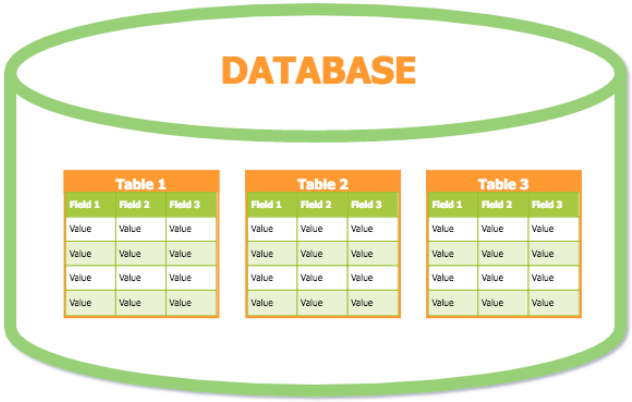
ISAD253SL - Databases

Lesson 7
Relational Algebra



Dileeka Alwis
Lecturer, School of Computing, NSBM

Relational Algebra (Relational Operations)

- A procedural query language, which takes instances of relations as input and yields instances of relations as output.
- A collection of operations that are used to manipulate queries from relations.
  - Restriction
  - Projection
  - Union
  - Intersection
  - Difference
  - Product
  - Join
  - Division

Restriction

- Returns a subset of rows in a table that satisfy a particular condition.

<u>Syntax</u>

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

Example

PNO	PNAME	COLOUR	WEIGHT	PRICE
P1	Nut	Red	12	.07
P2	Bolt	Green	17	.12
P3	Screw	Blue	17	.05
P4	Screw	Red	14	.02
P5	Cam	Blue	12	1.19
P6	Cog	Red	19	2.03

$$\sigma_{\texttt{COLOUR='Red'}}(\texttt{Product})$$

SQL Restriction

$\sigma_{\texttt{COLOUR='Red'}}(\texttt{Product})$

```
SELECT *
FROM Product
WHERE colour = 'Red'
```

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

Display the details of employees working in department 5.

$$\sigma_{\texttt{Dno=5}}\ (\texttt{Employee})$$

```
SELECT *
FROM Employee
WHERE Dno = 5
```

Exercises

$\sigma_{Sex=M}$ (Employee)

$\sigma_{(Salary>30000 \text{ and } Sex=F)}$ (Employee)

$\sigma_{(dno=4 \text{ and } salary>25000) \text{ or } (dno=5 \text{ and } salary>30000)}$ (Employee)

Projection

- Select certain fields from the table and discard the other fields.

<u>Syntax</u>

$\Pi_{\text{<attribute list>}}$ (<relation name>)

Example

PNO	PNAME	COLOUR	WEIGHT	PRICE
P1	Nut	Red	12	.07
P2	Bolt	Green	17	.12
P3	Screw	Blue	17	.05
P4	Screw	Red	14	.02
P5	Cam	Blue	12	1.19
P6	Cog	Red	19	2.03

$$\Pi_{\text{PNAME, PRICE}}(\text{Product})$$

SQL Projection

$$\Pi_{\text{PNAME, PRICE}}(\text{Product})$$

```
SELECT pname, price
FROM Product
```

Example

PNO	PNAME	COLOUR	WEIGHT	PRICE
P1	Nut	Red	12	.07
P2	Bolt	Green	17	.12
P3	Screw	Blue	17	.05
P4	Screw	Red	14	.02
P5	Cam	Blue	12	1.19
P6	Cog	Red	19	2.03

- List names of all red colour products

SQL

$$\Pi_{\texttt{PNAME}}(\sigma_{\texttt{COLOUR='Red'}}(\texttt{Product}))$$

```
SELECT pname
FROM Product
WHERE colour = 'Red'
```

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

Display ssn & name of employees working in department 5.

$$\Pi_{\texttt{ssn,Fname,Minit,Lname}}(\sigma_{\texttt{Dno=5}}(\texttt{Employee}))$$

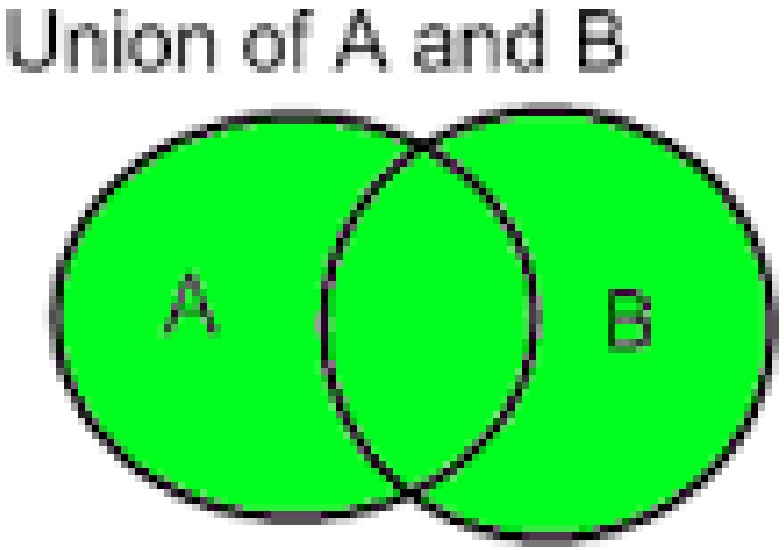
```
SELECT ssn, Fname, Minit, Lname
FROM Employee
WHERE Dno = 5
```

UNION

- The result of this operation is a relation that includes **all** tuples that are either in A or in B or in both.

- Duplicate tuples are eliminated.

$$A \cup B$$

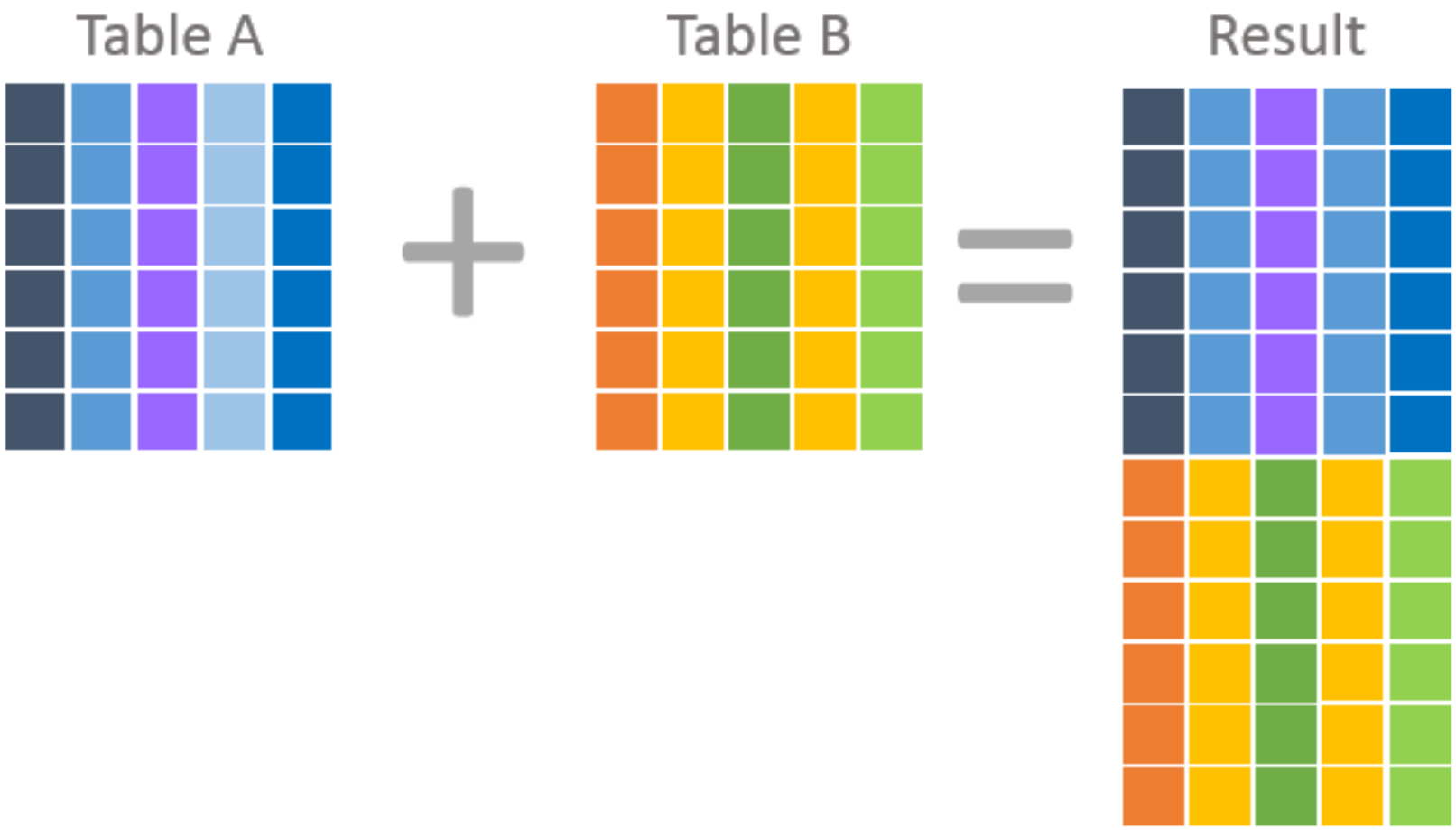


UNION

- Used to combine data from one or more tables into new rows.
  - Columns aren’t combined to create results, rows are combined.
  - The rows are in the same result.
  - Data from the first table is in one set of rows, and the data from the second table in another set.

- Typically used where you have two results whose rows you want to include in the same result set.

UNION



UNION

- Requirements to apply UNION:
  - The number of columns must be the same for both select statements.
  - The columns, in order, must be of the same data type.

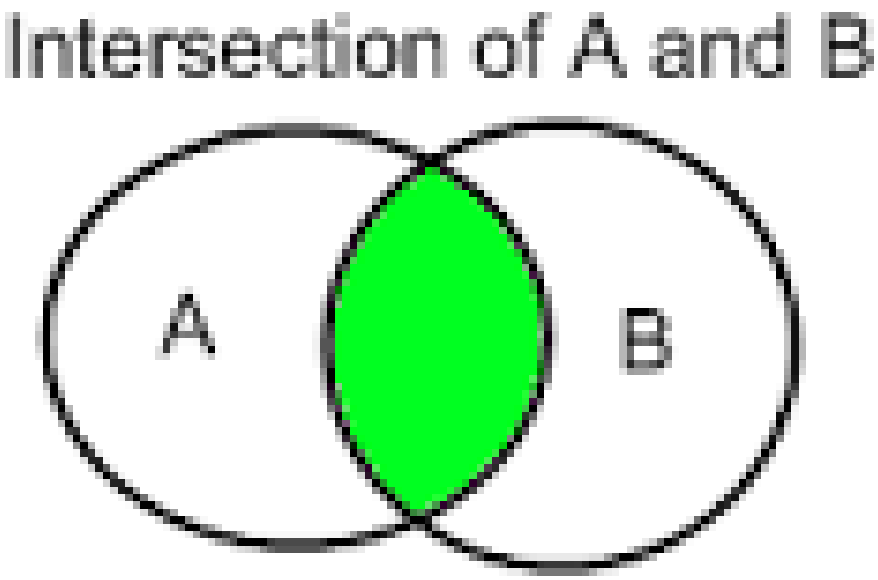
- When rows are combined duplicate rows are eliminated.
  - use the **ALL** keyword to keep all rows from both select statement's results.

Intersection

- The result of this operation is a relation that includes all tuples that are in **both** A and B.

- Duplicate tuples are eliminated.

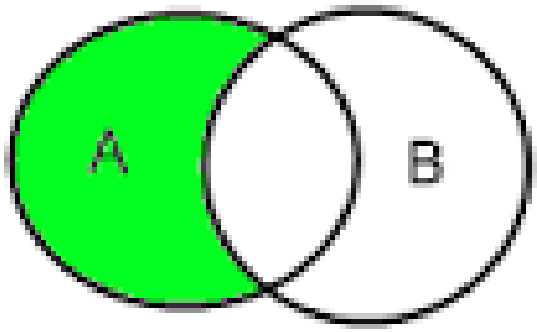
$$A \cap B$$



Difference

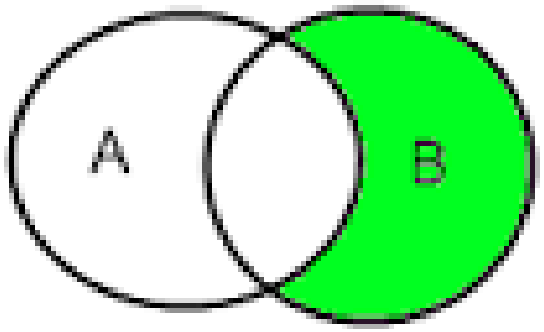
- The result of this operation is a relation that includes all tuples that are in R but not in S (R –S).

A - B



Difference A minus B

B - A



Difference B minus A

Example

A =

SNo	FName
S1	Susan
S2	Ramesh
S3	Johny
S4	Jimmy

B =

ENo	FName
E1	John
S1	Susan
E2	Francis
S4	Jimmy

Union ($A \cup B$) - S1, S2, S3, S4, E1, E2

Intersection ($A \cap B$) - S1, S4

Difference (A-B) - S2, S3

Difference (B-A) - E1, E2

SQL

- SELECT * FROM A
  UNION
  SELECT * FROM B

- SELECT * FROM A
  MINUS
  SELECT * FROM B

- SELECT * FROM A
  INTERSECT
  SELECT * FROM B

- SELECT * FROM B
  MINUS
  SELECT * FROM A

Exercise

EmpID	Name	Date Joined	Salary	DeptNo	Designation
E1	John	10/10/2013	75,000	D1	Executive
E2	Peter	01/04/2014	90,000	D2	Manager
E3	Ann	15/06/2014	85,000	D1	Manager
E4	Tom	28/01/2015	38,000	D3	Cashier
E5	Jimmy	06/12/2016	60,000	D1	Executive

Exercise

- Query A: Retrieve EmpID and Name of all Employees who are working in Department 1 from Employee table
- Query B: Retrieve EmpID and Name of all Managers from Employee table
- Find the following:

  $A \cup B$

  $A \cap B$

  $A - B$

  $B - A$

Join

- Combines rows from two or more tables when common field(s) satisfies a condition.

$$P \bowtie_{\theta} SP$$

equivalent to $\sigma_{\theta}(P \times SP)$

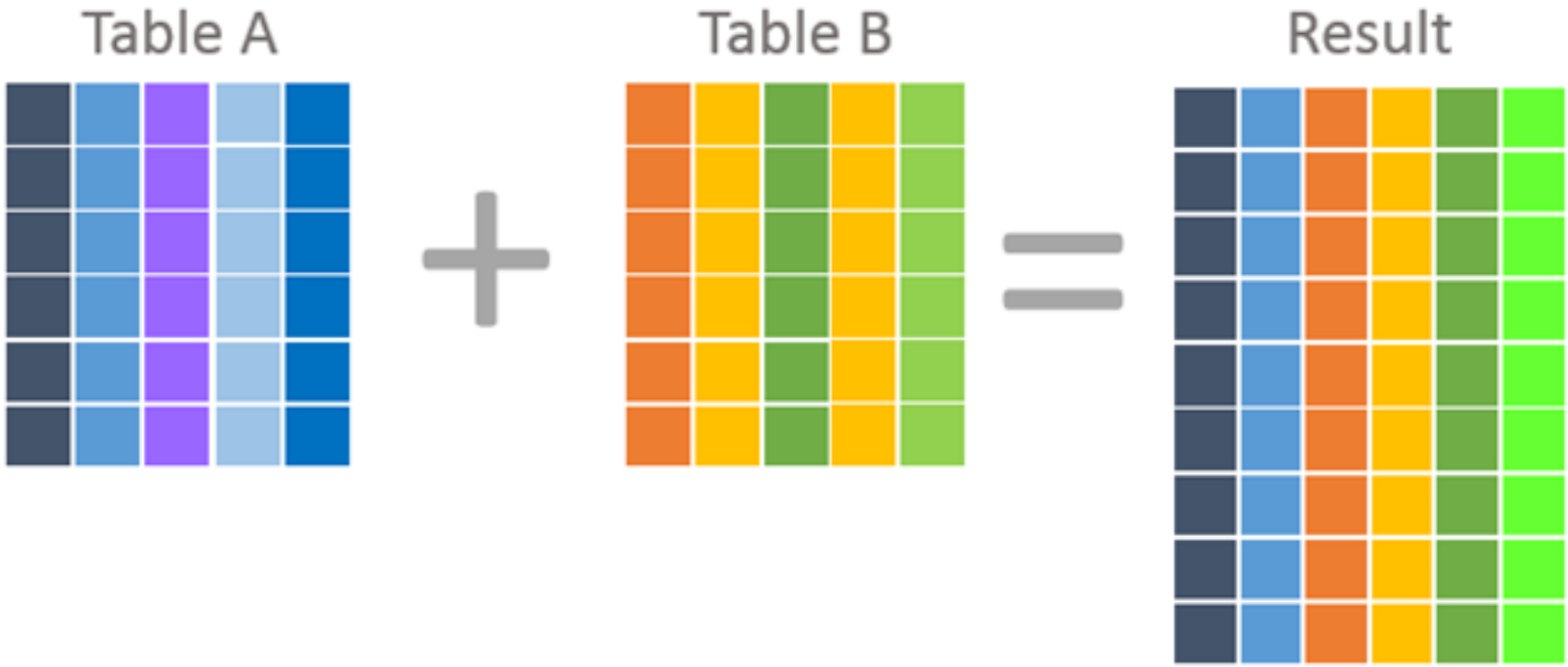
- Common columns are usually in a primary key - foreign key relationship (PK-FK)

JOIN

- Used to combine columns from different tables into new columns in a single result set.
  - Data from the first table is shown in one set of columns alongside the second table's column in the same row.
  - Each row in the result set contains columns from BOTH table tables.
  - Rows are created when columns from one table match columns from another.

- The foreign key in one table to look up column values by using the primary key in another.

JOIN



SQL Equijoin

- Performs a JOIN against equality or matching column(s) values of the associated tables.
- An equal sign (=) is used as comparison operator in the where clause to refer equality.

$$\text{Emp} \bowtie_{\text{Emp.DeptID=Dept.DeptID}} \text{Dept}$$

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

SQL Equijoin

Employee $\bowtie_{Dno=Dnumber}$ Department

```
SELECT *
FROM Employee, Department
WHERE Dno = Dnumber
```

SQL Equijoin

Employee $\bowtie_{\text{Dno=Dnumber}}$ Department

```
SELECT *
FROM Employee
JOIN  Department
ON Dno = Dnumber
```

SQL Natural Join

- A type of EQUI JOIN and is structured in such a way that, columns with the same name of associated tables will appear once only.

Guidelines

- The associated tables have one or more pairs of identically named columns.
- The columns must be the same data type.
- Don’t use ON clause in a natural join.

SQL Natural Join

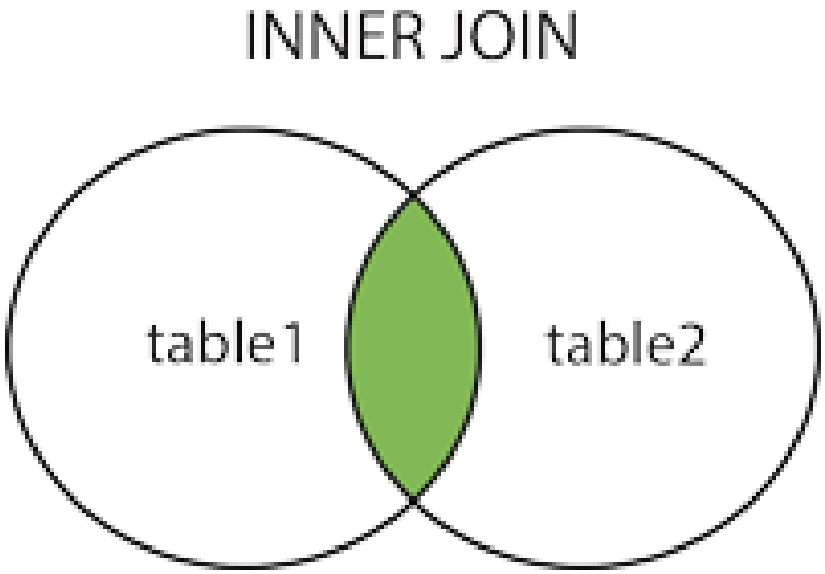
- Both fields should have the **same name** and **data type**.

Employee $\bowtie$ Department

```
SELECT *
FROM Employee
NATURAL JOIN Department
```

INNER JOIN

- Returns all rows from participating tables where the join condition is met.
- Displays only the rows that have a match in both the joined tables.




```
SELECT *
FROM  Customers AS C
INNER JOIN Orders AS O
ON C.CustomerId = O.CustomerId
```

```
SELECT *
FROM  Customers AS C
JOIN Orders AS O
ON C.CustomerId = O.CustomerId
```

OUTER JOIN

- Returns all rows from both the participating tables which satisfy the join condition along with rows which do not satisfy the join condition.

- Where join condition value is not present in the second table, results table padded out with null values.

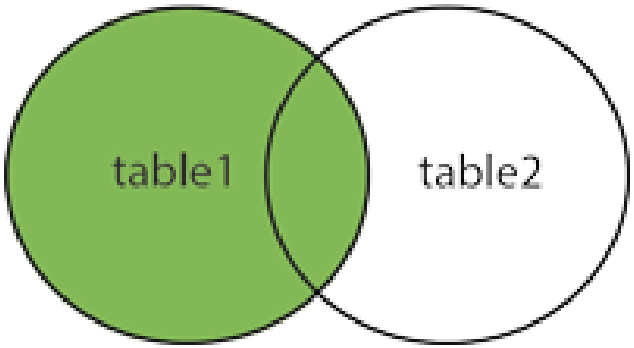
  – LEFT JOIN
  – RIGHT JOIN
  – FULL JOIN

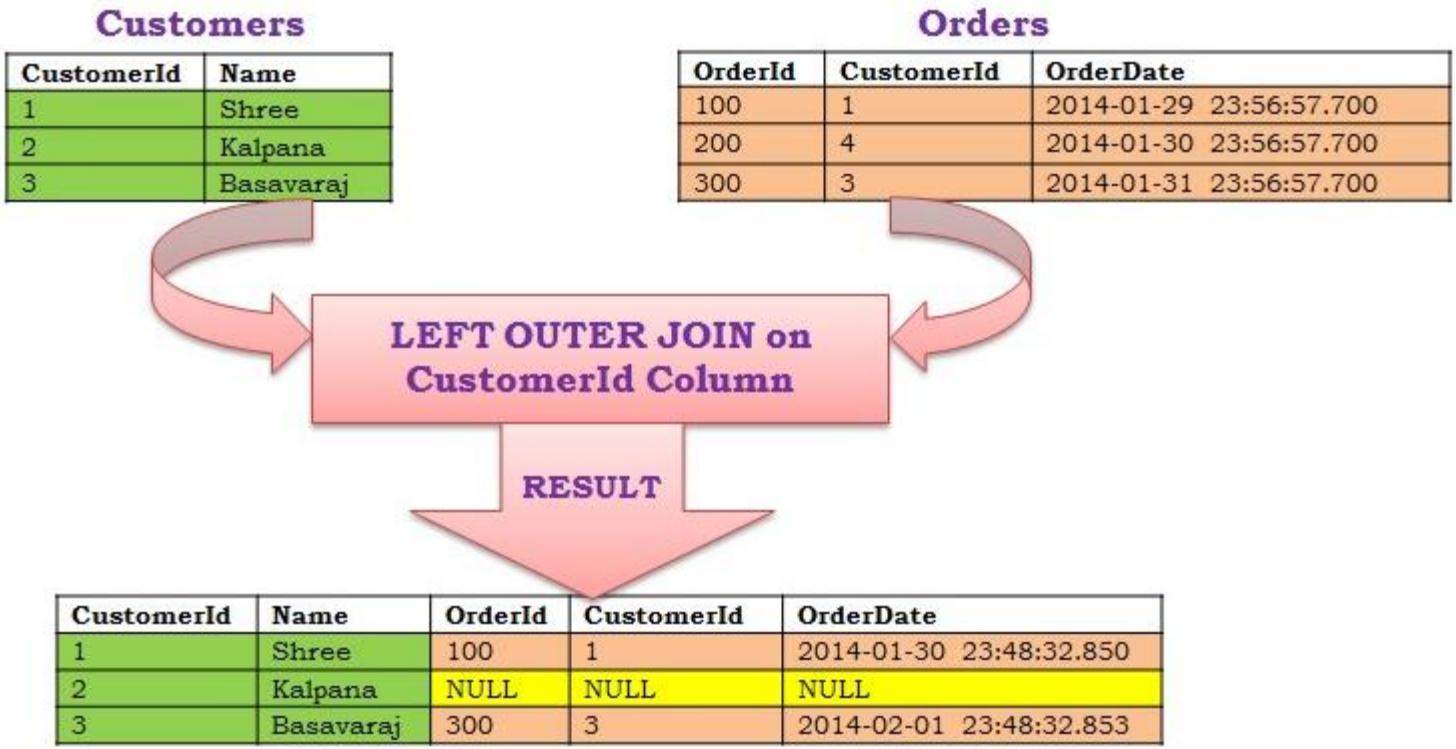
LEFT OUTER JOIN

- Return all rows from the left table, and the matched rows from the right table.

- The result is NULL in the right side when there is no match.

$$P \;⟕_{P.PNO=SP.PNO}\; SP$$





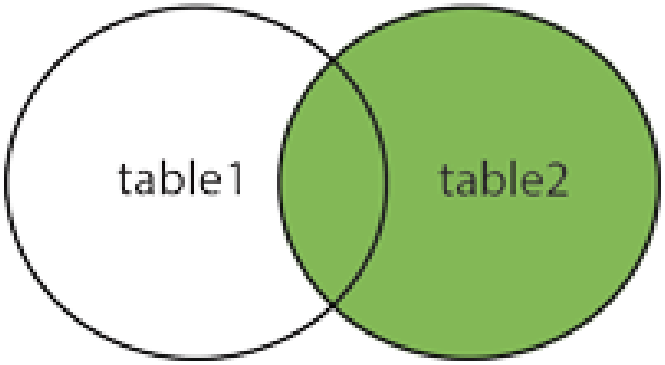

```
SELECT *
FROM  Customers AS C
LEFT OUTER JOIN Orders AS O
ON C.CustomerId = O.CustomerId
```

RIGHT OUTER JOIN

- Return all rows from the right table, and the matched rows from the left table.

- The result is NULL in the left side when there is no match.

$$P \bowtie_{P.PNO=SP.PNO} SP$$



Customers

CustomerId	Name
1	Shree
2	Kalpana
3	Basavaraj

Orders

OrderId	CustomerId	OrderDate
100	1	2014-01-29 23:56:57.700
200	4	2014-01-30 23:56:57.700
300	3	2014-01-31 23:56:57.700

RIGHT OUTER JOIN on CustomerId Column

RESULT

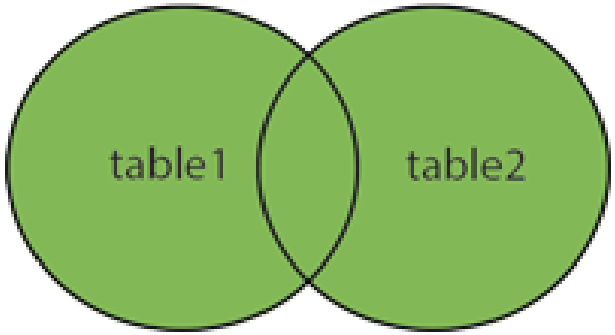
CustomerId	Name	OrderId	CustomerId	OrderDate
1	Shree	100	1	2014-01-30 23:48:32.850
NULL	NULL	200	4	2014-01-31 23:48:32.853
3	Basavaraj	300	3	2014-02-01 23:48:32.853

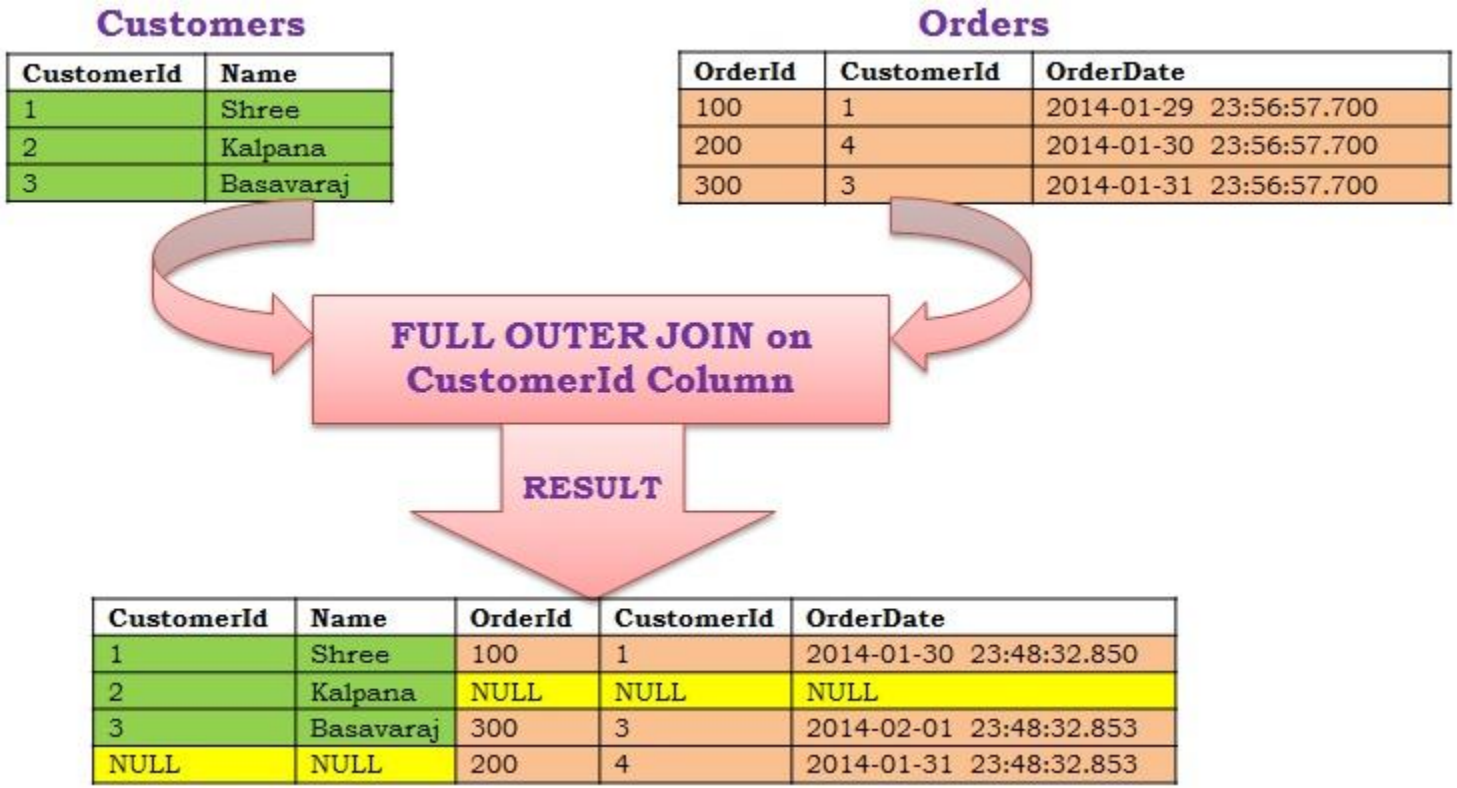
```
SELECT *
FROM  Customers AS C
RIGHT OUTER JOIN Orders AS O
ON C.CustomerId = O.CustomerId
```

FULL OUTER JOIN

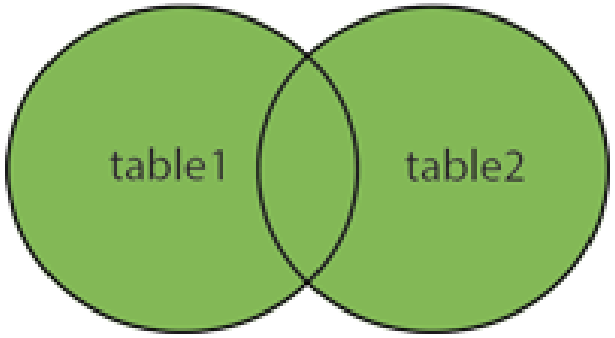
- Return all rows when there is a match in ONE of the tables.
- Returns all the rows from both tables whether it has been matched or not.
- Combines the result of both LEFT and RIGHT joins.

$$P \bowtie_{P.PNO=SP.PNO} SP$$






```
SELECT *
FROM Customers AS C
FULL OUTER JOIN Orders AS O
ON C.CustomerId = O.CustomerId
```




```
SELECT * FROM  Customers AS C
LEFT OUTER JOIN Orders AS O
ON C.CustomerId = O.CustomerId
UNION
SELECT * FROM  Customers ASC
RIGHT OUTER JOIN Orders AS O
ON C.CustomerId = O.CustomerId
```

SELF JOIN

- A table is joined to itself (Unary relationships), especially when the table has a foreign key which references its own primary key.
- Can be viewed as a join of two copies of the same table.
- Self join is used to retrieve the records having some relation or similarity with other records in the same table.
- Used where the same table needs to be visited twice.

EmployeeId	Name	ManagerId
1	Shree	1
2	Kalpana	1
3	Basavaraj	2
4	Monty	2

If we need to get the name of the Employee and his Manager name for each employee in the Employee Table.

RESULT:

EmployeeId	Employee Name	Manager Name
1	Shree	Shree
2	Kalpana	Shree
3	Basavaraj	Kalpana
4	Monty	Kalpana

SELF JOIN

$$E \bowtie_{E.ManagerID=M.EmployeeID} M$$

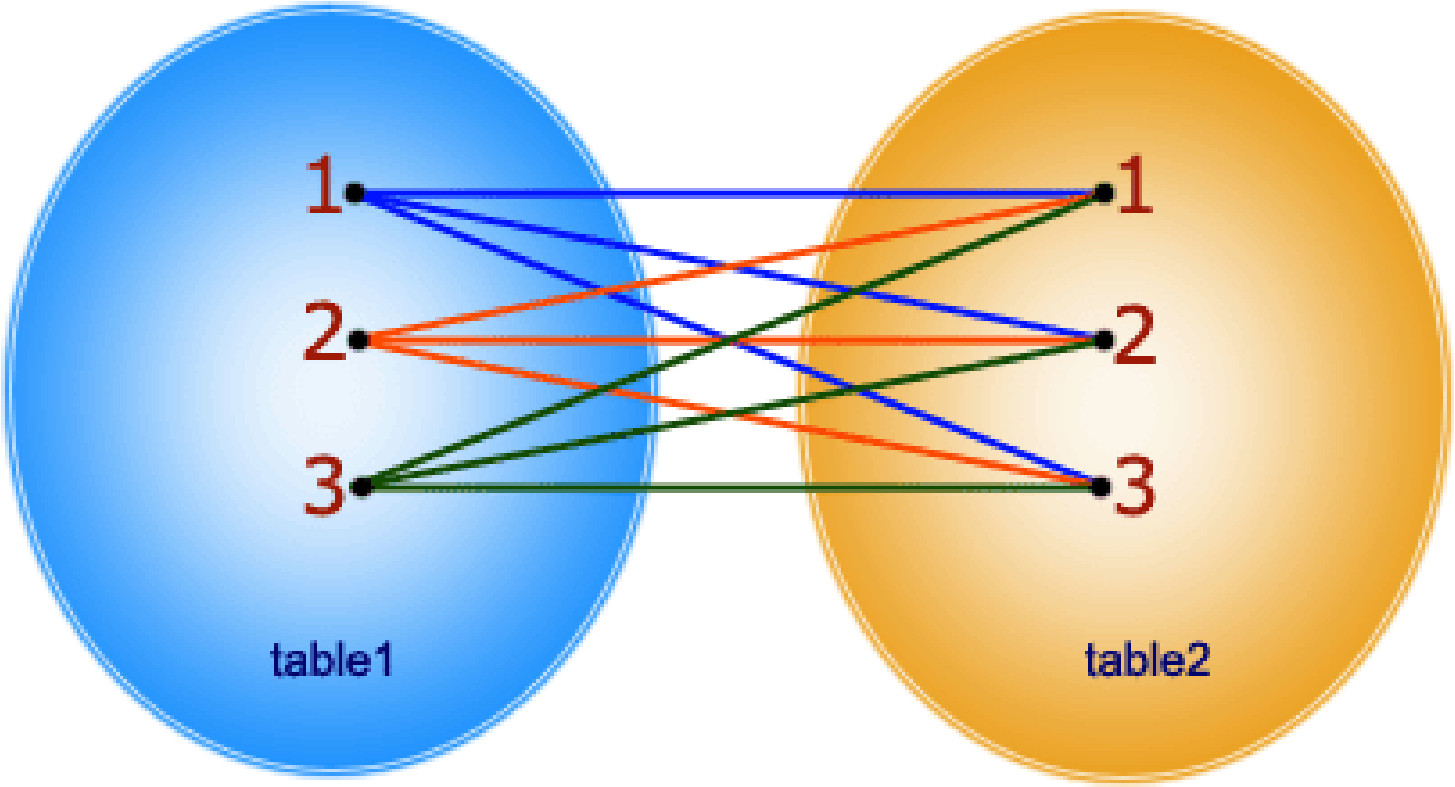
```
SELECT E.EmployeeId, E.Name AS 'Employee Name',
        M.Name AS ‘Manager Name'
FROM Employee AS E
JOIN Employee AS M
ON E.ManagerId = M.EmployeeId
```

Product

- Returns all possible combinations of rows from two tables.
- Produces **Cartesian product** of the tables that are involved in the join.
- The size of a Cartesian product is the number of the rows in the first table multiplied by the number of rows in the second table.

$$A \times B$$

SELECT * FROM table1 CROSS JOIN table2;



In CROSS JOIN, each row from 1st table joins with all the rows of another table. If 1st table contain x rows and y rows in 2nd one the result set will be x * y rows.

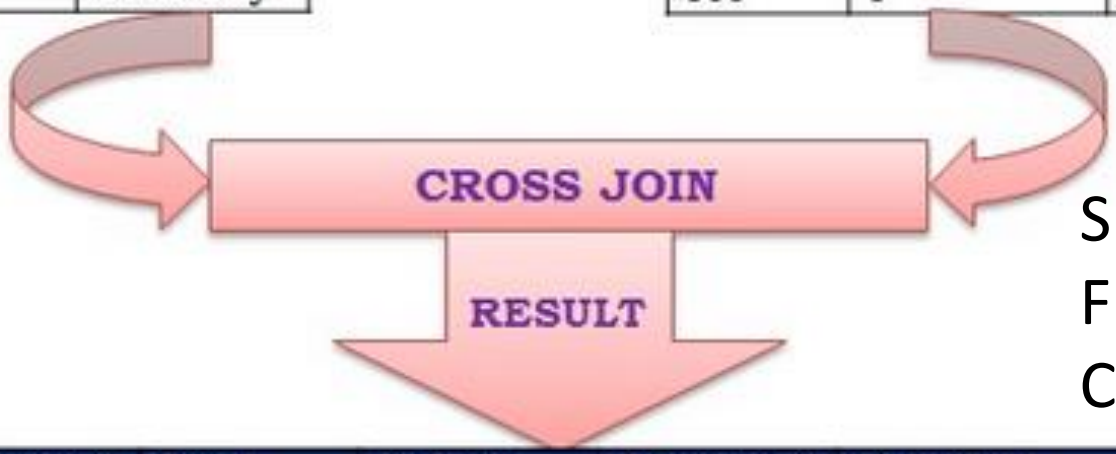
CROSS JOIN

Customers

CustomerId	Name
1	Shree
2	Kalpana
3	Basavaraj

Orders

OrderId	CustomerId	OrderDate
100	1	2014-01-29 23:56:57.700
200	4	2014-01-30 23:56:57.700
300	3	2014-01-31 23:56:57.700




```
SELECT *
FROM  Customers
CROSS JOIN Orders
```

CustomerId	Name	OrderId	CustomerId	OrderDate
1	Shree	100	1	2014-01-30 23:48:32.850
2	Kalpana	100	1	2014-01-30 23:48:32.850
3	Basavaraj	100	1	2014-01-30 23:48:32.850
1	Shree	200	4	2014-01-31 23:48:32.853
2	Kalpana	200	4	2014-01-31 23:48:32.853
3	Basavaraj	200	4	2014-01-31 23:48:32.853
1	Shree	300	3	2014-02-01 23:48:32.853
2	Kalpana	300	3	2014-02-01 23:48:32.853
3	Basavaraj	300	3	2014-02-01 23:48:32.853

DIVISION

- Returns dividend rows that relate to **all** specified rows in divisor table.

$$A \div B$$

Z = no of tuples in relation A

X = no of tuples in relation B

and X < Z

- For a tuple ‘t’ to appear in the result of the division, the value in ‘t’ must appear in ‘A’ in combination with every tuple in ‘B’.

A

SNO	PNO
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4

B1

PNO
P2

B2

PNO
P2
P4

B3

PNO
P1
P2
P4

A/B1

SNO
S1
S2
S3
S4

A/B2

SNO
S1
S4

A/B3

SNO
S1

Example

A

SNO	PNO
S1	P1
S2	P3
S3	P3
S5	P1
S5	P3

÷

B

PNO
P1
P3

=

SNO
S5

"Which supplier(s) supplies **ALL** products?"

SQL Division

```
SELECT Sno
FROM A A1
WHERE NOT EXISTS
(SELECT *
 FROM B
 WHERE NOT EXISTS
  (SELECT *
   FROM A A2
   WHERE A1.Sno = A2.Sno
   AND A2.Pno = B.Pno))
```

Thank You