Basis Data Tugas Pertemuan 12

Kelompok 10

Athillah Naufal Al-Falah – 2023071002

Daffa Ma'ruf – 2023071008

6 Quiz Knowledge Check

1. Database optimization involves optimizing only the data retrieval statements that are executed against a database.
a. True
b. ~~False~~

2. Which of the following guidelines should you follow when optimizing SQL select statements? Select all that apply.
Use OUTER JOIN instead of INNER JOIN wherever possible.
Avoid using leading wildcards in predicates.
~~Avoid using trailing wildcards in predicates.~~
~~Avoid using unnecessary columns in the SELECT clause.~~
~~Use INNER JOIN instead of OUTER JOIN wherever possible.~~

3. Identify the key characteristics of a Primary Index. Select all that apply.
1 point
A Primary Index is created using the CREATE INDEX syntax.
A Primary Key is also known as a Non-Clustered index.
~~A Primary Key is also known as a Clustered index.~~
~~A Primary Index is created automatically when a table is created.~~

4. The following query is executed on a table named Clients that contains the fields ClientID, FullName and ContactNumber.

```
SELECT ContactNumber
FROM Clients
WHERE FullName='Client Name here';
```

On which field within the table should a Secondary Index be created to optimize this query?
~~a. FullName~~
b. ContactNumber
c. ClientID

5. What keyword must be added before the following query to view its MySQL query execution plan?

```
EXPLAIN SELECT *
FROM Orders
WHERE ClientID='Cl1';
```

7 Praktikum SELECT statement optimization in MySQL

```sql
-- Buat Database Lucky_Shrub
CREATE DATABASE Lucky_Shrub;
-- Pilih & Gunakan Database Lucky_Shrub
USE Lucky_Shrub;

-- Buat Tabel Orders
CREATE TABLE Orders(
    OrderID INT NOT NULL,
    ClientID VARCHAR(10) DEFAULT NULL,
    ProductID VARCHAR(10) DEFAULT NULL,
    Quantity INT DEFAULT NULL,
    Cost DECIMAL(6,2) DEFAULT NULL,
    Date DATE DEFAULT NULL,
    PRIMARY KEY (OrderID)
);

-- Buat Tabel Employees
CREATE TABLE Employees(
    EmployeeID INT DEFAULT NULL,
    FullName VARCHAR(100) DEFAULT NULL,
    Role VARCHAR(50) DEFAULT NULL,
    Department VARCHAR(255) DEFAULT NULL
);

-- Masukkan Data ke Tabel Orders
INSERT INTO Orders (OrderID, ClientID, ProductID, Quantity, Cost, Date) VALUES
(1, 'C11', 'P1', 10, 500, '2020-09-01'),
(2, 'C12', 'P2', 5, 100, '2020-09-05'),
(3, 'C13', 'P3', 20, 800, '2020-09-03'),
(4, 'C14', 'P4', 15, 150, '2020-09-07'),
(5, 'C13', 'P3', 10, 450, '2020-09-08'),
(6, 'C12', 'P2', 5, 800, '2020-09-09'),
(7, 'C11', 'P4', 22, 1200, '2020-09-10'),
(8, 'C13', 'P1', 15, 150, '2020-09-10'),
```

```
    (9, 'C11', 'P1', 10, 500, '2020-09-12'),
    (10, 'C12', 'P2', 5, 100, '2020-09-13'),
    (11, 'C11', 'P2', 15, 80, '2020-09-12'),
    (12, 'C11', 'P1', 10, 500, '2022-09-01'),
    (13, 'C12', 'P2', 5, 100, '2022-09-05'),
    (14, 'C13', 'P3', 20, 800, '2022-09-03'),
    (15, 'C14', 'P4', 15, 150, '2022-09-07'),
    (16, 'C13', 'P3', 10, 450, '2022-09-08'),
    (17, 'C12', 'P2', 5, 800, '2022-09-09'),
    (18, 'C11', 'P4', 22, 1200, '2022-09-10'),
    (19, 'C13', 'P1', 15, 150, '2022-09-10'),
    (20, 'C11', 'P1', 10, 500, '2022-09-12'),
    (21, 'C12', 'P2', 5, 100, '2022-09-13'),
    (22, 'C12', 'P1', 10, 500, '2021-09-01'),
    (23, 'C12', 'P2', 5, 100, '2021-09-05'),
    (24, 'C13', 'P3', 20, 800, '2021-09-03'),
    (25, 'C14', 'P4', 15, 150, '2021-09-07'),
    (26, 'C11', 'P3', 10, 450, '2021-09-08'),
    (27, 'C12', 'P1', 20, 1000, '2022-09-01'),
    (28, 'C12', 'P2', 10, 200, '2022-09-05'),
    (29, 'C13', 'P3', 20, 800, '2021-09-03');

    -- Masukkan Data ke Tabel Employees
    INSERT INTO Employees (EmployeeID, FullName, Role, Department) VALUES
    (1, 'Seamus Hogan', 'Manager', 'Management'),
    (2, 'Thomas Eriksson', 'Assistant', 'Sales'),
    (3, 'Simon Tolo', 'Executive', 'Management'),
    (4, 'Francesca Soffia', 'Assistant', 'Human Resources'),
    (5, 'Emily Sierra', 'Accountant', 'Finance'),
    (6, 'Greta Galkina', 'Accountant', 'Finance'),
    (7, 'Maria Carter', 'Executive', 'Human Resources'),
    (8, 'Rick Griffin', 'Manager', 'Marketing');
```

```sql
-- Task 1: Optimized SELECT statement
SELECT OrderID, ProductID, Quantity, Date
FROM Orders;

-- Task 2: Create index and optimized query
CREATE INDEX IdxClientID ON Orders(ClientID);
EXPLAIN SELECT * FROM Orders WHERE ClientID = 'Cl1';

-- Task 3: Add ReverseFullName column, populate it, create index, and optimize query
-- Step 1: Add new column
ALTER TABLE Employees ADD COLUMN ReverseFullName VARCHAR(100);

-- Step 2: Populate ReverseFullName column
UPDATE Employees
SET ReverseFullName = CONCAT(
    SUBSTRING_INDEX(FullName, ' ', -1),
    ' ',
    SUBSTRING_INDEX(FullName, ' ', 1)
);

-- Step 3: Create index
CREATE INDEX IdxReverseFullName ON Employees(ReverseFullName);

-- Optimized query using trailing wildcard
SELECT * FROM Employees WHERE ReverseFullName LIKE 'Tolo%';
```

8 Self Review SELECT Statement Optimization In MySQL

1.

Question 1

In the first task, you optimized the following SELECT query to extract data from the **`Orders`** table:

```sql
SELECT * FROM Orders;
```

What action did you take to optimize this query?

~~a. You reduced the number of rows by adding filter criteria.~~

b. You added an index to the OrderID column.

c. You added an index to the ProductID column.

d. You rewrote the query to avoid the use of *.

2.

Question 2

In the second task, you helped Lucky Shrub retrieve the order placed by the client with the ID of Cl1. You performed this task by creating an index on the Orders table to help optimize the following query:

```sql
SELECT *
FROM Orders
WHERE ClientID='Cl1';
```

On which column in the Orders table did you create the index?

~~a. ClientID column~~

b. OrderID column

c. Quantity column

d. ProductID column

3.

Question 3

In the second task, you reviewed a query EXPLAIN plan before optimizing the query. The plan indicated NULL values in the **`possible_keys`** and **`keys`** columns. This suggests that there is no index in the targeted table which can be used to perform the search.

~~a. True~~

b. False

4.

Question 4

In the third task you performed three steps on the Orders table and then rewrote the SELECT query to use a trailing wildcard.

In the third task, Lucky Shrub used the following SELECT query, which contains a leading wildcard, to find the details of the employee whose last name is 'Tolo':

```
SELECT *
FROM Employees
WHERE FullName LIKE '%Tolo';
```

You helped to optimize this query by replacing the leading wildcard with a trailing wildcard. Why is the use of a trailing wildcard a more optimal approach?

a. A trailing wild card provides more accurate results when the SQL query is executed.

~~b. With a trailing wild card, MySQL can make use of an index created on the column to which the wildcard is assigned.~~

c. A trailing wildcard needs to match a smaller number of rows when compared to a leading wildcard.

d. A trailing wild card takes less time to identify matches when compared to a leading wildcard.