UNIT-2

Relational Data Model, Relational Algebra and Calculus

PRELIMINARIES

In defining relational algebra and calculus, the alternative of referring to fields by position is more convenient than referring to fields by name: Queries often involve the computation of intermediate results, which are themselves relation instances, and if we use field names to referto fields, the definition of query language constructs must specify the names of fields for all intermediate relation instances.

We present a number of sample queries using the following schema:

Sailors (sid: integer, sname: string, rating: integer, age: real)

Boats (bid: integer, bname: string, color: string)

Reserves (sid: integer, bid: integer, day: date)

The key fields are underlined, and the domain of each field is listed after the field name. Thus sidis the key for Sailors, bid is the key for Boats, and all three fields together form the key for Reserves. Fields in an instance of one of these relations will be referred to by name, or positionally, using the order in which they are listed above.

RELATIONAL ALGEBRA

Relational algebra is one of the two formal query languages associated with the re- lational model. Queries in algebra are composed using a collection of operators. A fundamental property is that every operator in the algebra accepts (one or two) rela-tion instances as arguments and returns a relation instance as the result.

Each relational query describes a step-by-step procedure for computing the desired answer, based on the order in which operators are applied in the query.

Selection and Projection

Relational algebra includes operators to *select* rows from a relation (σ) and to *project* columns (π).These operationsallow usto manipulate dataina single relation. Con -sider the instance of the Sailors relation shown in Figure 4.2, denoted as *S2*. We can retrieve rows corresponding to expert sailors by using the σ operator. The expression,

$\sigma_{rating>8}(S2)$

The selection operator σ specifies the tuples to retain through a *selection condition*. In general, the selection condition is a boolean combination (i.e., an expression using the logical connectives

A and ∨) of*terms* that have the form*attribute* op *constant* or *attribute1* op *attribute2*, where op isone ofthe comparison operators <, <=, =, =, >=, or >.

The projection operator π allows us to extractcolumns from a relation; for example, we can find out allsailor names and ratings byusing π. The expression $\pi_{sname,rating}(S2)$

Supposethatwewantedtofindoutonlytheagesofsailors.Theexpression

$\pi_{age}(S2)$

asingletuplewith*age=35.0*appearsintheresultoftheprojection.Thisfollowsfrom

the definition of a relation as a *set* of tuples. However, our discussion of relational algebra and calculus assumes that duplicate elimination is always done so that relations are always sets of tuples.

SetOperations

The following standard operations on set sareal so available inrelational algebra:*union*(*U*),

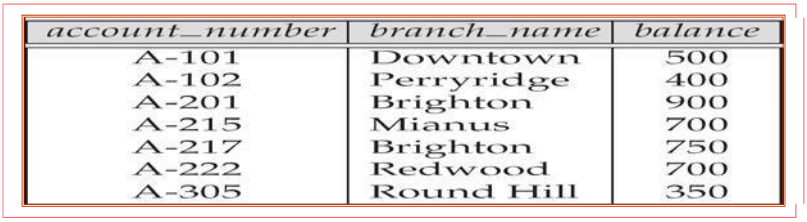
intersection(∩),*set-difference*(−),and*cross-product*(×).

- Union: *R u S* returns a relation instance containing all tuples that occur in *either* relation instance *R* or relation instance *S* (or both). *R* and *S* must be *union-compatible*, and the schemaofthe resultis defined to be identicalto the schema of*R*.
- Intersection: $R \cap S$ returns a relation instance containing all tuples that occur in *both* *R* and *S*.Therelations*R* and*S* must beunion-compatible, andtheschemaoftheresult isdefinedto beidentical to the schema of *R*.
- Set-difference: $R - S$ returns a relation instance containing all tuples that occur in *R*but not in *S*. The relations *R* and *S* must be union-compatible, and the schema of the result is defined tobe identicalto the schema of *R*.
- Cross-product:$R \times S$returnsarelation instancewhoseschemacontainsallthefieldsof *R*(inthesame orderastheyappearin*R*)followedbyallthefieldsof*S*

(inthesameorderastheyappearin*S*). Theresultof$R \times S$containsonetuple $\langle r, s \rangle$ (the concatenation of tuples *r* and *s*) for each pair of tuples $r \in R$, $s \in S$. The cross-product opertion is sometimes called Cartesian product.

RELATIONALMODEL

Relational model is simple model is simple model in which database is represented as acollection of “relations”



account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

where each relation is represented by two-dimensional table

The relational model was founded byE.F.Coddof the IBM in 1972.The basicconcept in the relational model is that of a relation.

Properties:

- Itiscolumnhomogeneous.Inotherwords,inanygivencolumnofatable,all items are of the same kind.
- Each item is a simple number or a character string. That is a table must be in first normal form.
- Allrowsofatableare distinct.
- Theorderingofrowswithinatableis immaterial.

- Thecolumnofatableareassigneddistinctnamesandtheorderingofthese columns in immaterial.

Domain,attributestuplesandrelational:

Tuple:

Eachrowinatablerepresentsarecordandiscalledatuple.Atable containing ‘n’ attributes in a record is called is called n-tuple.

Attributes:

The name of each column in a table is used to interpret its meaning and is called anattribute.Each table is called a relation.

Intheabovetable,account_number,branchname,balanceareattributes.

Domain:

A domain is a set of values that can be given to an attributes. So every attribute in a table has a specific domain. Values to these attributes can not be assigned outside their domains.

Relation:

Arelationconsistof

- **Relationalschema**
- **Relationinstance**

Relationalschema:

Arelationalschemaspecifiestherelation'name,itsattributesandthedomainofeach attribute. If R is the name of a relation and A1,A2,… and is a list of attributes representing R then R(A1,A2,…,an) is called a relational schema. Each attribute in this relational schema takes a value from some specific domain called domain(Ai).

Example:

PERSON(PERSON_IDinteger,NAME:STRING,AGE:INTEGER,ADDRESS:string)

Total number of attributes in a relation denotes the degree of a relation.since the PERSON relation schemea contains four attributes ,so this relation is of degree 4.

RelationInstance:

Arelationalinstancedenotedasrisacollectionoftuplesforagivenrelational schema at a specific point of time.

A relation state r to the relations schema R(A1,A2…,An) also denoted by r® is a set of n-tuples

R{t1,t2,…tm}

Whereeachn-tupleisanorderedlistofnvalues

T=<v1,v2,….vn>

Whereeachvibelongstodomain(Ai)orcontainsnullvalues.

Therelationschemaisalsocalled'intension'andtherelationstateisalsocalled 'extension'.

Eg:

Relationschemaforstudent:

STUDENT(rollno:strinhg,name:string,city:string,age:integer)

Relationinstance:

Student:

Rollno	Name	City	Age
101	Sujit	Bam	23
102	kunal	bbsr	22

Keys:

Superkey:

Asuperkeyisanattributeorasetofattributesusedtoidentifytherecordsuniquelyin a relation.

Forexample,customer-id,(cname,customer-id),(cname,telno)

Candidatekey:

Super keys of a relation can contain extra attributes . candidate keys are minimal super keys. i.e, such a key contains no extraneous attribute. An attribute is called extraneous if even after removing it from the key, makes the remainingattributes still has the properties of a key.

InarelationR,acandidatekey forRisasubsetofthesetofattributesofR,which have the following properties:

- *Uniqueness:* notwodistincttuples inRhavethesamevalues for thecandidatekey
- *Irreducible:* Nopropersubsetofthecandidatekeyhasthe *uniquenesspropertythat isthecandidatekey.*
- *Acandidatekey'svaluesmustexist.Itcan'tbe null.*
- *The values of a candidate key must be stable. Its value can not change outside the control of the system.*

Eg:(cname,telno)

Primarykey:

The primary key is the candidate key that is chosen by the database designer as the principal means of identifying entities with in an entity set. The remaining candidate keys if any are called ***alternate key*.**

RELATIONALCONSTRAINTS:

Therearethreetypesofconstraintsonrelationaldatabasethat include

- DOMAINCONSTRAINTS
- KEYCONSTRAINTS
- INTEGRITYCONSTRAINTS

DOMAINCONSTRAINTS:

Itspecifiesthateachattributeinarelationanatomicvaluefromthecorresponding domains. The data types associated with commercial RDBMS domains include:

- Standardnumericdatatypesforinteger
- Real numbers
- Characters
- Fixedlengthstringsandvariablelength strings

Thus,domainconstraintsspecifiestheconditionthatwetoputoneachinstanceofthe relation. So the values that appear in each column must be drawn from the domain associated with that column.

Rollno	Name	City	Age
101	Sujit	Bam	23
102	kunal	bbsr	22

Keyconstraints:

This constraints states that the key attribute value in each tuple msut be unique .i.e, no two tuples contain the same value for the key attribute.(null values can allowed)

Emp(empcode,name,address).hereempcodecanbeunique

Integrityconstraints:

Therearetwotypesofintegrityconstraints:

- Entityintegrityconstraints
- Referentialintegrityconstraints

Entityintegrityconstraints:

Itstatesthatnoprimarykeyvaluecanbenullandunique.Thisisbecausetheprimarykey is used to identify individual tuple in the relation. So we will not be able to identify the records uniquely containing null values for the primary key attributes. This constraint is specified on one individual relation.

Referentialintegrityconstraints:

Itstatesthatthetupleinonerelationthatreferstoanotherrelationmustrefertoan existing tuple in that relation. This constraints is specified on two relations .
Ifacolumnisdeclaredasforeignkeythatmustbeprimarykeyofanother table.

Department(deptcode,dname) Heret
hedeptcodeistheprimarykey.
Emp(*empcode*,name,city,*deptcode*).
Here the deptcode is foreign key.

CODD'SRULES

Rule 1:Theinformation Rule.

"All information in a relational data base is represented explicitlyat the logical level and in exactly one way - by values in tables."

Everythingwithinthedatabaseexistsintablesandisaccessedviatableaccess routines.

Rule2 : Guaranteed accessRule.

"Each and every datum (atomic value) in a relational data base is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name."

To access any data-item you specify which column within which table it exists, there isno reading of characters 10 to 20 of a 255 byte string.

Rule 3: Systematic treatment ofnullvalues.

"Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type."

If data does not exist or does not apply then a value of NULL is applied, this is understood by the RDBMS as meaning non-applicable data.

Rule4: Dynamicon-line catalogbasedon therelationalmodel.

"The data base description is represented at the logical level in the same wayas-ordinary data,sothatauthorizeduserscanapplythesamerelationallanguagetoitsinterrogationas they apply to the regular data."

The Data Dictionary is held within the RDBMS, thus there is no-need for off-line volumes to tell you the structure of the database.

Rule5:Comprehensivedatasub-language Rule.

"A relational system may support several languages and various modes of terminal use (for example, the fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and that is comprehensive in supporting all the following items

- DataDefinition

- ViewDefinition
- DataManipulation(Interactiveandbyprogram).
- IntegrityConstraints
- Authorization.

Every RDBMS should provide a language to allow the user to query the contents of the RDBMS and also manipulate the contents of the RDBMS.

Rule 6 :.ViewupdatingRule

"Allviewsthataretheoreticallyupdateablearealsoupdateablebythesystem."

Notonlycantheusermodifydata,butsocantheRDBMSwhentheuserisnotlogged-in.

Rule7:High-levelinsert,updateanddelete.

"The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data."

The user should be able to modifyseveral tables bymodifyingthe view to which theyact as base tables.

Rule8 :Physicaldataindependence.

"Application programs and terminal activities remain logicallyunimpaired whenever any changes are made in either storage representations or access methods."

Theusershouldnotbeawareofwhereoruponwhichmediadata-filesare stored

Rule9 : Logicaldataindependence.

"Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit un-impairment are made to the base tables."

User programs and the user should not be aware of any changes to the structure of the tables (such as the addition of extra columns).

Rule10:Integrityindependence.

"Integrity constraints specific to a particular relational data base must be definable in the relational data sub-language and storable in the catalog, not in the application programs."

If a column only accepts certain values, then it is the RDBMS which enforces these constraints and not the user program, this means that an invalid value can never be entered into this column, whilst if the constraints were enforced via programs there is always a chance that a buggyprogram might allow incorrect values into the system.

Rule11:Distribution independence.

"A relational DBMS has distribution independence."

The RDBMS may spread across more than one system and across several networks, however to the end-user the tables should appear no different to those that are local.

Rule12:Non-subversionRule.

"If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity Rules and constraints expressed in the higher level relational language (multiple-records-at-a-time)."

Tuple Relational Calculus (TRC) in DBMS

Tuple Relational Calculus (TRC) is a non-procedural query language used in relational database management systems (RDBMS) to retrieve data from tables. TRC is based on the concept of tuples, which are ordered sets of attribute values that represent a single row or record in a database table.

TRC is a declarative language, meaning that it specifies what data is required from the database, rather than how to retrieve it. TRC queries are expressed as logical formulas that describe the desired tuples.

Syntax: The basic syntax of TRC is as follows:

{ t | P(t) }

where t is a tuple variable and P(t) is a logical formula that describes the conditions that the tuples in the result must satisfy. The curly braces {} are used to indicate that the expression is a set of tuples.

For example, let's say we have a table called "Employees" with the following attributes:

Employee ID

Name

Salary

Department ID

To retrieve the names of all employees who earn more than $50,000 per year, we can use the following TRC query:

{ t | Employees(t) ∧ t.Salary > 50000 }

In this query, the "Employees(t)" expression specifies that the tuple variable t represents a row in the "Employees" table. The "∧" symbol is the logical AND operator, which is used to combine the condition "t.Salary > 50000" with the table selection.

The result of this query will be a set of tuples, where each tuple contains the Name attribute of an employee who earns more than $50,000 per year.

TRC can also be used to perform more complex queries, such as joins and nested queries, by using additional logical operators and expressions.

While TRC is a powerful query language, it can be more difficult to write and understand than other SQL-based query languages, such as Structured Query Language (SQL). However, it is useful in certain applications, such as in the formal verification of database schemas and in academic research.

Tuple Relational Calculus is a non-procedural query language, unlike relational algebra. Tuple Calculus provides only the description of the query but it does not provide the methods to solve it. Thus, it explains what to do but not how to do it.

Tuple Relational Query

In Tuple Calculus, a query is expressed as

$\{t| P(t)\}$

where t = resulting tuples,

P(t) = known as Predicate and these are the conditions that are used to fetch t. Thus, it generates a set of all tuples t, such that Predicate P(t) is true for t.

P(t) may have various conditions logically combined with OR (∨), AND (∧), NOT(¬).

It also uses quantifiers:

$\exists\ t \in r\ (Q(t))$ = ”there exists” a tuple in t in relation r such that predicate Q(t) is true.

$\forall\ t \in r\ (Q(t))$ = Q(t) is true “for all” tuples in relation r.

Domain Relational Calculus (DRC)

Domain Relational Calculus is similar to Tuple Relational Calculus, where it makes a list of the attributes that are to be chosen from the relations as per the conditions.

{<a1,a2,a3,.....an> | P(a1,a2,a3,.....an)}

where a1,a2,…an are the attributes of the relation and P is the condition.

Domain Relational Calculus in DBMS

Database management systems (DBMS) employ the non−procedural query language known as Domain Relational Calculus (DRC). DRC focuses simply on what data to collect without outlining the techniques for retrieval, as opposed to Relational Algebra, which provides methods and procedures for fetching data. It offers a declarative method of database querying.

Syntax

```
{ <x1, x2, ..., xn> | P(x1, x2, ..., xn) }
```

Here,

<x1, x2, ..., xn> refers to the resulting domain variable

P(x1, x2, ..., xn) refers to the condition equivalent to the predicate calculus.

Example 1

This example shows us to solve the query which is how to find the names of students who are 20 years old from the given table.

Students

ID	Name	Age
1	John	20
2	Sarah	22
3	Emily	19
4	Michael	21

DRC Expression

{<Name> | $\exists$ ID, Age (<ID, Name, Age> ϵ Students ∧ Age = 20)}

Output

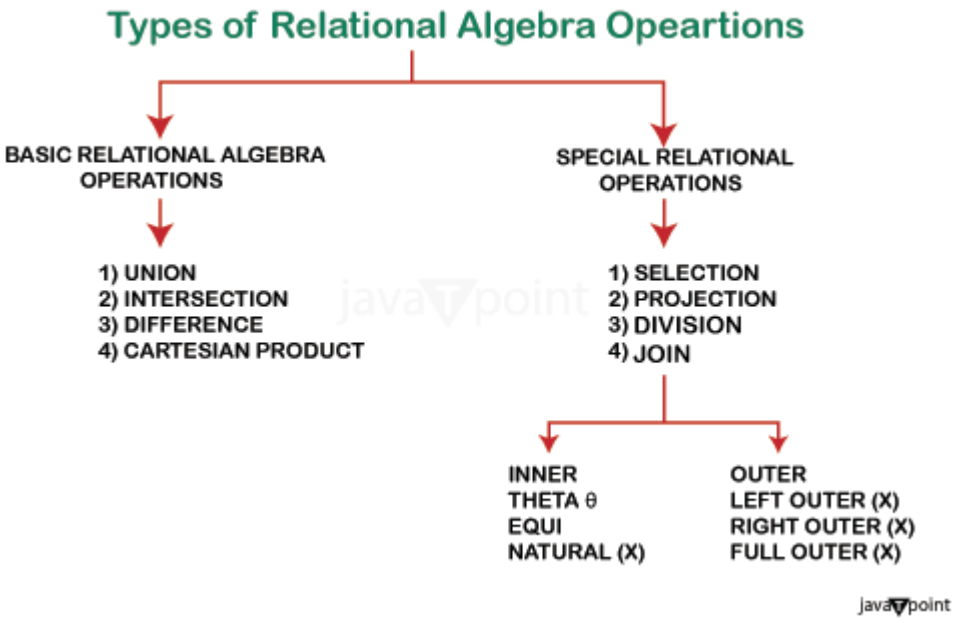
Name

John

Relational Algebra

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.

It is also a conceptual language. By this we mean that the queries made in it are not run on the computer so it is not used as a business language. However, this knowledge allows us to understand the optimization and query execution of RDBMS.



Basic Set Oriented Operations: It is also known as traditional set oriented operations. This operation is derived from the mathematical set theory. Following operations include in basic set oriented operations. All operations are binary operations which means that operations applies to pair of relations.

- UNION
- INTERSECTION
- DIFFERENCE
- CARTERSION PRODUCT

Following conditions are satisfied is both the relations are union

- All the relations must have the same attribute

- Each column in the first relation must have the same data type as the corresponding column in the second relation. The names of the corresponding attributes need not be the same.

For example: Consider two relations P and Q such that degrees of P and Q are m and n. Degree must be equal then m=n.

Special Relational Operations: These operations focus on the structure of the tuples. These operations not only add power to algebra but also simplify general questions that are too long to express using set oriented operations. Special relational operations include the following operations:

- JOIN
- SELCTION
- Projection
- Division

Let's explain each one by one in detail.

1. Select Operation:

- The select operation also known as restriction operation results in a new relation that contains those rows of relation that satisfy a specified condition.
- The select operation selects tuples that satisfy a given predicate.
- It is denoted by sigma (σ).

1. Notation: σ p(r)

Where:

σ is used for selection prediction
r is used for relation
p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like =, ≠, ≥, <, >, ≤.

For example: LOAN Relation

Input:

1. σ BRANCH_NAME="perryride" (LOAN)

Output:

Features of Select operation:

- It is a unary operation because works only a single relation.
- The resulting table has the same degree as that of the original table. This means that the number of columns in both the relations is same.
- The number of rows of the resulting relations is always less than or equal to the original relation.
- It operates on each row of the relation independently.

2. Project Operation:

- This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.
- It is denoted by ∏ and the attributes to be retrieved appear as subscripts separated by commas and relations name is given in parenthesis following the PI.

1. Notation: ∏ A1, A2, An (r)

Where

A1, **A2**, **A3** is used as an attribute name of relation **r**.

Example: CUSTOMER RELATION

Input:

1. ∏ **NAME**, CITY (CUSTOMER)

Output:

Features of Project operation:

- It is a unary operation i.e. it can operate only a single relation.
- The degree of the resulting relation is equal to the number of attributes specified in the attribute list
- If the attribute list contains a primary key attribute then the number of tuples in the resulting relation is equal to the number of tuples in the original relation.

- **Non-commutative:** It does not hold the commutative property.
- **Duplicate Elimination:** This operation removes the duplicate rows from the table which results in a valid relation knows as duplicate elimination.

3. Union Operation:

- The Union operation of two relations results in a new relation containing rows from both relations with duplicates removed.
- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
- It eliminates the duplicate tuples. It is denoted by ∪.

1. Notation: R ∪ S

A union operation must hold the following condition:

- R and S must have the attribute of the same number.
- Duplicate tuples are eliminated automatically.

Features of Union operation:

- Input relations must be union compatible.
- **Commutativity:** This means that the result of (R ∪ S) is same as that of (S ∪ R).
- **Associativity:** This mean that R ∪ (S ∪ O) = (R ∪ S) ∪ O where R, S and O are relations.

Input:

1. ∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

4. Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection ∩.

1. Notation: R ∩ S

Features of Intersection operation:

- Input relations must be union compatible.
- **Commutativity:** This means that result of (R ∩ S) is same as that of (S ∩ R).

- **Associativity:** This mean that R ∩ (S ∩ O) = (R ∩ S) ∩ O where R, S and O are relations.

Example: Using the above DEPOSITOR table and BORROW table

5. Difference:

- The difference of two relations results in a new relation that contains tuples that occur in the first relation but not in the second relation.
- Suppose there are two tuples R and S. The set difference operation contains all tuples that are in R but not in S.
- It is denoted by intersection minus (-).

1. Notation: R - S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1. ∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)

Features of Set Difference operation:

- Input relations must be union compatible.
- They are not commutative. This means that the result of R - Sis not the same as the result S - P.
- They are not associative. This means that result of Q - (R - S) is not the same as the result of (Q - S) - R where Q, S and R are relations.
- Intersection can be expressed as difference (-) operations:

(R ∩ S) = R - (R - S)

But writing the equation with a single intersection operation is more convenient than involving a pair of difference operations. Here R and S unions are favorable.

6. Cartesian product

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by X.

- It is a binary relation which means that it always operates on two relations.

Properties of Cartesian product operation:

- The relations to which Cartesian product operation is applied need not necessary to be union compatible.
- The total number of rows in the result operation is equal to the product of the number of rows in the first and second relations, i.e.

Total number of tuples of data = Total number of tuples of E + Total number of tuples of D

- The resulting relation may have duplicate properties if some properties of the two relations are defined on common domains.
- The resulting degree of action is equal to the sum of the degrees of all relations

Degree of E = Degree of E + Degree of D

- **Commutativity:** This means that result of E X D is same as that of D X
- **Associativity:** This mean that E X (D X F) = (E X D) X F where E, D and F are relations.

7. Rename Operation:

The rename operation is used to rename the output relation. It is denoted by **rho** (ρ). If no rename operation is applied then the names of the attributes in a resulting relation are the same as those in the original relation and in the same order.

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

1. ρ(STUDENT1, STUDENT)

Note: Apart from these common operations Relational algebra can be used in Join operations.

8. Join Operation:

A join operation combines two or more relations to form a new relation such that new relation contains only those tuples from different relations that satisfy the specified criteria. It forms a new relation which contains all the attributes from both the joined relations whose tuples are those defined by the restrictions applies i.e. the join condition applied on two participating relations. The join is performed on two relations, who have one or more

attributes in common. These attributes must be domain compatible i.e. they have same data type. It is a binary operation. The Join operation is denoted by a Join symbol. The general from of representing a join operation on two relations P and Q is

1. P⋈ <join_condion> Q
   When we use equality operator in the join condition then such a join is called EQUI Join. It is mostly commonly used Join.

   In the above both table, suppose we want to know the employee information with department name in which each employee is working. Now the employee information is in the Emp relation and Department name information is in dept relation. So to retrieve the columns from both the tables at same time, we need to join the EMP and DEPT relations. The relations can be joined over the column Dept_ID that exist in EMP relation and the Dept_no that exist in the DEPT relation and are domain compatible. Thus the result of EQUI Join where the condition is that Dept_Id attributes values the EMP relation should be equal to the Dept_No attribute values in the DEPT relation, **the result is shown below.**

   Another is known as natural join in which there is no need to explicitly name the columns. A join is performed by joining all columns from the first relation of any column to another relation with the same name. The result of natural join is as

   The natural join may also been referred to as INNER JOIN. Another type of JOIN in which a relation is joined to itself by comparing values with a column of the relation is called a self-join.

   In the EMP relation, the attribute EMP_ID shows employee'code, ENAME and Mang_Id under which employee is working. In this, some employee are not having Mang_Id i.e. their value is null because they act as a manager itself.

   Features of Join Operation:

   - Like the cartersion product, join operations are commutative. Using this property, we can choose which relation can be the inner and which one the outer while joining two relations. If P and Q are two relations then,

P⋈ Q = Q ⋈ P

- Rows whose join attribute is null do not appear in the resulting relation.
- We can also join more than two relations by increasing the complexity.
- Joins are generally used when a relationship exists between relations such as where the join condition is based on the primary key and foreign key columns.
- Join is a very powerful operator and together with projection form a base for normalization i.e. theoretical support for designing database relations.
- If the attributes on which the join is performed have the same name in both the relations then renaming is necessary for the EQUIJOIN operation and unnecessary for the NATURAL JOIN operation because the former i.e. EQUI JOIN both exist as a result of the common attribute relation but the latter In i.e., natural additive consequent relations have only one common property.

Division Operation:

The division operation results in a new relation such that every tuple appearing in the resulting relation must exist in the dividend relation. In the combination of each tuple in the denominator relation. It is a binary operation that operates on **Subject Relation "÷" Course Relation**

The resultant relation is: Result Relation

Properties of Division Relation:

- It is a binary operation as it operators on two relations
- The division operation is suited to queries that include the phrase "for all".
- It is very rarely used in database applications.