Restricting and Sorting Row

Upon completing this chapter we should understand:

1. How to limit the rows retrieved by a query.

2. How to Sort the rows retrieved by a query

At times when retrieving data, we will need to display data in a fashion that when presented to the user it is usable or readable for the person requesting the information.

The possibility exists that that person does not need to know all of the information available in the table(s).

To accomplish this we will use the WHERE clause added to the basic SQL statement we have already learned. Some people will equate the WHERE clause to the same functionality as an IF statement.

The basic format of the WHERE clause is as follows:

SELECT column names

FROM tablename

WHERE the condition checks as TRUE.

Example:

We have a table. EMP, that consists of the columns of: EMPNO, ENAME, JOB, and DEPTNO. We want to select the rows from the EMP table BUT only the rows for the employees that are assigned to Department 10.

```
SELECT empno, ename, job, deptno
FROM emp
WHERE deptno = 10;
```

In this example only the people that work in department 10 will e displayed.

When we add the WHERE clause we have to understand that the purpose is to LIMIT the rows returned.

The WHERE clause can compare values in columns, literal values, arithmetic expression, or functions.

The WHERE clause consists of three (3) elements:

1. The column name
2. The comparison operator
3. Column name, constant, or list of values.

Remember that ORACLE is case sensitive when using a constant where MySQL and MSSQL are not.

Working with Character strings and Dates.

Character strings and date values are enclosed on single quotes.

Character values are case sensitive (ORACLE) and date values are FORMAT sensitive.

The date format are:

In ORACLE: DD-MON-YY

In MySQL: YYYY-MM-DD

In MSSQL: YYYY-MM-DD

Character strings and dates have to be enclosed in single quotes however number constants do not.

The comparison operators are the same that are used in programming languages such as C++ and Java.

Equal to	=
Greater Than	>
Greater than OR equal to	>=
Less Than	<
Less than OR equal to	<=
Not Equal to	<>

Example of using a column to column comparison:

We want to display from the EMP table, all employees who have a commission (comm) that is greater than their salary.

```
SELECT ename, sal, comm
FROM emp
WHERE sal <= comm;
```

In this example the value contained in the two columns are compared and IF the condition (<=) checks out as TRUE the requested information is returned in the data set.

We have some other comparison operators that we can also use. There are four (4) of them:

BETWEEN and	Used to check a range of values.
In	Used to see if the value is in a list.
Like	Used to match a character pattern.
IS NULL	Used to check for a NULL value.

Using the BETWEEN operator.

The BETWEEN operator is use to see if a value in the specified range. Using the BETWEEN operator is inclusive, meaning that the lowest limit and the highest limit are also returned.

```
SELECT ename, sal
FROM emp
WHERE sal BETWEEN 1000 and 1500;
```

This will return all salaries of 1000 or more but less than or equal to 1500. The lower limit must be specified first.

Using the IN operator.

We are checking to see if the value in the specified column is in the LIST that is provided. When using the IN operator we could also use a series of OR statement to accomplish that same thing, however using the IN operator reduces code and allows for easier modification.

```
SELECT empno, ename, sal, mgr
FROM EMP
where mgr IN (7902, 7566, 7788);
```

The only records returned are the employees who have a manager with a manager number enclosed in the parenthesis.

Using the LIKE operator

The like operator use wildcard character so search for a pattern when we do not know the exact value we are searching for. The two wildcard character are: % and _ (Underscore).

The % sign represents any sequence of ZERO or more characters.

The _ (underscore) represent any SINGLE character.

The search conditions can contain either literal characters or numbers.

Using the % sign as a wildcard character.

```
SELECT ename
FROM emp
WHERE ename LIKE ' %S%';
```

This statement will return all rows of employees that have an S in the name.

```
SELECT ename
FROM emp
WHERE ename LIKE ' S%';
```

This statement will return all rows of employees that have an S as the first character in their name.

Using the _ (underscore) wildcard character.

The underscore can be used in the LIKE statement as a placeholder. Each underscore represent one (1) character in the search pattern. I the following example we will get data back where the letter “A” is in the second position of their name.

```
SELECT ename
FROM emp
WHERE ename LIKE ‘_A%’;
```

If we had used the % instead of the underscore we would get back a result set of anyone who has an “A” anywhere in the name, a totally different results.

Using the IS NULL operator.

The IS NULL operator tests for values that are null. A null value is defined as: UNAVAILABLE, UNASSIGNED, UNKNOW, or INAPPLICABLE.

Since we do not know what the value is we cannot test the column for equality since a NULL value cannot be equal or unequal to ANY value.

If we want to display the employee information for all employees who are not entitled to get commission we would do the following:

```
SELECT ename, job, comm
FROM emp
WHERE comm IS NULL;
```

Logical Operators

The use of logical operators allow us to combines the results of two components to produce a single result set based on them or to invert the results of a single condition.

The logical operators are:

AND

OR

NOT

When using the logical operators we have to consider the ‘TRUTH TABLE’.

The AND operator

When we use the AND operator BOTH condition have to be TRUE to return a TRUE value. In the following example both sides have to be TRUE to return TRUE.

```
SELECT empno, ename, job, sal
FROM emp
WHERE sal >= 1100
and job = 'CLERK';
```

If either column value if FALSE the entire expression will be FALSE. So if the value of sal is 1500 and the job is SALESMAN, the row will not be returned.

The OR operator

When we use the OR operator EITHER condition has to be TRUE to return a TRUE value. In the following example either side have to be TRUE to return TRUE.

```
SELECT empno, ename, job, sal
FROM emp
WHERE sal >= 1100
OR job = 'CLERK';
```

If either column value if TRUE the entire expression will be TRUE. So if the value of sal is 1500 and the job is SALESMAN, the row will be returned.

The NOT operator

When we use the NOT operator we negate or reverse the values of the field being checked.

```
SELECT empno, ename, job, sal
FROM emp
WHERE job not in ('CLERK')
OR sal < 1100;
```

If either column value if TRUE the entire expression will be TRUE. So if the value of sal is greater than 1100 and the job is NOT equal to CLERK, the row will be returned.

Rules of Precedence

Everything has a certain order in which they are evaluated. This is referred to as the Rules of PRECEDENCE. The order in SQL is as follows:

1. All comparison operators
2. NOT
3. AND
4. OR

This rule can be overridden with the use of parenthesis. Anything INSIDE the parenthesis is evaluated first with the comparisons inside the parenthesis being evaluated in the above order.

ORDER BY clause

The order by clause is used to display the result set in an order that has meaning to the user. It is always the LAST clause in the SELECT statement. We can use either column names or alias in the order by clause. The components of the order by clauses are:

```
SELECT desire columns
FROM tablename
WHERE clause
ORDER BY columnname [ASC|DESC];
```

ASC is Ascending order (The DEFAULT) from lowest to highest.

DESC is Descending order from highest to lowest.

More ORDER BY clause

We can also sort by multiple columns. We can also order by columns that are not included in the SELECT statement. To do this we specify two columns in the order by separated with a comma. We do not have to sort both columns the same, one can be ASC and the other DESC.