Cognizant | Academy

Business Focused. Learner Centric.

Oracle Version 12C

Restricting and Sorting data

Enabling Objectives

After completing this chapter, in the next 60 minutes you will be able to :

- Identify at least 2 operators in different types of operators from the given list.

- Define how to Limit the rows retrieved by Select statements
- Implement sorting of the records based on the given scenario.

- Describe formatting in SQL*Plus

Key Topics

- SQL Operators
- Restricting
- Sorting

SQL Operators

Comparison Operators

- Comparison operators are used in conditions that compare one operand with another.
- The result of a comparison can be TRUE (or) FALSE (or) NULL.

Comparison operator	Name
=	equals operator
!=, <>	not equals operator
<	less than operator
>	greater than operator
<=	less than or equals operator
>=	greater than or equals operator

Logical Operators

- *Logical operators* are used for manipulating the results of one or more conditions.
- In SQL, all logical operators evaluate to TRUE, FALSE, or NULL (UNKNOWN).

Operator	Description
NOT	Returns TRUE if the condition returns FALSE. Returns FALSE if the return values is TRUE.
AND	Used to combine two conditions. Returns TRUE if both condition are met. Returns FALSE if either of it is FALSE.
OR	Returns TRUE if one of the condition returns TRUE. Returns FALSE if both are FALSE.

Arithmetic Operators

Arithmetic operators are used to manipulate numeric operands, which are columns storing numeric values.

Operator	Description
+(monadic)	Makes operand positive
-(monadic)	Makes operand negative
/	Division(Used with Number and Date)
*	Multiplication
+	Addition (numbers and dates)
-	Subtraction (numbers and dates)

List of Operators

Range Operators

- Compares with given Range
- A range can be defined with lower and upper limits

Operator	Description
BETWEEN AND	Checks whether the operand value falls within a range.
NOT BETWEEN AND	Checks whether the operand value does not falls within a range.

List of Operators

String Operators

- Compares with given Wild card characters.

Operator	Description
LIKE /NOT LIKE	The LIKE operator is used for wild card matching. _ is used for single character.

Restricting
(Limit the rows retrieved)

List of Operators

Set Operators

- Matches the given list of values.

Operator	Description
IN	Equivalent to comparing the operand value with a list of values and if any match happens it returns true.
NOT IN	Equivalent to comparing the operand value with a list of values and if any match happens it returns true.

Where Clause

- This clause is used with Select statements when restricted set of records are retrieved from database.
- Multiple conditions can be specified using any of the standard logical operators.
- It works on row level.

Limit the rows retrieved

- The following elements can be used with the WHERE clause:
  - Column names
  - All Operators
  - List of values

```
SELECT column_list|*
FROM table_name
WHERE condition
```

Limit the rows retrieved

Employees Table:

Examples

```
select  e_name, e_code
from employees
where e_salary between 10000 and  20000 ;
```

Result

Retrieves names and codes for all employees having salary in the range of 10000 and 20000.

Limit the rows retrieved

Employees Table:

Examples

```
select e_code, join_date
from employees
where city in ('Kolkata', 'Chennai', 'Bangalore');
```

Result

Retrieves codes and join dates for all employees who are posted in either Kolkata or Chennai or Bangalore.

Employees Table:

Examples

```
select e_name, e_code, join_date
from employees
where e_salary >=30000 and city not in ('Kolkata');
```

Result

Retrieves names, codes and join dates for all employees who are having a salary greater than or equal to 30000 and posted in cities other than Kolkata.

Character Comparison Operator: ‘like’

Examples

```
select  *  from employees
where city like  ‘K%’;
```

Result

Retrieve the details of all employees posted in cities that begin with ‘K’.

```
select  *  from employees
where e_name not like ‘_a%’;
```

Result

Retrieve the details of all employees whose second letter in name does not contain ‘a’.

Character Comparison Operator: ‘like’

Examples

```
select  * from employees
where join_date like  ’%Jan%’;
```

Result

Retrieve the details of all employees who were recruited in the month of January

```
select * from employees
where job_code like ‘Sa\_%' escape '\';
```

Result

Retrieves records of employees having job_code beginning with ‘Sa_’

Use ‘null ‘ in where clause

The predicates *: is null* and *is not null* are used for fetching data based on null values.

Example

```
select e_name, e_code, city
from employees
where phone_no is not null;
```

Result

Retrieve the codes of those employees who have a phone number .

Use ‘null ‘ in where clause

Example

```
select e_code from employees
where join_date is null;
```

Result

Retrieve the codes of those employee records whose join_date field is blank.

Eliminating duplicates: ‘distinct’

The keyword ***distinct*** is used with Select statements to retrieve unique data entries from the column specified after it.

Example

```
select distinct city from employees;
```

Result

Retrieves list of all cities where the employees are placed without repeating them.

Sorting

Sorting records : ‘order by’

order by clause is used to sort the output in ascending or descending order.

- Default order is ascending (ASC):
  - Date Values earliest first
  - Numeric Values Lowest First
  - Character Values alphabetically

- For arranging in descending order, DESC is used

Sorting records : ‘order by’

- Column names or Column index can be specified
- ORDER BY should be the last clause in a SELECT statement
- Can specify multiple column names.
- Order by clause first sorts the retrieved data by the first column, then the next one and so forth.

Sorting records : ‘order by’

Examples

```
select * from employees
where city in (‘Kolkata’, ’Chennai’, ’Bangalore’)
order by salary;
```

Result

Retrieve the records of employees placed in either Kolkata, Chennai or Bangalore and display them in ascending order of salary.

Sorting records : ‘order by’

Examples:

```
select * from employees
order by city, join_date
```

Result

Display the records of employees ordered by both city and join_date columns.;

```
select e_name, city
from employees
order by 2;
order by city, join_date
```

Result

Displays names and city of employees in the ascending order of city.

Label columns

Column Aliasing or Labeling Columns :

- Renames a column heading
- Mainly used with calculated columns
- Immediately follows column name, optionally AS keyword can be used between column name and alias
- Requires double quotation marks if it contains embedded spaces or special characters or is case sensitive
- Alias names can be used in the order by clause

Concatenate column contents

The Concatenation Operator: ||

- Used to join columns to other columns, arithmetic expressions and literal strings
- Resultant column is a character expression

Concatenation Operator

Consider a sample table : Employees

phone_no	city	e_code	e_name	e_salary	join_date
5678	Kolkata	12	Smith	10000	11-Jan-01
3456	Kolkata	34	John	30000	01-Nov-02
6543	Chennai	8	Sam	13000	10-Jan-01

Example

```
select e_code||'   '||e_name||'  '||e_salary "Employee Details"
from employees;
```

Result

Retrieves values from e_code, e_name and salary columns ,concatenates them with blank spaces embedded and a column alias "Employee Details"

Employee Details
12 Smith 10000
34 John 30000
8 Sam 13000

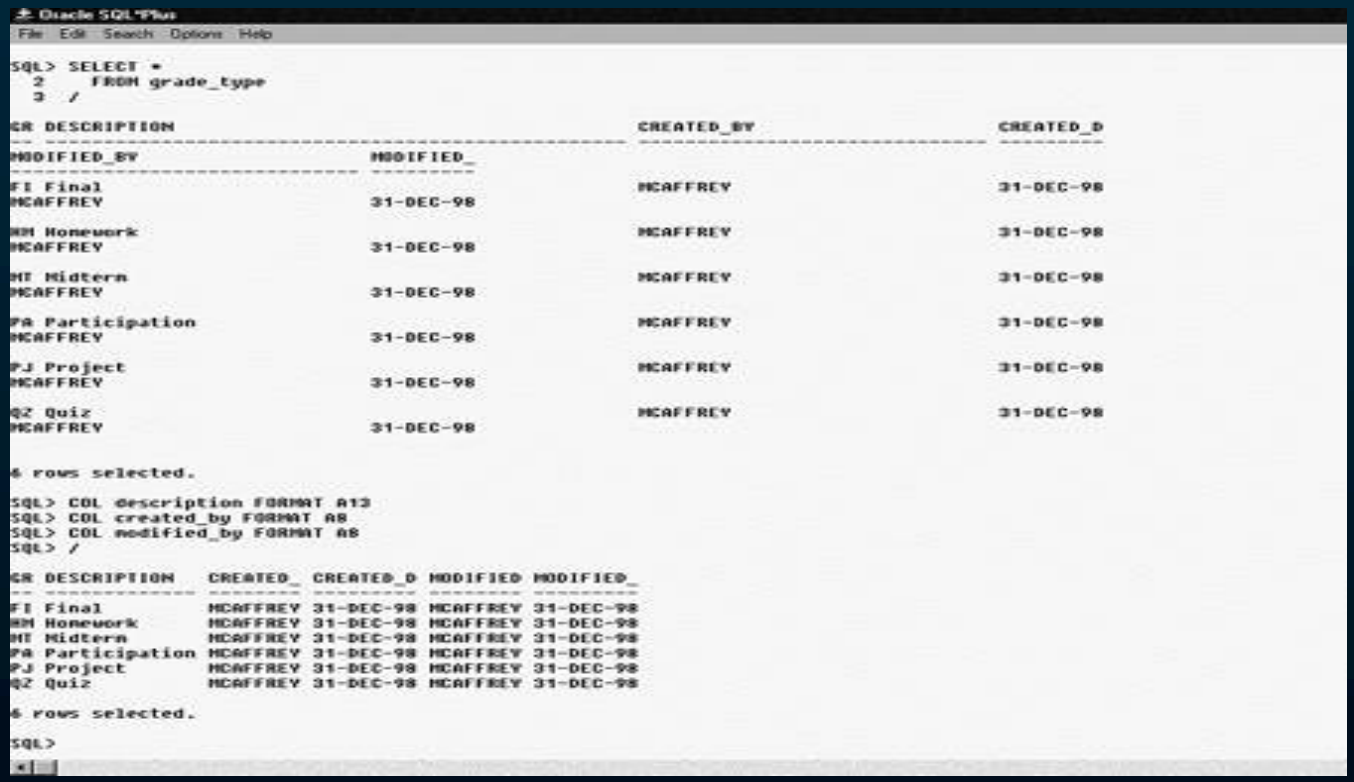
Formatting in SQL*Plus

Formatting in SQL*Plus

- In **SQL***Plus interface the result set is difficult to read as data "wraps" itself onto the next line.

Data is wrapped onto new line

Column command used for formatting the output



Formatting in SQL*Plus

- SQL*Plus ***Column*** command with ***format*** attribute is used to format specific columns

COLUMN column_name FORMAT format_model

Formatting in SQL*Plus

- The format stays in place until it is cleared or SQL*Plus is exited.

  Example:

  COLUMN description FORMAT A13

  Result

  The description column is formatted to display a maximum of 13 characters

Formatting in SQL*Plus

Example:

```
COLUMN e_salary FORMAT $99,999.99
```

Result

Retrieves values from e_salary column.

9 represents numeric digits, $ is prefixed before all values.

```
e_salary
--------------------
$10,000.00
$30,000.00
$13,000.00
```

Formatting in SQL*Plus

Examples

CLEAR COLUMNS Clears all column formatting.

```
COLUMN   e_name HEADING 'EMPLOYEE NAME'
COLUMN  e_salary HEADING 'MONTHLY SALARY'
select e_name, e_salary from employees
```

Result

Retrieves employee names and salary with the column labels as specified by COLUMN formatting.

User defined variables

- Can be included in select statements to make the query more interactive
- Are preceded by & or &&

Example

```
select * from employees where e_code = &code;
```

Result

When the query is executed the user is prompted to enter a value for &code.

The users input is substituted in the query and then executed.

Every time the user executes the query, he will be prompted for input.

User defined variables

Example

```
select * from employees where e_code = &&code;
```

Result

When the query is executed the user is prompted to enter a value for &&code.

The users input is substituted in the query and then executed.

The user will be prompted only the first time the variable is referenced.

User defined variables

Example

```
select * from employees where e_code = &code;
```

Result

Enter value for code: 55 ---------- user prompted for input
old 1: select * from e where e_code=&code
new 1: select * from e where e_code=55
OUTPUT DISPLAYED -------record of employee 55 displayed

If user executes this query again, he will be prompted for value of e_code

User defined variables

Example

SQL> **select * from employees where e_code = &&code;**

Enter value for code: 55 ---------- user prompted for input

old 1: select * from e where e_code=&&code

new 1: select * from e where e_code=55

*** OUTPUT DISPLAYED*** ---------- record of employee 55 displayed

If user executes this query again, he will not be prompted for value of e_code

Case Statement

- Used to perform certain tasks based on a condition:

CASE expression
WHEN value | Boolean expression ***THEN*** return value
ELSE return value
END

Example

```
select e_name, case city
when 'Kolkata' then 'EAST'
when 'Bangalore' then 'SOUTH'
else 'OTHERS'
end "Region" from employees
```

Result

Displays the employee names along with the region in where they are placed.
Region is logically computed based on the value in city.

Sysdate

sysdate is a built in function that returns current system date and time.

- The value displayed is in the default format :dd-mon-yy

Example

```
select e_name from employees
where (sysdate – join_date) <30
```

Result

Retrieves the names of all employees hired less than 30 days ago.

sysdate –join_date returns the difference in terms of days .

Test Your Understanding

- Which SQL construct can be used to locate rows which have NULL values?

- State whether TRUE or FALSE:

  select e_name, e_salary from employees order by 2 desc

  is same as

  select e_name, e_salary from employees order by e_salary desc

- What is the role of the function sysdate?

Restate Objectives

In this session we learnt the following:

- Identify at least 2 operators in different types of operators from the given list.
- Define how to Limit the rows retrieved by Select statements
- Implement sorting of the records based on the given scenario.
- Describe formatting in SQL*Plus

You have successfully completed – Restricting and Sorting Data