

Data Filtering with WHERE clause

Introduction

In SQL where clause is used to filter the rows returned by a SELECT, UPDATE or DELETE statement based on the specific condition given by the user. The condition provided on the where clause must be met by the rows to be included on the result query. The basic syntax of WHERE clause is:

Select column1, column2,

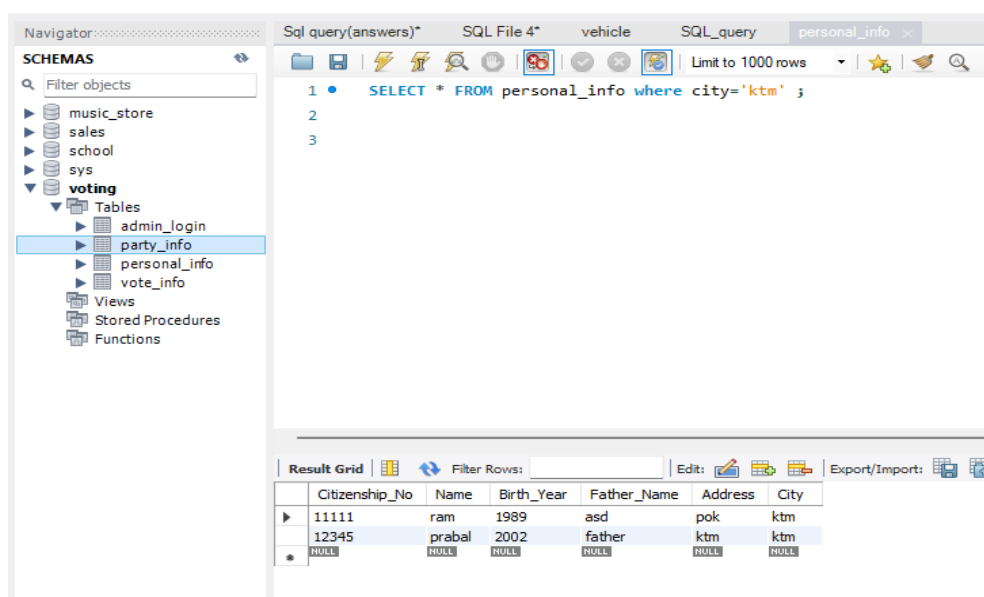
From table_name

WHERE condition;

For example, If we want to retrieve all the records from the personal_info table where the city column is equal to 'ktm'.

```
SELECT * FROM personal_info
```

```
Where city = 'ktm';
```



The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane displays a tree view of database objects, including tables like 'music_store', 'sales', 'school', 'sys', and 'voting'. The 'personal_info' table is selected. The main pane shows a SQL query: `SELECT * FROM personal_info where city='ktm' ;`. Below the query, the 'Result Grid' displays the following data:

Citizenship_No	Name	Birth_Year	Father_Name	Address	City
11111	ram	1989	asd	pok	ktm
12345	prabal	2002	father	ktm	ktm
NULL	NULL	NULL	NULL	NULL	NULL

WHY TO USE WHERE CLAUSE ?

There might be many questions why WHERE clause is required and why to use it. The answer is simple .If we use :

```
SELECT * from table_name;
```

This query returns all the rows present in the table. So using the where clause allows to selectively retrieve only the specific records that meet certain conditions hence avoiding unnecessary data retrieval.

Common Operators used in Where clause

1. Comparison operators

Comparison operators are used to compare values in a database and determine the relationship. Some of the commonly used comparison operators along with examples are:

a. Equals to (=):

The equals to operator are used to retrieve the rows where the specified column is equal to a particular value. For example:

```
SELECT * from customers
```

```
WHERE
```

```
Id=10;
```

The above query returns all the details of customer having id 10 from the customer table.

b. Not Equal to (<> or !=):

The not equal to operators are used to retrieve rows where the specified column is not equal to a particular value. For example:

```
SELECT * FROM customers
WHERE
category_id != 4;
```

The above query returns all the rows except the rows having value 4 as category_id from the customers table.

c. Greater Than (>):

The greater than operator is used to retrieve rows where the specified column has value greater than a particular value.

```
SELECT * FROM customers
WHERE
category_id > 2;
```

The above query returns all the rows having category_id greater than 2 (i.e 3,4,...) from customers table.

d. Less Than (<):

The less than operator is used to retrieve rows where the specified column has value less than the particular value.

```
SELECT * FROM customers
WHERE
category_id < 4;
```

The above query returns all the rows having category_id less than 4 (i.e 3,2,...) from customers table.

e. Greater than or Equal to(>=)

The greater than or equal to operator is used to retrieve rows where the specified column has value greater than or equal to a particular value.

```
SELECT * FROM customers
WHERE
category_id >= 2;
```

The above query returns all the rows having category_id greater than or equal to 2 (i.e 2,3,4,...) from customers table.

f. Less Than or Equal to (<=);

The less than or equal to operator is used to retrieve rows where the specified column has value less than or equal to a particular value.

```
SELECT * FROM customers
WHERE
category_id <=4;
```

The above query returns all the rows having category_id less than 4 or equal to 4 (i.e 4,3,2,...) from customers table.

2. LIKE operator:

The like operator is used to search for some specific pattern in a column. There are mainly 2 types of wildcards often used with the like operator.

They are:

The percent sign % represents zero, one, or multiple characters.
For example:

```
SELECT * from customers  
WHERE name like 'A%';
```

The above query returns all the rows where the name of the customer begins with the letter 'A' from the customer table.

The underscore sign _ represents one, single character.

```
SELECT * from customers  
WHERE name like '__am%';
```

The above query returns all the rows where the name of the customer have 'a' and 'm' as second and third letter respectively.

3. IN operator:

The IN operator is used to specify multiple values in a WHERE clause.

```
SELECT * from customers  
WHERE customer_id in ( 1,2,3,4);
```

The above query retrieves all the rows from the customer table where the customer_id is either 1 or 2 or 3 or 4.

4. Between operator :

The Between operator is used to display the records when a columns value fall within the given range. For example:

```
SELECT * FROM customers  
Where customer_id between 10 and 20;
```

The above query retrieves all the rows from the customer table where the customer_id is between 10 and 20.

5. Logical Operators :

The logical operators are used to combine multiple conditions to filter for the data retrieval.

a. AND operator

The And operators retrieves only those rows which satisfies all the conditions.

```
SELECT * from customers  
Where customer_id = 23 AND purchase_amount >20000;
```

The above query retrieves the customer with id 23 and purchase amount greater than 20000.

b. OR operator

The Or operators retrieves those rows which satisfies any one of the specified conditions .

```
SELECT * from customers  
Where customer_id = 23 OR purchase_amount >20000;
```

The above query retrieves the customer with id 23 or the customer who have purchase amount greater than 20000.

c. NOT operator

```
SELECT * FROM customers  
Where not customer_id >20;
```

The above query retrieves all the rows where customer_id is not greater than 20.

Conclusion

The WHERE clause is an important and powerful tool in data analysis as it helps to extract meaningful insight by allowing the existence of simple comparison logic to complex logic. Where clause also help to optimize the query performance and contribute to overall efficiency of data retrieval process.

