ‘C’

σGRADE=‘C’ (STUDENT)

ID	NAME	CONTACT	CGPA	GRADE
1235	SYAM	8000000987	7.8	C
1238	SURABHI	7056078091	7.2	C

Step 2: Select the attributes names and contact

$$\Pi_{NAME,GRADE}\ (\sigma_{GRADE='C'}\ (STUDENT))$$

NAME	CONTACT
SYAM	8000000987
SURABHI	7056078091

Union:

Union (U):

Let A, B are two union compatible relations.

Then A U B,

Returns a relation instance containing all tuples that occur in *either relation instance A or relation instance B (or both)* with the identical schema to A.

Union (AU B):

Union Compatibility –

- ❑ Two relations have same number of attributes.
- ❑ Corresponding attributes have same domains.

Intersection (∩): Let A, B are two union compatible relations.

Then A∩ B,

Returns a relation instance containing alltuples that occur in *both relations* with the identical schema to A.

Difference (–) :

Let A, B are two union compatible relations.

Then A –B,

Returns a relation instance containing alltuples that occur in *A but not in B* with the identical schema to A.

Cross product / Cartesian product (X):

Let A, B are relations.

Then A X B,

- ❑ Returns a relation instance with the schema that contains all attributes A and

B.

- The result of *A X B* contains all the tuple *(1 , s) (the concatenation of tuples A and B) for each pair* of tuples *l E A, s E B.*

Renaming (ρ) -

- Rename the given relation or attributesnames.
- ρ $_{\text{new\_name}}$ (relation_name).
- ρ new_name(List of attributes) (relation_name).

Join (⋈) –

Natural Join (⋈)

Joining of two relations *A and B, denoted A* ⋈ *B, in* which we pair only those tuples from *A and Athat agree in whatever attributes* are common tothe schemas of *A and B.*

NaturalJoin(⋈)

Student:

ID	NAME	DEPT
1234	RADHA	CSE
1235	KRISHNA	ECE
1236	SHANTHI	CSE

Marks:

ID	SUB-1	SUB-2	SUB-3
1234	78	89	88
1235	56	95	70
1237	90	97	96

Result:-

ID	NAME	DEPT	SUB-1	SUB-2	SUB-3
1234	RADHA	CSE	78	89	88
1235	KRISHNA	ECE	56	95	70

Relational algebra Queries:-

Writing relational algebra expressions for queries:

- ❑ Find the list of employee id numbers who hassalary above 30000.

  Emp(id, name, dept, salary)

Writing relational algebra expressions for queries:

- ❑ Find the list of employee id numbers and nameswho has salary above 30000 and below 60000.

  Emp(id, name, dept, salary)

Writing relational algebra expressions for queries:

- ❑ Find the list of dept that has lessthan 10employees.

  Emp(id, name, dept, salary)

Writing relational algebra expressions for queries:

- ❑ Find the list of id, name, dept of the studentswho has grade ‘A’.

  Students(id, name, dept, contact)Marks(id,

  percent, cgpa, grade)

Division (/) –

Let A(Z) / B(X) where the attributes of *B are a subset of the attributes of A; that is, X Z.* $\subseteq$ *Let Y be the set of* attributes of *A that are not attributes of B; that is, Y = Z – X (and hence Z = X Y).* $\cup$

Then,

Division Produces a relation *R(Y) that includes all tuples t[X] inA(Z) that appear in A in combination* with every tuple from *B(X),*

Intersection:

- Intersection can be derived in terms of ***difference***.

$$A \cap B = A - (A - B)$$

Question:

List out the subject names and ids that are taughtin both semester 1 and 2 in the year e1?

subjects(id, name, dept, sem, year)

Question:

What is the largest salary among the employs ?

emp(id, name, dept, salary)

$\Pi_{salary}(emp) - \Pi_{salary}(emp \bowtie_{(emp.salary < e.salary)} \rho_{e}(emp))$

Theta Join ($\bowtie_c$)

The resultant relation schema consist of allthe attributes of given two relations.

RELATIONAL CALCULUS:

➔ It is alternative (or) contrast to the Relational Algebra.

➔ It is a Non-procedural language and more declarative compare with Realtional Algebra.

➔ User-define Queries in terms of “what they want” and not in terms of “how they want”.

➔ It also contains variables,constants,comparision operators,Logical operators and quantifiers.

➔ It is divided into two parts:

(a) Tuple Relational Calculus(TRC).
(b) Domain Relational Calculus(DRC).

(a) Tuple Relational Calculus(TRC):

- ➔ It is a Non-procedural language which specifies "set of tuples" in a relation.
- ➔ It was proposed by EF codd in 1972.
- ➔ It can select tuples with range of values or tuples with certain attributes.

  Syntax:

  {t | p(t)},it means set of all tuples 't' such that predicate is true for tuple 't'.

  r(t) means tuple 't' from relation 'r'.
- ➔ We can specify column name using "dot" operator along with tuple 't' is "t.A" (or) t[A].

- ➔ The predicate calculus 'P' contains

  (a)variables,constants.
  (b) comparision operator(Relational operator).
  (c)connectivities(Logical operator)-V , ^ , | , ~
  (d)Implications(=>): p=>q that is ~pVq
  (e)Quantifier: for all and there exist.

➔ Examples for **“TRC”** is **“SQL”**.

Ex-1:select the tuple from student relation such that student tuples will have age>20.

O/P: {t | student(t) ^ t.age>20}

Ex-2:select tuple from student relation such that student tuple will have marks greater than 70.

O/P: {t | student(t) ^ t.marks>70}

➔ The variable used in tuple are called “tuple variable”.

Boundary Variable and PriVariables:

➔ **Boundary Variable** are with there exist and for all.

➔ **PriVariables** are without there exixt and for all.
Ex-1 and Ex-2 are prevariables.

(b) Domain Relational Calculus:

➔ It is a Non-procedural language which is based on domain of attributes and not based on tuple values.

➔ Example for **“DRC”** is: **QBE**(Query by language).

➔ The only difference between TRC and DRC is TRC concentrate on tuple values but DRC concentrate on selecting attributes.

Syntax: {<a1,a2,....,an> | p(a1,a2,....,an) }

where a1,a2,....,an are attributes and P is a predicate logic with several operators.

➔ **Ex-1:** select sid,age of student whose age is greater than 20.

{< sid,age > | $\exists$ student ^ age>20 }

➔ **Ex-2:** select employee id, esalary of employee whose salary is greater than 50000.

{ <eid,esalary> | $\exists$ employee ^ salary > 50000}