

UNIT-2

RELATIONAL MODEL

Introduction:

- The Relational model proposed by EF codd in 1970.
- In this model,data is in the form of tables with rows and 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 single record of a table(relation).

Ex: from above table
5 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 relation and its attributes.

Ex: from above table
Student(sid,name,age,marks)

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 updatation etc.

Ex:In above table
Student with 5 tuples

f.Degree of relation: It describes no. of attributes in a relation and determines its degree.

Ex: from above table degree is 4

g.Cardinality ratio: It describes no. of rows or records in a relation.

Ex: from above table cardinality is 5.

RELATIONAL ALGEBRA:

- ➔ It is a procedural language which takes relation as an output.
- ➔ It uses operators to perform queries that can be unary or binary.

Operations in Relational Algebra:

(1)SIMPLE OPERATORS:

(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 the duplicate rows from the table.
- ➔ The no. of columns and datatype must be same in both the tables.

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

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) INTERSECTION:

- ➔ 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) MINUS:

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

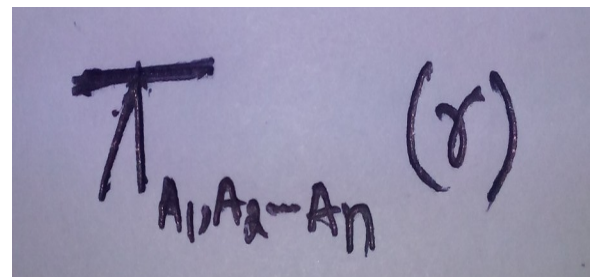
Ex: select *from first minus select *from second;

sid	name
500	sanju

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

SYNTAX:

Where A_1, A_2, \dots, A_n 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:

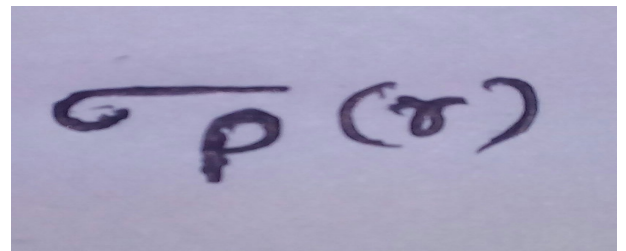
sid	name
1	a
2	b
3	c
4	d

(c)SELECTION:

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

SYNTAX:

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



Ex:

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

(a)Natural 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:

➔ 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

(a)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 returns 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 returns 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

RELATIONAL CALCULUS:

- ➔ It is alternative (or) contrast to the Relational Algebra.
- ➔ It is a Non-procedural language and more declarative compare with Relational Algebra.

- ➔ User-define Queries in terms of “what they want” and not in terms of “how they want”.
- ➔ It also contains variables, constants, comparison 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:
- $\{t \mid p(t)\}$, 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) comparison operator (Relational operator).
 - (c) connectivities (Logical operator) - \vee, \wedge, \mid, \sim
 - (d) Implications (\Rightarrow): $p \Rightarrow q$ that is $\sim p \vee q$
 - (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: $\{t \mid \text{student}(t) \wedge t.\text{age} > 20\}$

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

O/P: $\{t \mid \text{student}(t) \wedge t.\text{marks} > 70\}$

→ 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 exist 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: $\{ \langle a_1, a_2, \dots, a_n \rangle \mid p(a_1, a_2, \dots, a_n) \}$

where a_1, a_2, \dots, a_n are attributes and P is a predicate logic with several operators.

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

$$\{ \langle \text{sid}, \text{age} \rangle \mid \exists \text{ student} \wedge \text{age} > 20 \}$$

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

$$\{ \langle \text{eid}, \text{esalary} \rangle \mid \exists \text{ employee} \wedge \text{salary} > 50000 \}$$