RELATIONAL MODEL

The relational data model was first introduced by Ted Codd of IBM Research in 1970

Relational Model Concepts

- The relational model represents the database as a collection of relations
- A Relation is thought of as a table of values
- Each row in the table represents a collection of related data values.
- The table name and column names are used to interpret the meaning of the values in each row.
- The column names specify how to interpret the data values in each row.
- In the formal relational model terminology, **a row is called a tuple**, **a column header is called an attribute**, and **the table is called a relation**. The data type describing the types of values that can appear in each column is represented by a **domain** of possible values.

Domains:

- A domain D is a set of atomic values i.e. each value in the domain is indivisible
- Specifying a domain means specifying data types for columns from which the data values forming the domain are drawn.
- Ex: Names: The set of character strings that represent names of persons.
  Mobile_number: The set of ten-digit valid phone numbers

Relation Schema:

- A **relation schema R**, denoted by **$R(A_1, A_2, ..., A_n)$,** is made up of a **relation name R and a list of attributes, $A_1, A_2, ..., A_n$.**
- The value of Each **attribute A_i** comes from its **domain D** in the **relation schema R.**
- **D** is called the **domain of A_i** and is denoted by **$dom(A_i)$.**
- **A relation schema is used to describe a relation. *R*** *is called the* ***name of this relation.***

The **degree (or arity) of a relation** is the **number of attributes** present in the relation.

For example, ***R(A_1, A_2, A_3) has degree 3 and R(A_1, A_2, A_3, …, A_n) has degree n.***

A Relation schema can be represented as follows

STUDENT(Rollno , Name, dob, gender, course, grade)

Using the data type of each attribute, the definition is sometimes written as:

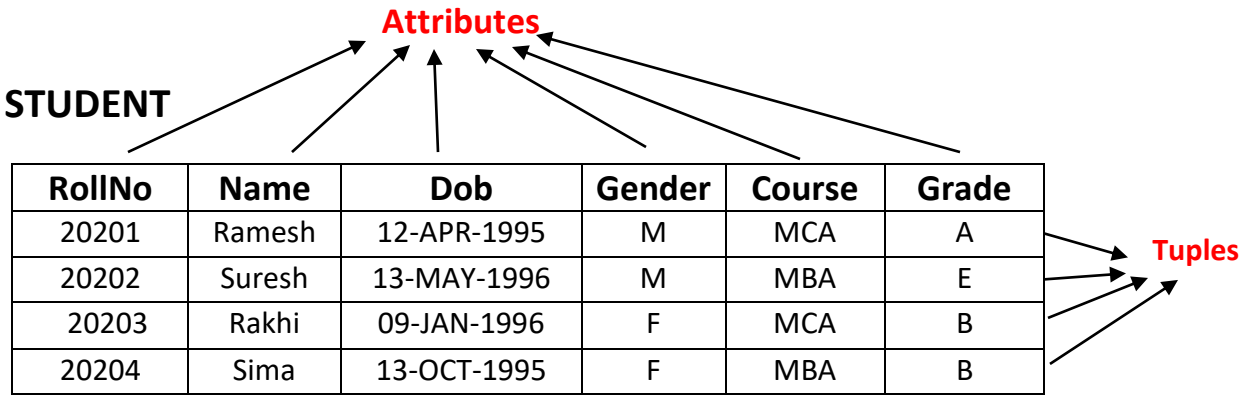
STUDENT (Rollno : number ,
Name : string, Ssn: string,
Dob : date,
Gender : String
Course : string,
Grade : string

Relation or Relation State:

A relation (or relation state) r of the relation schema **R(A_1, A_2, ..., A_n),** also denoted by **r(R), is a set of n-tuples r = {t_1, t_2, ..., t_m}.**

Each **n-tuple t** is an **ordered list of n values t =<v_1, v_2, ..., v_n>**, where each value **vi**, ***1 ≤ i ≤ n***, is an element of **dom (A_i)** or is a special **NULL** value.

Example :



RollNo	Name	Dob	Gender	Course	Grade
20201	Ramesh	12-APR-1995	M	MCA	A
20202	Suresh	13-MAY-1996	M	MBA	E
20203	Rakhi	09-JAN-1996	F	MCA	B
20204	Sima	13-OCT-1995	F	MBA	B

Mathematically, **A relation (or relation state) r(R) of degree n on the domains dom(A1), dom(A2), ..., dom(An), can be defined as a subset of the Cartesian product of the domains that define R:**

i.e. $r(R) \subseteq (dom(A_1) \times dom(A_2) \times ... \times dom(A_n))$

NULL Values:

NULL values are not zeros or empty strings. NULL values are used for the attributes in a tuple in the following situations

- **value unknown,**
- **value exists but is not available,**
- **or value undefined i.e. value of attribute does not apply to this tuple**

For example, consider student's passport information. Some students have a valid passport but other students have not. In this scenario, the value **NULL** is used for the students having no valid passport.

Relational Model Notation:

- A relation schema **R** of **degree n** is denoted by **R (A_1, A_2, ..., A_n).**
- The uppercase letters **Q, R, S** denote relation names.
- The lowercase letters **q, r, s** denote relation states.
- The letters **t, u, v** denote tuples.
- In general, the **name of a relation schema** such as **STUDENT** also indicates the **current set of tuples in that relation—the current relation state**—whereas **STUDENT (Rollno, Name, ...)** refers only to the **relation schema.**
- **An attribute A** can be qualified with the **relation name R to which it belongs** by using the dot notation **R.A**—for example, **STUDENT.Name** or **STUDENT.course** . This is because the same name may be used for two

attributes in different relations. *However, all attribute names in a particular relation must be distinct.*

- An **n-tuple t** in a relation **r(R)** is denoted by **t = <v_1, v_2, ..., v_n>**, where **v_i** is the value corresponding to attribute **A_i**.
- The following notation refers to component values of tuples:
  Both **t[A_i]** and **t.A_i** refer to the value **v_i** in t for attribute **A_i**.

Relational Model Constraints:

Constraints are basically various restrictions on data that can be specified on a relational database.

Types of Constraints:

- **Domain Constraints:**
  Domain constraints specify that within each tuple, the value of each attribute must be an atomic value from the domain that attribute.
- **Key Constraints:**
  A relation is defined as a set of tuples. By definition, all elements of a set are distinct. Hence, all tuples in a relation must also be distinct.

  This means that no two tuples can have the same combination of values for all their attributes *i.e. uniqueness property of tuples.*

  Usually, there are other subsets of attributes of a relation schema R with the property that no two tuples in any relation state r of R (*r(R)*) should have the same combination of values for these attributes.

  These subset attributes of a relation are known as **key** of that relation.

Let us consider a relation $R(A_1, A_2, A_3, \ldots, A_n)$.

A subset of attribute K of R is known as key of R ,

if for any two tuples t_i and t_j ,

$$t_i[k] \neq t_j[k] \text{ , where } i \neq j \text{ and } 1 \leq i \leq n$$

Types of Keys:

- **Superkey**
- **Key**
- **Candidate Key**
- **Primary Key**
- **Alternate keys**

Superkey:

A **superkey** specifies a uniqueness constraint that no two distinct tuples in any state of a relation can have the same value for the Superkey.

Consider a relation $R(A_1, A_2, A_3, \ldots, A_n)$.

A subset of attribute SK of R is known as Superkey of R ,

if for any two tuples t_i and t_j ,

$$t_i[sk] \neq t_j[sk] \text{ , where } i \neq j \text{ and } 1 \leq i \leq n$$

- Every relation has **at least one default superkey**—***the set of all its attributes.***
- A superkey can have redundant attributes i.e. removal of attributes from the Superkey set still has the uniqueness property.

Note:

- Key is a minimal superkey .i.e. a key can't contain redundant attributes.
- Every key is a superkey but not the *vice versa*.
- The key attributes in a relation are always underlined.

  e.g. In the relation **R ($\underline{A_1}$, A_2, $\underline{A_3}$, …, $\underline{A_i}$, $\underline{A_j}$…., A_n)**

  keys are **{A_1}. {A_3}, {A_i, A_j}**
- If a key contain s more than one attribute, it is called as **composite key**. In the above relation R , **{A_i, A_j} is a composite key.**

Candidate Key:

If more than one key is present in a relation, each key is known as a candidate key.

Primary Key:

If more than one candidate keys are present in a relation, **One of the Candidate key** is chosen as **primary key** with the following properties.

- A primary key is one or more than one attributes of a relation that is used to uniquely identify each tuple of that relation. It means that, the Primary key has tuple uniqueness property.
- It does not allow duplicate values.
- It does not allow **NULL values.**
- Only one primary key is allowed for a table.

Alternate keys:

All keys other than primary key in a relation are known **as Alternate keys.**

Integrity Constraints:

Integrity Constraints (IC) are of two types.

- Entity Integrity
- Referential Integrity

Entity Integrity Constraint:

The **entity integrity constraint** states that **no primary key value can be NULL.** This is because the primary key value is used to identify individual tuples in a relation. Having NULL values for the primary key implies that some tuples cannot be identified uniquely.

Referential Integrity Constraint:

Key constraints and entity integrity constraints are specified on individual relations. The **referential integrity constraint** is specified between two relations and is used to maintain the consistency among tuples in the two relations.

The **referential integrity constraint is** defined by the concept of foreign ***key.***

Foreign Key:

- A **Foreign Key** is a set of attributes whose value is derived from the primary key of some other table (or of same table) provided that the datatype and size of primary key and foreign key must be same.
- The table in which foreign key is defined is known as Child table or Dependent table.

- The table that defines Primary Key which is referenced by the Foreign key is known as Parent table or Master table.

In a more formal way, the Foreign Key can be defines as follows,

A foreign key specifies a referential integrity constraint between the two relation schemas $R1$ and $R2$.

A set of attributes FK in relation schema $\boldsymbol{R_1}$ is a **foreign key** of $\boldsymbol{R_1}$ that **references** relation $\boldsymbol{R_2}$ if it satisfies the following rules:

- The attributes in FK have the same domain(s) as the primary key attributes PK of R_2; the attributes FK are said to **reference** or **refer to** the relation R_2.
- A value of FK in a tuple t_1 of the current state $r_1(R_1)$ either occurs as a value of PK for some tuple t_2 in the current state $r_2(R_2)$ *or is NULL*.

  In the former case, we have $t_1[FK] = t_2[PK]$, and we say that the tuple t_1 **references** or **refers to** the tuple t_2.

In this definition, $\boldsymbol{R_1}$ is called the **referencing relation** and $\boldsymbol{R_2}$ is the **referenced relation**. If these two conditions hold, a **referential integrity constraint** from $\boldsymbol{R_1}$ to $\boldsymbol{R_2}$ is said to hold.

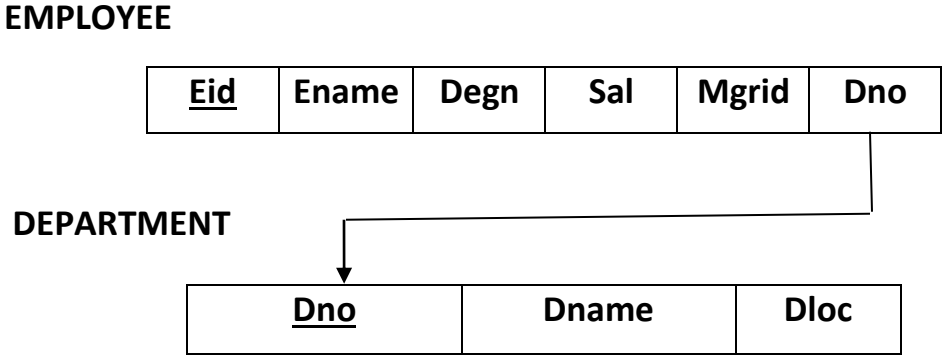
In a database of many relations, there are usually many referential integrity constraints.

For example consider two relations

EMPLOYEE*(Eid,Ename,Degn,Sal,Mgrid,Dno)* and
Depertment*(Dno,Dname,Dloc)*

In the **EMPLOYEE** relation, the attribute **Dno** refers to the department for which an employee works. Hence, we designate **Dno** to be a foreign key of **EMPLOYEE** referencing the **DEPARTMENT** relation.

This means that a value of **Dno** in any tuple t_1 of the **EMPLOYEE** relation must match a value of the **primary key** of **DEPARTMENT**—the **Dno** attribute—in some tuple t_2 of the **DEPARTMENT** relation, or the value of **Dno *can be NULL if the employee does not belong to a department or will be assigned to a department later.***



We can *diagrammatically display referential integrity constraints* by drawing a directed arc from each foreign key to the relation it references (*as shown above*). For clarity, the arrowhead may point to the primary key of the referenced relation.

Note:

A database state that does not obey all the integrity constraints is called an **invalid state**, and a state that satisfies all the constraints in the defined set of integrity constraints IC is called a **valid state.**

RELATIONAL ALGEBRA

→ The basic set of operations for relational model is the relational algebra

→ These operations specify the retrieval request operations

→ The relational algebra operations produce new relations which have been formed from one or more relations.

SELECT OPERATION

→ The SELECT operation is used to select a subset of tuples from a relation that satisfies a selection condition.

→ The SELECT operation can be visualized as a horizontal partition of the relation into two sets of tuples i.e. those tuples that satisfies the condition are selected and those tuples that don't satisfy the condition are discarded

Syntax

$$\sigma_{\langle \text{selection cond}^n \rangle}(R)$$

Ex: consider the relation Emp (eid, ename, degn, sal, dno)
find the employees who are working under dept no. 4.

$$\sigma_{Dno = 4}(Emp)$$

→ The selection condition is of the form

attribute name ⟨comparison operator⟩ Constant value

or,

attribute name ⟨comparison operator⟩ attribute name

the comparison operator is one of the operator from the set $\{=, <, \leq, >, \geq, \neq\}$

→ The selection condition is a Boolean Expression

→ More than one condition of the selection operation can be combined by the Boolean operators and, or, not to form a general selection condition.

Ex find the employees who are working under dept. number 4 and getting salary more than 25000/- per month.

Ans:

$$\sigma_{(Dno = 4) \wedge (sal > 25000)}(EMP)$$

Ex: Find the employees who are managers or getting more than 30000 as salary.

$$\sigma_{(dsgn = \text{'manager'}) \vee (sal > 3000)} (EMP)$$

Note

(I) The SELECT operation is unary operation (i.e. it is applied to a single relation)

(II) The SELECT operation is commutative

i.e. $\sigma_{cond1}(\sigma_{cond2}(R)) = \sigma_{cond2}(\sigma_{cond1}(R))$

(III) $deg(\sigma_c(R)) = deg(R)$

(IV) The no. of tuples obtained from SELECT operation is always less than or equals to the no. of tuples in R.

i.e. $|\sigma_c(R)| \leq |R|$

PROJECT OPERATION

↳ The project operation selects certain columns from a table and discards other columns.

↳ By project operation, certain attributes of a relation can be retrieved.

↳ The result of project operation can be visualized as a vertical partition of a relation.

Syntax

$$\Pi_{\langle \text{attribute list} \rangle}(R)$$

EX: find the names, degn and salary of all employees

Ans:

$$\Pi_{\text{name, degn, sal}}(EMP)$$

EX: find the eid, degn and salary of employees who are working under dept number 4

Ans

$$R_1 \leftarrow \sigma_{dno=4}(EMP)$$

$$RESULT \leftarrow \Pi_{eid, degn, sal}(R_1)$$

} sequence of operations.

OR,

$$\Pi_{eid, degn, sal}\left(\sigma_{dno=4}(EMP)\right)$$

Note

(I) The degree of project operation is equal to the number of attributes specified in the attribute list.

(II) The project operation removes duplicate tuples, if the attribute list contains only non-key attributes.

(III) $\Pi_{list\,1}\left(\Pi_{list\,2}^{(R)}\right) = \Pi_{list\,1}^{(R)}$

(if list 1 $\subseteq$ list 2)

(IV) It is a unary operator.

RENAME OPERATION

→ The rename operation is used to rename (I) the relation name
Or (II) the attribute name
Or (III) both the relation name and attribute name.

→ The rename operator is represented by the symbol ρ (rho)

Consider the relation $R(A_1, A_2, \ldots, A_n)$

$\rho_S(R)$ ← Renames the relation R to S

$\rho_{B_1, B_2, \ldots B_n}(R)$ ← Renames the attributes of R to $B_1, B_2, \ldots B_n$

$\rho_{S(B_1, B_2, \ldots B_n)}(R)$ ← Renames the attributes name and the relation name.

SET OPERATIONS IN RELATIONAL ALGEBRA

- → UNION (∪)
- → INTERSECTION (∩)
- → SET DIFFERENCE (or MINUS) (−)

The Union (∪), Intersection (∩) and minus (−) operations are binary operations

These set operations are applied on two relations provided the condition that, these two relations are Union compatible

UNION Compatibility

Two relations $R(A_1, A_2, \cdots A_n)$ and $S(B_1, B_2, \cdots B_m)$ are Union compatible

if (i) $deg(R) = deg(S)$ i.e. $n = m$

(ii) $dom(R \cdot A_i) = dom(S \cdot B_i)$, $1 \leq i \leq n$

This means that the two relations have the same number of attributes and each corresponding pair of attributes has the same domain.

UNION (∪)

$R \cup S$: $R \cup S$ is a relation that includes all the tuples that are either in R or in S or in both R and S. (Duplicate tuples are eliminated)

Mathematically $R \cup S = \{ t \mid t \in R \text{ or } t \in S \}$

INTERSECTION (∩)

$R \cap S$: $R \cap S$ includes all tuples that are both in R and S i.e. $R \cap S$ is a relation that contains tuples common to both R and S.

Mathematically $R \cap S = \{ t \mid t \in R \text{ and } t \in S \}$

Set Difference or Minus (−)

$R - S$: $R - S$ is a relation that includes all those tuples that are present in R but not in S.

Mathematically $R - S = \{ t \mid t \in R , t \notin S \}$

Note

(i) Union and intersection are commutative

i.e. $R \cup S = S \cup R$ and $R \cap S = S \cap R$

(ii) Union and intersection are associative

i.e. $(R \cup S) \cup T = R \cup (S \cup T)$

and $(R \cap S) \cap T = R \cap (S \cap T)$

(iii) Minus is not commutative

i.e. $R - S \neq S - R$

Ex: find eid, ename and degn of employees who are working under dept number 4 or dept number 5.

Ans:

$$R_1 \leftarrow \Pi_{eid, ename, degn} \left(\sigma_{dno=4} (Emp) \right)$$

$$R_2 \leftarrow \Pi_{eid, ename, degn} \left(\sigma_{dno=5} (Emp) \right)$$

$$RESULT \leftarrow R_1 - R_2$$

CARTESIAN PRODUCT OR CROSS PRODUCT (×)

- It is a binary operation
- The relations on which the cross product operation is applied don't have to be union compatible

- R × S is a new relation which includes the combinations of every tuple from one relation (R) with every other tuple from another relation (S)

Let $R(A_1, A_2, \ldots A_n)$ and $S(B_1, B_2, \ldots B_m)$ be two relations.

- then $R \times S = Q(A_1, A_2, \ldots A_n, B_1, B_2, \ldots B_m)$

i.e. $deg(R \times S) = deg(R) + deg(S)$

- Let R contains n_R no. of tuples and S contains n_S no. of tuples.

Then R × S contains $n_R * n_S$ no. of tuples.

i.e. if $|R| = n_R$ and $|S| = n_S$

$$\Rightarrow |R \times S| = |n_R * n_S|$$

Example

Let EMP1 (eid, ename, dno) and DEPT1 (dno, dname) be two relations.

EMP1

eid	ename	dno
e01	Ramesh	1
e02	Suresh	2
e03	Sruti	2
e04	Sham	3

DEPT1

dno	dname
1	finance
2	Research
3	Marketing

EMP1 X DEPT1

eid	ename	dno	dno	dname
e01	Ramesh	1	1	finance
e01	Ramesh	1	2	Research
e01	Ramesh	1	3	Marketing
e02	Suresh	2	1	finance
e02	Suresh	2	2	Research
e02	Suresh	2	3	Marketing
e03	Sruti	2	1	finance
e03	Sruti	2	2	Research
e03	Sruti	2	3	Marketing
e04	Sham	3	1	Finance
e04	Sham	3	2	Research
e04	Sham	3	3	Marketing

Example

Retrieve the eid, ename, degn of employees who are working under Research dept.

eid, ename, degn

Ans

$R_1 \leftarrow EMP \times DEPT$

$R_2 \leftarrow \sigma_{dname = 'Research'} (R_1)$

$RESULT \leftarrow \Pi_{eid, ename, degn} (R_2)$

JOIN OPERATION

→ Denoted by the symbol $\bowtie$

→ $R \bowtie_c S$ is a relation which combines tuples from both the relation R and S provided they satisfy a join condition. (Here c is the join condition)

→ Join operation is equivalent to performing cross product followed by a selection condition

i.e. $R \bowtie_c S = \sigma_c (R \times S)$

→ It is a very useful operation in Relational Algebra.

General form

The general form of a join operation on two relations $R(A_1, A_2, \cdots A_n)$ and $S(B_1, B_2, \cdots B_m)$ is

$$R \underset{\langle \text{join condition} \rangle}{\bowtie} S$$

→ The result of this join operation Q (say) contains combination of tuples from R and S when the combination satisfy the join condition.

→ Q has $(m+n)$ attributes in the following order

$$Q(A_1, A_2, \cdots A_n, B_1, B_2, \cdots B_m)$$

Depending upon the type of condition, we have the following types of joins

(i) Theta Join

(ii) Equi Join

(iii) Natural Join

Note The join operation is used to process relationship among relations.

Theta JOIN

Let $R(A_1, A_2, \cdots A_n)$ and $S(B_1, B_2, \cdots B_m)$ be two relations.

The θ join of R and S can be represented as

$$R \bowtie_{\theta} S$$

where θ is the join condition.

θ is of the form $cond_1 \wedge cond_2 \wedge \cdots \wedge cond_n$

where condition ($cond$) is of the form

$$A_i \ \langle \text{comparison operator} \rangle \ B_j$$

Comparison operator is one of the operator from the set $\{ =, >, \geq, <, \leq, \neq \}$

and $dom(A_i) = dom(B_j)$

i.e. the attributes on which comparison operator is applied must have the same domain.

EQUI JOIN

If Equality (=) is the only comparison operator used in the join condition, the join is called as Equi join.

Example

Let Emp (eid, ename, degn, sal, dno) and

DEPT (dno, dname, dloc)

Emp

eid	ename	degn	sal	dno
e01	Ramesh	Manager	35000	1
e02	Suresh	Executive	20000	2
e03	Sruthi	Executive	22000	2
e04	Rani	Sr. Manager	45000	3

DEPT

Dno	Dname	Dloc
1	finance	Bhubaneswar
2	Research	Bangalore
3	Marketing	Mumbai

EMP ⋈ DEPT

Emp.Dno = DEPT.Dno

eid	ename	degn	sal	dno	dno	dname	dloc
e01	Ramesh	Manager	35000	1	1	Finance	Bhubaneswar
e02	Suresh	Executive	20000	2	2	Research	Bangalore
e03	Sruthi	Executive	22000	2	2	Research	Bangalore
e04	Rani	Sr. Manager	45000	3	3	Marketing	Mumbai

Consider the following relations

Emp (eid, ename, degn, mgrid, sal, dno)

and DEPT (dno, dname, dloc)

Q: find eid, degn, salary of all employees who are working under 'Research' dept

Ans

$$R_1 \leftarrow \sigma_{Dname = \text{'Research'}} (EMP \bowtie_{Emp.DNO = DEPT.DNO} DEPT)$$

$$RESULT \leftarrow \Pi_{eid, degn, sal} (R_1)$$

Q: find the employees (eid, ename, sal) who are located in Bhubaneswar and getting salary less than 20000.

Ans

$$R_1 \leftarrow EMP \bowtie_{DNO} DEPT$$

$$R_2 \leftarrow \sigma_{(DLOC = \text{'Bhubaneswar'}) \wedge (sal < 2000)} (R_1)$$

$$RESULT \leftarrow \Pi_{eid, ename, sal} (R_2)$$

Natural Join

→ It is represented by * symbol.

→ It is a form of equi join

→ Natural join can be performed on two relations if there is atleast one common attribute exists between the two relations with the same name and same domain.

→ Natural join is basically an equi join followed by removal of superfluous (unnecessary) attributes.

Let $R(A_1, A_2, A_3, \cdots A_n)$ and $S(B_1, B_2, A_3, \cdots B_m)$ be two relations where attribute A_3 is common in both R and S (i.e. same name A_3 in both R and S and $dom(R.A_3) = dom(S.A_3)$)

$$R * S = Q(A_1, A_2, A_3, \cdots, A_n, B_1, B_2, \cdots, B_m)$$

→ Join condition is not specified in the natural join (*) operation as it is a variation of equi join operation and the equality condition is performed on the common attribute.

Ex: Consider the two relations

Emp (eid, ename, degn, sal, dno)

and DEPT (dno, dname, dloc)

where a common attribute 'dno' is present in both the relations and dom (Emp.dno) = dom (dept.dno)

Emp

eid	ename	degn	sal	dno
e01	Ramesh	manager	35000	1
e02	Suresh	Executive	20000	2
e03	Sham	manager	40000	2
e04	Sheli	clerk	21000	3

DEPT

dno	dname	dloc
1	Finance	Bhubaneswar
2	Research	Bangalore
3	Marketing	NewDelhi

EMP * DEPT

eid	ename	degn	sal	dno	dname	dloc
e01	Ramesh	Manager	35000	1	Finance	Bhubaneswar
e02	Suresh	Executive	20000	2	Research	Bangalore
e03	Sham	Manager	40000	2	Research	Bangalore
e04	Sheli	clerk	21000	3	Marketing	New Delhi

JOIN

Join are of two types

- Inner Join (Theta join, Equi join, Natural join)
- Outer join

Outer JOIN

Outer join are of 3-types

- Left Outer join
- Right outer join
- full outer join

Left Outer JOIN

→ Represented by 

→ It keeps all the tuples of left relation in the result.

→ for some tuple t in the left relation, if no matching tuple found in right relation, then the attributes of right relation are made NULL in the result.

consider the following example

R

col 1	col 2
A	1
B	2
D	3
F	4
E	5

S

col 1	col 2
A	A1
C	A2
D	A3
E	A4

R ⋈ S

R.col1 = S.col1

col 1	col 2	col 1	col 2
A	1	A	A1
B	2	NULL	NULL
D	3	D	A3
f	4	NULL	NULL
E	5	E	4

Right Outer Join

→ Represented by ⟖

→ It keeps all the tuple of the right relation in the result.

→ If some tuple t in the right relation, if no matching tuple found in left relation then the attribute of ~~left relation~~ left relation are made NULL

R

col1	col2
A	1
B	2
D	3
F	4
E	5

S

col1	col2
A	A_1
C	A_2
D	A_3
E	A_4

R ⟖ S

R.col1 = S.col1

col1	col2	col1	col2
A	1	A	A_1
NULL	NULL	C	A_2
D	3	D	A_3
E	5	E	A_4

FULL OUTER JOIN

→ Represented by ⟗

→ All tuples of both the relations are in the result. for non-matching tuples NULL will be supplied

R

col1	col2
A	1
B	2
D	3
F	4
E	5

S

col1	col2
A	A1
C	A_2
D	A_3
E	A_4

R ⟗ S

R.col1 = S.col1

col1	col2	col1	col2
A	1	A	A1
B	2	NULL	NULL
NULL	NULL	C	A_2
D	3	D	A_3
f	4	NULL	NULL
E	5	E	A_4

Division operation

Consider two relation R and S in which R has exactly two fields x and y and S has just one field y with the same domain as in R.

We define division operation R/S (or R ÷ S) as the set of all x values such that, for every y value in S there is a tuple $\langle x, y \rangle$ in R.

i.e. Let R(x, y) and S(y) be two relations such that dom(R.y) = dom(S.y)

R/S is the set of all x values in R such that $\forall y \in S$, $\exists$ a tuple $\langle x, y \rangle$ in R.

Example

EMP21

eid	DNO
e1	D2
e2	D1
e3	D2
e2	D2
e4	D3
e2	D3
e5	D1

DEPT21

DNO
D1
D2
D3

EMP21 / DEPT21

Eid
e2

Example

Consider the relations Emp (eid, ename, degn, sal, dno) and DEPT (DNO, Dname, Dloc). find the eid of employees who works for all the departments.

Ans:

$$R_1 \leftarrow \Pi_{eid, dno}(EMP)$$

$$R_2 \leftarrow \Pi_{dno}(DEPT)$$

$$RESULT \leftarrow R_1 / R_2$$

Q: find the eid and names of employees who works for all the departments

Ans

$$R_1 \leftarrow \Pi_{eid, dno}(Emp)$$

$$R_2 \leftarrow \Pi_{dno}(DEPT)$$

$$R_3 \leftarrow R_1 / R_2$$

$$RESULT \leftarrow \Pi_{eid, ename}(R_3 \bowtie Emp)$$

Note:

Complete Set of Relational Algebra Operations

The set of Relational Algebra operations $\{\sigma, \pi, \cup, -, \times\}$ is a complete set i.e. any other relational algebra operations can be expressed as a sequence of operations from this set.

e.g. $R \cap S \equiv (R \cup S) - ((R - S) \cup (S - R))$

$R \underset{\text{cond}^n}{\bowtie} S \equiv \sigma_{\text{cond}^n}(R \times S)$