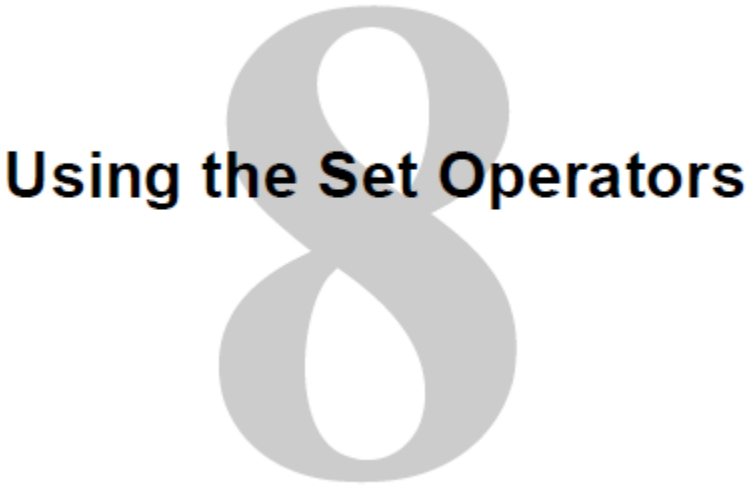
Les08-Set Operators



Objectives

1 Describe Set Operators

2 Use set operators to combine multiple queries into a single query

3 Control order of rows returned

Topics Covered

Set Operator Types and rules

Tables Used in the lesson
UNION

UNION ALL

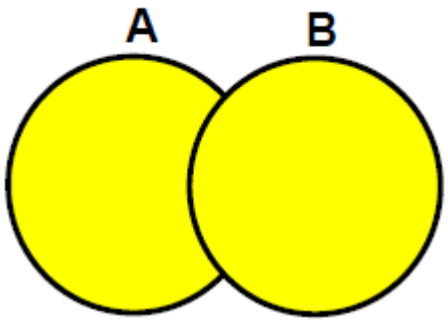
INTERSECT

MINUS

Matching SELECT statements
ORDER BY with set operators

Types - Union

UNION



UNION of all the rows in A
With ALL the rows in B
With NO DUPLICATES

RESULT is the Yellow- but duplicates not showing twice

EXAMPLE:

JOB_HISTORY ➔Table keeps history of when an employee changes jobs

Records start date and end date of employees that switch jobs
Employees who are still in the same job will not appear here
The current job is shown in the EMPLOYEE table. Again, this shows history.

SELECT * FROM job_history;

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

10 rows selected
```

EMPLOYEE table contains employee information.

This example, only show the employee_id, job_id, department_id ➔ the common attributes.

SELECT employee_id, job_id, department_id
FROM employees;

```
EMPLOYEE_ID JOB_ID     DEPARTMENT_ID
----------- ---------- -------------
        100 AD_PRES                90
        101 AD_VP                  90
        102 AD_VP                  90
        103 IT_PROG                60
        104 IT_PROG                60
        107 IT_PROG                60
        124 ST_MAN                 50
        141 ST_CLERK               50
        142 ST_CLERK               50
        143 ST_CLERK               50
        144 ST_CLERK               50
        149 SA_MAN                 80
        174 SA_REP                 80
        176 SA_REP                 80
        178 SA_REP
        200 AD_ASST                10
        201 MK_MAN                 20
        202 MK_REP                 20
        205 AC_MGR                110
        206 AC_ACCOUNT            110
20 rows selected.
```

Result of the UNION of both tables using just employee_id and job_id

```
SELECT employee_id, job_id
FROM   employees
UNION
SELECT employee_id, job_id
FROM   job_history
ORDER BY job_id;     ← added order by for readability
```

```
EMPLOYEE_ID       JOB_ID
----------------------- ----------
100               AD_PRES
101               AC_ACCOUNT
101               AC_MGR
101               AD_VP
102               AD_VP
102               IT_PROG
103               IT_PROG
104               IT_PROG
107               IT_PROG
114               ST_CLERK
122               ST_CLERK
124               ST_MAN
141               ST_CLERK
142               ST_CLERK
143               ST_CLERK
144               ST_CLERK
149               SA_MAN
174               SA_REP
176               SA_MAN
176               SA_REP
178               SA_REP
200               AC_ACCOUNT
200               AD_ASST
201               MK_MAN
201               MK_REP
202               MK_REP
205               AC_MGR
206               AC_ACCOUNT
```

28 rows selected ← Notice the 28 rows

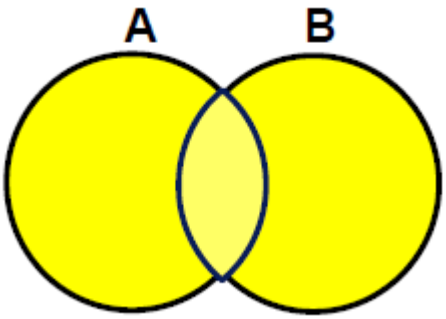
```
Employees       20
Job_history     10
TOTAL           30 rows
```

Since only produced 28 rows, then 2 rows must be duplicates and not shown.

WHERE ARE THE DUPLICATES ?

Types – Union All

8-4



UNION of ALL the rows in A and B including duplicates

RESULT is all the Yellow and duplicates showing twice

ENTER the code using the same 2 tables to see the result.

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

```
EMPLOYEE_ID          JOB_ID
-------------------- ----------
100                  AD_PRES
101                  AD_VP
101                  AC_ACCOUNT
101                  AC_MGR
102                  IT_PROG
102                  AD_VP
103                  IT_PROG
104                  IT_PROG
107                  IT_PROG
114                  ST_CLERK
122                  ST_CLERK
124                  ST_MAN
141                  ST_CLERK
142                  ST_CLERK
143                  ST_CLERK
144                  ST_CLERK
149                  SA_MAN
174                  SA_REP
176                  SA_REP     Was a Sales Representative
176                  SA_MAN     Became a Sales Manager
176                  SA_REP     Went back to a Sales Rep
178                  SA_REP
200                  AD_ASST    Looks like the same here
200                  AD_ASST
200                  AC_ACCOUNT
201                  MK_REP
201                  MK_MAN
202                  MK_REP
205                  AC_MGR
206                  AC_ACCOUNT
30 rows selected
```

Change the code and add in DEPARTMENT_ID

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

What was the result?

How many duplicates, if any?

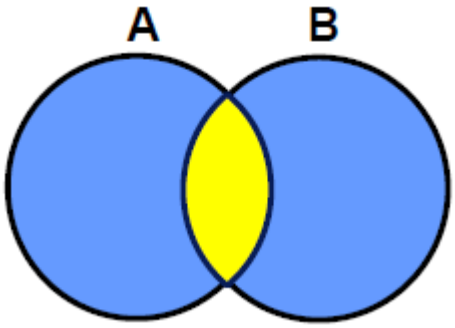
Why?

ANSWER:

Look at the former duplicate 200 – notice different department_id

Types – Intersect

INTERSECT



The rows in common to both tables only

A intersect B
same as
B intersect A

RESULT is the Yellow

Change the previous SQL to an INTERSECT

- finds the common rows (or duplicates)

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

```
EMPLOYEE_ID JOB_ID     DEPARTMENT_ID
----------- ---------- -------------
        176 SA_REP                80
```

Do the same but without department_id

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

```
EMPLOYEE_ID JOB_ID
----------- ----------
        176 SA_REP
        200 AD_ASST
```

What did this tell you?

SELECT * FROM JOB_HISTORY;

```
EMPLOYEE_ID START_DAT END_DATE  JOB_ID     DEPARTMENT_ID
----------- --------- --------- ---------- -------------
        102 13-JAN-93 24-JUL-98 IT_PROG               60
        101 21-SEP-89 27-OCT-93 AC_ACCOUNT           110
        101 28-OCT-93 15-MAR-97 AC_MGR               110
        201 17-FEB-96 19-DEC-99 MK_REP                20
        114 24-MAR-98 31-DEC-99 ST_CLERK              50
        122 01-JAN-99 31-DEC-99 ST_CLERK              50
        200 17-SEP-87 17-JUN-93 AD_ASST               90
        176 24-MAR-98 31-DEC-98 SA_REP                80
        176 01-JAN-99 31-DEC-99 SA_MAN                80
        200 01-JUL-94 31-DEC-98 AC_ACCOUNT            90
10 rows selected.
```

TITLES and ORDER BY

```
SELECT     employee_id as "Emp#",  job_id as "Job Title"
FROM       employees
UNION ALL
SELECT     employee_id, job_id
FROM       job_history
ORDER BY 1, 2
```

```
      Emp# Job Title
---------- ----------
       100 AD_PRES
       101 AC_ACCOUNT
       101 AC_MGR  …. Etc for 30 rows
```

What if use 3 columns in table 1 and 2 in table 2?

```
SELECT     employee_id as "Emp#",  job_id as "Job Title", department_id
FROM       employees
UNION ALL
SELECT     employee_id, job_id
FROM       job_history
ORDER BY 1, 2
```

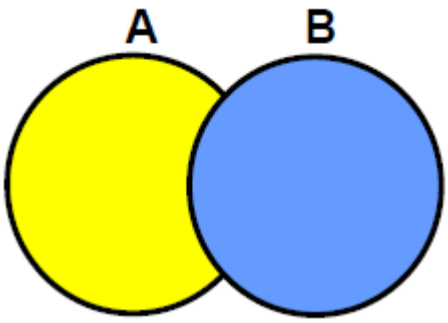
Can't make the comparison properly

SQL Error: ORA-01789: query block has incorrect number of result columns
01789. 00000 - "**query block has incorrect number of result columns**"

Types – Minus

8-4

MINUS



Rows in the first query A
That are not in second query B

RESULT is the Yellow

PRECEDENCE – equal – evaluated left to right

Caution recommended. Use brackets with INTERSECT

Alternate way of saying it is ➔ Those rows that are unique to the first query

```
SELECT employee_id, job_id
FROM   employees
MINUS
SELECT employee_id, job_id
FROM   job_history
ORDER BY 1, 2
```

Predict how many rows?

Table A or employees has 20-0 ➔ means all 20 rows with no duplicates
The intersect of A and B is 2 rows duplicated ➔ result so far is 20 – 0 – 2 = 18

A bit more

Give a list of department_id, location_id, hire_date.

That requires 2 tables, EMPLOYEES and DEPARTMENT

Using a JOIN

```
SELECT E.department_id, location_id, hire_date
FROM   employees E, departments D
WHERE       E.department_id = D.department_id
```

```
DEPARTMENT_ID LOCATION_ID HIRE_DATE
------------- ----------- ---------
           10        1700 17-SEP-87
           20        1800 17-FEB-96
           20        1800 17-AUG-97
           50        1500 16-NOV-99
           50        1500 17-OCT-95
           50        1500 29-JAN-97
           50        1500 15-MAR-98
           50        1500 09-JUL-98
           60        1400 03-JAN-90
           60        1400 21-MAY-91
           60        1400 07-FEB-99
           80        2500 29-JAN-00
           80        2500 11-MAY-96
           80        2500 24-MAR-98
           90        1700 17-JUN-87
           90        1700 21-SEP-89
           90        1700 13-JAN-93
          110        1700 07-JUN-94
          110        1700 07-JUN-94
```

19 rows selected

SAME EXAMPLE but using UNION

Display department ID, location ID and hire date for all members

To use SET operators you need the same number of columns

Need 3 columns in employees

Need same 3 columns in departments

PROBLEM:

- ⇨ Need hire_date from employees but it doesn't have a location_id in employees
- ⇨ Need location_id from departments but it doesn't have a date to match with

SOLUTION

Because the expressions in the SELECT lists of the queries must match in number,
- use the dummy columns and the data type conversion functions to comply with this rule.

You must match the data type when columns do not exist in one or the other table
- use the TO_CHAR or any other conversion function to get the same data type

```
SELECT department_id, TO_NUMBER (null) as location, hire_date
FROM   employees
UNION
SELECT department_id, location_id, TO_DATE (null)
FROM   departments;

DEPARTMENT_ID   LOCATION HIRE_DATE
------------- ---------- ---------
           10       1700
           10            17-SEP-87
           20       1800
           20            17-FEB-96
           20            17-AUG-97
           50       1500
           50            17-OCT-95
           50            29-JAN-97
           50            15-MAR-98
           50            09-JUL-98
           50            16-NOV-99
           60       1400
           60            03-JAN-90
           60            21-MAY-91
           60            07-FEB-99
           80       2500
           80            11-MAY-96
           80            24-MAR-98
           80            29-JAN-00
           90       1700
           90            17-JUN-87
           90            21-SEP-89
           90            13-JAN-93
          110       1700
          110            07-JUN-94
          190       1700
                        24-MAY-99

27 rows selected.
```

Note the location because TO_NUMBER (null) does not make a good column heading

From a diffeent example. Notice that the blanks are nulls

	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 SELECT statements

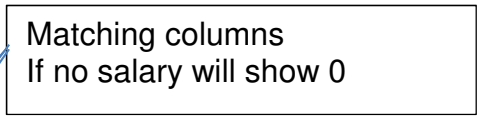
8-25

EXAMPLE 1:
Display all employees their job id and salary.

What are the problems?
Employees have several jobs and to display all the jobs requires a join to the job_history table
But … the job_history table does not have salary

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

EMPLOYEE_ID JOB_ID         SALARY
----------- ---------- ----------
        100 AD_PRES         24000
        101 AC_ACCOUNT          0
        101 AC_MGR              0
        101 AD_VP           17000
        102 AD_VP           17000
        102 IT_PROG             0   ← means, no record of salary of previous job
        103 IT_PROG          9000
        104 IT_PROG          6000
        107 IT_PROG          4200
        114 ST_CLERK            0   ← what does this mean?
        122 ST_CLERK            0
        124 ST_MAN           5800
        141 ST_CLERK         3500
        142 ST_CLERK         3100
        143 ST_CLERK         2600
        144 ST_CLERK         2500
        149 SA_MAN          10500
        174 SA_REP          11000
        176 SA_MAN              0   ← means, no record of salary of previous job
        176 SA_REP              0
        176 SA_REP           8600
        178 SA_REP           7000
        200 AC_ACCOUNT          0
        200 AD_ASST             0
        200 AD_ASST          4400
        201 MK_MAN          13000
        201 MK_REP              0
        202 MK_REP           6000
        205 AC_MGR          12000
        206 AC_ACCOUNT       8300
```



30 rows ← 20 + 10 - 0

Rules or Guidelines

8-5

- The expressions in the `SELECT` lists must match in number.
    - If you select 3 columns in A, then must have 3 columns in B

- 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.

Other

- Duplicate rows are automatically eliminated except in `UNION ALL`.

- Column names from the first query are the ones that appear in the result.

- The output is sorted in ascending order by default except in `UNION ALL`.