

RESTRICTING AND SORTING DATA -- CHAPTER 2 --

- Limiting the rows selected involves the use of the special WHERE clause as follows:

```
SELECT    [DISTINCT] { * | column [alias], ... }
FROM      table
WHERE     condition(s);
```

- The condition above refers to any combination of column names, expressions, constants, and a comparison operator (>, <, =, >=, <=, <> (not equal) etc.).

```
SQL> run
 1 select ename, job, deptno
 2 from emp
 3* where job='SALESMAN'
```

ENAME	JOB	DEPTNO
ALLEN	SALESMAN	30
WARD	SALESMAN	30
BOBBY	SALESMAN	
TURNER	SALESMAN	30
BLACK	SALESMAN	10

- Note that 'SALESMAN' in the where clause IS case-sensitive!
- The default DATE format is: DD-MON-YY [01-JAN-95].
- Oracle stores dates in an internal numeric format, representing the century, year, month, day, hours, minutes, and seconds.
- Number values should not be enclosed in quotation marks as string literals should.

```
SQL> r
 1 select ename, job, deptno
 2 from emp
 3 where job='SALESMAN' AND
 4*      deptno=30
```

ENAME	JOB	DEPTNO
ALLEN	SALESMAN	30
WARD	SALESMAN	30
TURNER	SALESMAN	30

- Note the use of the AND operator to further restrict the selection criteria.
- Other comparison operators:
 - BETWEEN ...AND... Between two values (inclusive)

- IN(list) Match any of a list of values
- LIKE Match a character pattern
- IS NULL Is a null value.

- The following example shows how to use the BETWEEN operator...

```
SQL> run
  1  select ename, job, deptno
  2  from emp
  3* where deptno between 30 AND 50
```

ENAME	JOB	DEPTNO
ALLEN	SALESMAN	30
WARD	SALESMAN	30
BLAKE	MANAGER	30
TURNER	SALESMAN	30
MOLITOR	ANALYST	40
SHANK	ANALYST	40

6 rows selected.

- In this case, we use two constants, 30 and 50, but we could have used a reference to a value stored in the table. It is also very important to note the order of the 30 and 50. The left of the 'AND' keyword represents the lower limit (30) while the number at the right of the 'AND' represents the upper limit. For this reason, a statement like this: ...WHERE DEPTNO BETWEEN 50 AND 30; would find no rows since there would be no intersection between upper and lower value.
- The following is an example on how to use the IN operator...

```
SQL> run
  1  select ename, job, deptno
  2  from emp
  3* where job IN ('SALESMAN', 'MANAGER', 'victor3434')
```

ENAME	JOB	DEPTNO
ALLEN	SALESMAN	30
WARD	SALESMAN	30
JONES	MANAGER	20
BOBBY	SALESMAN	
BLAKE	MANAGER	30
CLARK	MANAGER	10
TURNER	SALESMAN	30
BLACK	SALESMAN	10

- Note that as long as one of the fields is found, the whole row is selected.
- Now look at the following example using the LIKE operator...

```
SQL> select ename
      2  from emp
      3  where ename LIKE 'S%';
```

```
ENAME
-----
SMITH
SCOTT
SIMMSON
SHANK
```

- Note that the % symbol is a 'Wildcard' character...in other words, in the example above, all those values whose ename start with an 'S' followed by zero or more characters, regard less of what they are. Note also that the pattern to be matched IS case-sensitive meaning that those of ename that start with a lower-case 's' will not be matched!

```
SQL> r
      1  select empno, ename
      2  from emp
      3* where ename LIKE 'S_'
```

```
EMPNO ENAME
-----
3434 SI
```

- In the example above, we use "_" to match a single character (as opposed as any number of them with '%'). SI is the only ename that matches this description, despite that there are other values of ename that start with a capital 'S'.
- Further, these two operators ('_', and '%') can be combined in a single statement...

```
SQL> select ename
      2  from emp
      3  where ename like '_A%';
```

```
ENAME
-----
WARD
MARKELL
HAYES
HAYES2
```

- Which returns all those names whose second letter is a capital 'A'.
- Finally, in the event that we are looking specifically for patterns that contain the symbols '_' or '%', we can use the ESCAPE identifier which allows us to specify any character as an ESCAPE character in order to match those special symbols.

SEE BELOW...

```
SQL> r
1 select empno, ename, job
2 from emp
3* where ename like '%*_%' ESCAPE '*'
```

```
      EMPNO ENAME      JOB
-----
      3535 VIC_SANC
```

- Here, we designate '*' to be our escape character. Note that the escape character must be of type character string and must have a length of 1.
- The following example shows how to use the NULL operator...

```
SQL> select empno, ename, job
2 from emp
3 where job IS NULL;
```

```
      EMPNO ENAME      JOB
-----
      3434 SI
      3535 VIC_SANC
```

- In this case, we select those items having a NULL value in the JOB field. Recall that some of these columns must contain a value and NULL values are not allowed...such is the case with rows representing primary key (and possibly foreign key values). Testing for NULL cannot be done with the '=' operator!
- Logical operators AND, OR, and NOT can be applied to further restrict search criteria. Here's an example using the OR operator...(all of these can be used in combination).

```
SQL> r
1 select empno, ename, job
2 from emp
3 where empno=3434 or
4* job is null
```

```
      EMPNO ENAME      JOB
-----
      3434 SI
      3535 VIC_SANC
```

- Below is an example using the NOT operator...

```
SQL> r
1 select empno, ename, job
2 from emp
3 where not empno=3434 or
4* job is null
```

EMPNO	ENAME	JOB
7369	SMITH	ANALYST
7499	ALLEN	SALESMAN
7521	WARD	SALESMAN
7566	JONES	MANAGER
321	BOBBY	SALESMAN

...

- Note that list goes on and on...but I truncated it here...
- Here's a few other examples of proper syntax using the NOT operator...

- ...WHERE job NOT IN('CLERK', 'ANALYST')
- ...WHERE sal NOT BETWEEN 1000 AND 1500
- ...WHERE ename NOT LIKE '%A%'
- ...WHERE IS NOT NULL

- There are rules of precedence when using logical operators in conjunction. In order of most to least importance, these operators are evaluated as follows: All comparison operators-->NOT-->AND-->OR.
- However, it is always a good idea to use parentheses to force the evaluation of certain pairs of items and it is also easier to read.
- ORDER BY clause can be used to sort the rows. If you use the ORDER BY clause, it must be used at the end of all SQL statements. You can specify an expression or an alias to sort. Here's the syntax:

```

SELECT      expression
FROM        table
[WHERE      condition(s)]
[ORDER BY   {column, expression}[ASC|DESC]];

```

- where ORDER BY specifies the order in which the retrieved rows are displayed. ASC orders the rows in ascending order (this is the default order) and DESC (orders rows in descending order).
- The following is an example of using the ORDER BY clause...

```

SQL> r
1  select empno, ename, job
2  from emp
3  where empno > 3000 and
4         empno < 8000
5* order by empno desc

```

EMPNO	ENAME	JOB
7876	ADAMS	ANALYST
7844	TURNER	SALESMAN
7839	KING	PRESIDENT

```

7096 BLACK      SALESMAN
7044 MOLITOR    ANALYST
3535 VIC_SANC
3434 SI

```

7 rows selected.

- Notice that we use the 'DESC' keyword in order to list the results in descending order according to the empno field.
- Note that we can also sort using a user-defined column alias as follows:

```

SQL> select empno, ename, sal*12 annsal
2  from emp
3  order by annsal;

```

EMPNO	ENAME	ANNSAL
8555	HAYES	1188
7499	ALLEN	12000
7521	WARD	12000
7844	TURNER	12000
...		

- Here, we have annsal as the column alias and it is ordering in ascending order, since it is the default manner in which ORDER BY orders rows.
- Further, it is possible to generate a list of values that are sorted using a column that is not included in the select list (i.e., a column that exists on the table, but is not displayed in the results).
- Finally, it is possible to sort based on multiple columns...

```

SQL> run
1  select ename, deptno, sal
2  from emp
3* order by deptno, sal DESC

```

ENAME	DEPTNO	SAL
MARKELL	2	1500
H	3	1500
KING	10	3000
CLARK	10	2000
BLACK	10	1000
JONES	20	2000
SCOTT	20	2000
SMITH	20	1500
...		