USING THE SET OPERATORS

Set operators are used to join the results of two *ormore* SELECT statements.The SET operators available in Oracle 11g are UNION,UNION ALL,INTERSECT,and MINUS.

The UNION set operator returns the combined results of the two SELECT statements.Essentially,it removes duplicates from the results i.e. only one row will be listed for each duplicated result.To counter this behavior,use the UNION ALL set operator which retains the duplicates in the final result.INTERSECT lists only records that are common to both the SELECT queries; the MINUS set operator removes the second query's results from the output if they are also found in the first query's results. INTERSECT and MINUS set operations produce unduplicated results.

All the SET operators share the same degree of precedence among them.Instead,during query execution, Oracle starts evaluation from left to right or from top to bottom.If explicitly parentheses are used, then the order may differ as parentheses would be given priority over dangling operators.

Points to remember -

- Same number of columns must be selected by all participating SELECT statements.Column names used in the display are taken from the first query.
- Data types of the column list must be compatible/implicitly convertible by oracle. Oracle will not perform implicit type conversion if corresponding columns in the component queries belong to different data type groups.For example, if a column in the first component query is of data type DATE, and the corresponding column in the second component query is of data type CHAR,Oracle will not perform implicit conversion, but raise ORA-01790 error.
- Positional ordering must be used to sort the result set. Individual result set ordering is not allowed with Set operators. ORDER BY can appear once at the end of the query. For example,
- UNION and INTERSECT operators are commutative, i.e. the order of queries is not important; it doesn't change the final result.
- Performance wise, UNION ALL shows better performance as compared to UNION because resources are not wasted in filtering duplicates and sorting the result set.
- Set operators can be the part of sub queries.
- Set operators can't be used in SELECT statements containing TABLE collection expressions.
- The LONG, BLOB, CLOB, BFILE, VARRAY,or nested table are not permitted for use in Set operators.For update clause is not allowed with the set operators.

UNION

When multiple SELECT queries are joined using UNION operator, Oracle displays the combined result from all the compounded SELECT queries,after removing all duplicates and in sorted order *ascendingbydefault*, without ignoring the NULL values.

Consider the below five queries joined using UNION operator.The final combined result set contains value from all the SQLs. Note the duplication removal and sorting of data.

```
SELECT 1 NUM FROM DUAL
UNION
SELECT 5 FROM DUAL
UNION
SELECT 3 FROM DUAL
UNION
SELECT 6 FROM DUAL
UNION
SELECT 3 FROM DUAL;
```

```
NUM
-------
1
3
5
6
```

To be noted, the columns selected in the SELECT queries must be of compatible data type. Oracle throws an error message when the rule is violated.

```
SELECT TO_DATE('12-OCT-03') FROM DUAL
UNION
SELECT '13-OCT-03' FROM DUAL;

SELECT TO_DATE('12-OCT-03') FROM DUAL
       *
ERROR at line 1:
ORA-01790: expression must have same datatype as corresponding expression
```

UNION ALL

UNION and UNION ALL are similar in their functioning with a slight difference. But UNION ALL gives the result set without removing duplication and sorting the data. For example,in above query UNION is replaced by UNION ALL to see the effect.

Consider the query demonstrated in UNION section. Note the difference in the output which is generated without sorting and deduplication.

```
SELECT 1 NUM FROM DUAL
UNION ALL
SELECT 5 FROM DUAL
UNION ALL
SELECT 3 FROM DUAL
UNION ALL
SELECT 6 FROM DUAL
UNION ALL
SELECT 3 FROM DUAL;

NUM
-------
1
5
3
6
3
```

INTERSECT

Using INTERSECT operator, Oracle displays the common rows from both the SELECT statements, with no duplicates and data arranged in sorted order *ascendingbydefault*.

For example,the below SELECT query retrieves the salary which are common in department 10 and 20.As per ISO SQL Standards, INTERSECT is above others in precedence of evaluation of set operators but this is not still incorporated by Oracle.

```
SELECT SALARY
FROM employees
WHERE DEPARTMENT_ID = 10
INTRESECT
SELECT SALARY
FROM employees
WHERE DEPARTMENT_ID = 20

SALARY
---------
1500
```

```
1200
2000
```

MINUS

Minus operator displays the rows which are present in the first query but absent in the second query, with no duplicates and data arranged in ascending order by default.

```
SELECT JOB_ID
FROM employees
WHERE DEPARTMENT_ID = 10
MINUS
SELECT JOB_ID
FROM employees
WHERE DEPARTMENT_ID = 20;

JOB_ID
-------------
HR
FIN
ADMIN
```

Matching the SELECT statement

There may be the scenarios where the compound SELECT statements may have different count and data type of selected columns. Therefore, to match the column list explicitly, NULL columns are inserted at the missing positions so as match the count and data type of selected columns in each SELECT statement. For number columns, zero can also be substituted to match the type of the columns selected in the query.

In the below query, the data type of employee name *varchar2* and location id *number* do not match. Therefore, execution of the below query would raise error due to compatibility issue.

```
SELECT DEPARTMENT_ID "Dept", first_name "Employee"
FROM employees
UNION
SELECT DEPARTMENT_ID, LOCATION_ID
FROM departments;

ERROR at line 1:
ORA-01790: expression must have same datatype as corresponding expression
```

Explicitly, columns can be matched by substituting NULL for location id and Employee name.

```
SELECT DEPARTMENT_ID "Dept", first_name "Employee", NULL "Location"
FROM employees
UNION
SELECT DEPARTMENT_ID, NULL "Employee", LOCATION_ID
FROM departments;
```

Using ORDER BY clause in SET operations

The ORDER BY clause can appear only once at the end of the query containing compound SELECT statements.It implies that individual SELECT statements cannot have ORDER BY clause. Additionally, the sorting can be based on the columns which appear in the first SELECT query only. For this reason, it is recommended to sort the compound query using column positions.

The compund query below unifies the results from two departments and sorts by the SALARY column.

```
SELECT employee_id, first_name, salary
FROM employees
WHERE department_id=10
UNION
SELECT employee_id, first_name, salary
```

```
FROM employees
WHERE department_id=20
ORDER BY 3;
```