CST 204 Database Management Systems

B.Tech. CSE

Semester IV

Viswajyothi College of Engineering and Technology

MODULE 2

Relational Model

Syllabus of Module 2

- Structure of Relational Databases - Integrity Constraints, Synthesizing ER diagram to relational schema
- Introduction to Relational Algebra - select, project, cartesian product operations, join - Equi-join, natural join. query examples
- Introduction to Structured Query Language (SQL), Data Definition Language (DDL), Table definitions and operations – CREATE, DROP, ALTER, INSERT, DELETE, UPDATE.

Introduction to Relational Algebra

Relational algebra: Relational algebra is a **procedural query language**, which takes instances of relations as input and yields instances of relations as output.

- It uses operators to perform queries. An operator can be either **unary or binary.**
- They accept relations as their input and the ouput is also realtion(a temporary table holding the data asked for by the user)

Relational Calculus : Relational calculus is a **non-procedural query language**, and instead of algebra, it uses mathematical predicate calculus.

Relational Algebra Operations

Unary Relational Operations

- SELECT (symbol: σ)
- PROJECT (symbol: π)
- RENAME (symbol: ρ)

Relational Algebra Operations From Set Theory

- UNION (∪)
- INTERSECTION (∩)
- DIFFERENCE (-)
- CARTESIAN PRODUCT (x)

Binary Relational Operations

- JOIN (⋈)
- DIVISION (/)

1. Unary Relational Operations:

- SELECT
- PROJECT
- RENAME

SELECT operation (σ)

- The SELECT operation is used to choose a subset of the tuples from a relation that satisfies a **selection condition**
- SELECT operation restricts the tuples in a relation to only those tuples that satisfy the condition.
- Example : Select the EMPLOYEE tuples whose department is 4, or those whose salary is greater than $30,000

$$\sigma_{Dno=4}(EMPLOYEE)$$
$$\sigma_{Salary>30000}(EMPLOYEE)$$

- In general, the SELECT operation is denoted by

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

where R is the table name

- 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>

- Degree of the relation from SELECT operation is its number of attributes(columns). It is the same as the degree of *R*.
- SELECT operation is **commutative**

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

- Clauses can be connected by the standard Boolean operators **AND**, **OR**, and **NOT** to form a general selection condition
- **Example:** 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.

- Employee table

EMPLOYEE

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
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

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

- Output

(a)

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
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

PROJECT operation (Π)

- PROJECT operation selects certain columns from the table and discards the other columns.
- If we are interested in only certain attributes(columns) of a relation, we use the PROJECT operation to project the relation over these attributes only.
- The general form of PROJECT

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

- Example: list each employee's first and last name and salary

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

- 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>.
- The PROJECT operation *removes any duplicate tuples, so the result of the PROJECT operation is* a set of distinct tuples, and hence a valid relation.
- This is known as **duplicate elimination.**

- Example

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

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

EMPLOYEE

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
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

Sequences of Operations

- 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
- We must give names to the relations that hold the intermediate results.
- Example : Retrieve the first name, last name, and salary of all employees who work in department number 5. Here we must apply a SELECT and a PROJECT operation.

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

- Output

(a)

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

- Example- Intermediate Relation for the above expression

$$DEP5\_EMPS \leftarrow \sigma_{Dno=5}(EMPLOYEE)$$
$$RESULT \leftarrow \pi_{Fname,\ Lname,\ Salary}(DEP5\_EMPS)$$

RENAME operation : (ρ)

- The RENAME operation is used to rename a relation name or attribute name or both.
- It is denoted by any of the following methods

1. Rename relation
   - It renames relation R to S

$$\rho_S(R)$$

2. Rename attributes
   - It renames attributes of relation R to (B1,B2,..Bn)

$$\rho_{(B1,B2,...Bn)}(R)$$

3. Rename relation and its attributes
   - It renames relation R to S and attributes of relation R to (B1,B2,..Bn)

$$\rho_{S(B1,B2,...,Bn)}(R)$$

2. Relational Algebra Operations From Set Theory

- UNION (∪)
- INTERSECTION (∩),
- DIFFERENCE (-)
- CARTESIAN PRODUCT (x)

The UNION, INTERSECTION, and MINUS Operations

UNION:

- The result of this operation, denoted by R ∪ 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 ∩ 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.

Note: For performing all the above operation both the tables should have the same attributes.

Example:

Retrieve the Social Security numbers of all employees who either work in department 5 or directly supervise an employee who works in department 5, we can use the UNION operation as follows.

$$\text{DEP5\_EMPS} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})$$
$$\text{RESULT1} \leftarrow \pi_{\text{Ssn}}(\text{DEP5\_EMPS})$$
$$\text{RESULT2(Ssn)} \leftarrow \pi_{\text{Super\_ssn}}(\text{DEP5\_EMPS})$$
$$\text{RESULT} \leftarrow \text{RESULT1} \cup \text{RESULT2}$$

- UNION operation produces the tuples that are in either RESULT1 or RESULT2 or both, while eliminating any duplicates.

(b)

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

$$DEP5\_EMPS \leftarrow \sigma_{Dno=5}(EMPLOYEE)$$
$$RESULT1 \leftarrow \pi_{Ssn}(DEP5\_EMPS)$$
$$RESULT2(Ssn) \leftarrow \pi_{Super\_ssn}(DEP5\_EMPS)$$
$$RESULT \leftarrow RESULT1 \cup RESULT2$$

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Union compatibility or Type compatibility

- Two relations R(A1, A2, …, An) and S(B1, B2, …, Bn) are said to be union compatible (or type compatible)
  - if they have the same degree n and if dom(Ai) = dom(Bi) for 1 <=i<=n.
  - This means that the two relations have the same number of attributes and each corresponding pair of attributes has the same domain.

- Two union-compatible relations.

(a) 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

(b)

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

- STUDENT ∩ INSTRUCTOR

(a) 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

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

- STUDENT – INSTRUCTOR

(a) **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

(d)

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

- INSTRUCTOR – STUDENT

(a) **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

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

The UNION, INTERSECTION, and MINUS Properties

- UNION and INTERSECTION are commutative operations

$$R \cup S = S \cup R \quad \text{and} \quad R \cap S = S \cap R$$

- Both UNION and INTERSECTION can be treated as n-ary operations applicable to any number of relations because both are also associative operations.

$$R \cup (S \cup T) = (R \cup S) \cup T \quad \text{and} \quad (R \cap S) \cap T = R \cap (S \cap T)$$

- 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

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

CARTESIAN PRODUCT OR CROSS JOIN (×)

- This is also a binary set operation, but the relations on which it is applied **do not have to be** union compatible.
- The result of R(A1, A2, …, An) × S(B1, B2, …, Bm) is a relation Q with degree n + m attributes Q(A1, A2,…, An, B1, B2, …, Bm), in that order.
- The resulting relation Q has one tuple for each combination of tuples—one from R and one from S.
- Hence, if R has nR tuple, and S has nS tuples, then R × S will have nR * nS tuples.

n-ary CARTESIAN PRODUCT

- It produces new tuples by concatenating all possible combinations of tuples from n underlying relations.

- **Example:** Loan x Borrower

Loan

Branch_name	Loan_number	Amount
SBI Db Road	L-15	1500
SBI City	L-16	2500
SBI Db Road	L-17	2000
SBI City	L-18	1900

Borrower

Customer_name	Loan_number
amit	L-15
sumit	L-16
raj	L-17
rajesh	L-19

Result:

```
mysql> select * from Loan,Borrower;
```

Branch_name	Loan_number	Amount	Customer_name	Loan_number
SBI Db Road	L-15	1500	amit	L-15
SBI City	L-16	2500	amit	L-15
SBI Db Road	L-17	2000	amit	L-15
SBI City	L-18	1900	amit	L-15
SBI Db Road	L-15	1500	sumit	L-16
SBI City	L-16	2500	sumit	L-16
SBI Db Road	L-17	2000	sumit	L-16
SBI City	L-18	1900	sumit	L-16
SBI Db Road	L-15	1500	raj	L-17
SBI City	L-16	2500	raj	L-17
SBI Db Road	L-17	2000	raj	L-17
SBI City	L-18	1900	raj	L-17
SBI Db Road	L-15	1500	rajesh	L-19
SBI City	L-16	2500	rajesh	L-19
SBI Db Road	L-17	2000	rajesh	L-19
SBI City	L-18	1900	rajesh	L-19

Q. Write a relational algebra query and from above the relation Loan and Borrower, select all the customers whose Branch_name is “SBI City”.

Relational algebra Query: $\sigma_{\text{Branch\_name}=\text{“SBI City”}}(\text{Loan x Borrower})$

Result:

```
mysql> select * from Loan, Borrower where Branch_name='SBI City';
+-------------+-------------+--------+---------------+-------------+
| Branch_name | Loan_number | Amount | Customer_name | Loan_number |
+-------------+-------------+--------+---------------+-------------+
| SBI City    | L-16        |   2500 | amit          | L-15        |
| SBI City    | L-18        |   1900 | amit          | L-15        |
| SBI City    | L-16        |   2500 | sumit         | L-16        |
| SBI City    | L-18        |   1900 | sumit         | L-16        |
| SBI City    | L-16        |   2500 | raj           | L-17        |
| SBI City    | L-18        |   1900 | raj           | L-17        |
| SBI City    | L-16        |   2500 | rajesh        | L-19        |
| SBI City    | L-18        |   1900 | rajesh        | L-19        |
+-------------+-------------+--------+---------------+-------------+
```

Q. Write a relational algebra query and from above the relation Loan and Borrower. Display Branch_name, Customer_name, Amount of the customers whose Branch_name is “SBI City”.

Relational algebra query:

$\Pi_{\text{Branch\_name, Customer\_name, Amount}}(\sigma_{\text{Branch\_name=”SBI City”}}(\text{Loan x Borrower}))$

Result:

```
mysql> select Branch_name,Customer_name,Amount from Loan, Borrower where Branch_name='SBI City';
+-------------+---------------+--------+
| Branch_name | Customer_name | Amount |
+-------------+---------------+--------+
| SBI City    | amit          |   2500 |
| SBI City    | amit          |   1900 |
| SBI City    | sumit         |   2500 |
| SBI City    | sumit         |   1900 |
| SBI City    | raj           |   2500 |
| SBI City    | raj           |   1900 |
| SBI City    | rajesh        |   2500 |
| SBI City    | rajesh        |   1900 |
+-------------+---------------+--------+
```

Q. Write a relational algebra query and sql from above the relation Loan and Borrower. Display Customer_name, Amount where Loan_number is equal in both relation.

Relational algebra query:

$$\Pi_{\text{Customer\_name, Amount}} (\sigma_{\text{Loan.Loan\_number} = \text{Borrower.Loan\_number}} (\text{Loan x Borrower}))$$

```
mysql> select Customer_name,Amount from Borrower,Loan where Borrower.Loan_number=Loan.Loan_number;
+---------------+--------+
| Customer_name | Amount |
+---------------+--------+
| amit          |   1500 |
| sumit         |   2500 |
| raj           |   2000 |
+---------------+--------+
```

Binary Relational Operations

- JOIN (⋈)
- DIVISION (/)

The JOIN Operation (⋈)

- The JOIN operation, denoted by ⋈ , is used to combine related tuples from two relations into single "longer" tuples.
- This operation is very important for any relational database with more than a single relation because it allows us to process relationships among relations.

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.
- We do this by using the JOIN operation and then projecting the result over the necessary attributes, as follows.

EMPLOYEE

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
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

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

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

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	...

Figure 6.6
Result of the JOIN operation $DEPT\_MGR \leftarrow DEPARTMENT \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE$.

RESULT

Dname	Lname	Fname
Research	Wong	Franklin
Administration	Wallance	Jennifer
Headquaters	Borg	James

- Thus the two operations

  1. RESULT1 ← DEPARTMENT x EMPLOYEE

  2. RESULT2 ← $\sigma_{Ssn=Mgr\_ssn}(RESULT1)$

can be replaced with a single JOIN operation as follows:

- RESULT ← $DEPARTMENT \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE$

- The **JOIN** operation can be specified as a **combination of CARTESIAN PRODUCT** operation followed by a **SELECT** operation as shown above.

Difference between JOIN and CARTESIAN PRODUCT

- 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.
- The join condition is specified on attributes from the two relations R and S and is evaluated for each combination of tuples.
- Each tuple combination for which the join condition evaluates to TRUE is included in the resulting relation Q as a single combined tuple.

THETA JOIN (θ)

- Theta join is a join which combines the tuples from different relations according to the given theta condition.
- The join condition in theta join is denoted by **theta(θ) symbol.** θ (theta) is one of the comparison operators $\{=, <, \leq, >, \geq, \neq\}$.
- **Notation:** $R1 \bowtie_{\theta} R2$
- Where R1 and R2 are relations such that they don't have any common attribute.

Example : Theta Join (θ)

- We have two tables: Student(S_id,Name,Std,Age) and Courses(Class,C_name).

$$\text{Student} \bowtie_{\text{Student.Std}=\text{Course.Class}} \text{Course}$$

- The above join operations check if the 'Std' attribute in Student is equal to the values of the 'Class' attribute of the Course table. If these values are equal then it is included in the resulting table.

Student

S_id	Name	Std	Age
1	Andrew	5	25
2	Angel	10	30
3	Anamika	8	35

Course

Class	C_name
10	Foundation C
5	C++

Student $\bowtie_{\theta}$ Course

S_id	Name	Std	Age	Class	C_name
1	Andrew	5	25	5	C++
2	Angel	10	30	10	Foundation C

EQUIJOIN

- 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.
- The previous example which we gave in the theta join is also an example of equi-join.

NATURAL JOIN

- NATURAL JOIN requires that the two join attributes (or each pair of join attributes) have the same name in both relations.
- NATURAL JOIN, denoted by *, is used to eliminate the duplicate attributes in an EQUIJOIN condition.
- 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, rename operation has to be applied first to convert the name of the join attribute of one relation same as that of the join attribute in second relation.
- ***Notation: R1* R2*** where R1 and R2 are two relations.
- Example: We have two tables of Student (S_id, Name, Class, Age, C_id) and Courses(C_id,C_name). Now, we will perform natutral join on both the tables i.e Student * Course

Student * Course

Student

S_id	Name	Class	Age	C_id
1	Andrew	5	25	11
2	Angel	10	30	11
3	Anamika	8	35	22

Course

C_id	C_name
11	Foundation C
21	C++

Student ⋈ Course

S_id	Name	Class	Age	C_id	C_name
1	Andrew	5	25	11	Foundation C
2	Angel	10	30	11	Foundation C

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 $R_1 \bowtie_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$
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 $R_1 *_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$ 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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Different Types of SQL JOINs

- INNER JOIN
  - Returns records that have matching values in both tables

  1. Theta join
  2. EQUI join
  3. Natural join

- OUTER JOIN
  - In an outer join, along with tuples that satisfy the matching criteria, we also include some or all tuples that do not match the criteria.

  1. Left Outer JOIN
  2. Right Outer Join
  3. Full Outer Join

Left Outer Join ($A \mathbin{⟕} B$)

- In the left outer join, operation allows keeping all tuple in the left relation.
- if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.

- We have two tables: Student(S_id, Name, Class, Age, C_type) and Courses(C_type, C_name).

Student⋈Course

Student

S_id	Name	Class	Age	C_type
1	Andrew	5	25	A
2	Angel	10	30	A
3	Anamika	8	35	C

Course

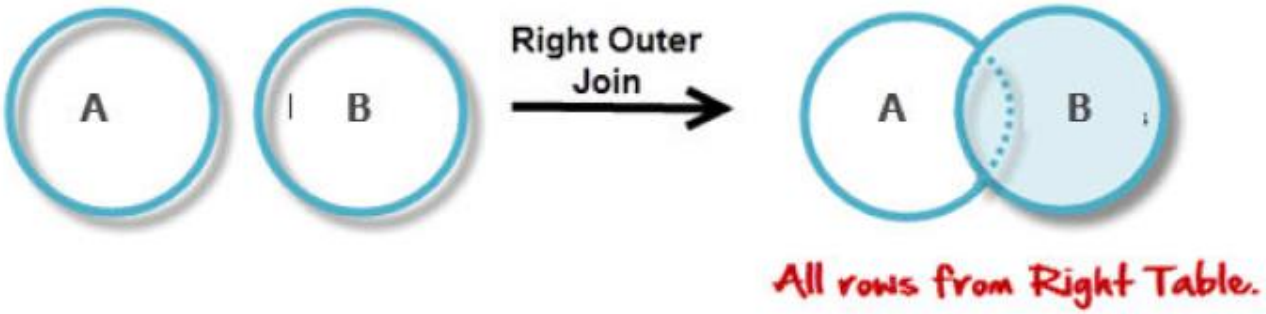
C_type	C_name
A	Foundation C
B	C++

Student ⋈ Course

S_id	Name	Class	Age	C_type	C_name
1	Andrew	5	25	A	Foundation C
2	Angel	10	30	A	Foundation C
3	Anamika	8	35	C	-

Right Outer Join ($A \bowtie B$)

- In the right outer join, operation allows keeping all tuple in the right relation.
- However, if there is no matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values.



- We have two tables: Student(S_id, Name, Class, Age, C_type) and Courses(C_type, C_name).

  Student⋈Course

Student

S_id	Name	Class	Age	C_type
1	Andrew	5	25	A
2	Angel	10	30	A
3	Anamika	8	35	C

Course

C_type	C_name
A	Foundation C
B	C++

Student ⋈ Course

S_id	Name	Class	Age	C_type	C_name
1	Andrew	5	25	A	Foundation C
2	Angel	10	30	A	Foundation C
-	-	-	-	B	C++

Full Outer Join (⟗)

- Full Outer Join is a type of join in which all the tuples from the left and right relation which are having the same value on the common attribute. Also, they will have all the remaining tuples which are not common on in both the relations.
- ***Notation: R1 ⟗ R2*** where R1 and R2 are relations.

- We have two tables: Student(S_id, Name, Class, Age, C_type) and Courses(C_type, C_name).

  Student ⋈ Course

Student

S_id	Name	Class	Age	C_type
1	Andrew	5	25	A
2	Angel	10	30	A
3	Anamika	8	35	C

Course

C_type	C_name
A	Foundation C
B	C++

Student ⋈ Course

S_id	Name	Class	Age	C_type	C_name
1	Andrew	5	25	A	Foundation C
2	Angel	10	30	A	Foundation C
3	Anamika	8	35	C	-
-	-	-	-	B	C++

SEMI JOIN

- Semi-Join matches the rows of two relations and then show the matching rows of the relation whose name is mentioned to the left side of ⋉ Semi Join operator.

Semi join

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Production

Dept

DeptName	Manager
Sales	Bob
Sales	Thomas
Production	Katie
Production	Mark

Employee ⋉ Dept

Name	EmpId	DeptName
Sally	2241	Sales
Harriet	2202	Production

ANTI JOIN (▷)

- **Anti-join** between two tables returns rows from the first table where no matches are found in the second table. It is opposite of a **semi-join**. An **anti-join** returns one copy of each row in the first table for which no match is found.

Anti join

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Production

Dept

DeptName	Manager
Sales	Sally
Production	Harriet

Employee ▷ Dept

Name	EmpId	DeptName
Harry	3415	Finance
George	3401	Finance

Division Operator (÷): Division operator A÷B can be applied if and only if:

- Attributes of B is proper subset of Attributes of A.
- The relation returned by division operator will have attributes = (All attributes of A – All Attributes of B)
- The relation returned by division operator will return those tuples from relation A which are associated to every B's tuple.

DIVISION Operation

- **Example1:** Retrieve the names of employees who work on all the projects that 'John Smith' works on.
  - query using the DIVISION operation, proceed as follows.
  - First, retrieve the list of project numbers that 'John Smith' works on in the intermediate relation SMITH_PNOS:

$$SMITH \leftarrow \sigma_{Fname='John' \text{ AND } Lname='Smith'}(EMPLOYEE)$$
$$SMITH\_PNOS \leftarrow \pi_{Pno}(WORKS\_ON \bowtie_{Essn=Ssn} SMITH)$$

  - Next, create a relation that includes a tuple <Pno, Essn> whenever the employee whose Ssn is Essn works on the project whose number is Pno in the intermediate relation SSN_PNOS:

$$SSN\_PNOS \leftarrow \pi_{Essn,\ Pno}(WORKS\_ON)$$

Figure 3.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

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
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

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

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

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

DEPENDENT

Essn	Dependent_name	Sex	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

EMPLOYEE

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
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

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

- Finally apply DIVISION operation to the two relations, which gives the required employees SSN(Social Security Number)

(a)

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30

SMITH_PNOS

Pno
1
2

SSNS

Ssn
123456789
453453453

- Then to get the result, $RESULT \leftarrow \pi_{Fname,\ Lname}(SSNS * EMPLOYEE)$

- **Example :** (b) $T \leftarrow R \div S$

(b)

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

T

B
b1
b4

- **Query 1.** Retrieve the name and address of all employees who work for the 'Research' department.

$$RESEARCH\_DEPT \leftarrow \sigma_{Dname='Research'}(DEPARTMENT)$$
$$RESEARCH\_EMPS \leftarrow (RESEARCH\_DEPT \bowtie_{Dnumber=Dno} EMPLOYEE)$$
$$RESULT \leftarrow \pi_{Fname,\ Lname,\ Address}(RESEARCH\_EMPS)$$

As a single in-line expression, this query becomes:

$$\pi_{Fname,\ Lname,\ Address}\ (\sigma_{Dname='Research'}(DEPARTMENT \bowtie_{Dnumber=Dno}(EMPLOYEE))$$

- This query could be specified in other ways; for example, the order of the JOIN and
- SELECT operations could be reversed, or the JOIN could be replaced by a NATURAL JOIN after renaming one of the join attributes to match the other join attribute name.

- **Query 2.** For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

$$\text{STAFFORD\_PROJS} \leftarrow \sigma_{\text{Plocation='Stafford'}}(\text{PROJECT})$$
$$\text{CONTR\_DEPTS} \leftarrow (\text{STAFFORD\_PROJS} \bowtie_{\text{Dnum=Dnumber}} \text{DEPARTMENT})$$
$$\text{PROJ\_DEPT\_MGRS} \leftarrow (\text{CONTR\_DEPTS} \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE})$$
$$\text{RESULT} \leftarrow \pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(\text{PROJ\_DEPT\_MGRS})$$

- We first select the projects located in Stafford, then join them with their controlling departments, and then join the result with the department managers.
- Finally, we apply a project operation on the desired attributes

- **Query 3.** Make a list of project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

$$\text{SMITHS(Essn)} \leftarrow \pi_{\text{Ssn}}\,(\sigma_{\text{Lname='Smith'}}(\text{EMPLOYEE}))$$
$$\text{SMITH\_WORKER\_PROJS} \leftarrow \pi_{\text{Pno}}(\text{WORKS\_ON} * \text{SMITHS})$$
$$\text{MGRS} \leftarrow \pi_{\text{Lname, Dnumber}}(\text{EMPLOYEE} \bowtie_{\text{Ssn=Mgr\_ssn}} \text{DEPARTMENT})$$
$$\text{SMITH\_MANAGED\_DEPTS(Dnum)} \leftarrow \pi_{\text{Dnumber}}\,(\sigma_{\text{Lname='Smith'}}(\text{MGRS}))$$
$$\text{SMITH\_MGR\_PROJS(Pno)} \leftarrow \pi_{\text{Pnumber}}(\text{SMITH\_MANAGED\_DEPTS} * \text{PROJECT})$$
$$\text{RESULT} \leftarrow (\text{SMITH\_WORKER\_PROJS} \cup \text{SMITH\_MGR\_PROJS})$$

Queries

Assume the following relations:

BOOKS(DocId, Title, Publisher, Year)

STUDENTS(StId, StName, Major, Age)

AUTHORS(AName, Address)

borrows(DocId, StId, Date)

has-written(DocId, AName)

describes(DocId, Keyword)

1. List the year and title of each book.
2. List all information about students whose major is CS
3. List all students with the books they can borrow
4. List all books published by McGraw-Hill before 1990
5. List the name of students who are older than 30 and who are not studying CS
6. .Rename ANamein the relation AUTHORS to Name.

Answers

List the year and title of each book.

$\pi_{Year, Title}(BOOKS)$

List all information about students whose major is CS

$\sigma_{Major = 'CS'}(STUDENTS)$

List all students with the books they can borrow

$STUDENTS \times BOOKS$

List all books published by McGraw-Hill before 1990

$\sigma_{Publisher = \text{'McGraw-Hill'} \textbf{ AND } Year<1990}(BOOKS)$

List the name of students who are older than 30 and who are not studying CS

$\pi_{StName}(\sigma_{Age>30}(STUDENTS) - \pi_{StName}(\sigma_{Major='CS'}(STUDENTS))$

Rename Aname in the relation AUTHORS to Name

$\rho_{(Name, Address)}(AUTHORS)$