PostgreSQL

Lesson 4: PostgreSQL - Joins and Subqueries

Lesson Objectives

In this lesson, you will learn about:

- Joins:
  - CROSS JOINS
  - INNER JOIN
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN
- Subqueries
  - Single row
  - Multirow
- Set operators
  - Union, Union All, Intersect, Except

JOINS

- To combine data from two or more tables in a database we can use Joins
- We can combine fields from two tables by using values common to each
- Types of joins in PostgreSQL:
  - CROSS JOINS
  - INNER JOIN
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN

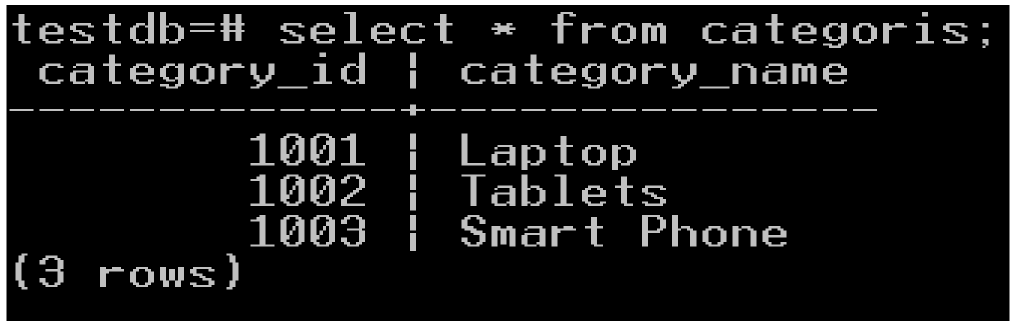
CROSS JOIN

- CROSS JOIN – Every row from first table matches with every row in second table
- If input tables have x and y rows then result will have x*y rows
- So if we have to show employee table along with department names

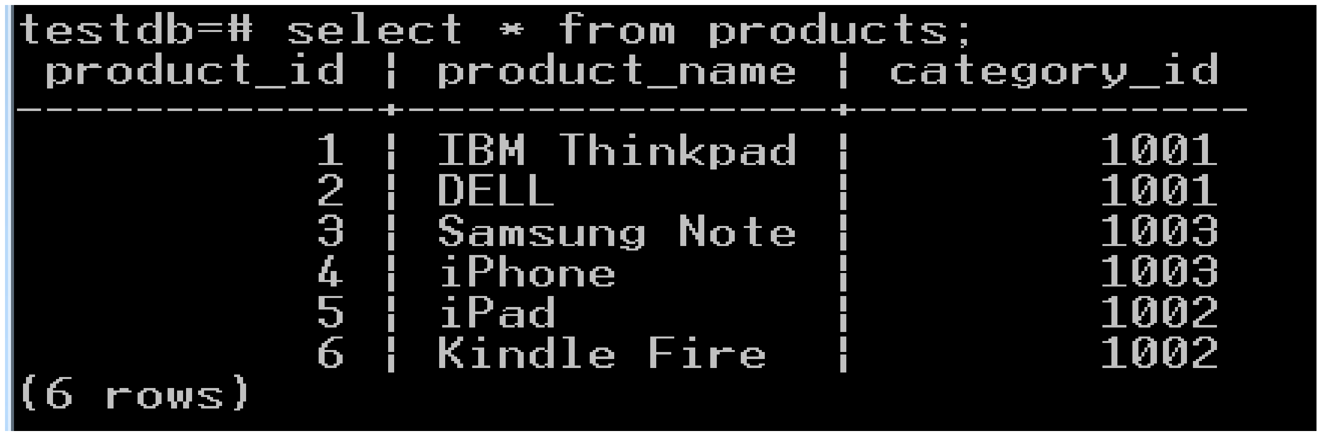
```
SELECT empid, name, dname FROM employee CROSS JOIN department;
```

NATURAL JOIN

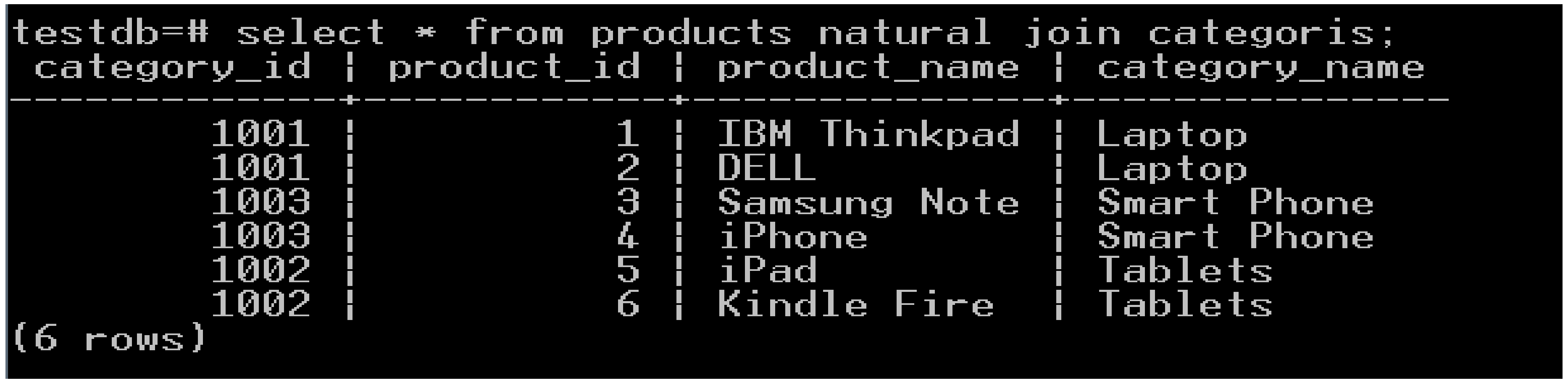
- NATURAL JOIN – Natural join can be used to access the data from two or more tables
- A natural join is a join that creates an implicit join based on the same column names in the joined tables



```
testdb=# select * from categoris;
 category_id | category_name
-------------+---------------
        1001 | Laptop
        1002 | Tablets
        1003 | Smart Phone
(3 rows)
```



```
testdb=# select * from products;
 product_id | product_name   | category_id
------------+----------------+-------------
          1 | IBM Thinkpad   |        1001
          2 | DELL           |        1001
          3 | Samsung Note   |        1003
          4 | iPhone         |        1003
          5 | iPad           |        1002
          6 | Kindle Fire    |        1002
(6 rows)
```



```
testdb=# select * from products natural join categoris;
 category_id | product_id | product_name   | category_name
-------------+------------+----------------+---------------
        1001 |          1 | IBM Thinkpad   | Laptop
        1001 |          2 | DELL           | Laptop
        1003 |          3 | Samsung Note   | Smart Phone
        1003 |          4 | iPhone         | Smart Phone
        1002 |          5 | iPad           | Tablets
        1002 |          6 | Kindle Fire    | Tablets
(6 rows)
```

INNER JOIN

- INNER JOIN :
- A INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate
- The query compares each row of table1 with each row of table2 to find all pairs of rows, which satisfy the join-condition
- When the join-condition is satisfied, column values for each matched pair of rows of table1 and table2 are combined into a result row
- INNER JOIN is the most common type of join and is the default type of join. You can use INNER keyword optionally
- Example:

```
 SELECT empid, name, dname FROM employee INNER JOIN department
ON employee.deptno=department.deptid;
```

LEFT OUTER JOIN

- OUTER JOIN :
- Outer join is extension of INNER join
- PostgreSQL supports 3 types of OUTER joins:
  - LEFT
  - RIGHT
  - FULL
- LEFT OUTER JOIN
- First INNER join is performed
- Then, for each row in table T1 that does not satisfy the join condition with any row in table T2, a row is added with null values in columns of T2
- Example:

```
SELECT empid, name, dname FROM department
LEFT OUTER JOIN employee ON employee.deptno=department.deptid;
```

RIGHT OUTER JOIN, FULL OUTER JOIN

- This is similar to LEFT OUTER Join, only that it adds null values to match the table on right side
- Example:

```
SELECT empid, name, dname FROM employee
RIGHT OUTER JOIN department ON employee.deptno=department.deptid;
```

- FULL OUTER JOIN:
- Rows on right side table that do not satisfy the join condition with table on left side, a null value is added
- Also, rows on left side that do not satisfy the join condition with table on right side, a null value is added
- So it is called as full outer join

```
SELECT empid, name, dname FROM employee
FULL OUTER JOIN department ON employee.deptno=department.deptid;
```

Subqueries

- How can we get data about employees working in same department as ‘Divya’?
- This can be solved in following way:
  - Which department does ‘Divya’ work?

```
SELECT deptno FROM employee where name=‘Divya’;            => 30
```

  - Who all are working in department 30?

```
SELECT * FROM employee where deptno=30;
```

- Running 2 queries gives required solution.
- This can be done using 1 query using subqueries:

```
SELECT * FROM employee
where deptno=(SELECT deptno FROM employee where name=‘Divya’);
```

Subqueries

- Subquery is a query with in a query – inner query or nested query
- Subquery returns data which in turn is used to restrict the rows in outer query
- Subquery must follow some rules:
  - It must be enclosed in a parentheses
  - Subquery can use only one column in select clause unless multiple columns are in main query
  - Order by clause cannot be used in the subquery
  - Subquery that returns single value can use single row operators like >,<,=, <= and >=
  - Subquery that returns multiple values can use multi row operators like IN, ANY and ALL

Subqueries – returning multiple values

- If subquery returns multiple values, then we have to use multirow operators
- Multirow operators are as follows:
  - IN
  - ANY
  - ALL

  - IN – to match every value in the return list
  - Example: Get data about employees earning same salaries as that in department 10

```
SELECT * FROM employee
where salary  IN (SELECT salary FROM employee where deptno=10);
```

Subqueries – returning multiple values

- ANY – to check if row value is > or < any one value in the return list
- Example: Get data about employees earning salaries > any employee from department 10

```
SELECT * FROM employee
where salary  >any (SELECT salary FROM employee where deptno=10);
```

- ALL – to check if row value is > or < all the values in return list
- Example: Get data about employees earning salaries > all employees from department 10

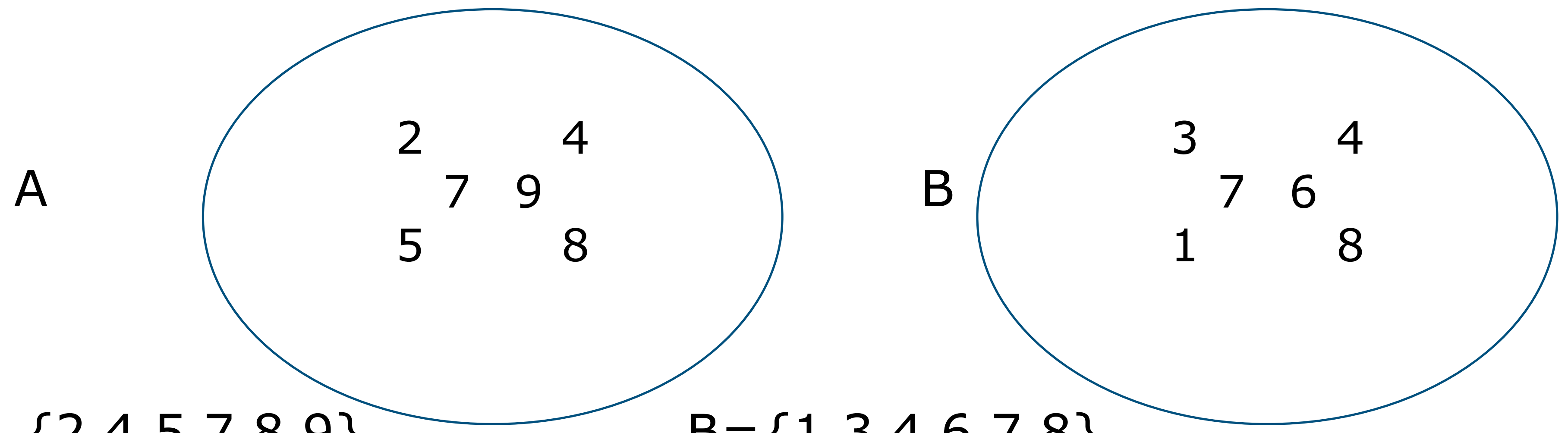
```
SELECT * FROM employee
where salary  >all (SELECT salary FROM employee where deptno=10);
```

SET Operators

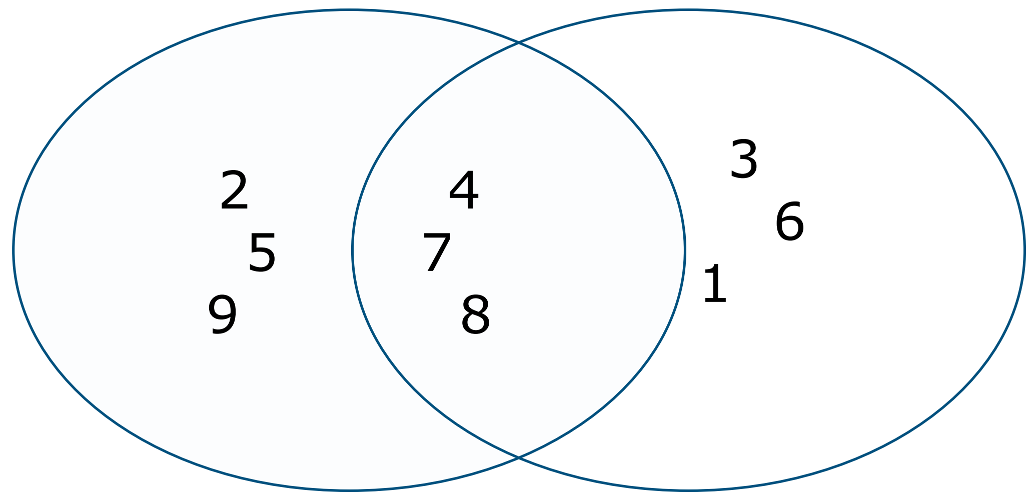
- Types of SET operators in PostgreSQL:
  - UNION
  - INTERSECT
  - EXCEPT

UNION Operators

- According to SET theory



- A = {2,4,5,7,8,9}   B={1,3,4,6,7,8}



A UNION B = {1,2,3,4,5,6,7,8,9}

A UNIONALL B={1,2,3,4,4,5,6,7,7,8,8,9}

UNION operator

- UNION Operator:
- It combines result set of two or more queries into a single result
- Rules applied to the queries using UNION operator:
  - Both queries must return the same number of columns.
  - The corresponding columns in the queries must have compatible data types
  - Example: show employee names who work in department 10 and also those who get salary >30000

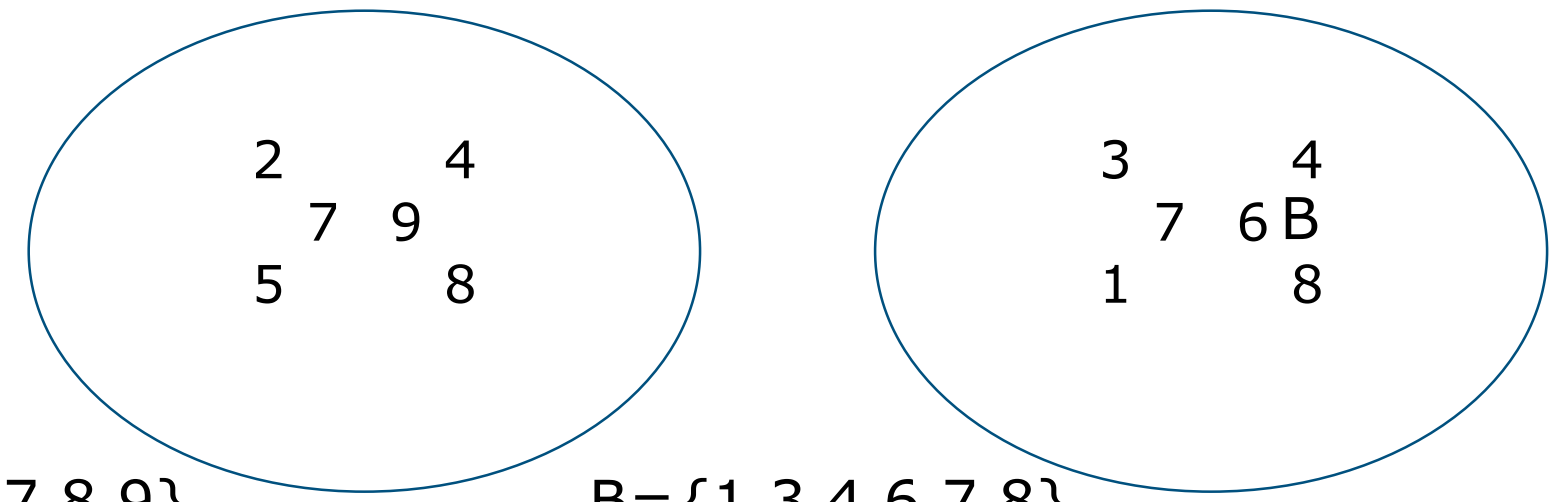
```
SELECT empid, name FROM employee where deptno=10
UNION
SELECT empid, name FROM employee where salary >30000;
```

UNION operator

- UNION removes all duplicate rows
- If you need to include duplicate rows also then we have to use UNION ALL
- If you have to sort the data then use ORDER BY clause in the last query

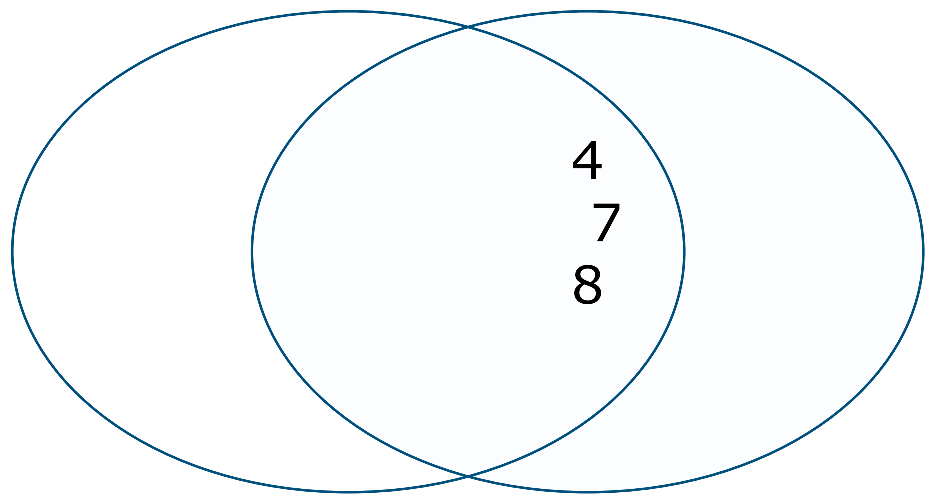
INTERSECT Operators

- According to SET theory



A = {2,4,5,7,8,9}

B={1,3,4,6,7,8}



A INTERSECT B = {4, 7,8}

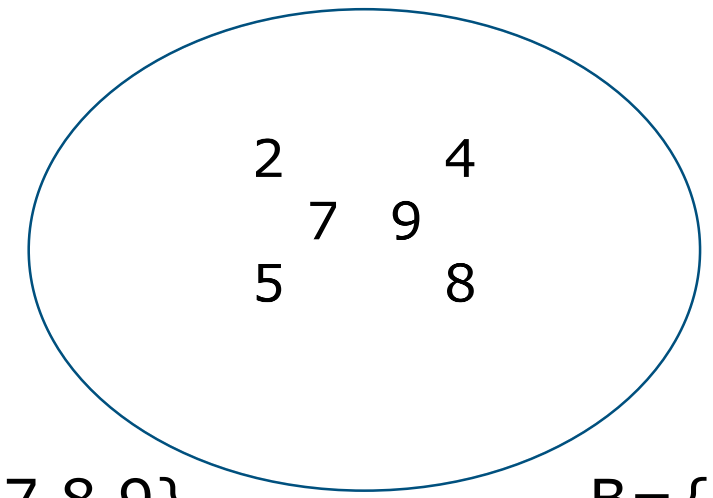
INTERSECT operator

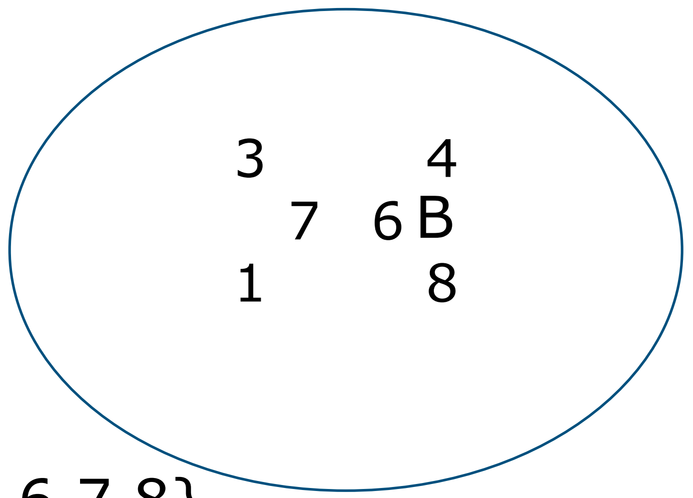
- INTERSECT Operator:
- It combines result set of two or more queries into a single result
- It returns all the rows in both result sets i.e. common rows in both

```
 SELECT empid, name, deptno, salary FROM employee where deptno=10
UNION
SELECT empid, name, deptno, salary FROM employee where salary >30000;
```

EXCEPT Operators

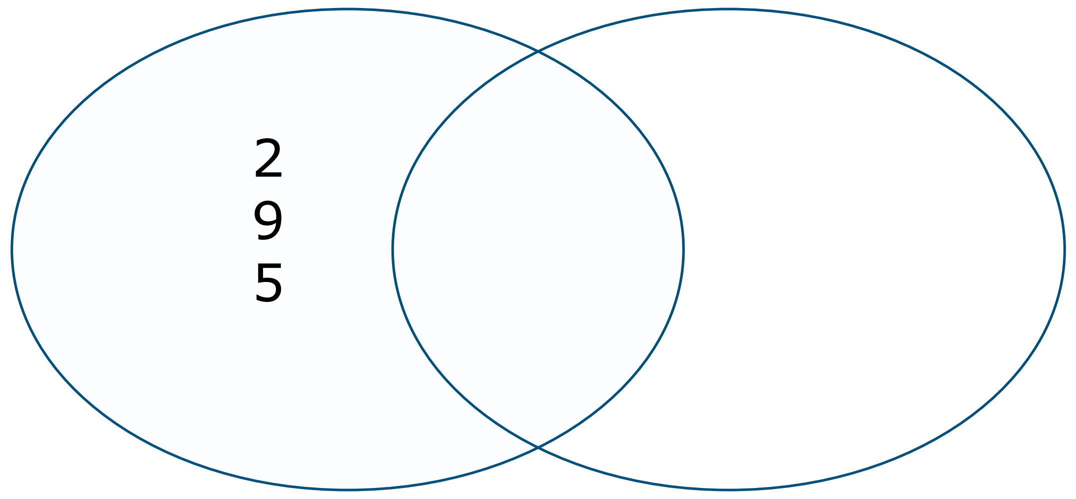
- According to SET theory





A = {2,4,5,7,8,9}

B={1,3,4,6,7,8}



A INTERSECT B = {4, 7,8}

EXCEPT operator

- EXCEPT Operator:
- It combines result set of two or more queries into a single result
  - It returns all the rows in both result sets i.e. common rows in both
  - EXCEPT operator to return the rows in the first query that do not appear in the output of the second query
- To combine the queries using EXCEPT operator, you must obey the following rules:
  - The number of columns and their orders must be the same in the two queries.
  - The data types of the respective columns must be compatible.

```
 SELECT empid, name, deptno, salary FROM employee where deptno=10
EXCEPT
SELECT empid, name, deptno, salary FROM employee where salary >30000;
```

Demo

Using all Join types

Create subqueries

Using set operators

Lab

Lab 3

Summary

In this lesson, you have learn about:

- Joins are used to fetch data from more then one table
- Different types of joins are inner join, outer join – left, right and full, cross join and self join
- Subquery is a query within a query. Inner query returns a result which is used by outer query.
- Single row sub query uses single row operator and multirow operator uses multirow operators
- Set operators are used to combine results of two different queries

Review Question

Question 1: If we try to fetch data from two different tables without writing join condition then we call it as ________.

Question 2: We need result set from first query which does not exist in second query, then we use which of the following operator?

- UNION
- UNION ALL
- INTERSECT
- EXCEPT

People matter, results count.

About Capgemini

A global leader in consulting, technology services and digital transformation, Capgemini is at the forefront of innovation to address the entire breadth of clients' opportunities in the evolving world of cloud, digital and platforms. Building on its strong 50-year heritage and deep industry-specific expertise, Capgemini enables organizations to realize their business ambitions through an array of services from strategy to operations. Capgemini is driven by the conviction that the business value of technology comes from and through people. It is a multicultural company of 200,000 team members in over 40 countries. The Group reported 2016 global revenues of EUR 12.5 billion.

Visit us at

www.capgemini.com