

8

Using the Set Operators

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe set operators
- Use a set operator to combine multiple queries into a single query
- Control the order of rows returned

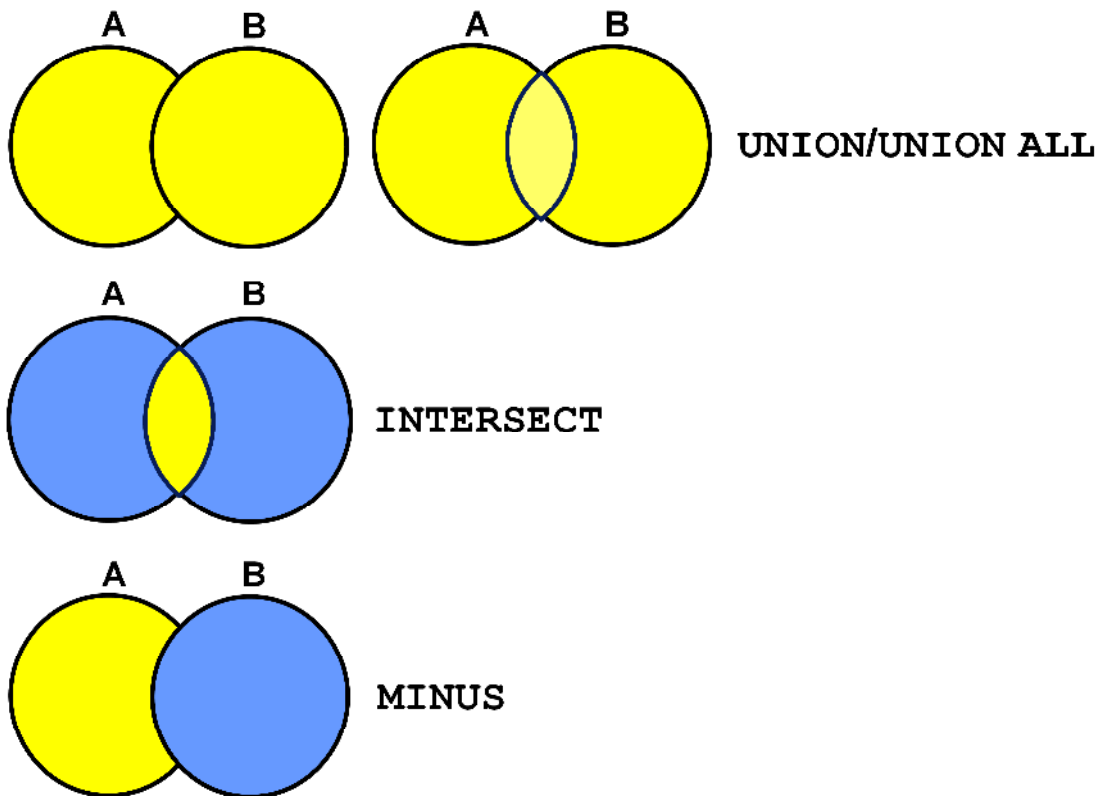
Objectives

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

Lesson Agenda

- **Set Operators: Types and guidelines**
- Tables used in this lesson
- UNION and UNION ALL operator
- INTERSECT operator
- MINUS operator
- Matching the SELECT statements
- Using the ORDER BY clause in set operations

Set Operators



8 - 4

Copyright © 2007, Oracle. All rights reserved.

ORACLE

Set Operators

Set operators combine the results of two or more component queries into one result. Queries containing set operators are called *compound queries*.

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

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.

Set Operator Guidelines

- The expressions in the `SELECT` lists must match in number.
- The data type of each column in the second query must match the data type of its corresponding column in the first query.
- Parentheses can be used to alter the sequence of execution.
- `ORDER BY` clause can appear only at the very end of the statement.

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 `SELECT` list of the queries in the compound query may not be exactly the same. The column in 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.

The Oracle Server and Set Operators

- Duplicate rows are automatically eliminated except in `UNION ALL`.
- Column names from the first query appear in the result.
- The output is sorted in ascending order by default except in `UNION ALL`.

The 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, then the returned values have the `CHAR` data type of that length. If the queries select values of `CHAR` with different lengths, then 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, then the returned values have the `VARCHAR2` data type.

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

Lesson Agenda

- Set Operators: Types and guidelines
- **Tables used in this lesson**
- UNION and UNION ALL operator
- INTERSECT operator
- MINUS operator
- Matching the SELECT statements
- Using the ORDER BY clause in set operations

Tables Used in This Lesson

The tables used in this lesson are:

- **EMPLOYEES:** Provides details regarding all current employees
- **JOB_HISTORY:** Records the details of the start date and end date of the former job, and the job identification number and department when an employee switches jobs

ORACLE

8 - 8

Copyright © 2007, Oracle. All rights reserved.

Tables Used in This Lesson

Two tables are used in this lesson. They are the **EMPLOYEES** table and the **JOB_HISTORY** table. You are already familiar with the **EMPLOYEES** table that stores employee details such as a unique identification number, email address, job identification (such as **ST_CLERK**, **SA_REP**, and so on), salary, manager and so on.

Some of the employees have been with the company for a long time and have switched to different jobs. This is monitored using the **JOB_HISTORY** table. When an employee switches jobs, the details of the start date and end date of the former job, the **job_id** (such as **ST_CLERK**, **SA_REP**, and so on), and the department are recorded in the **JOB_HISTORY** 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
```

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)

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




```
DESCRIBE job_history
```

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)

Tables Used in This Lesson (continued)

```
SELECT * FROM job_history;
```

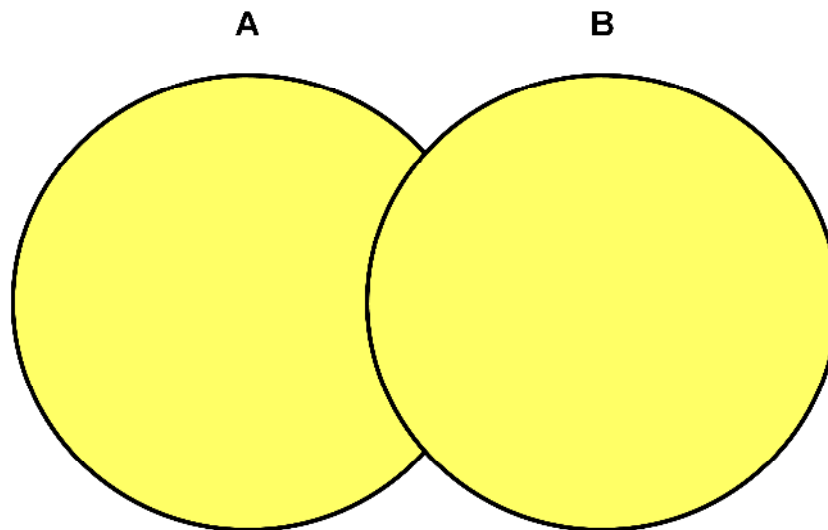
	 EMPLOYEE_ID	START_DATE	END_DATE	 JOB_ID	 DEPARTMENT_ID
1	102	13-JAN-93	24-JUL-98	IT_PROG	60
2	101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
3	101	28-OCT-93	15-MAR-97	AC_MGR	110
4	201	17-FEB-96	19-DEC-99	MK_REP	20
5	114	24-MAR-98	31-DEC-99	ST_CLERK	50
6	122	01-JAN-99	31-DEC-99	ST_CLERK	50
7	200	17-SEP-87	17-JUN-93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

Lesson Agenda

- Set Operators: Types and guidelines
- Tables used in this lesson
- **UNION and UNION ALL operator**
- INTERSECT operator
- MINUS operator
- Matching the SELECT statements
- Using the ORDER BY clause in set operations

ORACLE

UNION Operator



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

ORACLE

8 - 13

Copyright © 2007, Oracle. All rights reserved.

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

Display the current and previous job details of all employees.
Display each employee only once.

```
SELECT employee_id, job_id
FROM employees
UNION
SELECT employee_id, job_id
FROM job_history;
```

EMPLOYEE_ID	JOB_ID
1	100 AD_PRES
2	101 AC_ACCOUNT
...	
22	200 AC_ACCOUNT
23	200 AD_ASST
24	201 MK_MAN
...	

ORACLE

8 - 14

Copyright © 2007, Oracle. All rights reserved.

Using the UNION Operator

The UNION operator eliminates any duplicate records. If records that occur in both the EMPLOYEES and the JOB_HISTORY tables are identical, the records are displayed only once. Observe in the output shown in the slide that the record for the employee with the EMPLOYEE_ID 200 appears twice because the JOB_ID is different in each row.

Consider the following example:

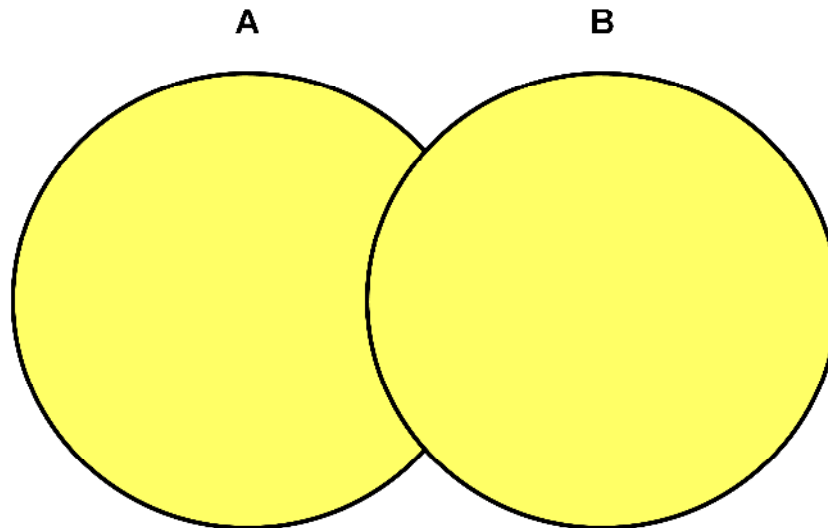
```
SELECT employee_id, job_id, department_id
FROM employees
UNION
SELECT employee_id, job_id, department_id
FROM job_history;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100 AD_PRES	90
2	101 AC_ACCOUNT	110
...		
22	200 AC_ACCOUNT	90
23	200 AD_ASST	10
24	200 AD_ASST	90
...		

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.

ORACLE

8 - 16

Copyright © 2007, Oracle. All rights reserved.

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 employee_id, job_id, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100 AD_PRES	90
...		
16	144 ST_CLERK	50
17	149 SA_MAN	80
18	174 SA_REP	80
19	176 SA_REP	80
20	176 SA_MAN	80
21	176 SA_REP	80
22	178 SA_REP	(null)
...		
30	206 AC_ACCOUNT	110

ORACLE

8 - 17

Copyright © 2007, Oracle. All rights reserved.

Using the UNION ALL Operator

In the example, 30 rows are selected. The combination of the two tables totals to 30 rows. The UNION ALL operator does not eliminate duplicate rows. UNION returns all distinct rows selected by either query. UNION ALL returns all rows selected by either query, including all duplicates.

Consider the query in the slide, now written with the UNION clause:

```
SELECT employee_id, job_id, department_id
FROM employees
UNION
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

The preceding query returns 29 rows. This is because it eliminates the following row (because it is a duplicate):

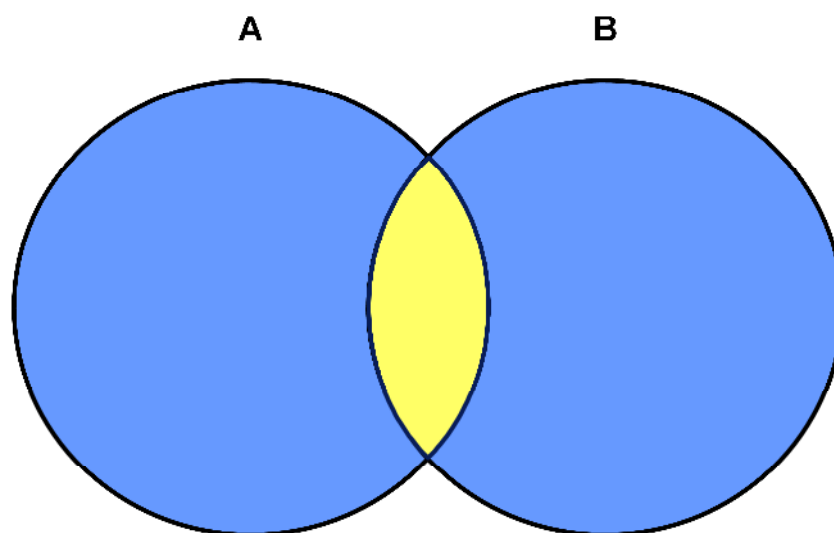
176 SA_REP	80
------------	----

Lesson Agenda

- Set Operators: Types and guidelines
- Tables used in this lesson
- UNION and UNION ALL operator
- INTERSECT operator
- MINUS operator
- Matching the SELECT statements
- Using ORDER BY clause in set operations

ORACLE

INTERSECT Operator



The **INTERSECT** operator returns rows that are common to both queries.

ORACLE

8 - 19

Copyright © 2007, Oracle. All rights reserved.

INTERSECT Operator

Use the **INTERSECT** operator to return all rows that are common to multiple queries.

Guidelines

- The number of columns and the data types of the columns being selected by the **SELECT** statements in the queries must be identical in all the **SELECT** statements used in the query. The names of the columns, however, need not be identical.
- Reversing the order of the intersected tables does not alter the result.
- **INTERSECT** does not ignore **NULL** values.

Using the INTERSECT Operator

Display the employee IDs and job IDs of those employees who currently have a job title that is the same as their previous one (that is, they changed jobs but have now gone back to doing the same job they did previously).

```
SELECT employee_id, job_id
FROM   employees
INTERSECT
SELECT employee_id, job_id
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

ORACLE

8 - 20

Copyright © 2007, Oracle. All rights reserved.

Using the INTERSECT Operator

In the example in this slide, the query returns only those records that have the same values in the selected columns in both tables.

What will be the results if you add the DEPARTMENT_ID column to the SELECT statement from the EMPLOYEES table and add the DEPARTMENT_ID column to the SELECT statement from the JOB_HISTORY table, and run this query? The results may be different because of the introduction of another column whose values may or may not be duplicates.

Example:

```
SELECT employee_id, job_id, department_id
FROM   employees
INTERSECT
SELECT employee_id, job_id, department_id
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	176	SA_REP	80

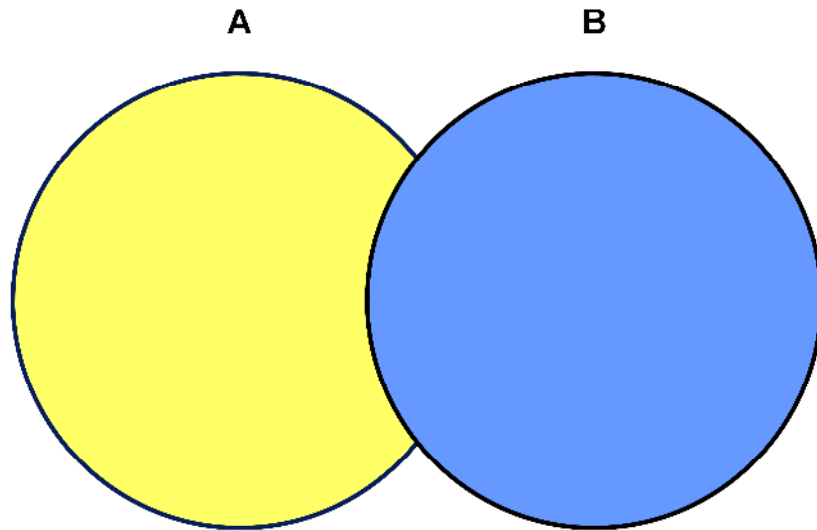
Employee 200 is no longer part of the results because the EMPLOYEES.DEPARTMENT_ID value is different from the JOB_HISTORY.DEPARTMENT_ID value.

Lesson Agenda

- Set Operators: Types and guidelines
- Tables used in this lesson
- UNION and UNION ALL operator
- INTERSECT operator
- **MINUS operator**
- Matching the SELECT statements
- Using the ORDER BY clause in set operations

ORACLE

MINUS Operator



The **MINUS** operator returns all the distinct rows selected by the first query, but not present in the second query result set.

ORACLE

8 - 22

Copyright © 2007, Oracle. All rights reserved.

MINUS Operator

Use the **MINUS** operator to return all distinct rows selected by the first query, but not present in the second query result set (the first **SELECT** statement **MINUS** the second **SELECT** statement).

Note: The number of columns must be the same and the data types of the columns being selected by the **SELECT** statements in the queries must belong to the same data type group in all the **SELECT** statements used in the query. The names of the columns, however, need not be identical.

Using the MINUS Operator

Display the employee IDs of those employees who have not changed their jobs even once.

```
SELECT employee_id
FROM employees
MINUS
SELECT employee_id
FROM job_history;
```

	EMPLOYEE_ID
1	100
2	103
3	104
4	107
5	124

...

14	205
15	206

ORACLE

Using the MINUS Operator

In the example in the slide, the employee IDs in the JOB_HISTORY table are subtracted from those in the EMPLOYEES table. The results set displays the employees remaining after the subtraction; they are represented by rows that exist in the EMPLOYEES table, but do not exist in the JOB_HISTORY table. These are the records of the employees who have not changed their jobs even once.

Lesson Agenda

- Set Operators: Types and guidelines
- Tables used in this lesson
- UNION and UNION ALL operator
- INTERSECT operator
- MINUS operator
- **Matching the SELECT statements**
- Using ORDER BY clause in set operations

ORACLE

Matching the SELECT Statements

- Using the `UNION` operator, display the location ID, department name, and the state where it is located.
- You must match the data type (using the `TO_CHAR` function or any other conversion functions) when columns do not exist in one or the other table.

```
SELECT location_id, department_name "Department",  
       TO_CHAR(NULL) "Warehouse location"  
FROM departments  
UNION  
SELECT location_id, TO_CHAR(NULL) "Department",  
       state_province  
FROM locations;
```

ORACLE

8 - 25

Copyright © 2007, Oracle. All rights reserved.

Matching the SELECT Statements

Because the expressions in the `SELECT` lists of the queries must match in number, you can use the dummy columns and the data type conversion functions to comply with this rule. In the slide, the name, Warehouse location, is given as the dummy column heading. The `TO_CHAR` function is used in the first query to match the `VARCHAR2` data type of the `state_province` column that is retrieved by the second query. Similarly, the `TO_CHAR` function in the second query is used to match the `VARCHAR2` data type of the `department_name` column that is retrieved by the first query.

The output of the query is shown:

	LOCATION_ID	Department	Warehouse location
1	1400	IT	(null)
2	1400	(null)	Texas
3	1500	Shipping	(null)
4	1500	(null)	California
5	1700	Accounting	(null)
6	1700	Administration	(null)
7	1700	Contracting	(null)
8	1700	Executive	(null)
9	1700	(null)	Washington

■ ■ ■

Matching the SELECT Statement: Example

Using the `UNION` operator, display the employee ID, job ID, and salary of all employees.

```
SELECT employee_id, job_id, salary
FROM   employees
UNION
SELECT employee_id, job_id, 0
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID	SALARY
1	100	AD_PRES	24000
2	101	AC_ACCOUNT	0
3	101	AC_MGR	0
4	101	AD_VP	17000
5	102	AD_VP	17000
...			
29	205	AC_MGR	12000
30	206	AC_ACCOUNT	8300

ORACLE

Matching the SELECT Statement: Example

The `EMPLOYEES` and `JOB_HISTORY` tables have several columns in common (for example, `EMPLOYEE_ID`, `JOB_ID`, and `DEPARTMENT_ID`). But what if you want the query to display the employee ID, job ID, and salary using the `UNION` operator, knowing that the salary exists only in the `EMPLOYEES` table?

The code example in the slide matches the `EMPLOYEE_ID` and `JOB_ID` columns in the `EMPLOYEES` and `JOB_HISTORY` tables. A literal value of 0 is added to the `JOB_HISTORY` `SELECT` statement to match the numeric `SALARY` column in the `EMPLOYEES` `SELECT` statement.

In the results shown in the slide, each row in the output that corresponds to a record from the `JOB_HISTORY` table contains a 0 in the `SALARY` column.

Lesson Agenda

- Set Operators: Types and guidelines
- Tables used in this lesson
- UNION and UNION ALL operator
- INTERSECT operator
- MINUS operator
- Match the SELECT statements
- Using the ORDER BY clause in set operations

Using the ORDER BY Clause in Set Operations

- The ORDER BY clause can appear only once at the end of the compound query.
- Component queries cannot have individual ORDER BY clauses.
- ORDER BY clause recognizes only the columns of the first SELECT query.
- By default, the first column of the first SELECT query is used to sort the output in an ascending order.

ORACLE

8 - 28

Copyright © 2007, Oracle. All rights reserved.

Using the ORDER BY Clause in Set Operations

The ORDER BY clause can be used only once in a compound query. If used, the ORDER BY clause must be placed at the end of the query. The ORDER BY clause accepts the column name or an alias. By default, the output is sorted in ascending order in the first column of the first SELECT query.

Note: The ORDER BY clause does not recognize the column names of the second SELECT query. To avoid confusion over column names, it is a common practice to ORDER BY column positions.

For example, in the following statement, the output will be shown in ascending order of the job_id.

```
SELECT employee_id, job_id,salary
FROM   employees
UNION
SELECT employee_id, job_id,0
FROM   job_history
ORDER BY 2;
```

If you omit the ORDER BY, then by default the output will be sorted in the ascending order of employee_id. You cannot use the columns from the second query to sort the output.

Summary

In this lesson, you should have learned how to use:

- `UNION` to return all distinct rows
- `UNION ALL` to return all rows, including duplicates
- `INTERSECT` to return all rows that are shared by both queries
- `MINUS` to return all distinct rows that are selected by the first query, but not by the second
- `ORDER BY` only at the very end of the statement

ORACLE

8 - 29

Copyright © 2007, Oracle. All rights reserved.

Summary

- The `UNION` operator returns all the distinct rows selected by each query in the compound query. Use the `UNION` operator to return all rows from multiple tables and eliminate any duplicate rows.
- Use the `UNION ALL` operator to return all rows from multiple queries. Unlike the case with the `UNION` operator, duplicate rows are not eliminated and the output is not sorted by default.
- Use the `INTERSECT` operator to return all rows that are common to multiple queries.
- Use the `MINUS` operator to return rows returned by the first query that are not present in the second query.
- Remember to use the `ORDER BY` clause only at the very end of the compound statement.
- Make sure that the corresponding expressions in the `SELECT` lists match in number and data type.

Practice 8: Overview

In this practice, you create reports by using:

- The `UNION` operator
- The `INTERSECTION` operator
- The `MINUS` operator

Practice 8: Overview

In this practice, you write queries using the set operators.

Practice 8

1. The HR department needs a list of department IDs for departments that do not contain the job ID ST_CLERK. Use the set operators to create this report.

	DEPARTMENT_ID
1	10
2	20
3	60
4	80
5	90
6	110
7	190

2. The HR department needs a list of countries that have no departments located in them. Display the country ID and the name of the countries. Use the set operators to create this report.

	COUNTRY_ID	COUNTRY_NAME
1	DE	Germany

3. Produce a list of jobs for departments 10, 50, and 20, in that order. Display the job ID and department ID by using the set operators.

	JOB_ID	DEPARTMENT_ID
1	AD_ASST	10
2	ST_MAN	50
3	ST_CLERK	50
4	MK_MAN	20
5	MK_REP	20

4. Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (that is, they changed jobs but have now gone back to doing their original job).

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

Practice 8 (continued)

5. The HR department needs a report with the following specifications:

- Last name and department ID of all employees from the EMPLOYEES table, regardless of whether or not they belong to a department
- Department ID and department name of all departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them

Write a compound query to accomplish this.

	A Z	LAST_NAME	A Z	DEPARTMENT_ID	A Z	TO_CHAR(NULL)
1		Abel		80	(null)	
2		Davies		50	(null)	
3		De Haan		90	(null)	
4		Ernst		60	(null)	
5		Fay		20	(null)	
6		Gietz		110	(null)	
7		Grant		(null)	(null)	
8		Hartstein		20	(null)	
9		Higgins		110	(null)	
10		Hunold		60	(null)	
11		King		90	(null)	
12		Kochhar		90	(null)	
13		Lorentz		60	(null)	
14		Matos		50	(null)	
15		Mourgos		50	(null)	
16		Rajs		50	(null)	
17		Taylor		80	(null)	
18		Vargas		50	(null)	
19		Whalen		10	(null)	
20		Zlotkey		80	(null)	
21	(null)			10	Administration	
22	(null)			20	Marketing	
23	(null)			50	Shipping	
24	(null)			60	IT	
25	(null)			80	Sales	
26	(null)			90	Executive	
27	(null)			110	Accounting	
28	(null)			190	Contracting	