

Laporan Praktikum
Mata Kuliah Pemrograman WEB



Pertemuan 5
“CRUD: Create, Read, Update & Delete”

Dosen Pengampu :
Willdan Aprizal Arifin, S.Pd., M.Kom.

Disusun Oleh :
Dzakiya Fikri Murtianingsih
2300313

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

I. PENDAHULUAN

Pada pertemuan ke-5 Mata Kuliah Pemrograman WEB ini, kami belajar mengenai CRUD: Create, Read, Update, dan Delete yang dilakukan di aplikasi Visual Code dengan alat atau aplikasi tambahan yang dibutuhkan, yaitu Node JS. CRUD biasa digunakan pada database untuk membuat, menampilkan, menambahkan, dan menghapus sebuah data. Sebelumnya kami telah mempelajari CRUD di PHPMyAdmin, namun kali ini kami mempraktikkannya di Visual Code dengan cara yang hampir sama.

II. ALAT DAN BAHAN

Alat yang digunakan pada praktikum ini adalah sebagai berikut:

1. Laptop/komputer.
2. Aplikasi Visual Code.
3. Aplikasi Node JS.

III. LANGKAH KERJA

Langkah-langkah yang dilakukan adalah sebagai berikut:

1. Buka aplikasi Visual Code, tambahkan folder baru, kemudian beralih ke aplikasi XAMPP untuk mengaktifkan MySQL. Setelah itu cek NPM dengan cara ketik “npm init -y” pada terminal, namun apabila tidak bisa, bisa dilakukan di CMD.

```
C:\Users\Lenovo L390>npm init -y
Wrote to C:\Users\Lenovo L390\package.json:

{
  "name": "lenovo-l390",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

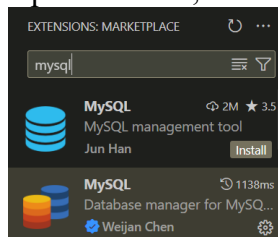
2. Ketika sudah mengecek NPM-nya, kemudian install beberapa alat dengan mengetikkan tulisan “npm install express mysql2 body-parser ejs” seperti pada gambar.

```
C:\Users\Lenovo L390> npm install express mysql2 body-parser ejs
added 93 packages, and audited 94 packages in 14s

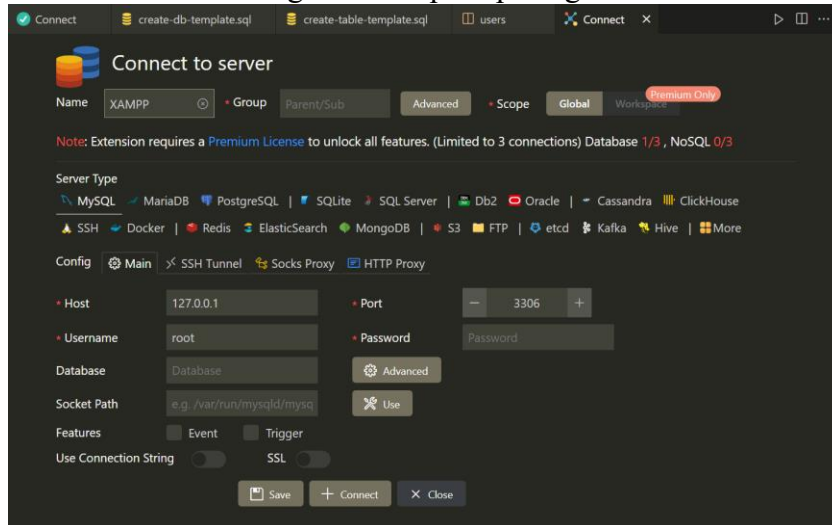
16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

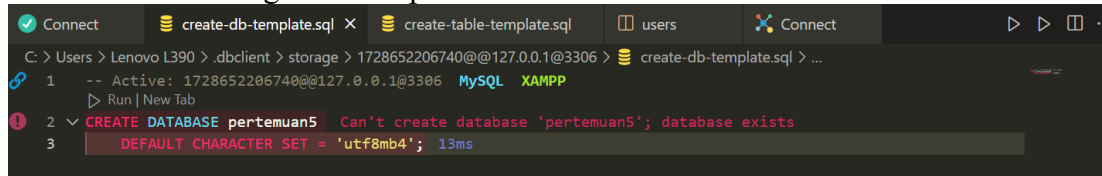
3. Apabila sudah, install extensions MySQL (pilih yang by Weijian Chen).



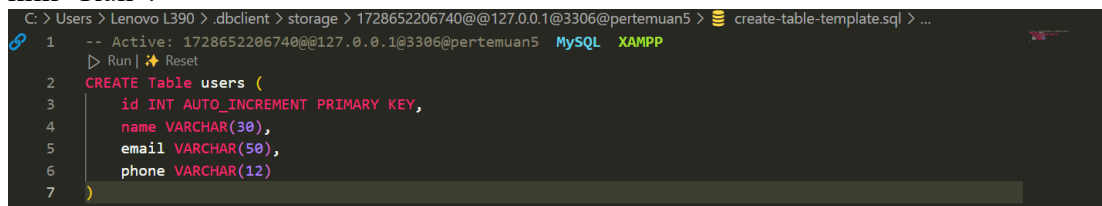
4. Connect to server dengan cara seperti pada gambar.



5. Create database dengan nama “pertemuan5”.



6. Kemudian create table, sesuaikan dengan format seperti pada gambar, setelah itu klik “Run”.



Keterangan:

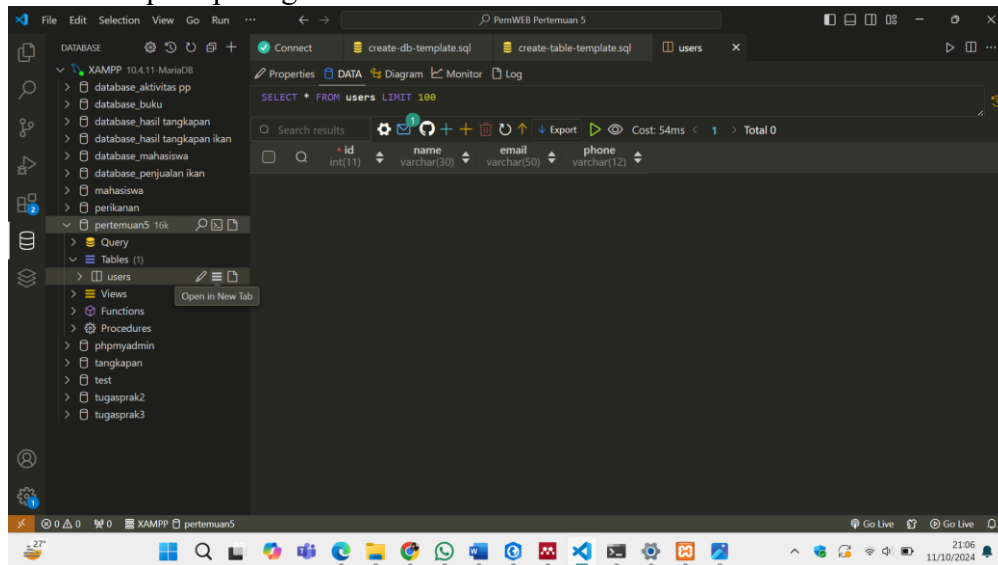
INT = tipe data bentukan.

AUTO INCREMENT = data yang dimasukkan langsung otomatis menyesuaikan.

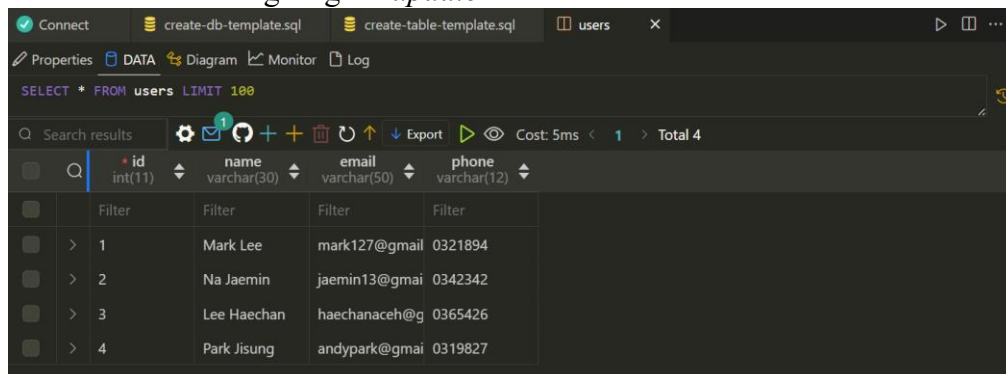
PRIMARY KEY = kata kunci yang nantinya tidak boleh sama dengan yang lainnya dan harus unik.

VARCHAR = tipe data yang digunakan untuk menyimpan karakter alfanumerik dengan panjang yang bervariasi.

7. Setelah diklik “Run” maka akan muncul tabel yang sudah dibuat sebelumnya, yaitu tabel “users”. Kemudian pilih “Open New Tab” yang nantinya akan diarahkan ke tab baru seperti pada gambar.



8. Menambahkan atribut pada tabel dengan cara klik 2 kali bagian yang ingin diisi (kecuali id). Nanti id akan otomatis terisi sesuai dengan jumlah atribut yang dimasukkan dan langsung ter-update.



9. Mengkonekkan MySQL.



10. Mengetes apakah berjalan atau tidak.

```
connection.connect((err) => {  
  if (err){  
    console.error("Terjadi kesalahan dalam koneksi ke MySQL:", err.stack);  
    return;  
  }  
  console.log("Koneksi MySQL berhasil dengan id" + connection.threadId)  
});
```

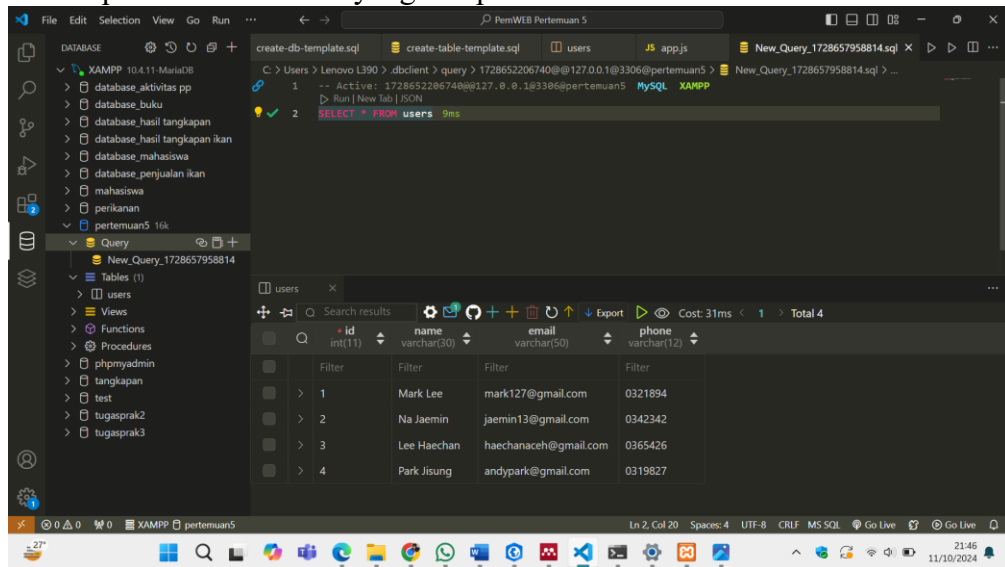
11. View engine yang nantinya digunakan untuk routing (create, read, update, dan delete).

```
app.set('view engine', 'ejs');
```

12. Ketikkan kode seperti pada gambar di bawah, kemudian copy “SELECT * FROM users” ke bagian Query.

```
create-db-template.sql create-table-template.sql users JS app.js  
JS app.js > app.length('/') callback > query  
26  
27 app.length('/', (req, res) => {  
28   const query = 'SELECT * FROM users';  
29 })
```

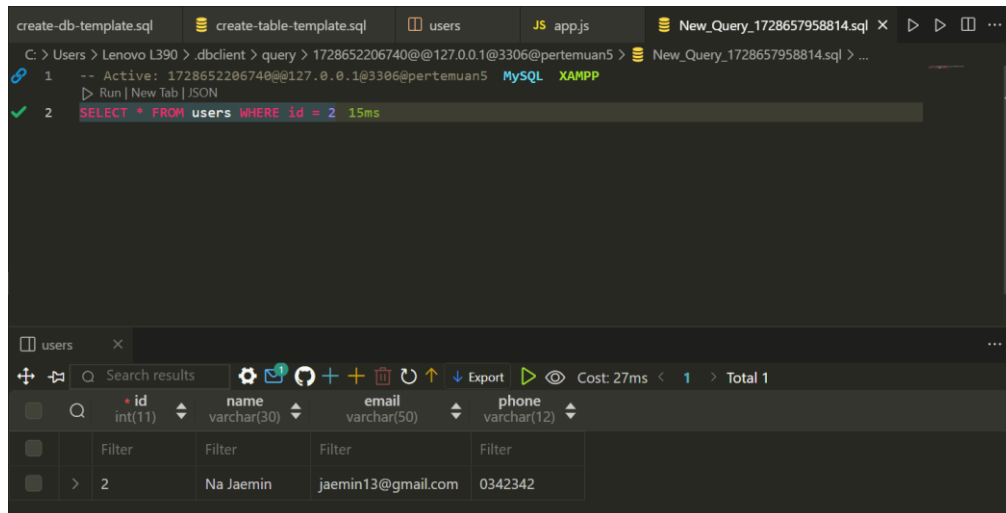
13. Copy-an “SELECT * FROM users” ke Query tersebut kemudian di “Run” untuk menampilkan seluruh data yang ada pada tabel users.



The screenshot shows a web application interface with a sidebar on the left containing a database explorer. The main area displays the results of a SQL query. The query is 'SELECT * FROM users' and the results are shown in a table with 4 rows and 4 columns: id, name, email, and phone. The table data is as follows:

id	name	email	phone
1	Mark Lee	mark127@gmail.com	0321894
2	Na Jaemin	jaemin13@gmail.com	0342342
3	Lee Haechan	haechanaceh@gmail.com	0365426
4	Park Jisung	andypark@gmail.com	0319827

Atau bisa juga apabila ingin lebih spesifik ditambahkan “SELECT * FROM users WHERE id = 2”



17. Kode untuk menambahkan data pada tabel.

```
// CREATE
app.post('/add', (req, res) => {
  const {name, email, phone} = req.body;
  const query = 'INSERT INTO users (name, email, phone) VALUES (?, ?, ?)';
})
```

18. Menambahkan data pada tabel users.

The screenshot shows a MySQL command line interface where an INSERT statement is executed to add a new user. Below the command line, a table view displays the contents of the 'users' table, showing five rows of data including the newly added user.

```
INSERT INTO users (name, email, phone) VALUES ("Chong Chenle", "chenle22@gmail.com", "03482791")
```

id	name	email	phone
1	Mark Lee	mark127@gmail.com	0321894
2	Na Jaemin	jaemin13@gmail.com	0342342
3	Lee Haechan	haechanaceh@gmail.com	0365426
4	Park Jisung	andypark@gmail.com	0319827
5	Chong Chenle	chenle22@gmail.com	03482791

Koneksi telah berhasil:

```
PS C:\Users\Lenovo L390\Documents\Semester 3\PemWEB\PemWEB Pertemuan 5> node app.js
server berjalan di port 3000, buka web melalui http://localhost:3000
Koneksi MySQL berhasil dengan id12
```

19. Kode untuk melakukan update.

```
// UPDATE
app.get('/edit/:id', (req, res) => {
  const query = 'SELECT * FROM users WHERE id = ?';
  connection.query(query, [req.params.id], (err, result) => {
    if(err) throw err;
    res.render('edit', {user: result[0]});
  });
});

app.post('/update/:id', (req, res) => {
  const {name, email, phone} = req.body;
  const query = 'UPDATE users SET name = ?, email = ?, phone = ? WHERE id = ?';
  connection.query(query, [name, email, phone, req.params.id], (err, results) => {
    if(err) throw err;
    res.redirect('/');
  });
});
```

Hasil dari update-an:

The screenshot shows a MySQL command line interface where an UPDATE statement is executed to change the name and email of the user with id 1. Below the command line, a table view displays the contents of the 'users' table, showing that the user's name is now 'Lee Minhyung' and their email is 'leeminhyung@gmail.com'.

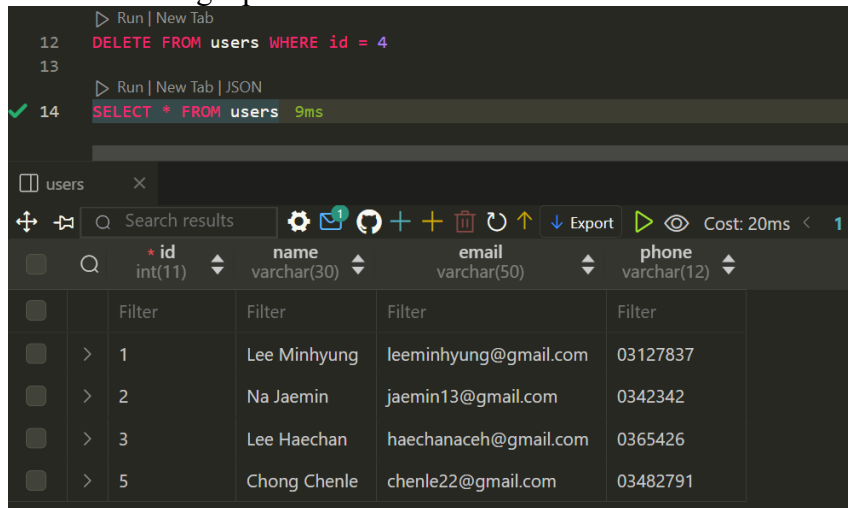
```
UPDATE users SET name = "Lee Minhyung", email = "leeminhyung@gmail.com", phone = "03127837" WHERE id = 1
```

id	name	email	phone
1	Lee Minhyung	leeminhyung@gmail.com	03127837
2	Na Jaemin	jaemin13@gmail.com	0342342

20. Kode untuk menghapus data

```
// DELETE
app.get('/delete/:id', (req, res) => {
  const query = 'DELETE FROM users WHERE id = ?';
  connection.query(query, [req.params.id], (err, result) => {
    if(err) throw err;
    res.render('edit', {user: result[0]});
  });
});
```

Hasil dari menghapus data:



The screenshot shows a database client interface. At the top, a SQL query is executed: `DELETE FROM users WHERE id = 4`. Below it, another query is shown: `SELECT * FROM users` with a execution time of 9ms. The main part of the screenshot is a table view of the 'users' table. The table has four columns: 'id' (int(11)), 'name' (varchar(30)), 'email' (varchar(50)), and 'phone' (varchar(12)). There are four rows of data displayed.

	* id int(11)	name varchar(30)	email varchar(50)	phone varchar(12)
>	1	Lee Minhyung	leeminhyung@gmail.com	03127837
>	2	Na Jaemin	jaemin13@gmail.com	0342342
>	3	Lee Haechan	haechanaceh@gmail.com	0365426
>	5	Chong Chenle	chenle22@gmail.com	03482791

IV. KESIMPULAN

Pada praktikum ini telah mempelajari bagaimana menambah, menghapus, menampilkan, dan mengupdate data pada database menggunakan Visual Code.