AROR UNIVERSITY
OF ART, ARCHITECTURE,
DESIGN & HERITAGE,
SUKKUR, SINDH

Department of Artificial Intelligence & Multimedia Gamming

CSC-207: Database Systems

Lab # 05: To Work with SQL Operators & Querying Database Tables

Objectives

1. WHERE clause
2. Arithmetic operators
3. Comparison operators
4. Logical operators
5. Concatenation Operator
6. SQL Operator Precedence

WHERE clause in SQL

- The WHERE clause is used to filter rows based on a specified condition.
- Syntax: SELECT columns FROM table WHERE condition;
- The condition is evaluated for each row, and only rows that satisfy the condition are included.
- The WHERE clause is used to filter records.
- The WHERE clause is used to extract only those records that fulfill a specified condition.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Note: The WHERE clause is not only used in SELECT statement, it is also used in UPDATE,

DELETE statement, etc.! (will learn in upcoming labs)

The following SQL statement selects all the employees from the FIRST_NAME "Ellen", in the "employees" table:

Example:

```
SELECT *
FROM employees
WHERE FIRST_NAME = 'Ellen';
```

If you want to get the opposite, the employees other than Ellen then query will be:

```
SELECT *
FROM employees
WHERE FIRST_NAME <> 'Ellen';
```

Also you can use "!=" at the place of "<>".

Text Field & Numeric Fields

SQL requires single quotes around text values (most database systems will also allow double quotes).

However, numeric fields should not be enclosed in quotes.

Syntax:

```
SELECT *
FROM employees
WHERE EMPLOYEE_ID = 103;
```

Operators in the WHERE clause

Operator	Type	Description
=	Comparison	Equal
>	Comparison	Greater than
<	Comparison	Less than
>=	Comparison	Greater than or equal

<=	Comparison	Less than or equal
<>	Comparison	Not equal. Note: In some versions of SQL this operator may be written as !=
AND	Logical	
OR	Logical	
NOT	Logical	
BETWEEN	Logical	Between a certain range
LIKE	Logical	Search for a pattern
IN	Logical	To specify multiple possible values for a column

Arithmetic operators in MySQL

Arithmetic Operators

+ Add

- Subtract

** Multiply*

/ Divide

% Modulo

```
-- Create a table named `ArithmeticExample`
CREATE TABLE ArithmeticExample (
   num1 INT,
   num2 INT
);
-- Insert some sample data into the table
INSERT INTO ArithmeticExample (num1, num2) VALUES (10, 5), (8, 4), (15, 3);
```

-- Display the current data in the table

```
SELECT * FROM ArithmeticExample;
```

Addition:

-- Add `num1` and `num2`

```
SELECT num1, num2, num1 + num2 AS Sum FROM ArithmeticExample;
```

Subtraction:

```
-- Subtract `num2` from `num1`
SELECT num1, num2, num1 - num2 AS Difference FROM ArithmeticExample;
```

Multiplication:

```
-- Multiply `num1` and `num2`
SELECT num1, num2, num1 * num2 AS Product FROM ArithmeticExample;
```

Division:

```
-- Divide `num1` by `num2`
SELECT num1, num2, num1 / num2 AS Quotient FROM ArithmeticExample;
```

Modulo:

```
-- Get the remainder of `num1` divided by `num2`
SELECT num1, num2, num1 % num2 AS Remainder FROM ArithmeticExample;
```

Comparison operators in MySQL

Comparison Operators

= Equal to

> Greater than

< Less than

>= Greater than or equal to

<= Less than or equal to

<> Not equal to

!= Not equal to

```
-- Create a table named `ComparisonExample`
CREATE TABLE ComparisonExample (
   value1 INT,
   value2 INT
);
```

```
-- Insert some sample data into the table
INSERT INTO ComparisonExample (value1, value2) VALUES (10, 5), (8, 8), (15, 20);

-- Display the current data in the table
SELECT * FROM ComparisonExample;
```

Equal to (=):

```
-- Check if `value1` is equal to `value2`
SELECT value1, value2 FROM ComparisonExample WHERE value1 = value2;
```

Greater than (>):

```
-- Check if `value1` is greater than `value2`
SELECT value1, value2 FROM ComparisonExample WHERE value1 > value2;
```

Less than (<):

```
-- Check if `value1` is less than `value2`
SELECT value1, value2 FROM ComparisonExample WHERE value1 < value2;
```

Greater than or equal to (>=):

```
-- Check if `value1` is greater than or equal to `value2`
SELECT value1, value2 FROM ComparisonExample WHERE value1 >= value2;
```

Less than or equal to (<=):

```
-- Check if `value1` is less than or equal to `value2`
SELECT value1, value2 FROM ComparisonExample WHERE value1 <= value2;
```

Not equal to (<> or !=):

```
-- Check if `value1` is not equal to `value2`
SELECT value1, value2 FROM ComparisonExample WHERE value1 <> value2;
```

Logical Operators

Logical Operators

ALL : TRUE if all of the subquery values meet the condition

AND : TRUE if all the conditions separated by AND is TRUE

ANY : TRUE if any of the subquery values meet the condition

BETWEEN : TRUE if the operand is within the range of comparisons

EXISTS : TRUE if the subquery returns one or more records

IN : TRUE if the operand is equal to one of a list of expressions

LIKE : TRUE if the operand matches a pattern

NOT : Displays a record if the condition(s) is NOT TRUE

OR : TRUE if any of the conditions separated by OR is TRUE

SOME : TRUE if any of the subquery values meet the condition

```
-- Create a table named `LogicalExampleTable`
CREATE TABLE LogicalExampleTable (
   id INT PRIMARY KEY,
   value INT
);
-- Insert some sample data into the table
INSERT INTO LogicalExampleTable (id, value) VALUES
   (1, 10),
   (2, 25),
   (3, 15),
   (4, 30),
   (5, 5);
```

ALL:

```
-- Display the current data in the table
SELECT * FROM LogicalExampleTable;
```

-- Check if all values are greater than 5

SELECT * FROM LogicalExampleTable WHERE value > ALL (SELECT 5);

AND:

-- Check if the value is greater than 10 AND less than 20

SELECT * FROM LogicalExampleTable WHERE value > 10 AND value < 20;

ANY/SOME:

-- Check if any value is greater than 25

SELECT * FROM LogicalExampleTable WHERE value > ANY (SELECT 25);

BETWEEN:

-- Check if the value is between 10 and 30

SELECT * FROM LogicalExampleTable WHERE value BETWEEN 10 AND 30;

EXISTS:

-- Check if there exists a value greater than 20

SELECT * FROM LogicalExampleTable WHERE EXISTS (SELECT VALUE WHERE value > 20);

IN:

-- Check if the value is in a list of specific values

SELECT * FROM LogicalExampleTable WHERE value IN (5, 15, 25);

LIKE:

-- Check if the value starts with '1'

SELECT * FROM LogicalExampleTable WHERE CAST(value AS CHAR) LIKE '1%';

NOT:

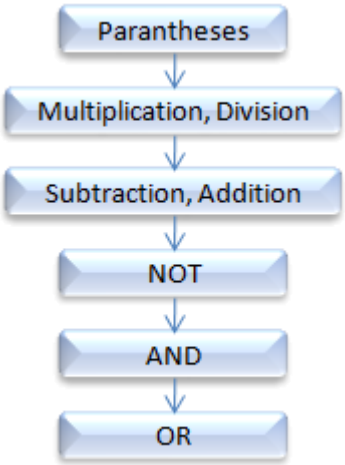
```
-- Check if the value is NOT equal to 5
SELECT * FROM LogicalExampleTable WHERE NOT value = 5;
```

OR:

```
-- Check if the value is either less than 10 OR greater than 30
SELECT * FROM LogicalExampleTable WHERE value < 10 OR value > 30;
```

SQL Operator Precedence

- Operator precedence describes the order in which operations are performed when an expression is evaluated.
- Operations with a higher precedence are performed before those with a lower precedence.
- Parentheses has the highest precedence and OR has the lowest.



Exercises (Class)

Add here all the tasks performed in lab.

Exercises (Weekly)

1. Write a query to display EMPLOYEE_ID, FIRST_NAME, and SALARY of employees whose SALARY is less than $3000.

2. Write a query to display FIRST_NAME, LASTNAME of all employees whose first name starts with letter 'A'.
3. Write a query to display FIRST_NAME, JOB_ID, DEPARTMENT_ID of employees who are either PU_CLERK or belongs to MANAGER_ID = 114.
4. Write a query to display EMPLOYEE_ID, FIRST_NAME, and SALARY of employees whose salaries lies in the range of $1500 to $3000;
5. Write a query to display EMPLOYEE_ID, FIRST_NAME, and SALARY of employees whose commission is empty.
6. Write a query to display first names of all employees that end with alphabet 'N'.
7. Write a query to display FIRST_NAME, JOB_ID, DEPARTMENT_ID of employees who are not PU_CLERK.
8. Write a query to display EMPLOYEE_ID, FIRST_NAME, and SALARY of those employees who do not have salaries of $3300, $3200, $2200.
9. Write a query to display names of those employees whose first name starts with 'A' and ends with 'N'.
10. Write a query to display the list of employee names that have letters 'LA' in their names.
11. Write a query to display the EMPLOYEE_ID, FIRST_NAME, and SALARY of employees. In that, the highest paid employee should display first and lowest paid should display last.
12. Write a query to display FIRST_NAME of employees that have "a" in the second position.
13. Write a query to display EMPLOYEE_ID, FIRST_NAME, and SALARY of employees whose salaries do not lies in the range of $1500 to $3000;
14. Write a query to display FIRST_NAME, LAST_NAME and DEPARTMENT_ID of all employees in departments 30 or 100 in ascending order.
15. Write a query to display FIRST_NAME, LAST_NAME and SALARY for all employees whose salary is not in the range $10,000 through $15,000 and are in department 30 or 100.
16. Write a query to display FIRST_NAME, LAS T_NAME and HIRE_DATE for all employees who were hired in 1987.
17. Write a query to display the LAST_NAME of employees whose LAST_NAME have exactly 6 characters.
18. Write a query to display FIRST_NAME, SALARY and PF (15% of salary) of all employees.
19. Write a query to display FIRST_NAME, SALARY and commission amount (% of salary) of all employees.
20. Write a query to display FIRST_NAME, SALARY and NET_SALARY after 500 deduction from salary of all employees;