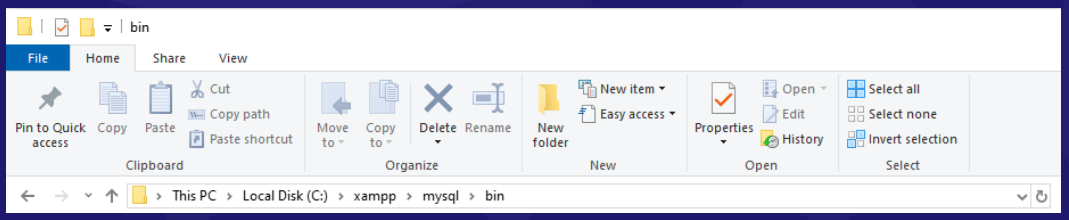
Data Manipulation Language (DML)

Modul Praktikum Sistem Manajemen Basis Data 2025

Tim Asisten
Sistem Manajemen
Basis Data 2025

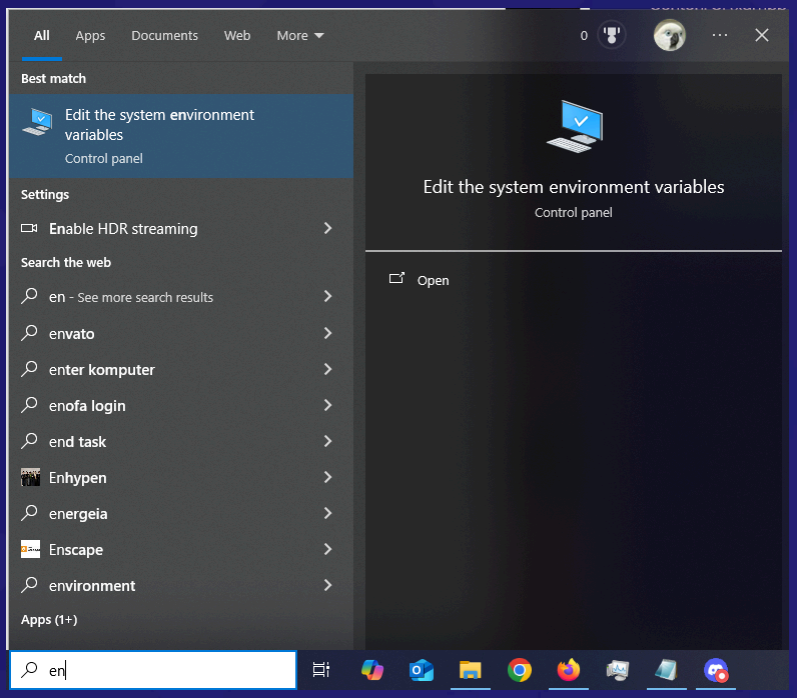
Cara Akses MySQL dengan CLI tanpa akses direktori MySQL/bin

1. Buka direktori instalasi Xampp/MySQL/bin.
   Contoh: C:\xampp\mysql\bin

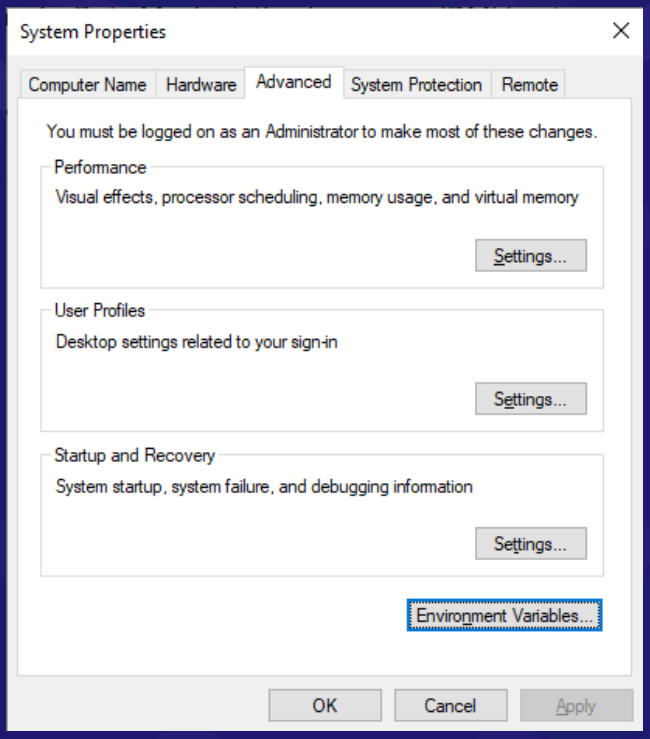


2. Klik address bar dan copy path tersebut.

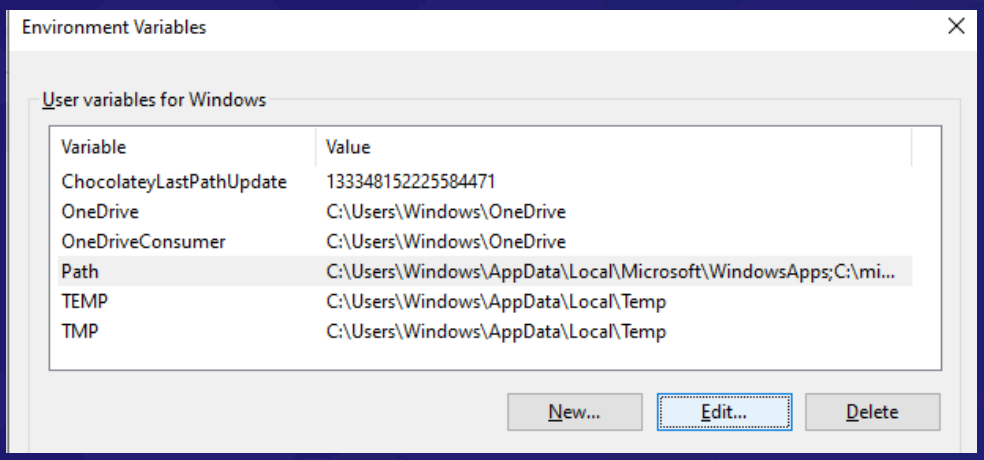
3. Tekan tombol windows dan ketik “environment” dan pilih “Edit the system environment variables”



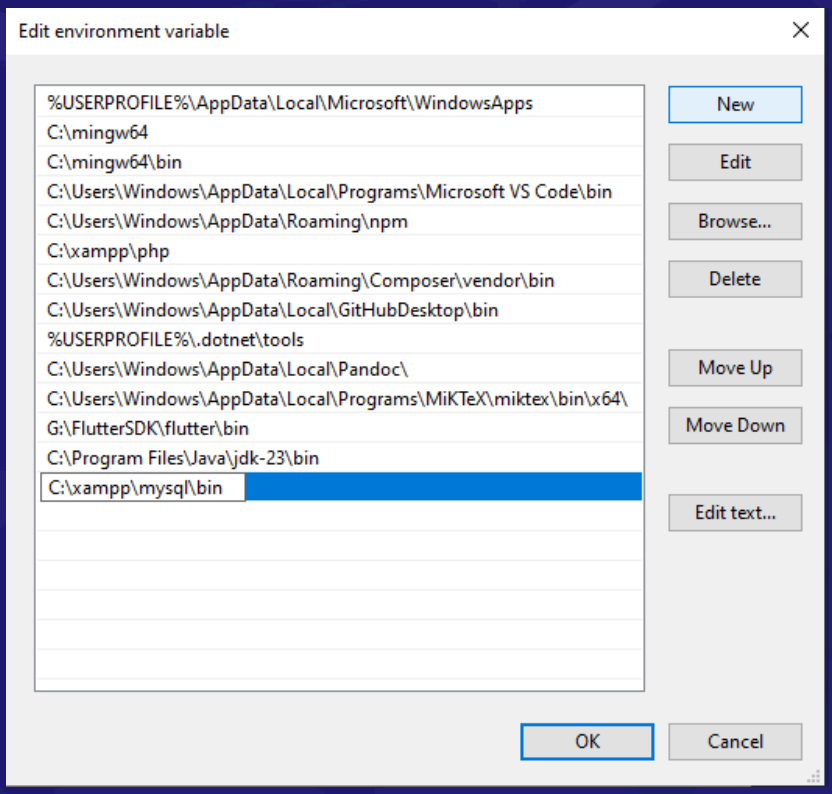
4. Klik “Environment Variables”



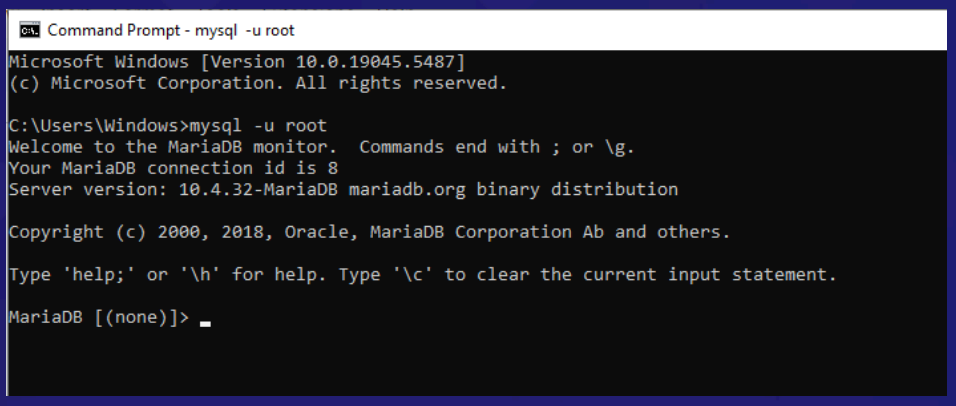
5. Pilih PATH dan klik “Edit....”



6. Klik New dan masukkan path yang kalian copy pada step 2 (pastikan tidak ada petik dua di path tersebut). Klik OK



7. Buka CMD dan ketik “mysql -u root”. Jika langkah-langkah dilakukan dengan benar, kalian dapat menjalankan MariaDB dari direktori Default (C:\Users\Windows) CMD.



Data Manipulation Language (DML)

Data Manipulation Language (DML) pada SQL digunakan untuk query dan melakukan manipulasi (insert, update, delete) isi tabel yang sudah dirancang pada Data Definition Language (DDL).

Etika penulisan Query DML:

Dalam penulisan query DML, ada beberapa hal yang menjadi etika dalam penulisannya agar lebih rapi, mudah dibaca, dan dipahami.

1. Setiap klausa pada statement harus mulai pada baris baru. Klausa seperti `SELECT`, `FROM`, `WHERE`, dan `ORDER BY` harus ditulis di baris baru agar struktur query lebih jelas.

```sql
SELECT
    first_name,
    last_name,
    salary
FROM
    employees
WHERE
    department = 'Sales'
ORDER BY
    salary DESC;
```

2. Awal dari setiap klausa harus sejajar dengan awal dari klausa lain

```sql
SELECT
    u.created_at,
    u.gender,
    u.age_at_registration
FROM
    users AS u
INNER JOIN
    payments AS p ON u.user_id = p.user_id
WHERE
    p.has_refund = FALSE
```

3. Jika sebuah klausa memiliki beberapa bagian, bagian-bagian tersebut harus berada pada baris yang berbeda dan diberi

indentasi di bawah awal klausa untuk menunjukan keterhubungan.

```sql
SELECT
    CASE postcode
        WHEN 'BN1' THEN 'Brighton'
        WHEN 'EH1' THEN 'Edinburgh'
    END AS city
FROM
    office_locations
WHERE
    country = 'United Kingdom'
    AND opening_time BETWEEN 8 AND 9
    AND postcode IN ('EH1', 'BN1', 'NN1', 'KW1');
```

4. Huruf besar digunakan untuk merepresentasikan reserved words (contoh: CREATE, SHOW, SELECT, dst)

```sql
SELECT
    created_at,
    gender,
    age_at_registration,
    province
FROM
    users
```

5. Huruf kecil digunakan untuk merepresentasikan user-defined words, seperti nama tabel, nama kolom, dst.cm

INSERT Statement

Query INSERT digunakan untuk menambahkan data ke dalam tabel. Terdapat dua cara Query INSERT:

1. Tidak menspesifikasikan kolom:

```
INSERT INTO nama_tabel
VALUES(value_1,value_2,value_3,...);
```

2. Dengan menspesifikasikan kolom:

```
INSERT INTO
nama_tabel(kolom_1,kolom_2,...)
VALUES(value_kolom_1, value_kolom_2,...);
```

Apabila menggunakan cara pertama, value harus dimasukkan secara berurutan, sesuai dengan kolom pada tabel. Berbeda dengan cara kedua, value dimasukkan sesuai dengan kolom yang dispesifikasikan.

Contoh:

1.

```
INSERT INTO
barang VALUES('', "Basket", 50000, 10);
```

2.

```
INSERT INTO
barang(nama_barang, stok, harga)
VALUES("Basket", 10, 50000);
```

UPDATE Statement

Query UPDATE digunakan untuk mengubah baris data (row) yang sudah dimasukkan ke dalam tabel, atau menambahkan data ke kolom jika NULL.

Query:

```
UPDATE nama_table
SET kolom_1 = value_1,
    kolom_2 = value_2,
    kolom_n = value_n
WHERE condition;
```

Artinya, kita mengupdate/memodifikasi kolom baris data yang memiliki kondisi yang sesuai pada WHERE.

Contoh:

```
UPDATE barang
SET nama_barang = "Bola Voli",
    stok = 12
WHERE id = 1;
```

DELETE Statement

Query DELETE digunakan untuk menghapus baris data (row) dalam tabel.

Query:

```
DELETE FROM nama_table
WHERE condition;
```

Hal yang perlu diperhatikan:

1. Jika **tidak menggunakan klausa WHERE**, data pada satu tabel tersebut akan **terhapus semua**.

2. **DELETE hanya dapat dilakukan untuk SATU TABEL** saja.

Contoh:

```
DELETE FROM barang
WHERE nama_barang = "Bola Voli";
```

SELECT Statement

Query SELECT digunakan untuk menampilkan data dari tabel berdasarkan kriteria spesifik.

Query:

```
SELECT list_kolom
FROM nama_tabel
WHERE condition
ORDER BY list_kolom ASC/DESC
GROUP BY list_kolom;
```

Contoh:

```
SELECT *
FROM barang
```

```sql
SELECT
    employee_id,
    first_name,
    last_name
FROM
    employees
WHERE
    department_id = 10;
```

Perbedaan Query DDL dan DML

Fungsi	DDL	DML
Add	Menambahkan Database: `CREATE DATABASE nama_db;`	Menambahkan Data: `INSERT INTO nama_tabel`

	Menambahkan Tabel: `CREATE TABLE nama_tabel;`	`(kolom_1, kolom_2) VALUES (value_1, value_2);`

Read	Melihat Daftar Database:
	`SHOW DATABASES;`
	Melihat Daftar Tabel:
	`SHOW TABLES;`
	Menampilkan Struktur Tabel:
	`DESC nama_tabel;`
		Membaca Isi Data:
		`SELECT * FROM nama_tabel;`

Delete	Menghapus Database:
	`DROP DATABASE nama_db;`
	Menghapus Tabel:
	`DROP TABLE nama_tabel;`
		Menghapus Data:
		`DELETE FROM nama_tabel WHERE condition;`

Update	Mengubah Struktur Tabel: `ALTER TABLE nama_tabel [ADD | DROP | RENAME | MODIFY | CHANGE COLUMN] condition;`	Mengubah Isi Data: `UPDATE nama_tabel SET kolom_1 = value_1, kolom_2 = value_2, kolom_n = value_n WHERE condition;`

OPERATOR

1. Operator Aritmatika

Operator	Deskripsi	Contoh
+	Penjumlahan	**`UPDATE barang SET stok = stok + 5 WHERE id = 2;`**
-	Pengurangan	**`UPDATE barang SET stok = stok - 5 WHERE id = 2;`**
*	Perkalian	**`UPDATE barang SET stok = stok * 2 WHERE id = 2;`**
/	Pembagian	**`UPDATE barang SET stok = stok / 2 WHERE id = 2; FROM barang`**

%	Modulo	`UPDATE barang SET stok = stok % 4 WHERE id = 2;`

2. Operator Perbandingan

Operator	Deskripsi	Contoh
=	Sama dengan	`SELECT * FROM barang WHERE nama_barang = "Shuttlecock"`
>	Lebih besar dari	`SELECT * FROM barang WHERE stok > 10`
<	Lebih kecil dari	`SELECT * FROM barang WHERE stok < 10`
>=	Lebih besar dari atau sama dengan	`SELECT * FROM barang WHERE stok <= 10`
<=	Lebih kecil dari atau sama dengan	`SELECT * FROM barang WHERE stok <= 10`
<>	Tidak sama dengan	`SELECT * FROM barang WHERE nama_barang <> "Basket"`

3. Operator Logika

Operator	Deskripsi	Contoh
ALL	Membandingkan nilai dengan **semua** hasil subquery (harus	`SELECT * FROM barang WHERE harga > ALL (SELECT harga FROM barang WHERE nama_barang = 'Shuttlecock');`

	memenuhi kondisi untuk semua nilai).	(select semua barang yang memiliki harga > Shuttlecock)
ANY	Membandingkan nilai dengan **setidaknya satu** hasil subquery (cukup ada satu yang memenuhi).	`SELECT * FROM barang WHERE harga > ANY (SELECT harga FROM barang WHERE stok < 10);` (Membandingkan dengan harga barang dengan stok lebih kecil dari 10. Jika ada setidaknya 1 barang dengan stok < 10 dengan harga lebih besar, maka muncul)
AND	TRUE apabila **semua** subquery yang dipisahkan oleh AND adalah benar.	`SELECT * FROM barang WHERE stok > 10 AND harga < 10000;`
OR	TRUE apabila **salah satu** atau **kedua** subquery yang dipisahkan oleh OR adalah benar.	`SELECT * FROM barang WHERE harga < 10000 OR stok < 10;`
NOT	Digunakan untuk **membalikkan kondisi** (TRUE menjadi FALSE dan sebaliknya).	`SELECT * FROM barang WHERE NOT (harga = 50000);`
IN	Digunakan untuk memeriksa apakah suatu nilai ada dalam **sekumpulan nilai** yang sudah ditentukan.	`SELECT * FROM barang WHERE nama_barang IN ('Basket', 'Voli', 'Raket');`

LIKE	Digunakan untuk mencari nilai yang sesuai dengan pola tertentu menggunakan wildcard: % → Mewakili **nol atau lebih** karakter. _ → Mewakili **satu** karakter saja. Notes: hanya bisa digunakan untuk string seperti CHAR, VARCHAR, TEXT, dll. Contoh penggunaan % = LIKE 'John%' Menampilkan "Johnny", "John Doe" LIKE '%sis%' Menampilkan "analisis", "rasisme", "sistem" LIKE '%si' Menampilkan "imunisasi", "nasi"	**SELECT *** **FROM barang** **WHERE nama_barang LIKE '%o%';**

	LIKE '__sis' Menampilkan "basis", "absis"	
EXISTS	Digunakan untuk memeriksa apakah sebuah **subquery menghasilkan data** (TRUE jika ada hasil).	`SELECT * FROM pelanggan WHERE EXISTS (SELECT 1 FROM booking WHERE booking.id_pelanggan = pelanggan.id);`
BETWEEN	Digunakan untuk mencari data yang berada dalam **rentang tertentu** (batas bawah dan atas termasuk).	`SELECT * FROM barang WHERE harga BETWEEN 10000 AND 80000;`
SOME	**Sama dengan ANY**, jadi bisa ditukar dengan ANY.	`SELECT * FROM barang WHERE harga > SOME (SELECT harga FROM barang WHERE stok < 10);`
LIMIT	Digunakan untuk mengambil sejumlah data terbatas.	`SELECT * FROM barang LIMIT 3;`
ORDER BY	Digunakan untuk mengurutkan hasil query berdasarkan kolom tertentu. ASC = terendah -> tertinggi DESC = tertinggi -> terendah	`SELECT * FROM barang ORDER BY stok DESC, nama_barang ASC;` Mengurutkan berdasarkan stok terbesar ke terkecil. Jika stok sama, maka diurutkan berdasarkan nama_barang secara alfabetis (A-Z).
GROUP BY	Digunakan untuk	`SELECT nama_barang,`

	mengelompokkan data berdasarkan kolom tertentu.	`SUM(stok) AS total_stok FROM barang` `GROUP BY nama_barang;`

FUNGSI AGREGAT

Fungsi agregat melakukan penghitungan pada sekumpulan nilai dan mengembalikan nilai tunggal, atau ringkasan.

Function	Deskripsi	Contoh
AVG	Menghitung **rata-rata** nilai dari kolom tertentu.	`SELECT AVG(harga) AS rata_rata_harga FROM barang;`
COUNT	Menghitung jumlah baris yang **tidak NULL** dalam kolom tertentu.	`SELECT COUNT(nama_barang) AS jumlah_nama_barang FROM barang;`
COUNT(*)	Menghitung **jumlah total baris** dalam tabel, termasuk yang **NULL**.	`SELECT COUNT(*) AS total_records FROM barang;`
MAX	Mengambil nilai **terbesar** dari kolom tertentu (bisa angka atau tanggal).	`SELECT MAX(harga) AS harga_tertinggi FROM barang;`
MIN	Mengambil nilai **terkecil** dari kolom tertentu (bisa angka atau tanggal).	`SELECT MIN(harga) AS harga_terendah FROM barang;`
SUM	Menjumlahkan **total nilai** dari suatu kolom numerik.	`SELECT SUM(harga * stok) AS total_nilai_inventori FROM barang;`

EXPORT DAN IMPORT DATABASE MENGGUNAKAN TERMINAL

- **Export**

  Di terminal di mana kita bisa mengakses MariaDB (C:\xampp\mysql\bin atau di mana saja asal sudah ditambahkan ke path environment variables), ketik perintah:

```
mysqldump -u root -p nama_database > path_file_akan_ditempatkan\nama_file.sql
```

  Contoh:

```
mysql -u root -p db_gor_olahraga > C:\Users\NASHWA\Downloads\prak_smdb.sql
```

- **Import**

  Sama seperti export, bedanya ada pada tanda '>'. Di Import, kita menggunakan '<'. Jangan lupa buat terlebih dahulu database nama_database yang akan diisi. Path yang digunakan adalah tempat di mana file .sql yang akan dipakai disimpan.

  Ketik perintah:

```
mysqldump -u root -p nama_database < path_file_yang_ditempatkan\nama_file.sql
```

  Contoh:

```
mysql -u root -p db_gor_olahraga < C:\Users\NASHWA\Downloads\prak_smdb.sql
```

Tugas

Penutup

Kami mengucapkan terima kasih kepada Tuhan, Bangsa, dan Almamater. Yang telah berkontribusi dalam pembuatan modul ini. Semoga pengetahuan dan pengalaman yang diperoleh dapat memberikan manfaat yang signifikan bagi masa depan. Aamiin.

Referensi

Asisten Praktikum Sistem Basis Data (2024). Modul Sistem Basis Data : Modul 4: SQL (DML)