**Practical : 11**

**Class 3**

**Advanced Database**

**Lecturer :Dr. Haider Haddad**

**Set Operations**

* **A UNION**

**A union used to Implement a Full Outer JOIN**

*We looked at how data can be retrieved from multiple tables using joins. In this lecturer we will discuss how data can be retrieved from multiple tables using set operations available in Microsoft® SQL Server® 2008.*

1. **Introducing Set Operations**

*A set is a collection of objects. In a relational database, a table is regarded*

*as a set of rows. In relational database tables, there are not supposed to be duplicate rows; however, if a primary key is not defined for a table, spurious duplicate rows can occur. Three explicit set operations are used in some versions of SQL—UNION, INTERSECT, and MINUS (for set difference). SQL Server 2008 allows the explicit use of the UNION and INTERSECT operations. Since the MINUS set operation is not directly available in SQL Server 2016, we will illustrate the MINUS operation by using the very common IN and NOT .. IN predicates, which enable us to accomplish the same result as using INTERSECT*

*and MINUS.*

*The format of a set statement in SQL Server is as follows:*

*First, we look at examples of the UNION operator , the following is the syntax for the general form of a UNION:*

**SELECT \***

**FROM TableA**

**UNION**

**SELECT \***

**FROM TableB**

**2- Unioning Constants or Variables**

*In SQL Server 2008, a group of SELECT statements can also be used to union constants or variables. You may want to use this technique to experiment with the UNION or other set operations. A union of number*

*sets is shown below:*

**SELECT col1=100, col2=200**

**UNION**

**SELECT col1=400, col2=500**

**UNION**

**SELECT col1=100\*3, col2=200\*3**

**UNION**

**SELECT 900, 400**

This will produce:

col1 col2

------- -----------

100 200

300 600

400 500

900 400

(4 row(s) affected)

Note **that the output here happens to be sorted by the first column.**

**3- Union Compatibility**

*For union compatibility, the three basic data types are numeric, string, and dates. All numeric columns are compatible with one another, all string columns are compatible with one another, and all date columns are compatible with one another. SQL will convert integers, floating-point numbers, and decimals into a numeric data type to make them compatible with one another. So, any numeric column (such as integers) can be unioned with any other numeric column (such as decimals). Likewise, any fixed-length character column and any variable-length character column will be converted to a character data type and take on the larger size of the character columns being unioned. Similarly, date columns will be combined to a date data type.*

*Union compatibility can happen in several ways:*

*\_ By union two tables*

*\_ By taking two subsets from a table and combining them*

*\_ By using two views from two tables with the columns chosen so that they are compatible.*

**Students**

STNO SNAME MAJOR CLASS

------ -------------------- ----- ------

2 Lineas ENGL 1

3 Mary COSC 4

8 Brenda COSC 2

10 Richard ENGL 1

13 Kelly MATH 4

14 Lujack COSC 1

15 Reva MATH 2

17 Elainie COSC 1

19 Harley WEB 2

20 Donald ACCT 4

24 Chris ACCT 4

34 Lynette WEB 1

49 Susan ENGL 3

62 Monica MATH 3

70 Bill WEB NULL

121 Hillary COSC 1

(16 row(s) affected)

*In SQL Server 2016, a binary union is performed with the UNION set operation. A UNION takes output from two (or more) queries .*

*Suppose we want to find the names of all students who are computer science (COSC) majors along with all students who are MATH majors from the Student table. We can write the following query that uses the UNION set operator:*

**SELECT sname**

**FROM Student**

**WHERE major= 'COSC'**

While executing the query:

sname

----------

Mary

Zelda

Brenda

Lujack

Elainie

Jake

Hillary

Brad

Alan

Jerry

(10 row(s) affected)

**SELECT sname**

**FROM Student**

**WHERE major= 'MATH'**

*This part virtually produces the following seven rows of output:*

sname

--------

Mario

Kelly

Reva

Monica

Sadie

Stephanie

Jake

(7 row(s) affected)

*SQL then combines the two virtual sets of results (***the UNION operation***),which includes throwing out any duplicates (an extra “Jake” in this case),leaving us with the following 16 rows of output:*

sname

-----------

Alan

Brad

Brenda

Elainie

Hillary

Jake

Jerry

Kelly

Lujack

Mario

Mary

Monica

Reva

Sadie

Stephanie

Zelda

1. row(s) affected)

**4- Similar Columns in Unions**

*Earlier we discussed that for a union to be successful there has to be union compatibility—the two sets being unioned have to have similar columns.*

*So what does “similar columns” mean?*

*If we wrote the earlier UNION example like this:*

**SELECT major**

**FROM Student**

**WHERE major= 'COSC'**

**UNION**

**SELECT sname**

**FROM Student**

**WHERE major= 'MATH'**

*we would get an output, but would the output be valid? The answer is NO.*

*Why?. These are not similar columns (though the data types of the two columns are compatible).*

* **The UNION ALL Operation**

*UNION ALL works almost exactly like UNION but does not expunge duplicates*

*nor sort the results. UNION ALL is more efficient in execution because it does not have to deal with sorting and row removal*

*The following is the same query previously shown for UNION but using UNION ALL instead of UNION:*

**SELECT sname**

**FROM Student**

**WHERE major= 'COSC'**

**UNION ALL**

**SELECT sname**

**FROM Student**

**WHERE major= 'MATH'**

This query results in 17 unsorted rows, including one duplicate, Jake (whereas

using UNION produced 16 rows with no duplicates):

sname

---------

Mary

Zelda

Brenda

Lujack

Elainie

Jake

Hillary

Brad

Alan

Jerry

Mario

Kelly

Reva

Monica

Sadie

Stephanie

Jake

(17 row(s) affected)

This result set is not sorted and contains two occurrences of Jake.

* **The IN and NOT .. IN Predicates**

*Although SQL Server 2008 does not have the MINUS (difference) operator*

*Parse , Here are some examples:*

Set ***A* = (dog, cat, bird, monkey)**

**Set *B* = (cat, monkey, deer)**

***A* – *B* = (dog, bird)**

***B* – *A* = (deer)**

A INTERSECT B = (cat, monkey)

*A* UNION *B* = (dog, cat, deer, bird, monkey)

1. **Using IN**

*The following is a simple example of an IN predicate with constants in a*

*SELECT statement:*

**SELECT sname, class**

**FROM Student**

**WHERE class IN (3,4)**

*In this example, “IN (3, 4)” is calleda subquery-set, where (3, 4) is the set in*

*which we are testing membership. This query says: “Find all student names*

*from the Student table where the class is in the set (3, 4).” It produces*

*the following 17 rows of output:*

sname class

-------------------- ------

Mary 4

Kelly 4

Donald 4

Chris 4

Jake 4

Susan 3

Monica 3

Phoebe 3

Holly 4

Rachel 3

Jerry 4

Cramer 3

Harrison 4

Francis 4

Losmith 3

Gus 3

Benny 4

(17 row(s) affected)

*The preceding query produces the same output as the following query:*

**SELECT sname, class**

**FROM Student**

**WHERE class = 3 OR class = 4**

In this example, we have a table called *suppliers* with the following data:

| supplier\_id | supplier\_name | City | State |
| --- | --- | --- | --- |
| 100 | Microsoft | Redmond | Washington |
| 200 | Google | Mountain View | California |
| 300 | Oracle | Redwood City | California |
| 400 | Kimberly-Clark | Irving | Texas |
| 500 | Tyson Foods | Springdale | Arkansas |
| 600 | SC Johnson | Racine | Wisconsin |
| 700 | Dole Food Company | Westlake Village | California |
| 800 | Flowers Foods | Thomasville | Georgia |
| 900 | Electronic Arts | Redwood City | California |

## Using the IN Condition with Character Values

SELECT \*

FROM suppliers

WHERE supplier\_name = 'Microsoft'

OR supplier\_name = 'Oracle'

OR supplier\_name = 'Flowers Foods';

Using IN statement

SELECT \*

FROM suppliers

WHERE supplier\_name IN ('Microsoft', 'Oracle', 'Flowers Foods');

| supplier\_id | supplier\_name | City | State |
| --- | --- | --- | --- |
| 100 | Microsoft | Redmond | Washington |
| 300 | Oracle | Redwood City | California |
| 800 | Flowers Foods | Thomasville | Georgia |

Output

1. **Using the IN Condition with Numeric Values**

SELECT \*

FROM supplier

WHERE customer\_id IN (100, 300, 800);

## Using the IN Condition with the NOT Operator

SELECT \*

FROM supplier

WHERE product\_name NOT IN ('Microsoft', 'Oracle', 'Flowers Foods');

**2- Using IN As a Subquery**

*We can expand the IN predicate’s sub query set part to be an actual query.*

*For example****, consider the following query that gives us the names of students***

***who have a grade of “A”:***

**SELECT Student.sname**

**FROM Student S**

**WHERE Student.stno IN(SELECT g.student\_number**

**FROM Grade\_report g**

**WHERE g.grade = 'A')**

**Go**

Student table

STNO SNAME MAJOR CLASS

------ -------------------- ----- ------

2 Lineas ENGL 1

3 Mary COSC 4

8 Brenda COSC 2

10 Richard ENGL 1

13 Kelly MATH 4

14 Lujack COSC 1

15 Reva MATH 2

Grade-Rep table

student\_number grade

------ -------- -----

2 A

3 A

8 A

10 B

13 B

14 C

15 A

Result

sname

-------

Lineas

Mary

Brenda

Reva

*The preceding query produces the following 14 rows of output:*

*Note the following about this query:*

*\_ “WHERE Student.stno” references the stno column in the Student table.*

*\_ “g.student number” is the student number column in the Grade report table.*

*\_ stno in the Student table and student number in the Grade report table have the same domain.*

*To make this command behave like a set operator (as if it were an INTERSECT operator), you can add the qualifier DISTINCT to the result set as follows:*

**SELECT DISTINCT (Student.sname)**

**FROM Student**

**WHERE Student.stno IN**

**(SELECT DISTINCT (g.student\_number)**

**FROM Grade\_report g**

**WHERE g.grade = 'A')**

**go**

*This produces the following 14 rows of output:*

sname

--------

Brenda

Cedric

Donald

Holly

Jerry

Jessica

Lineas

Lujack

Lynette

Mary

Richard

Sadie

Steve

Susan

(14 row(s) affected)

Here, SQL Server 2008 sorts the results for you and does not return duplicates.

Here we would have to use an IN with a sub query as discussed earlier:

**SELECT s.stno, s.sname**

**FROM Student AS s**

**WHERE (s.stno IN**

**(SELECT pno**

**FROM Dependent AS d))**

**go**

This gives us the following 19 rows of output:S

stno sname

------ ------

2 Lineas

10 Richard

14 Lujack

17 Elainie

20 Donald

34 Lynette

62 Monica

123 Holly

126 Jessica

128 Brad

132 George

142 Jerry

143 Cramer

144 Fraiser

145 Harrison

146 Francis

147 Smithly

153 Genevieve

158 Thornton

(19 row(s) affected)

**3- Using NOT .. IN**

*If you use the NOT .. IN predicate in your query, your query may perform poorly on large tables. The reason is that when NOT .. IN is used, no indexing can be used, because the NOT .. IN part of the query has to test the set with all values to find out what is not in the set. For smaller tables, no difference in performance will likely be detected. Nonetheless, we discuss how to use the NOT .. IN predicate in this section to understand the logical negative of the IN predicate, which will help to complete your overall understanding of the SQL language. Instead of using NOT .. IN, it is often preferable from a performance standpoint on large tables to use NOT EXISTS or outer join techniques.*

*Sometimes the NOT .. IN predicate may seem to more easily describe the desired outcome or may be used for a set difference. For an example, consider the following query:*

**SELECT sname, class**

**FROM Student**

**WHERE class IN (1, 3, 4)**

**go**

*This produces the following 28 rows of output:*

sname class

-------------------- ------

Lineas 1

Mary 4

Richard 1

Kelly 4

Lujack 1

Elainie 1

Donald 4

Chris 4

Jake 4

Lynette 1

Susan 3

Monica 3

Hillary 1

Phoebe 3

Holly 4

Steve 1

Brad 1

Rachel 3

George 1

Jerry 4

Cramer 3

Fraiser 1

Harrison 4

Francis 4

Losmith 3

Lindsay 1

Gus 3

Benny 4

(28 row(s) affected Compare the preceding query with the following query:

**SELECT sname, class**

**FROM Student**

**WHERE class NOT IN (2)**

**GO**

sname class

-------------------- ------

Lineas 1

Mary 4

Richard 1

Kelly 4

Lujack 1

Elainie 1

Donald 4

Chris 4

Jake 4

Lynette 1

Susan 3

Monica 3

Hillary 1

Phoebe 3

Holly 4

Steve 1

Brad 1

Rachel 3

(28 row(s) affected)

*As another example,* ***suppose you want the names of students who are not computer science (COSC) or math (MATH) majors. The query would be:***

**SELECT sname, major**

**FROM Student**

**WHERE major NOT IN ('COSC', 'MATH')**

*This produces the following output (28 rows):*

sname major

-------------------- -----

Lineas ENGL

Ken WEB

Romona ENGL

Richard ENGL

Harley WEB

Donald ACCT

Chris ACCT

Lynette WEB

Susan ENGL

Bill WEB

Phoebe ENGL

Holly WEB

Jessica WEB

Steve ENGL

Cedric ENGL

Rachel ENGL

George WEB

Cramer ENGL

Fraiser WEB

Harrison ACCT

Francis ACCT

Smithly ENGL

Sebastian ACCT

Losmith CHEM

Genevieve UNKN

Lindsay UNKN

Gus ART

Benny CHEM

(28 row(s) affected)

1. **The Difference Operation**

*Because SQL Server 2008 does not support the MINUS predicate, we will*

*show the set difference operation using a NOT.. IN predicate with twoexamples.*

*Suppose set A is the set of students in classes 2, 3, or 4 and set B is the set*

*of students in class = 2. We could use the NOT .. IN predicate to remove the students in set B from set A (a difference operation) by typing the following query*:

SELECT sname, class

FROM Student

WHERE class IN (2, 3, 4)

AND NOT class IN (2)

This produces the following output (17 rows):

sname class

-------------------- ------

Mary 4

Kelly 4

Donald 4

Chris 4

Jake 4

Susan 3

Monica 3

Phoebe 3

Holly 4

Rachel 3

Jerry 4

Cramer 3

Harrison 4

Francis 4

Losmith 3

Gus 3

Benny 4

(17 row(s) affected)

**Example 7.2**