Combining Table Expressions

Overview

Dr. E. F. Codd created the relational database model based on set theory and first-order predicate logic. A basic understanding of sets will enhance your understanding of complex SQL queries. SQL has directly incorporated some of the set operations from mathematical set theory. There are set union (UNION), set difference (EXCEPT), and set intersection (INTERSECT) operations.

The SQL standard has specified the use of the UNION keyword to combine results from multiple result sets. The ALL keyword is added after UNION to keep the system from eliminating duplicate data. Without the ALL keyword the system will eliminate any duplicate rows in the result sets. To combine result sets, each set must have the same number of columns, and each corresponding column must have comparable data types.

Although it is not used often, the UNION is a powerful tool to understand.

Learning objectives

- What Is a Set?
- Union
- Intersection
- Difference
- SQL Set Operations

Contents

1. What Is a Set?

Each table in a database is a set of information about a subject. A SQL query returns a result set; it can be as little as returning no data. Each row returned in a result set is known as a member of the set

2. Set operators

Set operators combine the results of two component queries into a single result. Queries containing set operators are called **compound queries**. Like joins, set operators combine data from two or more tables but there is a big difference. Joins try to combine columns from the base tables, however, set operators combine rows from two or more result sets. Generally, SET operations are applied on multiple SELECT statements. The records returned by each SELECT statement are treated as a SET of values and the final result is obtained depending on the SET operator used.

Three most common set operations:

1. **Union:** Used to combine two or more similar sets
    a. Places two or more sets of similar data together without repeating duplicate members. Members must have the same number and types of attributes
    b. SQL uses the UNION operation to combine the results of two or more SQL queries. Nearly all database systems support UNION.
2. **Intersection:** Used to find the common elements between two or more sets
    a. Simple intersection requires that members must match on all attributes. Each member must have the same number and types of attributes
    b. The INTERSECT operation can be used to return intersecting data of two SQL queries. Not all database systems support INTERSECT
    c. SQL uses intersection in the JOIN operation
        i. An intersection on single key values
        ii. Most database systems support JOIN
        iii. Only the key fields need to match to be included in the result set
3. **Difference**: Used to find items that are in one set but not another
    a. Remove members that match on all attributes from the result set. Each member must have the same number and types of attributes
        i. The EXCEPT operation returns the difference of two sets of data. Not all database systems support EXCEPT
    b. SQL returns differences using the OUTER JOIN operation
        i. Intersection on key values that includes the unmatched values of one or both of the sets of data
        ii. Most database systems support OUTER JOIN

SQL also has corresponding multiset operations, which are followed by the keyword ALL

- UNION ALL
- EXCEPT ALL
- INTERSECT ALL

Their results are multisets (duplicates are not eliminated). Basically, each tuple is considered as a different tuple when applying these operations.

3. Conditions for SET Operators

Two SELECT statements can be combined into a compound query by a SET operation if they satisfy the following conditions:

- The result set of both the queries must have the same number of columns.
- The data type of each column in the first result set must match with the data types of the columns of the second result set.

4. Restrictions on SET Operators

- The column names for the result data set will come from the first query.
- If you want to use the ORDER BY clause in the query involving SET operations, you must place the ORDER BY clause only once at the end of the compound query. The component queries can't have individual ORDER BY clauses.

Syntax:

```
<query 1> [SET OPERATOR]  <query2>
```

5. The UNION Operator

The UNION Operator is used to combine the union compatible results of two SELECT statements by listing all rows from the result of the first SELECT statement and all rows from the result of the other SELECT statement. If two or more rows are identical only one of them is shown (duplicates are eliminated from the result).

The syntax for a union operation

```
SELECT_statement_1
UNION [ALL]
 SELECT_statement_2
[UNION [ALL]
 SELECT_statement_3]...
[ORDER BY order_by_list]
```

Rules:

- Each result set must return the same number of columns.
- The corresponding columns in each result set must have compatible data types.
- The column names in the final result set are taken from the first SELECT clause.
- The UNION operator automatically eliminates duplicate rows from the composite result set (this behavior is similar to that obtained by adding the DISTINCT keyword to a regular SELECT query).
- To see all of the records (including duplicates) in the UNION, add the ALL keyword to the UNION operator, as in the example query:
  `SELECT * FROM a UNION ALL SELECT * FROM b;`
- The ORDER BY clause is added at the end of the SQL expression

Example 1: List project numbers for all projects that involve an employee whose last name is 'wong' as a worker or as a manager of the department that controls the project.

```
mysql> (SELECT p.pname
         FROM   project p, department d, employee e
         WHERE  p.dnum= d.dnumber
         AND d.mgrssn=e.ssn and e.lname='wong')
        UNION
       (SELECT p.pname
        FROM project p, works_on a, employee e
        WHERE  p.pnumber= a.pno
        AND a.essn = e.ssn and e.lname = 'wong');
+-----------------+
| pname           |
+-----------------+
| ProductX        |
| ProductY        |
| ProductZ        |
| Computerization |
| Reorganization  |
+-----------------+
5 rows in set (0.02 sec)

mysql> (SELECT p.pname
        FROM   project p, department d,employee e
        WHERE  p.dnum = d.dnumber
        AND d.mgrssn=e.ssn and e.lname='wong')
        UNION ALL
       (SELECT p.pname
         FROM   project p, works_on a, employee e
         WHERE  p.pnumber= a.pno
         AND a.essn = e.ssn and e.lname = 'wong');
+-----------------+
| pname           |
+-----------------+
| ProductX        |
| ProductY        |
| ProductZ        |
| ProductY        |
| ProductZ        |
| Computerization |
| Reorganization  |
+-----------------+
7 rows in set (0.00 sec)
```

Example 2: Make a list of all project numbers for projects that involve an employee whose last name is 'James', either as a worker or as a manager of the department that controls the project.

```
mysql> (SELECT DISTINCT 'Manager' AS James, p.pnumber
        FROM project p
        JOIN department d ON p.dnum = d.dnumber
        JOIN employee e ON d.mgrssn = e.ssn
        WHERE e.lname = 'James')
       UNION
       (SELECT DISTINCT 'Worker' As James, p.pnumber
        FROM project p
        JOIN works_on a ON p.pnumber= a.pno
        JOIN employee e ON a.essn = e.ssn
        WHERE e.lname = 'James');
```

```
+---------+---------+
| James   | pnumber |
+---------+---------+
| Manager |      61 |
| Manager |      62 |
| Manager |      63 |
| Manager |     100 |
| Worker  |      61 |
| Worker  |      92 |
+---------+---------+
6 rows in set (0.01 sec)
```

The first SELECT query retrieves the projects that involve a 'James' as manager of the department that controls the project, and the second retrieves the projects that involve a 'James' as a worker on the project. Notice that if several employees have the last name 'James', the project names involving any of them will be retrieved.

Applying the UNION operation to the two SELECT queries gives the desired result.

A union that simulates a full outer join

A full outer join returns unmatched rows from both the left and right tables. Although MySQL doesn't provide language for coding a full outer join, you can simulate a full outer join by coding a union that combines the result sets for a left outer join and a right outer join.

Example 3: This example returns all the rows from the department and employee tables even if these rows don't have matching columns in the other table.

```
mysql> (SELECT d.dname As department_name, d.dnumber AS d_department_number
        ,e.dno AS e_department_number, e.lname
        FROM department d
        LEFT JOIN employee e ON (d.dnumber = e.dno))
       UNION
       (SELECT d.dname As department_name, d.dnumber AS d_department_number
        ,e.dno AS e_department_number,e.lname
        FROM department d
        RIGHT JOIN employee e ON (d.dnumber = e.dno))
```

department_name	d_department_number	e_department_number	lname
Software	6	6	James
Software	6	6	Jones
Software	6	6	Mark
Software	6	6	Knight
Information System	5	5	Smith
Hardware	7	7	Wallis
Hardware	7	7	Zell
Hardware	7	7	Vile
Hardware	7	7	Brand
Hardware	7	7	Vos
Hardware	7	7	Carter
Software	6	6	Grace
Software	6	6	Chase
Information System	5	5	Wong
Hardware	7	7	Freed
Hardware	7	7	Bays
Hardware	7	7	Best
Hardware	7	7	Snedden
Information System	5	5	English
Software	6	6	Ball
Sales	8	8	Bender
Sales	8	8	Jarvis
Sales	8	8	King
Sales	8	8	Leslie
Sales	8	8	Kramer
Sales	8	8	Small
Sales	8	8	Head
Sales	8	8	Pataki
Sales	8	8	Drew
Sales	8	8	Reedy
Sales	8	8	Hall
Sales	8	8	Bacher
Information System	5	5	Narayan
Headquarters	1	1	Borg
Administration	4	4	Wallace
Administration	4	4	Jabbar
Administration	4	4	Zelaya
Human Resources	10	NULL	NULL
NULL	NULL	NULL	Freed
NULL	NULL	9	Dan
NULL	NULL	NULL	James
NULL	NULL	NULL	Borg

42 rows in set (0.01 sec)

1.1. The INTERSECT Operator

The INTERSECT Operator is used to combine the results of two SELECT statements that are union compatible by listing every row that appears in the result of both of the SELECT statements.

Returns rows selected that are common to both queries.

```
Query1 INTERSECT Query2;
```

Syntax:

```
  SELECT_statement_1
INTERSECT
  SELECT_statement_2
 [ORDER BY order_by_list]
```

Note: MySQL does not support INTERSECT operator.

1.2. The EXCEPT (MINUS) Operator

The EXCEPT Operator is used to combine the results of two SELECT statements that are union compatible by listing every row from the result of the first SELECT statement that does not appear in the result of the other SELECT statement.

Returns all distinct rows selected by the first query and are not by the second.

```
Query1 MINUS Query2;
```

Syntax:

```
  SELECT_statement_1
MINUS
  SELECT_statement_2
  [ORDER BY order_by_list]
```

Note: MySQL does not support EXCEPT operator.