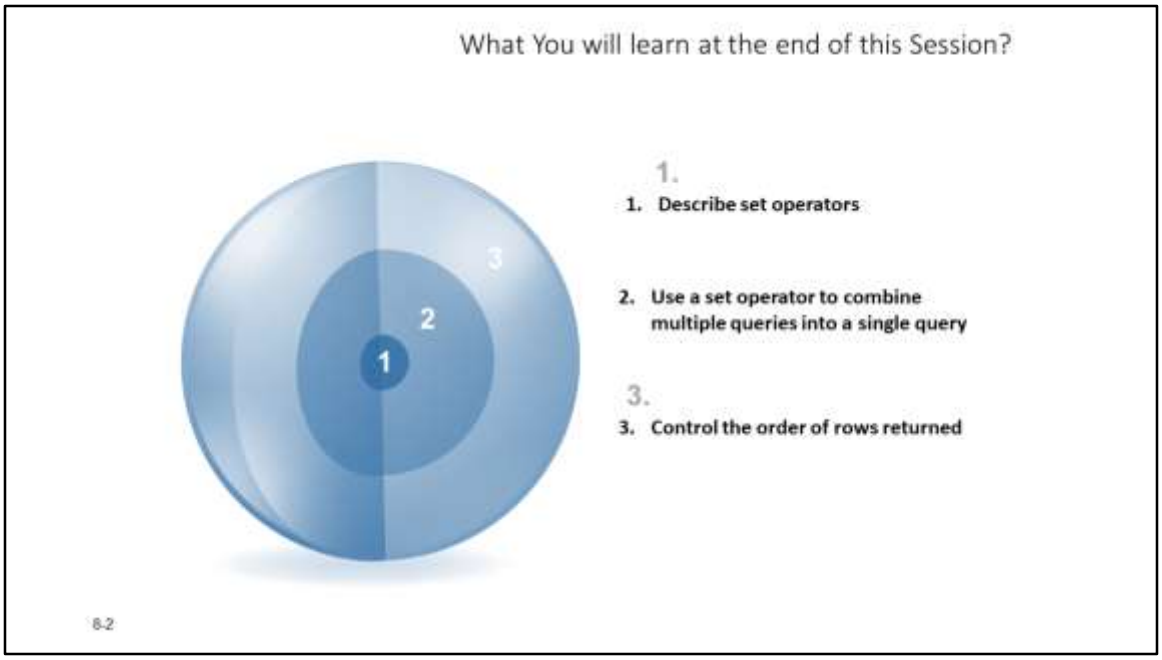
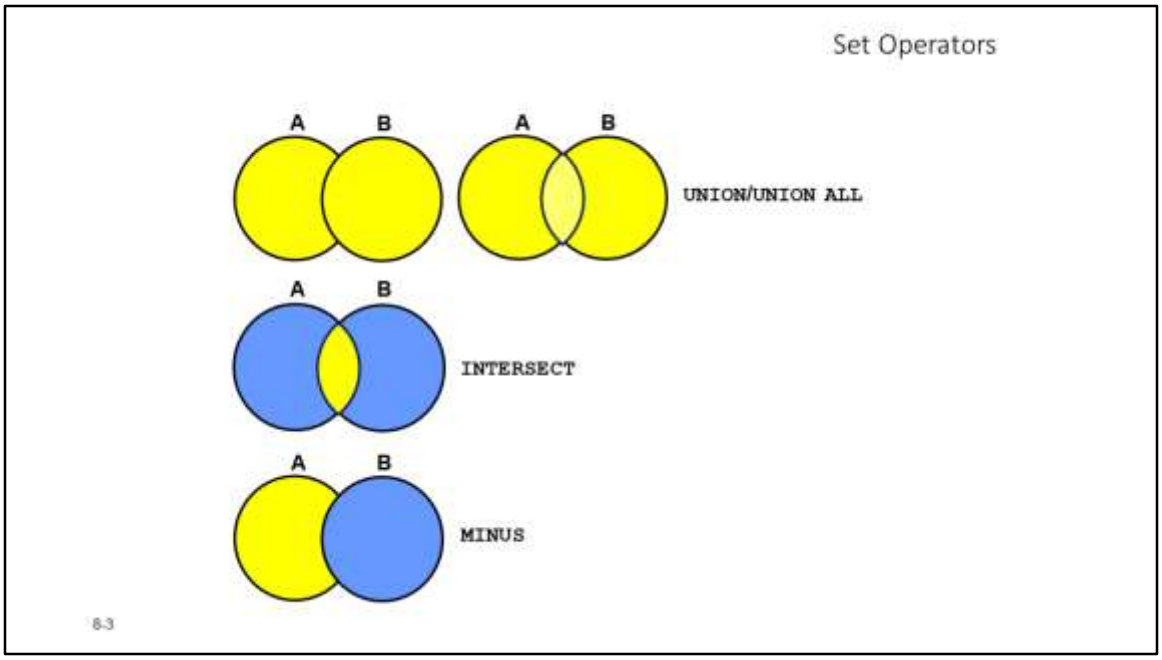
Using the Set Operators



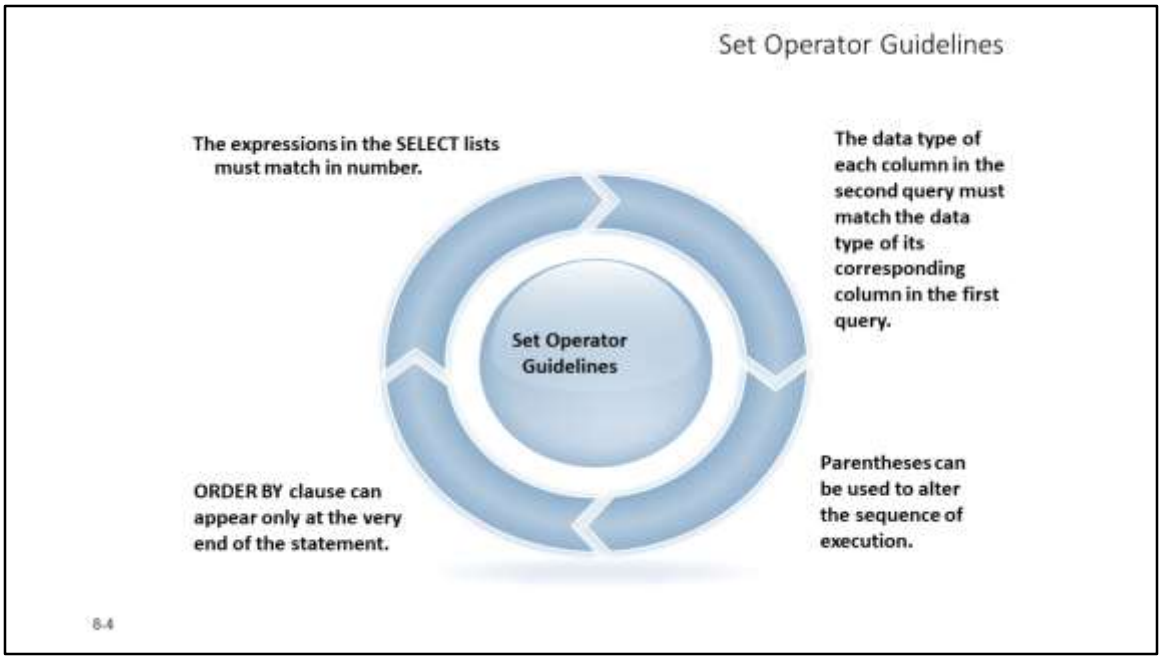
Objectives

In this lesson, you learn how to write queries by using set operators.



Set Operators :Set operators combine the results of two or more component queries into one result. Queries containing set operators are called *compound queries*. All set operators have equal precedence. If a SQL statement contains multiple set operators, the Oracle server evaluates them from left (top) to right (bottom)—if no parentheses explicitly specify another order. You should use parentheses to specify the order of evaluation explicitly in queries that use the `INTERSECT` operator with other set operators.

Operator	**Returns**
`UNION`	Rows from both queries after eliminating duplications
`UNION ALL`	Rows from both queries, including all duplications
`INTERSECT`	Rows that are common to both queries
`MINUS`	Rows in the first query that are not present in the second query

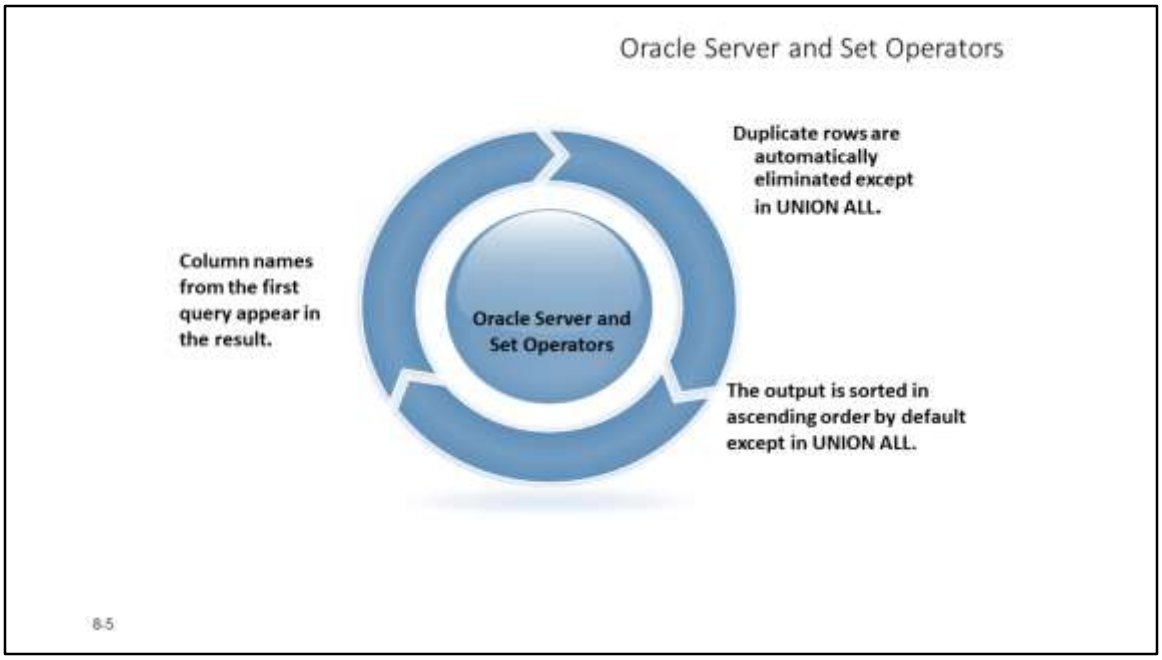


Set Operator Guidelines

The expressions in the `SELECT` lists of the queries must match in number and data type. Queries that use `UNION`, `UNION ALL`, `INTERSECT`, and `MINUS` operators in their `WHERE` clause must have the same number and data type of columns in their `SELECT` list. The data type of the columns in the `SELECT` list of the queries in the compound query may not be exactly the same. The column in the second query must be in the same data type group (such as numeric or character) as the corresponding column in the first query.

Set operators can be used in subqueries.

You should use parentheses to specify the order of evaluation in queries that use the `INTERSECT` operator with other set operators. This ensures compliance with emerging SQL standards that will give the `INTERSECT` operator greater precedence than the other set operators.



Oracle Server and Set Operators

When a query uses set operators, the Oracle server eliminates duplicate rows automatically except in the case of the `UNION ALL` operator. The column names in the output are decided by the column list in the first `SELECT` statement. By default, the output is sorted in ascending order of the first column of the `SELECT` clause.

The corresponding expressions in the `SELECT` lists of the component queries of a compound query must match in number and data type. If component queries select character data, the data type of the return values is determined as follows:

- If both queries select values of `CHAR` data type, of equal length, the returned values have the `CHAR` data type of that length. If the queries select values of `CHAR` with different lengths, the returned value is `VARCHAR2` with the length of the larger `CHAR` value.
- If either or both of the queries select values of `VARCHAR2` data type, the returned values have the `VARCHAR2` data type.

If component queries select numeric data, the data type of the return values is determined by numeric precedence. If all queries select values of the `NUMBER` type, the returned values have the `NUMBER` data type. In queries

using set operators, the Oracle server does not perform implicit conversion across data type groups. Therefore, if the corresponding expressions of component queries resolve to both character data and numeric data, the Oracle server returns an error.

Tables Used in This Lesson

•The tables used in this lesson are:

ORDERS: Provides details regarding all current orders.

ORDER_ITEMS: Records the details of the order in the form of product ID, price per unit, quantity ordered etc.

8-6

Tables Used in This Lesson

Two tables are used in this lesson: the ORDER table and the ORDER_ITEMS table.

The structure and data from the `EMPLOYEES` and `JOB_HISTORY` tables are shown on the following pages.

Tables Used in This Lesson (continued)

There have been instances in the company of people who have held the same position more than once during their tenure with the company. For example, consider the employee Taylor, who joined the company on 24-MAR-1998. Taylor held the job title `SA_REP` for the period 24-MAR-98 to 31-DEC-98 and the job title `SA_MAN` for the period 01-JAN-99 to 31-DEC-99. Taylor moved back into the job title of `SA_REP`, which is his current job title.

```
DESCRIBE employees
Name                          Null     Type
----------------------------- -------- --------------
EMPLOYEE_ID                   NOT NULL NUMBER(6)
FIRST_NAME                             VARCHAR2(20)
LAST_NAME                     NOT NULL VARCHAR2(25)
EMAIL                         NOT NULL VARCHAR2(25)
PHONE_NUMBER                           VARCHAR2(20)
HIRE_DATE                     NOT NULL DATE
JOB_ID                        NOT NULL VARCHAR2(10)
SALARY                                 NUMBER(8,2)
COMMISSION_PCT                         NUMBER(2,2)
MANAGER_ID                             NUMBER(6)
DEPARTMENT_ID                          NUMBER(4)

11 rows selected
```

Tables Used in This Lesson (continued)

```
SELECT employee_id, last_name, job_id, hire_date,
department_id
FROM employees;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
1	200	Whalen	AD_ASST	17-SEP-87	10
2	201	Hartstein	MK_MAN	17-FEB-96	20
3	202	Fay	MK_REP	17-AUG-97	20
4	205	Higgins	AC_MGR	07-JUN-94	110
5	206	Gietz	AC_ACCOUNT	07-JUN-94	110
6	100	King	AD_PRES	17-JUN-87	90
7	101	Kochhar	AD_VP	21-SEP-89	90
8	102	De Haan	AD_VP	13-JAN-93	90
9	103	Hunold	IT_PROG	03-JAN-90	60
10	104	Ernst	IT_PROG	21-MAY-91	60
11	107	Lorentz	IT_PROG	07-FEB-99	60
12	124	Mourgos	ST_MAN	16-NOV-99	50
13	141	Rajs	ST_CLERK	17-OCT-95	50
14	142	Davies	ST_CLERK	29-JAN-97	50
15	143	Matos	ST_CLERK	15-MAR-98	50
16	144	Vargas	ST_CLERK	09-JUL-98	50
17	149	Zlotkey	SA_MAN	29-JAN-00	80
18	174	Abel	SA_REP	11-MAY-96	80
19	176	Taylor	SA_REP	24-MAR-98	80
20	178	Grant	SA_REP	24-MAY-99	(null)

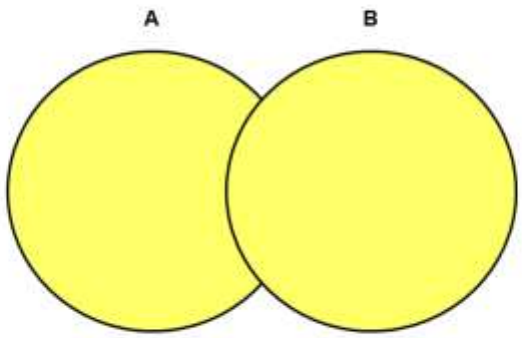
```
DESCRIBE job_history
Name                          Null     Type
----------------------------- -------- --------------
EMPLOYEE_ID                   NOT NULL NUMBER(6)
START_DATE                    NOT NULL DATE
END_DATE                      NOT NULL DATE
JOB_ID                        NOT NULL VARCHAR2(10)
DEPARTMENT_ID                          NUMBER(4)

5 rows selected
```

Tables Used in This Lesson (continued)

```
SELECT * FROM job_history;
```

	EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
1	102	13-JAN-93	24-JUL-98	IT_PROG	60
2	101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
3	101	28-OCT-93	15-MAR-97	AC_MGR	110
4	201	17-FEB-96	19-DEC-99	MK_REP	20
5	114	24-MAR-98	31-DEC-99	ST_CLERK	50
6	122	01-JAN-99	31-DEC-99	ST_CLERK	50
7	200	17-SEP-87	17-JUN-93	AD_ASST	90
8	176	24-MAR-98	31-DEC-98	SA_REP	80
9	176	01-JAN-99	31-DEC-99	SA_MAN	80
10	200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

UNION Operator



The UNION operator returns rows from both queries after eliminating duplications.

8-10

UNION Operator

The UNION operator returns all rows that are selected by either query. Use the UNION operator to return all rows from multiple tables and eliminate any duplicate rows.

Guidelines

- The number of columns being selected must be the same.
- The data types of the columns being selected must be in the same data type group (such as numeric or character).
- The names of the columns need not be identical.
- UNION operates over all of the columns being selected.
- NULL values are not ignored during duplicate checking.
- By default, the output is sorted in ascending order of the columns of the SELECT clause.

Using the UNION Operator

```
SELECT order_id
FROM orders
UNION
SELECT order_id
FROM order_items ORDER BY order_id;
```

	ORDER_ID
1	2354
2	2355
3	2356
4	2357
5	2358

...

73	2426
74	2427

...

8-11

Using the UNION Operator

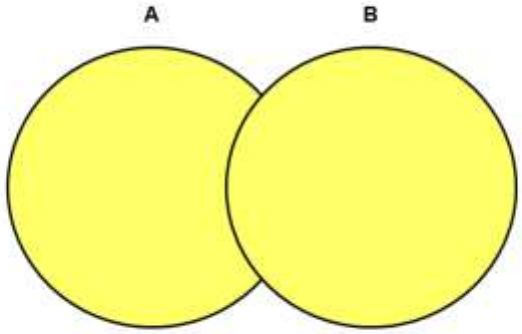
The UNION operator eliminates any duplicate records. If records that occur in both the ORDERS and the ORDERS_ITEMS tables are identical, the records are displayed only once.

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100	AD_PRES	90
...			
22	200	AC_ACCOUNT	90
23	200	AD_ASST	10
24	200	AD_ASST	90
...			
29	206	AC_ACCOUNT	110

Using the UNION Operator (continued)

In the preceding output, employee 200 appears three times. Why? Note the DEPARTMENT_ID values for employee 200. One row has a DEPARTMENT_ID of 90, another 10, and the third 90. Because of these unique combinations of job IDs and department IDs, each row for employee 200 is unique and, therefore, not considered to be a duplicate. Observe that the output is sorted in ascending order of the first column of the SELECT clause (in this case, EMPLOYEE_ID).

UNION ALL Operator



The UNION ALL operator returns rows from both queries, including all duplications.

8-13

UNION ALL Operator

Use the UNION ALL operator to return all rows from multiple queries.

Guidelines

The guidelines for UNION and UNION ALL are the same, with the following two exceptions that pertain to UNION ALL: Unlike UNION, duplicate rows are not eliminated and the output is not sorted by default.

Using the UNION ALL Operator

•Display the current and previous departments of all employees.

```
SELECT   order_id
FROM   orders
UNION ALL
SELECT   order_id
FROM   order_items
ORDER BY   order_id
```

	ORDER_ID
1	2354
2	2354
3	2354
4	2354
5	2354
...	
15	2355
16	2355
17	2355
18	2355
19	2355
20	2355

8-14

Using the UNION ALL Operator

In the example, 708 rows are fetched. Notice that all the duplicates are shown. Where as in the previous result, just 105 rows are fetched as duplicate rows were not shown.

176	SA_REP	80