Department of
Computer Science and Engineering

Title: Querying and Filtering data in MySQL Table (Extended)

Database System Lab

CSE 210

Green University of Bangladesh

1 Objective(s)

- To gather knowledge about Querying and filtering data with logic gates like AND OR as well as using limits.
- Learning about comparing tables in MySQL.
- To implement logic operations, comparisons and filtering data commands with limits in MySQL table.

2 Problem analysis

In SQL, all logical operators evaluate to TRUE , FALSE , or NULL (UNKNOWN). In MySQL, these are implemented as 1 (TRUE), 0 (FALSE), and NULL . ... Logical NOT. Evaluates to 1 if the operand is 0 , to 0 if the operand is nonzero, and NOT NULL returns NULL. MySQL provides a LIMIT clause that is used to specify the number of records to return. The LIMIT clause makes it easy to code multi page results or pagination with SQL, and is very useful on large tables. Returning a large number of records can impact on performance. MySQL NOT BETWEEN AND operator checks whether a value is not present between a starting and a closing expression. If expr is not greater than or equal to min and expr is not less than or equal to max, BETWEEN returns 1, otherwise, it returns 0. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

2.1 Logical Operators

In SQL, all logical operators evaluate to TRUE, FALSE, or NULL (UNKNOWN). In MySQL, these are implemented as 1 (TRUE), 0 (FALSE), and NULL. Most of this is common to different SQL database servers, although some servers may return any nonzero value for TRUE.

MySQL evaluates any nonzero, non-NULL value to TRUE. For example, the following statements all assess to TRUE:

- **NOT,** ! Logical NOT. Evaluates to 1 if the operand is 0, to 0 if the operand is nonzero, and NOT NULL returns NULL.

```
SELECT column1, column2, ... FROM table_name;NOT....
```

- **AND,** && Logical AND. Evaluates to 1 if all operands are nonzero and not NULL, to 0 if one or more operands are 0, otherwise NULL is returned.

```
SELECT column1, column2, ... FROM table_name;AND....
```

2.2 MySQL LIMIT (ORDER BY, ASC, DESC)

The LIMIT clause is used in the SELECT statement to constrain the number of rows to return. The LIMIT clause accepts one or two arguments. The values of both arguments must be zero or positive integers.

The following illustrates the LIMIT clause syntax with two arguments:

```
SELECT select_list
FROM table_name;
LIMIT [offset,] row_count;
```

2.3 Between, Not Between In, Not In

The SQL BETWEEN operator is used along with WHERE clause for providing a range of values. The values can be the numeric value, text value, and date.

```
SELECT Column(s)
FROM table_name;
WHERE column BETWEEN value1 AND value2;
```

3 Procedure (Implementation in MySQL)

1. **Using logical operators (AND, OR, NOT)):**

- **Insert Data:**

```
CREATE TABLE  ‘employees’(
emp_no            int(11)           NOT NULL,
birth_date        date              NOT NULL,
first_name        varchar(55)       NOT NULL,
last_name         varchar(55)       NOT NULL,
Gender            enum(‘M’,‘F’)     DEFAULT NULL,
Salary            int               NOT NULL,
Entry_date        datetime          NOT NULL,         DEFAULT current_timestamp(),
PRIMARY KEY(Emp_no)
);
```

- **Insert Multiple VALUES at a time:**

```
INSERT INTO employees (emp_no, birth_date, first_name,last_name, gender, salary)
VALUES            (1015312001,‘1989-08-28’, ‘Rina’ , ‘Khanam’ ,‘F’, 45000),
                  (1015312002, ‘1988-07-19’, ‘Sakib’ , ‘Hasan’ , ‘M’ , 67000),
                  (1015312003,‘1991-05-23’, ‘Sabbir’ , ‘Rahman’ , ‘M’, 32000);
```

- **Insert Single Values Must have same values as attributes number:**

```
INSERT INTO employees VALUES (1015312008, ‘1991-05-23’, ‘Sabbir’, ‘Rahman’, ‘M’, 24000, ‘2017-
11-11’);
INSERT INTO employees VALUES (1015312009, ‘1991-05-23’, ‘Sabbir’, ‘Rahman’, ‘M’, 25600, ‘2017-
11-11 21:44:35’);
```

- **MySQL AND operator examples:**

```
SELECT emp_no, first_name, last_name, salary, entry_date
FROM employees
WHERE first_name =‘Rina’ AND last_name = ‘Khanam’;
```

- **MySQL OR operator examples:**

```
SELECT emp_no, first_name, last_name, salary, entry_date
FROM employees
WHERE first_name =‘Rina’ OR last_name = ‘Khan’;
```

- **Operator precedence MySQL evaluates the OR operators after the AND operators:**

```
SELECT emp_no, first_name, last_name, salary, entry_date
FROM employees
WHERE first_name =‘Rina’ OR last_name = ‘Rahman’ AND salary <= 40000;
```

- **To change the order of evaluation, you use the parentheses, for example:**

```
SELECT emp_no, first_name, last_name, salary, entry_date
FROM employees
WHERE (first_name =‘Rina’ OR last_name = ‘Rahman’) AND salary <= 40000;
```

- **MySQL creates result for OR:**

```
SELECT emp_no, first_name, last_name, salary, entry_date
FROM employees
WHERE first_name =‘Rina’ OR last_name = ‘Rahman’;
```

2. **Using limit (ORDER BY, ASC, DESC)**

   - **Select the first 3 customers**

     ```
     SELECT emp_no, first_name, last_name, salary FROM 'employees' LIMIT 3 ;
     ```

   - **Select all attributes**

     ```
     SELECT emp_no, first_name, last_name, salary FROM 'employees' LIMIT 2,4 ;
     ```

   - **Find 4 records without first 2 records**

     ```
     SELECT emp_no, first_name, last_name, salary FROM 'employees' LIMIT 3 ;
     ```

   - **Using MySQL LIMIT to get the highest 3 values**

     ```
     SELECT emp_no, first_name, last_name, salary
     FROM 'employees'
     ORDER BY salary DESC LIMIT 3 ;
     ```

3. **Between, Not Between In, Not In:**

   - **MySQL IN examples Like OR operator**

     ```
     SELECT emp_no, first_name, last_name, salary, entry_date
     FROM employees
     WHERE salary IN (32000,40000);
     ```

   - **MySQL NOT IN examples**

     ```
     SELECT emp_no, first_name, last_name, salary, entry_date
     FROM employees
     WHERE salary NOT IN (32000,45000, 25600);
     ```

   - **MySQL BETWEEN examples**

     ```
     SELECT emp_no, first_name, last_name, salary, entry_date
     FROM employees
     WHERE salary BETWEEN 20000 AND 43000;
     ```

   - **MySQL BETWEEN to get exact values**

     ```
     SELECT emp_no, first_name, last_name, salary, entry_date
     FROM employees
     WHERE salary BETWEEN 25600 AND 42000;
     ```

   - **MySQL NOT BETWEEN to get exact values**

     ```
     SELECT emp_no, first_name, last_name, salary, entry_date
     FROM employees
     WHERE salary NOT BETWEEN 25600 AND 42000;
     ```

4. Using MySQL LIKE operator to select data based on patterns

   - MySQL LIKE examples
   - The percentage (%) wildcard allows you to match any string of zero or more characters.
   - The underscore (_) wildcard allows you to match any single character.
   - **Find employees name who has first name starting with 'm'**

     ```
     SELECT emp_no, first_name, last_name, salary, entry_date
     FROM employees
     WHERE first_name LIKE 'm%';
     ```

- **Find employees name who has first name ending with ‘r’**

```
SELECT emp_no, first_name, last_name, salary, entry_date
FROM employees
WHERE first_name LIKE ‘%r’;
```

- **Find employees name who has first name contains ‘bb’**

```
SELECT emp_no, first_name, last_name, salary, entry_date
FROM employees
WHERE first_name LIKE ‘%bb%’;
```

- **Find employees name who has first name contains first letter ‘r’ and fourth letter ‘a’**

```
SELECT emp_no, first_name, last_name, salary, entry_date
FROM employees
WHERE first_name LIKE ‘r__a’;
```

5. **Checking NULL values**

```
SELECT * FROM employees
WHERE gender IS NULL;
```

4 Discussion & Conclusion

Based on the focused objective(s) to understand about the knowledge of SELECT,WHERE, AND , BETWEEN, NOT BETWEEN and LIKE commands. The additional lab exercise made me more confident towards the fulfilment of the objectives(s)

5 Lab Task (Please implement yourself and show the output to the instructor)

- Task-1:

```
branch (branch_name, branch_city, assets)
customer (customer_id,customer_name, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)
```

1. Input multiple data existing bank database table from previous lab report.
2. Write a SQL query for searching customers who have 30000 to 50000 loan
3. Find the names of all branches located Between Dhaka and Cumilla.
4. To find all loan holders who have ‘J’ alphabets in their name or they have ‘M’ alphabets in the beginning of the names.

6 Lab Exercise (Submit as a report)

1. Input multiple data in any existing database table from previous lab report.
2. Query with primary key, query with condition, query with comparison operation.
3. Run all the queries using AND, OR, NOT, ORDER BY, ASC, DESC, Between, Not Between In, Not In, LIKE
4. Attach with query codes and with output screenshots in the report.

7 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.