Database Programming

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?

Purpose (cont.)

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

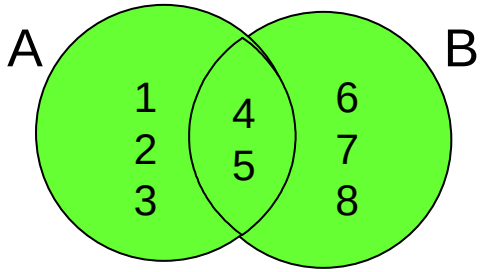
There are a few rules to remember when using SET operators:

- 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.

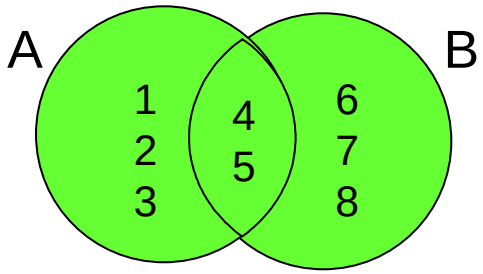
```
SELECT a_id
FROM   a
UNION
SELECT b_id
FROM   b;
```



The result of listing all elements in A and B eliminating duplicates is {1, 2, 3, 4, 5, 6, 7, 8}.

UNION (cont.)

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

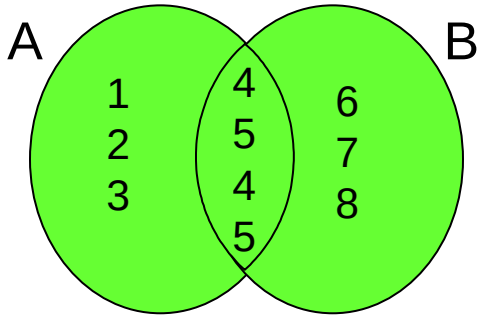


If you joined A and B you would get only {4, 5}. You would have to perform a full outer join to get the same list as above.

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;
```

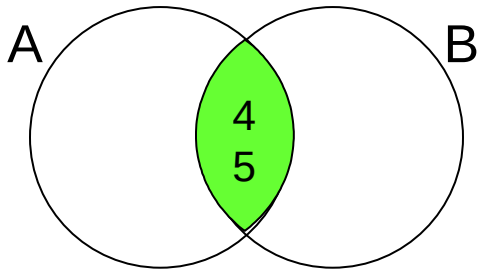


The result of listing all elements in A and B without eliminating duplicates is {1, 2, 3, 4, 5, 4, 5, 6, 7, 8}.

INTERSECT

The INTERSECT operator returns all rows common to both tables.

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

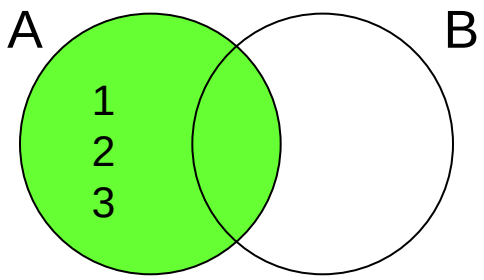


The result of listing all elements found in both A and B is {4, 5}.

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 datatype.

Set Operator Examples (cont.)

For example:

Table A contains a location id and a department name.

Table B contains a location id and a warehouse name.

The two tables have the location id in common, but one has department name and the other has warehouse name. You can use the TO_CHAR(NULL) function to create matching columns as shown below.

```
SELECT location_id, department_name "Department", TO_CHAR(NULL)
"Warehouse"
FROM departments
UNION
SELECT location_id, TO_CHAR(NULL) "Department", warehouse_name
FROM warehouses;
```

Set Operator Examples (cont.)

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 datatype 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 Operator Examples (cont.)

The WAREHOUSES table description:

Object Type **TABLE** Object **WAREHOUSES**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
WAREHOUSES	LOCATION_ID	Number	-	6	0	-	✓	-	-
	WAREHOUSE_NAME	Varchar2	50	-	-	-	✓	-	-

The WAREHOUSES table data:

LOCATION_ID	WAREHOUSE_NAME
1700	London
1800	Paris
2100	Copenhagen
4000	Shanghai
3000	Atlanta

Set Operator Examples (cont.)

```
Select location_id, department_name "Department", TO_CHAR(NULL)
"Warehouse"
FROM Departments
UNION
SELECT location_id, TO_CHAR(NULL) "Department", warehouse_name
FROM warehouses;
```

LOCATION_ID	Department	Warehouse
1400	IT	-
1500	Shipping	-
1700	Accounting	-
1700	Administration	-
1700	Contracting	-
1700	Executive	-
1700	-	London
1800	Marketing	-
1800	-	Paris
2100	-	Copenhagen
2500	Sales	-
3000	-	Atlanta
4000	-	Shanghai

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.

SET Operations ORDER BY (cont.)

Using the example on the previous slides, if you need to sort the rows returned by employee_id, you would need to add an ORDER BY clause.

```
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
```

HIRE_DATE	EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
17-JUN-87	100	-	-	AD_PRES	90
17-SEP-87	200	-	-	AD_ASST	10
21-SEP-89	101	-	-	AD_VP	90
03-JAN-90	103	-	-	IT_PROG	60
21-MAY-91	104	-	-	IT_PROG	60
13-JAN-93	102	-	-	AD_VP	90
07-JUN-94	205	-	-	AC_MGR	110
07-JUN-94	206	-	-	AC_ACCOUNT	110
17-OCT-95	141	-	-	ST_CLERK	50
16-NOV-99	124	-	-	ST_MAN	50
29-JAN-00	149	-	-	SA_MAN	80
-	101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
-	101	28-OCT-93	15-MAR-97	AC_MGR	110
-	102	13-JAN-93	24-JUL-98	IT_PROG	60
-	114	24-MAR-98	31-DEC-99	ST_CLERK	50
-	122	01-JAN-99	31-DEC-99	ST_CLERK	50
-	176	24-MAR-98	31-DEC-98	SA_REP	80
-	176	01-JAN-99	31-DEC-99	SA_MAN	80
-	200	17-SEP-87	17-JUN-93	AD_ASST	90
-	200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90
-	201	17-FEB-96	19-DEC-99	MK_REP	20

A partial query result is shown. Notice the rows from EMPLOYEES and JOB_HISTORY for the same employees are not displayed in any order. By default the first column in the first query is used to sort the rows.

SET Operations ORDER BY (cont.)

The same query ordered by employee_id gives the following result, and on this output you can see the entire employment history of one individual employee without scrolling up and down.

```
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
```

HIRE_DATE	EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
17-JUN-87	100	-	-	AD_PRES	90
21-SEP-89	101	-	-	AD_VP	90
-	101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
-	101	28-OCT-93	15-MAR-97	AC_MGR	110
13-JAN-93	102	-	-	AD_VP	90
-	102	13-JAN-93	24-JUL-98	IT_PROG	60
03-JAN-90	103	-	-	IT_PROG	60
21-MAY-91	104	-	-	IT_PROG	60
07-FEB-99	107	-	-	IT_PROG	60
-	114	24-MAR-98	31-DEC-99	ST_CLERK	50
-	122	01-JAN-99	31-DEC-99	ST_CLERK	50
16-NOV-99	124	-	-	ST_MAN	50
17-OCT-95	141	-	-	ST_CLERK	50
29-JAN-97	142	-	-	ST_CLERK	50
15-MAR-98	143	-	-	ST_CLERK	50
09-JUL-98	144	-	-	ST_CLERK	50
29-JAN-00	149	-	-	SA_MAN	80
11-MAY-96	174	-	-	SA_REP	80
24-MAR-98	176	-	-	SA_REP	80
-	176	24-MAR-98	31-DEC-98	SA_REP	80
-	176	01-JAN-99	31-DEC-99	SA_MAN	80

The ORDER BY used in SET operator queries can only be used in the LAST statement, and only columns from the first statement are valid in the ORDER BY clause.

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