Database Object Management using DDL by Alfan Nurrohim

1. Construct and analyze queries that create, alter, and drop tables
2. Construct and analyze queries that create, alter, and drop views
3. Construct and analyze stored procedures and functions
4. Given a scenario, choose between clustered and non-clustered indexes

Database Object Management using DDL

Database Object Management using Data Definition Language (DDL) adalah proses membuat, mengubah, dan menghapus objek-objek dalam sebuah basis data dengan menggunakan perintah-perintah yang disediakan oleh DDL. DDL adalah bagian dari SQL (Structured Query Language) yang digunakan untuk mendefinisikan struktur basis data.

---

1. Construct and analyze queries that create, alter, and drop tables

Create alter and drop tables by using proper ANSI SQL syntax: NULL and NOT NUL

```
-- Creating a Table with NULL and NOT NULL Constraints:
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    PhoneNumber VARCHAR(15) NULL,
    HireDate DATE NOT NULL
);

-- Altering a Table to Add/Modify NULL and NOT NULL Constraints:
ALTER TABLE Employees
ALTER COLUMN Email SET NOT NULL;

-- Modifying an existing NOT NULL column to allow NULL values
ALTER TABLE Employees
ALTER COLUMN PhoneNumber DROP NOT NULL;
```

```
-- Adding a UNIQUE constraint to an existing column
ALTER TABLE Employees
ADD CONSTRAINT UQ_Email UNIQUE (Email);

-- Dropping (deleting) an existing table
DROP TABLE Employees;
```

2. Construct and analyze queries that create, alter, and drop views

Create alter and drop tables by using proper ANSI SQL syntax: purpose of views

CREATE VIEW (Membuat Tampilan):

- `CREATE VIEW` digunakan untuk membuat tampilan (view) dalam basis data.
- Ini menciptakan tabel virtual yang merupakan hasil dari pernyataan SELECT. Tampilan tidak menyimpan data fisik; sebaliknya, mereka hanya menampilkan data dari satu atau lebih tabel yang ada.
- Tampilan hanya menyediakan tampilan yang dapat diquery dan digunakan untuk menyederhanakan akses ke data atau mengatur data untuk keperluan pengambilan data tertentu.
- Data dalam tampilan adalah dinamis dan berasal dari tabel yang ada dalam basis data.

```
CREATE VIEW EmployeeNames AS
SELECT EmployeeID, FirstName, LastName
FROM Employees;
```

ALTER VIEW (Mengubah Tampilan):

```
ALTER
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    VIEW view_name [(column_list)]
    AS select_statement;
```

DROP VIEW (Menghapus Tampilan):

Perintah SQL `DROP VIEW` digunakan untuk menghapus tampilan yang ada dari basis data. Ini menghapus tampilan virtual dan tidak memengaruhi tabel atau data asli yang digunakan dalam pembuatan tampilan tersebut.

```
DROP VIEW CustomerContacts;
```

3. Construct and analyze stored procedures and functions

Input and output parameters, return values, purpose of stored procedures

Stored procedure adalah objek database dalam MySQL yang berisi serangkaian pernyataan SQL. Mereka diciptakan, disimpan, dan dieksekusi dalam basis data. Stored procedure memberikan cara untuk mengelompokkan pernyataan SQL ke dalam unit yang dapat digunakan ulang, membuatnya lebih mudah untuk mengelola dan memelihara operasi database yang kompleks.

Poin-poin penting tentang stored procedure MySQL:

- Stored procedure dibuat dengan menggunakan pernyataan `CREATE PROCEDURE`.
- Mereka dapat menerima parameter input, melakukan manipulasi data, dan mengembalikan hasil.
- Stored procedure dapat dieksekusi dengan menggunakan pernyataan `CALL` atau dari kode aplikasi.

💡 **Parameter Stored Procedure MySQL:**

Stored procedure dalam MySQL dapat memiliki parameter, yang memungkinkan Anda untuk mengirimkan nilai ke dalam prosed saat dieksekusi. Parameter membuat stored procedure lebih fleksibel dan dapat digunakan kembali, karena mereka dapat beke dengan berbagai set data.

Ada dua jenis parameter dalam stored procedure MySQL:

```
-- Parameter Input (IN): Parameter ini memungkinkan Anda untuk mengirimkan nilai ke dalam pr
CREATE PROCEDURE GetEmployee(IN EmployeeID INT)
BEGIN
    SELECT FirstName, LastName
    FROM Employees
    WHERE EmployeeID = EmployeeID;
END;

-- Parameter Output (OUT): Parameter ini memungkinkan prosedur untuk mengembalikan nilai ke
CREATE PROCEDURE CalculateTax(IN Salary DECIMAL(10,2), OUT Tax DECIMAL(10,2))
BEGIN
    SET Tax = Salary * 0.15;
END;
```

💡 **Stored Procedure MySQL yang Mengembalikan Nilai Ganda:**

MySQL Stored Procedures that Return Multiple Values

Stored procedure MySQL dapat mengembalikan beberapa nilai dengan berbagai cara:

```
-- Menggunakan Set Hasil:
CREATE PROCEDURE GetEmployeesByDepartment(IN DepartmentName VARCHAR(50))
BEGIN
    SELECT FirstName, LastName
    FROM Employees
    WHERE Department = DepartmentName;
END;

-- Menggunakan Tabel Sementara:
CREATE PROCEDURE GetEmployeeData(IN DepartmentName VARCHAR(50))
BEGIN
    CREATE TEMPORARY TABLE TempEmployeeData AS
    SELECT FirstName, LastName
    FROM Employees
    WHERE Department = DepartmentName;

    SELECT * FROM TempEmployeeData;
END;
```

4. Given a scenario, choose between clustered and non-clustered indexes

When to use clustered vs. not-clustered indexes, syntax for creating indexes

Index adalah sebuah objek dalam sistem database yang dapat mempercepat proses pencarian query data. Resource CPU banyak digunakan untuk pencarian data atau pengaksesan query dengan metode table-scan, index membuat pencarian data menjadi lebih optimal karena tidak banyak menghabiskan resource CPU. Saat kita membuat index, MySQL akan menyimpan data dalam struktur data B-Tree.

Clustered Index adalah jenis index yang mengatur ulang jalan catatan penyimpanan dalam tabel. Tiap tabel hanya dapat memiliki satu clustered index. Ilustrasi nya mirip dengan direktori telepon, dimana nomor-nomor telepon disusun berdasarkan urutan abjad, dan

nomor telepon yang sesuai dapat ditemukan saat mencari nama tertentu. Dengan kata lain, clustered index berisi data yang diatur secara terorganisir.

Non-Clustered Index adalah jenis index yang bekerja dengan memberikan referensi pada data dalam suatu tabel. Tiap tabel dapat memiliki lebih dari satu non-clustered index. Ilustrasi nya mirip dengan buku yang berisi indeks utama dengan judul dan nomor halaman yang sesuai dalam urutan abjad. Indeks ini tidak mengandung data aktual, tetapi menyediakan informasi yang diperlukan untuk mencapai data aktual tersebut.

CLUSTERED VS NON-CLUSTERED INDEX

Clustered	Non-clustered
Mengatur ulang (reordering) suatu tabel	Memberikan referensi pada data dalam suatu tabel
Lebih cepat daripada non-clustered	Lebih lambat daripada clustered
Membutuhkan memori yang lebih sedikit	Membutuhkan memori yang lebih banyak
Dapat menyimpan data index dalam disk	Tidak dapat menyimpan data dalam disk
Satu tabel hanya satu index	Satu tabel dapat memiliki banyak index

```
-- membuat index dengan create
create tabel sellers
(
        id int not null auto_increment
        name varchar(100),
        email varchar(100),
        primary key (id),
        unique key email_unique (email),
        index name_index (name)
) engine = innoDB;

-- menambahkan dan menghapus index dengan alter
alter table sellers
add index name_index (name);

alter table sellers
drop index name_index:
```

PENGGUNAAN INDEX

Perlu Index	Tidak Perlu Index
Tabel berukuran besar	Tabel berukuran kecil
Kolom sering digunakan dalam kondisi tertentu (WHERE, JOIN, dll)	Kolom tidak sering digunakan sebagai kondisi dalam query
Kolom berisi nilai dengan jangkauan yang luas	Kolom berisi nilai dengan jangkauan yang sempit
Kolom berisi banyak nilai null	Table sering di-update

When To Use

Clustered Index

When to Use: Use a clustered index (primary key) when you need to physically order rows in a table based on a specific column, and when that column contains unique values. This is especially useful when you frequently perform range queries, sorting, or filtering

based on that column.

Non-Clustered Index

When to Use: Use a non-clustered index (secondary index) when you want to optimize queries for columns other than the clustered column, or when you need to support various query patterns on different columns. Non-clustered indexes are also suitable for optimizing sorting, filtering, and searching on columns, even if they are not stored in that order.