Relational Algebra

- Relational algebra is the basic set of operations for the relational model
- These operations enable a user to specify basic retrieval requests as relational algebra expressions.
- The result of an operation is a new relation, which may have been formed from one or more input relations

- The relational algebra is very important for several reasons

  - provides a formal foundation for relational model operations.

  - used as a basis for implementing and optimizing queries in the query processing and optimization modules that are integral parts of relational database management systems (RDBMSs)

  - some of its concepts are incorporated into the SQL standard query language for RDBMSs

Unary Relational Operations: SELECT and PROJECT

The SELECT Operation

- The SELECT operation denoted by **σ** (sigma) is used to select a subset of the tuples from a relation based on a **selection condition**
- The selection condition acts as a **filter**
- Tuples satisfying the condition are selected whereas the other tuples are discarded (filtered out)
- The SELECT operation can also be visualized as a horizontal partition of the relation into two sets of tuples

  - those tuples that satisfy the condition and are selected
  - those tuples that do not satisfy the condition are discarded

- In general, the select operation is denoted by

$$\sigma_{\text{<selection condition>}}(R)$$

where

- the symbol σ is used to denote the select operator
- the selection condition is a Boolean (conditional) expression specified on the attributes of relation R
- tuples that make the condition **true** are selected
  - appear in the result of the operation
- tuples that make the condition **false** are filtered out
  - discarded from the result of the operation

- The Boolean expression specified in <selection condition> is made up of a number of **clauses** of the form

  <attribute name> <comparison op> <constant value>

  or

  <attribute name> <comparison op> <attribute name>

where

<attribute name> is the name of an attribute of *R*,

<comparison op> is one of the operators {=, <, ≤, >, ≥, ≠}, and

<constant value> is a constant value from the attribute domain.

- Clauses can be connected by the standard Boolean operators ***and***, ***or***, and ***not*** to form a general selection condition

- Examples:
  - Select the EMPLOYEE tuples whose department number is 4:

$$\sigma_{DNO = 4} (EMPLOYEE)$$

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	gender	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

- Select the employee tuples whose salary is greater than $30,000:

$$\sigma_{SALARY > 30,000}(EMPLOYEE)$$

- Select the tuples for all employees who either work in department 4 and make over $25,000 per year, or work in department 5 and make over $30,000

$$\sigma_{(Dno=4\ \textbf{AND}\ Salary>25000)\ \textbf{OR}\ (Dno=5\ \textbf{AND}\ Salary>30000)}(EMPLOYEE)$$

- The result of a SELECT operation can be determined as follows:
  - The <selection condition> is applied independently to each individual tuple t in R
  - If the condition evaluates to TRUE, then tuple *t* is selected
  - All the selected tuples appear in the result of the SELECT operation
- The SELECT operator is **unary**; that is, it is applied to a single relation
- The **degree** of the relation resulting from a SELECT operation

  is the same as the degree of R
- The number of tuples in the resulting relation is always less than or equal to the number of tuples in R

- The SELECT operation is **commutative**; that is,

$$\sigma_{<cond1>}(\sigma_{<cond2>}(R)) = \sigma_{<cond2>}(\sigma_{<cond1>}(R))$$

Hence, a sequence of SELECTs can be applied in any order

- In SQL, the SELECT condition is specified in the WHERE clause of a query
- For example, the following operation:

$$\sigma_{Dno=4\ \textbf{AND}\ Salary>25000}(EMPLOYEE)$$

would to the following SQL query:

```
SELECT *  FROM EMPLOYEE
WHERE Dno=4 AND Salary>25000;
```

The PROJECT Operation

- The **PROJECT(π)** operation selects certain columns from the table and discards the other columns
- Used when we are interested in only certain attributes of a relation
- The result of the PROJECT operation can be visualized as a vertical partition of the relation into two relations:

  - one has the needed columns (attributes) and contains the result of the operation
  - the other contains the discarded columns.

- The general form of the PROJECT operation is

$$\pi_{<\text{attribute list}>}(R)$$

where

π (pi) - symbol used to represent the PROJECT operation,

<attributelist> - desired sublist of attributes from the attributes of relation R.

- The result of the PROJECT operation has only the attributes specified in <attribute list> in the same order as they appear in the list
- Hence, its **degree** is equal to the number of attributes in <attribute list>

- Example : to list each employee's first and last name and salary we can use the PROJECT operation as follows:

$$\pi_{\text{Lname, Fname, Salary}}(\text{EMPLOYEE})$$

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

- The result of the PROJECT operation is a set of distinct tuples, and hence a valid relation. This is known as **duplicate elimination**
- For example, consider the following PROJECT operation:

$$\pi_{\text{gender, Salary}}(\text{EMPLOYEE})$$

gender	Salary
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000

- The number of tuples in a relation resulting from a PROJECT operation is always less than or equal to the number of tuples in *R*
- Commutativity does not hold on PROJECT

$$\pi_{<list1>}(\pi_{<list2>}(R)) = \pi_{<list1>}(R)$$

as long as <list2> contains the attributes in <list1>; otherwise, the left-hand side is an incorrect expression.

- In SQL, the PROJECT attribute list is specified in the SELECT clause of a query
- For example, the following operation:

$\pi_{gender, Salary}(EMPLOYEE)$

would correspond to the following SQL query:

SELECT DISTINCT gender, Salary

FROM EMPLOYEE

Sequences of Operations and the RENAME Operation

- For most queries, we need to apply several relational algebra operations one after the other
- Either we can write the operations as a single **relational algebra expression** by nesting the operations, or we can apply one operation at a time and create intermediate result relations
- In the latter case, we must give names to the relations that hold the intermediate results
- For example, to retrieve the first name, last name, and salary of all employees who work in department number 5, we must apply a SELECT and a PROJECT operation

- We can write a single relational algebra expression, also known as an **in-line expression**, as follows:

$$\pi_{Fname,\ Lname,\ Salary}(\sigma_{Dno=5}(EMPLOYEE))$$

Fname	Lname	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

- Alternatively, we can explicitly show the sequence of operations, giving a name to each intermediate relation, as follows:

$$DEP5\_EMPS \leftarrow \sigma_{Dno=5}(EMPLOYEE)$$

$$RESULT \leftarrow \pi_{Fname,\ Lname,\ Salary}(DEP5\_EMPS)$$

DEP5_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston,TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston,TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble,TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

RESULT

Fname	Lname	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

- We can also use this technique to **rename** the attributes in the intermediate and result relations
- To rename the attributes in a relation, we simply list the new attribute names in parentheses

$$TEMP \leftarrow \sigma_{Dno=5}(EMPLOYEE)$$

$$R(First\_name, Last\_name, Salary) \leftarrow \pi_{Fname,\ Lname,\ Salary}(TEMP)$$

TEMP

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston,TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston,TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble,TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

R

First_name	Last_name	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

- If no renaming is applied, the names of the attributes in the resulting relation of a SELECT operation are the same as those in the original relation and in the same order
- For a PROJECT operation with no renaming, the resulting relation has the same attribute names as those in the projection list and in the same order in which they appear in the list
- We can also define a formal **RENAME** operation—which can rename either the relation name or the attribute names, or both—as a unary operator.

- The general RENAME operation when applied to a relation *R* of degree *n* is denoted by any of the following three forms:

  1. $\rho_{S(B1, B2, \ldots, Bn)}(R)$ — ρ (rho) – RENAME operator
  2. $\rho S(R)$ — S – new relation name
  3. $\rho_{(B1, B2, \ldots\ldots Bn)}(R)$ — $B_1, B_2, \ldots\ldots B_n$- new attribute names

- The first expression renames both the relation and its attributes
- The second renames the relation only and
- The third renames the attributes only
- If the attributes of R are $(A_1, A_2, \ldots, A_n)$ in that order, then each A_i is renamed as B_i.

- Renaming in SQL is accomplished by aliasing using **AS**, as in the following example:

```
SELECT E.Fname AS First_name,
       E.Lname AS Last_name,
       E.Salary AS Salary
       FROM EMPLOYEE AS E
       WHERE E.Dno=5,
```

STUDENT

USN	NAME	BRANCH	SEM	PHONE
4VVCS001	ARPITHA	CSE	5	9986786543
4VVCS004	CHANDAN	CSE	3	6364571234
4VVEC010	GIRISH	ECE	1	7782456723
4VVEE021	KEERTHI	EEE	7	9886235467

Write the query in relational algebra to perform the following operations

i) Display the details of the students studying in CSE branch
ii) Display the details of the students studying in 3 semester CSE branch.
iii) Display the USN and name of all the students.
iv) Display the name and sem and students. Resulting relation name should be student_details having the column heading as Student_Name,semester.
v) Display the USN and Name of the students studying in ECE branch.
vi) Display the USN and Name of the students studying in 7 semester EEE branch

Relational Algebra Operations from Set Theory

The UNION, INTERSECTION, and MINUS Operations

- **UNION:** The result of this operation, denoted by $R \cup S$, is a relation that includes all tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.
- **INTERSECTION:** The result of this operation, denoted by $R \cap S$, is a relation that includes all tuples that are in both R and S.
- **SET DIFFERENCE (or MINUS):** The result of this operation, denoted by $R - S$, is a relation that includes all tuples that are in R but not in S.

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

STUDENT ∪ INSTRUCTOR.

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

STUDENT $\cap$ INSTRUCTOR.

Fn	Ln
Susan	Yao
Ramesh	Shah

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

STUDENT – INSTRUCTOR.

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR – STUDENT.

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

- Example: to retrieve the Social Security numbers of all employees who either work in department 5 or directly supervise an employee who works in department 5

$$DEP5\_EMPS \leftarrow \sigma_{Dno=5}(EMPLOYEE)$$

$$RESULT1 \leftarrow \pi_{Ssn}(DEP5\_EMPS)$$

$$RESULT2(Ssn) \leftarrow \pi_{Super\_ssn}(DEP5\_EMPS)$$

$$RESULT \leftarrow RESULT1 \cup RESULT2$$

DEP5_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston,TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston,TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble,TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	gender	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

Single relational algebra expression,:

$$\text{Result} \leftarrow \pi_{Ssn} (\sigma_{Dno=5} (\text{EMPLOYEE})) \cup$$

$$\pi_{Super\_ssn} (\sigma_{Dno=5} (\text{EMPLOYEE}))$$

- These are **binary** operations; that is, each is applied to two sets (of tuples)
- When these operations are adapted to relational databases, the two relations on which any of these three operations are applied must have the same **type of tuples**; this condition has been called **union compatibility** or **type compatibility.**
- Two relations $R(A_1, A_2, ..., A_n)$ and $S(B_1, B_2, ..., B_n)$ are said to be **union compatible** (or **type compatible**) if they have the same degree n and if $dom(A_i) = dom(B_i)$ for $1 \leq i \leq n$.
- This means that the two relations have the same number of attributes and each corresponding pair of attributes has the same domain.

- Both UNION and INTERSECTION are *commutative operations*; that is,

  $R \cup S = S \cup R$ and $R \cap S = S \cap R$

- Both UNION and INTERSECTION can be treated as *n*-ary operations applicable to any number of relations because both are also *associative operations;* that is,

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

- The MINUS operation is *not commutative;* that is, in general,

  $R - S \neq S - R$

- INTERSECTION can be expressed in terms of union and set difference as follows:

  $R \cap S = ((R \cup S) - (R - S)) - (S - R)$

- In SQL, there are three operations—UNION, INTERSECT, and EXCEPT—that correspond to the set operations

The CARTESIAN PRODUCT (CROSS PRODUCT) Operation

- **CARTESIAN PRODUCT** operation—also known as **CROSS PRODUCT** or **CROSS JOIN** denoted by × is a binary set operation, but the relations on which it is applied do *not* have to be union compatible.
- In general, the result of $R(A_1, A_2, ..., A_n) \times S(B_1, B_2, ..., B_m)$ is a relation Q with degree $n + m$ attributes $Q(A_1, A_2, ..., A_n, B_1, B_2, ..., B_m)$, in that order.

Q= R x S

Example:

- Consider two relations STUDENT and DETAIL

STUDENT

SNO	FNAME	LNAME
1	Albert	Singh
2	Nora	Fatehi

DETAIL

ROLLNO	AGE
5	18
9	21

- On applying CROSS PRODUCT on STUDENT and DETAIL:

STUDENT × DETAILS

SNO	FNAME	LNAME	ROLLNO	AGE
1	Albert	Singh	5	18
1	Albert	Singh	9	21
2	Nora	Fatehi	5	18
2	Nora	Fatehi	9	21

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	gender	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPENDENT

Essn	Dependent_name	gender	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

- Example: suppose that we want to retrieve a list of names of each female employee's dependents.

  1. FEMALE_EMPS ← $\sigma_{gender='F'}$(EMPLOYEE)

FEMALE_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	gen	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	1968-07-19	3321Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291Berry, Bellaire, TX	F	43000	888665555	4
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

2. EMPNAMES ← $\pi_{Fname, Lname, Ssn}$(FEMALE_EMPS)

EMPNAMES

Fname	Lname	Ssn
Alicia	Zelaya	999887777
Jennifer	Wallace	987654321
Joyce	English	453453453

3. EMP_DEPENDENTS ← EMPNAMES × DEPENDENT

EMP_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Alicia	Zelaya	999887777	333445555	Alice	F	1986-04-05	...
Alicia	Zelaya	999887777	333445555	Theodore	M	1983-10-25	...
Alicia	Zelaya	999887777	333445555	Joy	F	1958-05-03	...
Alicia	Zelaya	999887777	987654321	Abner	M	1942-02-28	...
Alicia	Zelaya	999887777	123456789	Michael	M	1988-01-04	...
Alicia	Zelaya	999887777	123456789	Alice	F	1988-12-30	...
Alicia	Zelaya	999887777	123456789	Elizabeth	F	1967-05-05	...
Jennifer	Wallace	987654321	333445555	Alice	F	1986-04-05	...
Jennifer	Wallace	987654321	333445555	Theodore	M	1983-10-25	...
Jennifer	Wallace	987654321	333445555	Joy	F	1958-05-03	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...
Jennifer	Wallace	987654321	123456789	Michael	M	1988-01-04	...
Jennifer	Wallace	987654321	123456789	Alice	F	1988-12-30	...
Jennifer	Wallace	987654321	123456789	Elizabeth	F	1967-05-05	...
Joyce	English	453453453	333445555	Alice	F	1986-04-05	...
Joyce	English	453453453	333445555	Theodore	M	1983-10-25	...
Joyce	English	453453453	333445555	Joy	F	1958-05-03	...
Joyce	English	453453453	987654321	Abner	M	1942-02-28	...
Joyce	English	453453453	123456789	Michael	M	1988-01-04	...
Joyce	English	453453453	123456789	Alice	F	1988-12-30	...
Joyce	English	453453453	123456789	Elizabeth	F	1967-05-05	...

4. ACTUAL_DEPENDENTS ← $\sigma_{Ssn=Essn}$(EMP_DEPENDENTS)

ACTUAL_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...

5. RESULT ← $\pi_{Fname, Lname, Dependent\_name}$ (ACTUAL_DEPENDENTS)

RESULT

Fname	Lname	Dependent_name
Jennifer	Wallace	Abner

- The CARTESIAN PRODUCT creates tuples with the combined attributes of two relations.
- We can SELECT related tuples only from the two relations by specifying an appropriate selection condition after the Cartesian product
- In SQL, CARTESIAN PRODUCT can be realized by using the CROSS JOIN option in joined tables

Binary Relational Operations: JOIN

The JOIN Operation

- The **JOIN** operation, denoted by $\bowtie$ is used to combine related tuples from two relations into single "longer" tuples.
- The general form of a JOIN operation on two relations $R(A_1, A_2, ..., A_n)$ and $S(B_1, B_2, ..., B_m)$ is

$$R \bowtie_{\text{<join condition>}} S$$

- Example: retrieve the name of the manager of each department
- To get the manager's name, we need to combine each department tuple with the employee tuple whose Ssn value matches the Mgr_ssn value in the department tuple

$$DEPT\_MGR \leftarrow DEPARTMENT \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE$$
$$RESULT \leftarrow \pi_{Dname,\ Lname,\ Fname}(DEPT\_MGR)$$

DEPT_MGR

Dname	Dnumber	Mgr_ssn	...	Fname	Minit	Lname	Ssn	...
Research	5	333445555	...	Franklin	T	Wong	333445555	...
Administration	4	987654321	...	Jennifer	S	Wallace	987654321	...
Headquarters	1	888665555	...	James	E	Borg	888665555	...

- The result of the JOIN is a relation Q with n + m attributes Q(A_1, A_2, ..., A_n,B_1, B_2, ..., B_m in that order;
- Q has one tuple for each combination of tuples—one from R and one from S—whenever the combination satisfies the join condition. This is the main difference between CARTESIAN PRODUCT and JOIN.
- In JOIN, only combinations of tuples satisfying the join condition appear in the result, whereas in the CARTESIAN PRODUCT all combinations of tuples are included in the result.

- A general join condition is of the form

  <condition> **AND** <condition> **AND**...**AND** <condition>

  where each <condition> is of the form A_i θ B_j, A_i is an attribute of R, B_j is an attribute of S, Ai and B_j have the same domain, and θ (theta) is one of the comparison operators {=, <, ≤, >, ≥, ≠}.

- A JOIN operation with such a general join condition is called a **THETA JOIN**
- Tuples whose join attributes are NULL or for which the join condition is FALSE *do not* appear in the result.

Variations of JOIN: The EQUIJOIN and NATURAL JOIN

- The most common use of JOIN involves join conditions with equality comparisons only. Such a JOIN, where the only comparison operator used is =, is called an **EQUIJOIN**
- In the result of an EQUIJOIN we always have one or more pairs of attributes that have identical values in every tuple.
- For example the values of the attributes Mgr_ssn and Ssn are identical in every tuple of DEPT_MGR (the EQUIJOIN result) because the equality join condition specified on these two attributes requires the values to be identical in every tuple in the result.

- The standard definition of NATURAL JOIN(*) requires that the two join attributes (or each pair of join attributes) have the same name in both relations. If this is not the case, a renaming operation is applied first.
- Suppose we want to combine each PROJECT tuple with the DEPARTMENT tuple that controls the project.
- first we rename the Dnumber attribute of DEPARTMENT to Dnum—so that it has the same name as the Dnum attribute in PROJECT—and then we apply NATURAL JOIN:

$$PROJ\_DEPT \leftarrow PROJECT * \rho_{(Dname,\ Dnum,\ Mgr\_ssn,\ Mgr\_start\_date)}(DEPARTMENT)$$

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

- The same query can be done in two steps by creating an intermediate table DEPT as follows:

  DEPT ← $\rho_{(Dname,\ Dnum,\ Mgr\_ssn,Mgr\_start\_date)}$(DEPARTMENT)

  PROJ_DEPT ← PROJECT * DEPT

- The attribute Dnum is called the **join attribute** for the NATURAL JOIN operation, because it is the only attribute with the same name in both relations.

PROJ_DEPT

Pname	Pnumber	Plocation	Dnum	Dname	Mgr_ssn	Mgr_start_date
ProductX	1	Bellaire	5	Research	333445555	1988-05-22
ProductY	2	Sugarland	5	Research	333445555	1988-05-22
ProductZ	3	Houston	5	Research	333445555	1988-05-22
Computerization	10	Stafford	4	Administration	987654321	1995-01-01
Reorganization	20	Houston	1	Headquarters	888665555	1981-06-19
Newbenefits	30	Stafford	4	Administration	987654321	1995-01-01

- If the attributes on which the natural join is specified already have the same names in both relations, renaming is unnecessary.
- For example, to apply a natural join on the Dnumber attributes of DEPARTMENT and DEPT_LOCATIONS, it is sufficient to write

  DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS

DEPT_LOCS

Dname	Dnumber	Mgr_ssn	Mgr_start_date	Location
Headquarters	1	888665555	1981-06-19	Houston
Administration	4	987654321	1995-01-01	Stafford
Research	5	333445555	1988-05-22	Bellaire
Research	5	333445555	1988-05-22	Sugarland
Research	5	333445555	1988-05-22	Houston

- A single JOIN operation is used to combine data from two relations so that related information can be presented in a single table. These operations are also known as **inner joins**.
- Informally, an inner join is a type of match and combine operation defined formally as a combination of CARTESIAN PRODUCT and SELECTION.
- The NATURAL JOIN or EQUIJOIN operation can also be specified among multiple tables, leading to an n-way join.

- For example, consider the following three-way join:

$$((\text{PROJECT} \bowtie_{\text{Dnum=Dnumber}} \text{DEPARTMENT}) \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE})$$

- This combines each project tuple with its controlling department tuple into a single tuple, and then combines that tuple with an employee tuple that is the department manager.
- The net result is a consolidated relation in which each tuple contains this project-department-manager combined information.

- In SQL, JOIN can be realized in several different ways.
  - The first method is to specify the <join conditions> in the WHERE clause, along with any other selection conditions.

    Ex: select fname,lname,address from employee,department where dname='Research' and Dnumber=Dno;
  - The second way is to use a nested relation

    Ex: select fname,lname,address from employee where Dno IN (select Dnumber from department where Dname='Research');
  - Another way is to use the concept of joined tables

    Ex: select fname,lname,address from employee join department on Dno=Dnumber where Dname='Research' ;

Table 6.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{(\text{<join attributes 1>}), (\text{<join attributes 2>})} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{(\text{<join attributes 1>}), (\text{<join attributes 2>})} R_2$ OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Self-Learning Topic

Examples of Queries in Relational Algebra

Consider the two tables T1 and T2.

T1

P	Q	R
10	a	5
15	b	8
25	a	6

T2

A	B	C
10	b	6
25	C	3
10	b	5

Construct the results for the following operations

i) $T1 \bowtie_{T1.P=T2.A} T2$

ii) $T1 \bowtie_{T1.Q=T2.B} T2$

iii) $T1 \cup T2$

iv) $T1 \cap T2$

v) $T1 - T2$

vi) $T1 \bowtie_{T1.P=T2.A \text{ AND } T1.R=T2.C} T2$

vii) $T1 \times T2$

viii) $\pi_{P, Q} (\sigma_{Q=a} (T1))$