

### 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

8-2

## 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.

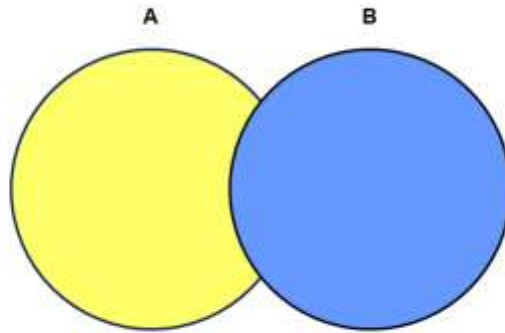
**Example:**

```
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

EMPLOYEES.DEPARTMENT\_ID value is different from the  
JOB\_HISTORY.DEPARTMENT\_ID value.

## MINUS Operator



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

8-3

### 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 order_id  
FROM orders  
MINUS  
SELECT order_id  
FROM order_items  
ORDER BY order_id;
```

	ORDER_ID
1	2380
2	2370
3	2373
4	2374
...	
56	2427
57	2425
58	2433
59	2441

8-4

### Using the MINUS Operator

In the example in the slide, the order\_id's in the ORDER\_ITEM table are subtracted from those in the ORDERS table. The results set displays the orders remaining after the subtraction; they are represented by rows that exist in the ORDERS table, but do not exist in the ORDER\_ITEMS table.

## 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;
```

8-5

## 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)

## Matching the SELECT Statement: Example

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

```
SELECT order_id, product_id  
FROM order_items  
UNION  
SELECT order_id, 0  
FROM orders  
ORDER BY order_id;
```

	ORDER ID	PRODUCT ID
1	2354	0
2	2354	3106
3	2354	3114
4	2354	3123
5	2354	3129
...		
50	2358	3181
57	2368	0
59	2361	0

8-6

## Matching the SELECT Statement: Example

The ORDERS and ORDER\_ITEMS tables have one column in common (for example, object\_id). But what if you want the query to display the order ID and the product ID, using the UNION operator, knowing that the product\_id exists only in the ORDER\_ITEMS table?

The code example in the slide matches the OBJECT\_ID and PRODUCT\_ID columns in the ORDERS and ORDER\_ITEMS tables. A literal value of 0 is added to the ORDERS table's SELECT statement to match the numeric PRODUCT\_ID column in the ORDER\_ITEM table's SELECT statement.

In the results shown in the slide, each row in the output that corresponds to a record from the ORDERS table contains a 0 in the PRODUCT\_ID column.

## 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.
- The 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.

8.7

### 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 `job_id`.

```
SELECT order_id, product_id
FROM   order_items
UNION
SELECT order_id, 0
FROM   orders
ORDER BY 2;
```

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



- Identify the set operator guidelines.
  - 1.The expressions in the SELECT lists must match in number.
  - 2.Parentheses may not be used to alter the sequence of execution.
  - 3.The data type of each column in the second query must match the data type of its corresponding column in the first query.
  - 4.The ORDER BY clause can be used only once in a compound query, unless a UNION ALL operator is used.

Answer: 1, 3

## Session Summary



1. **UNION** to return all distinct rows
2. **UNION ALL** to return all rows, including duplicates
3. **INTERSECT** to return all rows that are shared by both queries
4. **MINUS** to return all distinct rows that are selected by the first query, but not by the second
5. **ORDER BY** only at the very end of the statement

89

## Session 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:



8-10

## Practice 8: Overview

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