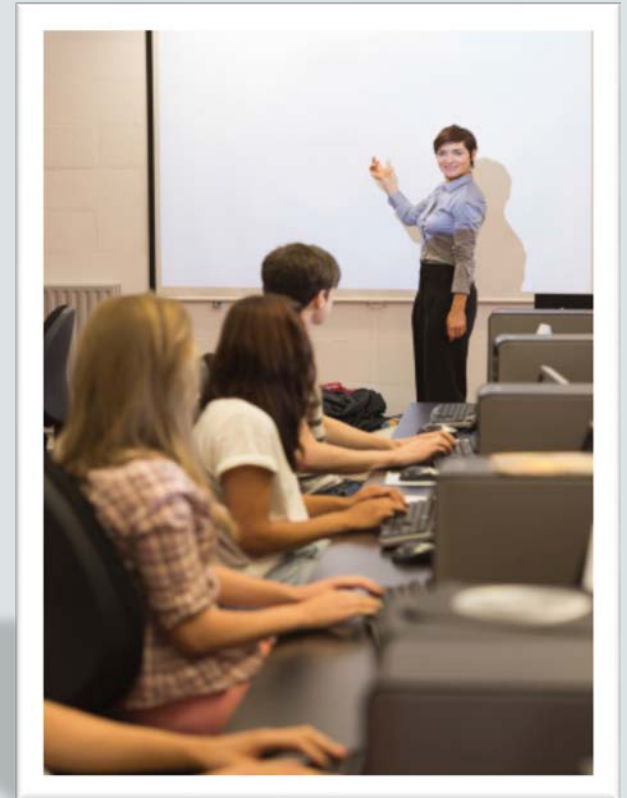




Database Programming with SQL

9-3

Using Set Operators



Objectives

This lesson covers the following objectives:

- Define and explain the purpose of Set Operators
- Use a set operator to combine multiple queries into a single query
- Control the order of rows returned using set operators

Purpose

- Set operators are used to combine the results from different SELECT statements into one single result output.
- Sometimes you want a single output from more than one table.
- If you join the tables, the rows that meet the join criteria are returned, but what if a join will return a result set that doesn't meet your needs?
- This is where SET operators come in.
- They can return the rows found in multiple SELECT statements, the rows that are in one table and not the other, or the rows common to both statements.

Setting the Stage

- In order to explain the SET operators, the following two lists will be referred to throughout this lesson:

$A = \{1, 2, 3, 4, 5\}$

$B = \{4, 5, 6, 7, 8\}$

- Or in reality: two tables, one called A and one called B.

A

| A_ID |
|------|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |

B

| B_ID |
|------|
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |

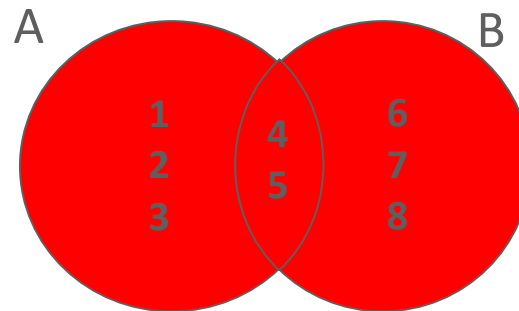
Rules to Remember

- There are a few rules to remember when using SET operators:
 - The number of columns and the data types of the columns must be identical in all of the SELECT statements used in the query.
 - The names of the columns need not be identical.
 - Column names in the output are taken from the column names in the first SELECT statement.
- So any column aliases should be entered in the first statement as you would want to see them in the finished report.

UNION

- The UNION operator returns all rows from both tables, after eliminating duplicates.

```
SELECT a_id
FROM a
UNION
SELECT b_id
FROM b;
```

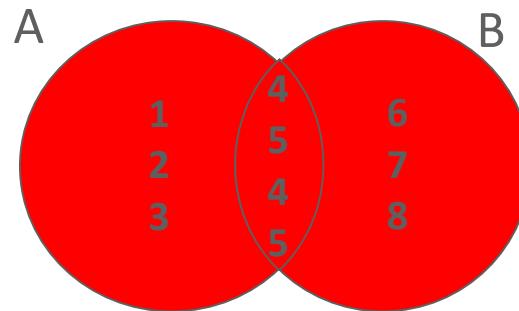


- The result of listing all elements in A and B eliminating duplicates is {1, 2, 3, 4, 5, 6, 7, 8}.
- If you joined A and B you would get only {4, 5}. You would have to perform a full outer join to get the same list as above.

UNION ALL

- The UNION ALL operator returns all rows from both tables, without eliminating duplicates.

```
SELECT a_id  
FROM a  
  UNION  ALL  
SELECT b_id  
FROM b;
```

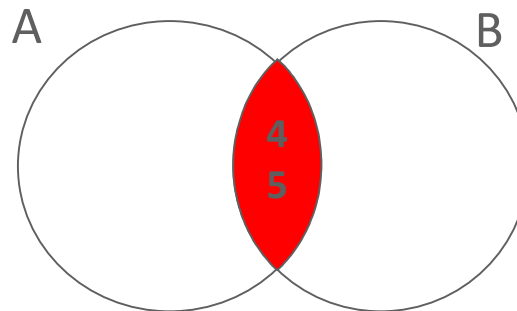


- The result of listing all elements in A and B without eliminating duplicates is {1, 2, 3, 4, 5, 4, 5, 6, 7, 8}.

INTERSECT

- The INTERSECT operator returns all rows common to both tables.

```
SELECT a_id
FROM a
  INTERSECT
SELECT b_id
FROM b;
```

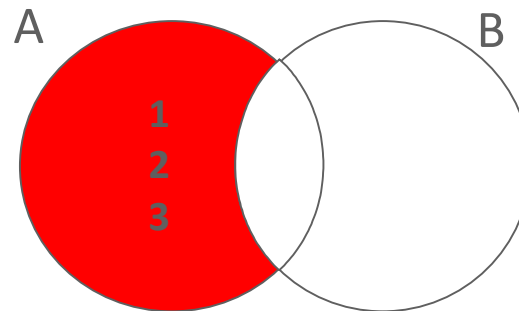


- The result of listing all elements found in both A and B is {4, 5}.

MINUS

- The MINUS operator returns all rows found in one table but not the other.

```
SELECT a_id  
FROM a  
  MINUS  
SELECT b_id  
FROM b;
```



- The result of listing all elements found in A but not B is {1, 2, 3}.
- The result of B MINUS A would give {6, 7, 8}.

Set Operator Examples

- Sometimes if you are selecting rows from tables that do not have columns in common, you may have to create your own columns in order to match the number of columns in the queries.
- The easiest way to do this is to include one or more NULL values in the select list.
- Remember to give each one a suitable alias and matching data type.

Set Operator Examples

- For example:
 - The employees table contains a hire date, employee id and a job id.
 - The job history table contains employee id and job id, but does not have a hire date column.
 - The two tables have the employee id and job id in common, but job history does not have a start date .
- You can use the TO_CHAR(NULL) function to create matching columns as in the next slide.

Set Operator Examples

```
SELECT hire_date, employee_id, job_id
FROM employees
UNION
SELECT TO_DATE(NULL), employee_id, job_id
FROM job_history;
```

| HIRE_DATE | EMPLOYEE_ID | JOB_ID |
|-------------|-------------|------------|
| 17-Jun-1987 | 100 | AD_PRES |
| 17-Sep-1987 | 200 | AD_ASST |
| 21-Sep-1989 | 101 | AD_VP |
| 03-Jan-1990 | 103 | IT_PROG |
| 21-May-1991 | 104 | IT_PROG |
| 13-Jan-1993 | 102 | AD_VP |
| 07-Jun-1994 | 205 | AC_MGR |
| 07-Jun-1994 | 206 | AC_ACCOUNT |
| 17-Oct-1995 | 141 | ST_CLERK |
| 17-Feb-1996 | 201 | MK_MAN |
| 11-May-1996 | 174 | SA_REP |
| 29-Jan-1997 | 142 | ST_CLERK |
| 17-Aug-1997 | 202 | MK_REP |
| 15-Mar-1998 | 143 | ST_CLERK |
| 24-Mar-1998 | 176 | SA_REP |
| 09-Jul-1998 | 144 | ST_CLERK |
| 07-Feb-1999 | 107 | IT_PROG |
| 24-May-1999 | 178 | SA_REP |
| 16-Nov-1999 | 124 | ST_MAN |
| 29-Jan-2000 | 149 | SA_MAN |
| - | 101 | AC_ACCOUNT |
| - | 101 | AC_MGR |
| - | 102 | IT_PROG |
| - | 114 | ST_CLERK |
| - | 122 | ST_CLERK |
| - | 176 | SA_MAN |
| - | 176 | SA_REP |
| - | 200 | AC_ACCOUNT |
| - | 200 | AD_ASST |
| - | 201 | MK_REP |

Set Operator Examples

- The keyword NULL can be used to match columns in a SELECT list.
- One NULL is included for each missing column.
- Furthermore, NULL is formatted to match the data type of the column it is standing in for, so TO_CHAR, TO_DATE, or TO_NUMBER functions are used to achieve identical SELECT lists.

SET Operations ORDER BY

- If you want to control the order of the returned rows when using SET operators in your query, the ORDER BY statement must only be used once, in the last SELECT statement in the query.
- Using the previous query example, we could ORDER BY employee_id to see the jobs each employee has held.

```
SELECT hire_date, employee_id, job_id
FROM employees
UNION
SELECT TO_DATE(NULL), employee_id, job_id
FROM job_history
ORDER BY employee_id;
```

SET Operations ORDER BY

```
SELECT hire_date, employee_id, job_id
FROM employees
UNION
SELECT TO_DATE(NULL), employee_id, job_id
FROM job_history
ORDER BY employee_id;
```

| HIRE_DATE | EMPLOYEE_ID | JOB_ID |
|-------------|-------------|------------|
| 17-Jun-1987 | 100 | AD_PRES |
| 21-Sep-1989 | 101 | AD_VP |
| - | 101 | AC_ACCOUNT |
| - | 101 | AC_MGR |
| 13-Jan-1993 | 102 | AD_VP |
| - | 102 | IT_PROG |
| 03-Jan-1990 | 103 | IT_PROG |
| 21-May-1991 | 104 | IT_PROG |
| 07-Feb-1999 | 107 | IT_PROG |
| - | 114 | ST_CLERK |
| ... | ... | ... |

SET Operations ORDER BY

- We could improve the readability of the output, by including the start date and end date columns from the job history table, to do this, we would need to match the columns in both queries by adding two more TO_DATE(NULL) columns to the first query.

```
SELECT hire_date, employee_id, TO_DATE(null) start_date,  
       TO_DATE(null) end_date, job_id, department_id  
FROM   employees  
UNION  
SELECT TO_DATE(null), employee_id, start_date, end_date, job_id,  
       department_id  
FROM   job_history  
ORDER BY employee_id;
```

SET Operations ORDER BY

| HIRE_DATE | EMPLOYEE_ID | START_DATE | END_DATE | JOB_ID | DEPARTMENT_ID |
|-------------|-------------|-------------|-------------|------------|---------------|
| 17-Jun-1987 | 100 | - | - | AD_PRES | 90 |
| 21-Sep-1989 | 101 | - | - | AD_VP | 90 |
| - | 101 | 21-Sep-1989 | 27-Oct-1993 | AC_ACCOUNT | 110 |
| - | 101 | 28-Oct-1993 | 15-Mar-1997 | AC_MGR | 110 |
| 13-Jan-1993 | 102 | - | - | AD_VP | 90 |
| - | 102 | 13-Jan-1993 | 24-Jul-1998 | IT_PROG | 60 |
| 03-Jan-1990 | 103 | - | - | IT_PROG | 60 |
| 21-May-1991 | 104 | - | - | IT_PROG | 60 |
| 07-Feb-1999 | 107 | - | - | IT_PROG | 60 |
| - | 114 | 24-Mar-1998 | 31-Dec-1999 | ST_CLERK | 50 |
| - | 122 | 01-Jan-1999 | 31-Dec-1999 | ST_CLERK | 50 |
| 16-Nov-1999 | 124 | - | - | ST_MAN | 50 |
| 17-Oct-1995 | 141 | - | - | ST_CLERK | 50 |
| 29-Jan-1997 | 142 | - | - | ST_CLERK | 50 |
| 15-Mar-1998 | 143 | - | - | ST_CLERK | 50 |
| ... | ... | ... | ... | ... | ... |

Terminology

Key terms used in this lesson included:

- INTERSECT
- MINUS
- SET operators
- TO_CHAR(null) – matching the select list
- UNION
- UNION ALL

Summary

In this lesson, you should have learned how to:

- Define and explain the purpose of Set Operators
- Use a set operator to combine multiple queries into a single query
- Control the order of rows returned using set operators

