SQL essentials

Mastering database queries:
a comprehensive guide to SQL

6

Learning session 06

Using set operators to combine the results of multiple queries

Instructor
Péter Fülöp
peteristvan.fulop@hcl.com

Introduction to set theory

The application of set theory in SQL

- **Set Theory** is a mathematical concept where a set is defined as a collection of elements. This collection adheres to **specific rules**, including the **absence of duplicates**, and **the elements themselves are not arranged** in any particular order.

- The basic principle of set theory is that **an element belongs to a set**.

- In SQL, **the outcome of a query can be viewed as a set**, illustrating the application of set theory.

- When **applying set operators in SQ**L, the SQL sets involved must adhere to two criteria: they should contain an equal number of columns, and the corresponding columns in each set must share the same datatype.

Set operators

UNION

This operator combines the result sets of two or more SELECT statements and eliminates duplicate rows. The result is a set that contains all distinct rows from the queries involved.

INTERSECT

The **INTERSECT set operator** is used to return the intersection of two or more queries. Essentially, it finds the common elements that exist in the result sets of both queries. **The number and order of columns** in the SELECT statements must be the same. **The data types of the corresponding columns** must be compatible or, ideally, the same.

EXCEPT

The **EXCEPT set operator** is used to return all rows from the first (left) query that are not found in the second (right) query. It effectively subtracts the result of one query from another.

Example 1: using the UNION set operator

**The company aims to assign team leads for employees who currently do not have one.
To facilitate this, a list should be created that includes both employees lacking a team lead and those who are already serving as team leads. Additionally, the list should feature a column titled 'is_team_lead', marked with 'Y' or 'N' to differentiate between the roles.**

Let's split our approach into **two distinct queries**, subsequently merging their results. **The first query** will focus on employees without a team lead, while the **second query** will composea list of existing team leads.
IMPORTANT: both queries should return the same columns in the same order.

SQL

```
query1
UNION
query2
```

Example 1: Using the UNION set operator

(Part 1) Create a list that includes all the employees lacking a team lead.

SQL

```sql
SELECT id, first_name, last_name, 'N' as is_team_lead
FROM employees
WHERE lead_employee_id IS NULL;
```

OUTPUT

	id	first_name	last_name	is_team_lead
1	16	Eva	Kiss	N
2	17	Tamas	Molnar	N
3	18	Zsuzsa	Papp	N
4	19	Laszlo	Varga	N
5	20	Noemi	Biro	N
6	10	Laszlo	Nagy	N

6 rows

Example 1: Using the UNION set operator

(Part 2) Create a list that includes all team leads.

SQL

```sql
SELECT emp.id, emp.first_name, emp.last_name, 'Y' as is_team_lead
FROM employees emp
         INNER JOIN (SELECT DISTINCT lead_employee_id FROM employees) AS tld
                    ON emp.id = tld.lead_employee_id;
```

OUTPUT

	id	first_name	last_name	is_team_lead
1	1	Gabor	Nagy	Y
2	2	Istvan	Kiss	Y
3	3	Zoltan	Toth	Y
4	4	Eva	Horvath	Y

10 rows

Example 1: Using the UNION set operator

Let's merge the results of the two queries using the UNION set operator.

SQL

```
(SELECT id, first_name, last_name, 'N' as is_team_lead
FROM employees
WHERE lead_employee_id IS NULL)

UNION

(SELECT emp.id, emp.first_name, emp.last_name, 'Y' as is_team_lead
FROM employees emp
         INNER JOIN (SELECT DISTINCT lead_employee_id FROM employees) AS tld
                    ON emp.id = tld.lead_employee_id)
ORDER BY last_name ASC, first_name ASC;
```

Example 2: Using the UNION set operator

The firm resolved to implement a salary hike of 10% for employees engaged in active projects, and a 5% raise for those not involved in active projects but who had their most recent salary review six months ago or more. The list should include employee ID, first names, last names, present salary, percentage of salary increment, and the amount of the raise. This list should be arranged alphabetically by last names and then by first names.

Let's break down our solution into **two separate queries** and then combine or merge the resulting rows. **The first query** will determine those employees who are going to receive 10% salary raise and **the second query** will determine employess receiving 5% salary raise. Both queries will return the same columns in the same order.

SQL

```
query1
UNION
query2
```

Example 2: Using the UNION set operator

The first query below solves the following section of the previous task:

(PART 1) The firm resolved to implement a salary hike of 10% for employees engaged in active projects. The list should include employee ID, first names, last names, present salary, percentage of salary increment, and the amount of the raise.

QUERY 1 SQL

```
SELECT emp. id, emp.first_name, emp.last_name, emp.salary,
    10 as salary_raise_percent, 0.1*salary as salary_raise
FROM employees emp
    INNER JOIN project_assignments as pra ON emp.id = pra.employee_id
    INNER JOIN projects as p ON pra.project_id = p.id
WHERE COALESCE(p.start_date, current_date + 1) <= current_date
    AND p.completion_date IS NULL
    AND p.cancellation_date IS NULL
    AND COALESCE(pra.start_date, current_date + 1) <= current_date
    AND pra.end_date IS NULL;
```

Example 2: Using the UNION set operator

The second query below solves the following section of the previous task:

(PART 2) The firm resolved to implement a 5% raise for those not involved in active projects but who had their most recent salary review six months ago or more. The list should include employee ID, first names, last names, present salary, percentage of salary increment, and the amount of the raise.

QUERY 2 SQL

```
SELECT emp.id, emp.first_name, emp.last_name, emp.salary, 5 as salary_raise_percent, 0.05 * salary
FROM employees emp
WHERE COALESCE(current_date - emp.last_salary_review_date, 0) >= 180
  AND emp.id NOT IN (SELECT emp1.id
                     FROM employees emp1
                              INNER JOIN project_assignments as pra1 ON emp1.id = pra1.employee_id
                              INNER JOIN projects as p1 ON pra1.project_id = p1.id
                     WHERE COALESCE(p1.start_date, current_date + 1) <= current_date
                       AND p1.completion_date IS NULL
                       AND p1.cancellation_date IS NULL
                       AND COALESCE(pra1.start_date, current_date + 1) <= current_date
                       AND pra1.end_date IS NULL)
```

Example 2: Using the UNION set operator

Combining the results of the queries shown on the previous two pages you should get the results below:

OUTPUT

	id	first_name	last_name	salary	salary_raise_percent	salary_raise
1	27	Andras	Farkas	3500	5	175
2	4	Eva	Horvath	4500	10	450
3	15	Istvan	Horvath	4500	5	225
4	23	Ferenc	Kiss	2900	5	145
5	2	Istvan	Kiss	3500	10	350
6	6	Andras	Kovacs	2600	10	260
7	11	Anna	Kovacs	3500	10	350
8	7	Anita	Molnar	3800	10	380
9	22	Anna	Nagy	3800	5	190
10	29	Bela	Nagy	3100	5	155
11	1	Gabor	Nagy	2500	10	250
12	10	Laszlo	Nagy	3200	10	320

18 rows

Example 2: refactoring - increasing code readability

The queries shown in example 2 can be restructured using Common Table Expressions (CTEs), which will greatly enhance the readability of the code.

QUERY REFACTORING (PART 1) SQL

```sql
WITH employees_on_active_project AS
    (SELECT emp1.id
     FROM employees emp1
              INNER JOIN project_assignments as pra1
              ON emp1.id = pra1.employee_id
              INNER JOIN projects as p1 ON pra1.project_id = p1.id
     WHERE COALESCE(p1.start_date, current_date + 1) <= current_date
       AND p1.completion_date IS NULL
       AND p1.cancellation_date IS NULL
       AND COALESCE(pra1.start_date, current_date + 1) <= current_date
       AND pra1.end_date IS NULL )
```

Example 2: refactoring - increasing code readability

Let us use the defined CTE in refactoring our queries.

QUERY REFACTORING (PART 2) SQL

```sql
WITH employees_on_active_project AS (...)
SELECT emp.id, emp.first_name, emp.last_name, emp.salary,
       10 as salary_raise_percent, 0.1 * salary as salary_raise
FROM employees emp
         INNER JOIN employees_on_active_project eoap ON emp.id = eoap.id
UNION
SELECT emp.id, emp.first_name, emp.last_name, emp.salary,
       5 as salary_raise_percent, 0.05 * salary
FROM employees emp
         LEFT OUTER JOIN employees_on_active_project eoap ON emp.id = eoap.id
WHERE COALESCE(current_date - emp.last_salary_review_date, 0) >= 180
  AND eoap.id IS NULL
ORDER BY last_name, first_name;
```

Retaining/removing duplicates

The UNION set operator eliminates duplicate rows; to retain duplicates, use UNION ALL instead.

SQL

```sql
select * from employees
UNION ALL
select * from employees
order by id;
```

OUTPUT

	id	first_name	middle_name	last_name	hire_date
1	1	Gabor	<null>	Nagy	2022-02-15
2	1	Gabor	<null>	Nagy	2022-02-15
3	2	Istvan	Peter	Kiss	2022-03-01
4	2	Istvan	Peter	Kiss	2022-03-01

58 rows

Using the INTERSECT set operator

Which languages are spoken in both Argentina and Chile?
Do not include generic terms such as 'indigenous', 'Other', or 'unspecified'. Provide the names of specific languages.
Order the list of languages in the alphabetical order.

Let's break down our solution into **two separate queries** and then intersect the resulting rows.

The first query will determine languages that are spoken in Argentina. The second query will detemine languages spoken in Chile.

SQL

```
query1
INTERSECT
query2
```

Using the INTERSECT set operator

(PART 1) Which languages are spoken in Argentina?
Do not include generic terms such as 'indigenous', 'Other', or 'unspecified'. Provide the names of specific languages.

SQL

```sql
SELECT name AS language
FROM languages
WHERE languages.code = (select code from countries where lower(name) = 'argentina')
      AND lower(languages.name) NOT IN ('indigenous', 'other', 'unspecified');
```

OUTPUT

	language
1	Spanish
2	Italian
3	English
4	German
5	French

5rows

Using the INTERSECT set operator

(PART 2) Which languages are spoken in Chile?
Do not include generic terms such as 'indigenous', 'Other', or 'unspecified'. Provide the names of specific languages.

SQL

```sql
SELECT name AS language
FROM languages
WHERE languages.code = (select code from countries where lower(name) = ‘chile’)
      AND lower(languages.name) NOT IN ('indigenous', 'other', 'unspecified');
```

OUTPUT

	language
1	Spanish
2	English

2 rows

Using the INTERSECT set operator

Let's intersect the resulting rows using the INTERSECT operator.

SQL

```
SELECT name AS language
FROM languages
WHERE languages.code = (select code from countries where lower(name) = ‘argentina’)
      AND lower(languages.name) NOT IN ('indigenous', 'other', 'unspecified')
INTERSECT
SELECT name AS language
FROM languages
WHERE languages.code = (select code from countries where lower(name) = ‘chile’)
      AND lower(languages.name) NOT IN ('indigenous', 'other', 'unspecified')
ORDER BY language ASC;
```

OUTPUT

	language
1	English
2	Spanish

2 rows

Using the EXCEPT set operator

Find the languages that are not the official language of any country.
Order the list of languages in the alphabetical order.

We can approach our solution in **two steps** by executing two separate queries and then combining the results using the EXCEPT set operator.
In the **first query**, we will identify languages that are spoken but are not recognized as official languages in at least one country. In the **second query**, we will specifically retrieve the official languages.

SQL

```
query1
EXCEPT
query2
```

Using the EXCEPT set operator

(PART 1) Which languages are official languages in at least one country.

SQL

```
SELECT name as language
FROM languages
WHERE official IS TRUE;
```

OUTPUT

	language
1	Turkic
2	Other
3	Greek
4	Other
5	unspecified
6	French

629 rows

Using the EXCEPT set operator

(PART 2) Which languages are spoken but are not recognized as official languages in at least one country.

SQL

```sql
SELECT name as language
FROM languages
WHERE official IS NOT TRUE;
```

OUTPUT

	language
1	Dari
2	Pashto
3	Albanian
4	Arabic
5	Berber or Tamazight
6	Catalan
7	Portuguese

294 rows

Using the EXCEPT set operator

Let's combine the resulting rows using the EXCEPT operator.

SQL

```sql
SELECT name AS language
FROM languages
WHERE official IS NOT TRUE
EXCEPT
SELECT name AS language
FROM languages
WHERE offcial IS TRUE
ORDER BY language ASC;
```

OUTPUT

	language
1	Akyem
2	Alsatian
3	Amerindian

253 rows