

DB FACADE (RAW SQL QUERIES)

1.1 Kompetensi Utama

- Diharapkan mahasiswa dapat memahami DB Facade pada laravel.

1.2 Kompetensi Penunjang

- Mengenalkan kepada mahasiswa tentang DB Façade
- Membuat query SQL menggunakan DB Facade pada laravel

1.3 Dasar Teori

1. Pengertian DB Facade

DB facade (Raw SQL Queries) adalah *facade* class bawaan Laravel yang berisi berbagai method untuk menjalankan query SQL. Di sebut sebagai raw (mentah), karena query langsung ditulis sebagaimana yang biasa diinput ke dalam mysqli extension atau PDO seperti 'SELECT * FROM mahasiswas', 'INSERT INTO mahasiswas...', 'UPDATE mahasiswas SET...', dst.

Raw query merupakan cara paling dasar di dalam Laravel, Raw query juga sangat familiar karena sudah biasa kita pakai ketika membuat aplikasi native. Untuk mempraktekkan DB Facade pastikan file migrasi dan controller mahasiswa pada pertemuan sebelumnya sudah dibuat. Untuk file **route/web.php** tambahkan beberapa route di bawah ini :

```
12 | Here is where you can register web routes for your application. These
13 | routes are loaded by the RouteServiceProvider and all of them will
14 | be assigned to the "web" middleware group. Make something great!
15 |
16 */
17
18 Route::get('/', function () {
19     return view('welcome');
20 });
21 Route::get('/insert-sql', [MahasiswaController::class, 'insertSql']);
22 Route::get('/insert-prepared', [MahasiswaController::class, 'insertPrepared']);
23 Route::get('/insert-binding', [MahasiswaController::class, 'insertBinding']);
24 Route::get('/update', [MahasiswaController::class, 'update']);
25 Route::get('/delete', [MahasiswaController::class, 'delete']);
26 Route::get('/select', [MahasiswaController::class, 'select']);
27 Route::get('/select-tampil', [MahasiswaController::class, 'selectTampil']);
28 Route::get('/select-view', [MahasiswaController::class, 'selectView']);
29 Route::get('/select-where', [MahasiswaController::class, 'selectWhere']);
30 Route::get('/statement', [MahasiswaController::class, 'statement']);
31
```

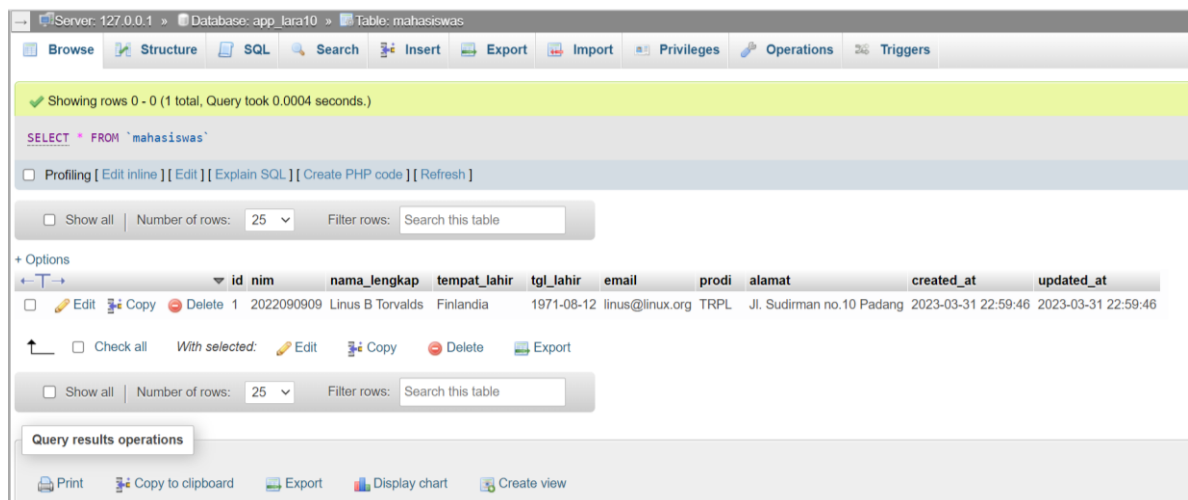
2. Input Data (DB::insert)

Perintah raw query pertama yang akan kita praktekan adalah proses input data menggunakan DB::insert() facade. Buat method insertSQL pada MahasiswaController dan lakukan query insert seperti di bawah ini :

```
app > Http > Controllers > MahasiswaController.php > MahasiswaController > insertSql
1  <?php
2  namespace App\Http\Controllers;
3
4  use Illuminate\Support\Facades\DB;
5
6  class MahasiswaController extends Controller {
7
8      public function insertSql()
9      {
10         $query=DB::insert("INSERT INTO mahasiswas(nim,nama_lengkap,tempat_lahir,
11         tgl_lahir,email,prodi,alamat,created_at,updated_at) VALUES ('2022090909',
12         'Linus B Torvalds','Finlandia','1971-08-12','linus@linux.org','TRPL',
13         'Jl. Sudirman no.10 Padang',now(),now())");
```

Dalam MahasiswaController ini kita memerlukan **DB facade**, sehingga butuh perintah untuk mengimport-nya, yakni dengan kode Use Illuminate\Support\Facades\DB.

Jalankan localhost:8000/insert-sql pada browser, untuk memastikan data berhasil tersimpan, silahkan buka phpmyadmin dan lihat isi tabel mahasiswas.



The screenshot shows the phpMyAdmin interface for the 'mahasiswas' table. The table contains one row of data for Linus B Torvalds. The interface includes a top navigation bar, a toolbar with various actions, and a table view with columns for personal and contact information.

	id	nim	nama_lengkap	tempat_lahir	tgl_lahir	email	prodi	alamat	created_at	updated_at
<input type="checkbox"/>	1	2022090909	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-03-31 22:59:46	2023-03-31 22:59:46

Terlihat data mahasiswa 'Linus B Torvalds' sudah berhasil diinput ke dalam tabel mahasiswas

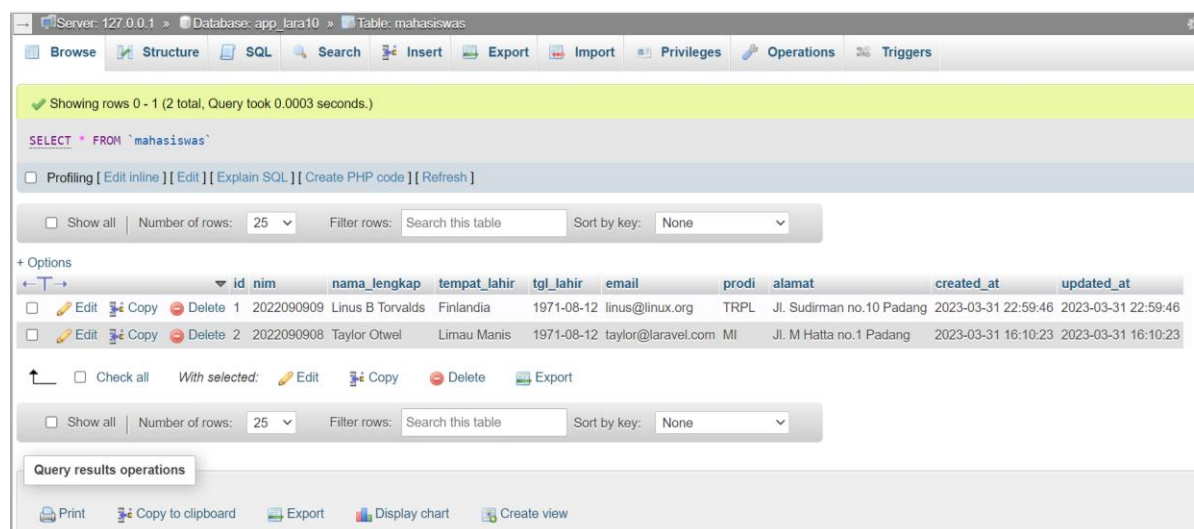
3. Membuat Prepared Statement

Yang kita lakukan pada method insertSQL di atas adalah menjalankan query "apa adanya" dan belum aman. Supaya lebih aman, query ini lebih baik ditulis dengan teknik **prepared statement**, dimana terdapat pemisahan antara perintah SQL dengan data yang akan diinput. Apalagi data nantinya banyak berasal dari inputan form yang sangat rawan diisi nilai-nilai aneh, termasuk resiko dari SQL Injection.

Untuk membuat prepared statement, caranya cukup mudah. Kita tetap tulis query seperti biasa ke dalam method DB::insert(), tapi kali ini mengganti nilai data menjadi *placeholder* dengan karakter tanda tanya " ? ". Kemudian input sebuah array di argument kedua method DB::insert() untuk menggantikan *placeholder* tadi. Berikut contoh penulisannya:

```
app > Http > Controllers > MahasiswaController.php > MahasiswaController
7
8     public function insertSql()
9     {
10         $query=DB::insert("INSERT INTO mahasiswa(nim,nama_lengkap,tempat_lahir,
            tgl_lahir,email,prodi,alamat,created_at,updated_at) VALUES ('2022090909',
            'Linus B Torvalds','Finlandia','1971-08-12','linus@linux.org','TRPL',
            'Jl. Sudirman no.10 Padang',now(),now())");
11     }
12     public function insertPrepared()
13     {
14         $query=DB::insert("INSERT INTO mahasiswa(nim,nama_lengkap,tempat_lahir,
            tgl_lahir,email,prodi,alamat,created_at,updated_at) VALUES
            (?, ?, ?, ?, ?, ?, ?, ?)",['2022090908','Taylor Otwell','Limau Manis',
            '1971-08-12','taylor@laravel.com','MI','Jl. M Hatta no.1 Padang',now(),
            now()]);
15     }
16
```

Jalankan method **insertPrepared()** dengan mengakses alamat: localhost:8000/insert-prepared, kemudian lihat hasilnya di phpmyadmin



The screenshot shows the phpMyAdmin interface for a database named 'app_lara10'. The 'Table: mahasiswa' is selected. The SQL tab is active, showing a query that inserted two rows. The 'Query results operations' section at the bottom shows the resulting data in a table.

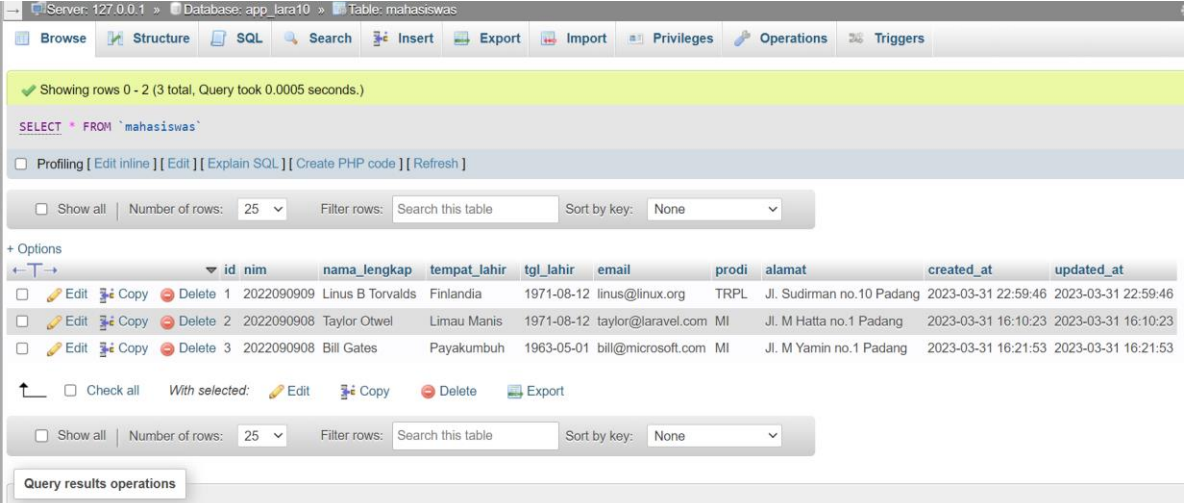
	id	nim	nama_lengkap	tempat_lahir	tgl_lahir	email	prodi	alamat	created_at	updated_at
<input type="checkbox"/>	1	2022090909	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-03-31 22:59:46	2023-03-31 22:59:46
<input type="checkbox"/>	2	2022090908	Taylor Otwell	Limau Manis	1971-08-12	taylor@laravel.com	MI	Jl. M Hatta no.1 Padang	2023-03-31 16:10:23	2023-03-31 16:10:23

4. Membuat Named Binding

Selain menggunakan placeholder berupa tanda tanya, method `DB::insert()` juga bisa memakai teknik **named parameter** seperti contoh berikut:

```
app > Http > Controllers > MahasiswaController.php > MahasiswaController > insertBinding
15         '1971-08-12', 'taylor@laravel.com', 'MI', 'Jl. M Hatta no.1 Padang', now(),
16         now()]);
17
18     public function insertBinding()
19     {
20         $query=DB::insert("INSERT INTO mahasiswa(nim,nama_lengkap,tempat_lahir,
21         tgl_lahir,email,prodi,alamat,created_at,updated_at) VALUES (:nim,
22         :nama_lengkap,:tempat_lahir,:tgl_lahir,:email,:prodi,:alamat,:created_at,
23         :updated_at)",
24         [
25             'nim'=>'2022090908',
26             'nama_lengkap' =>'Bill Gates',
27             'tempat_lahir'=>'Payakumbuh',
28             'tgl_lahir'=>'1963-05-1',
29             'email'=>'bill@microsoft.com',
30             'prodi'=>'MI',
31             'alamat'=>'Jl. M Yamin no.1 Padang',
32             'created_at'=>now(),
33             'updated_at'=>now()
34         ]
35     );
36 }
```

Jalankan method `insertBinding()` dengan mengakses alamat: `localhost:8000/insert-binding`, kemudian lihat hasilnya di phpmyadmin



The screenshot shows the phpMyAdmin interface for a database named 'app_lara10'. The 'mahasiswa' table is selected, and the 'SQL' tab is active. The table contains 3 rows of data. The query results are displayed below the table.

	id	nim	nama_lengkap	tempat_lahir	tgl_lahir	email	prodi	alamat	created_at	updated_at
<input type="checkbox"/>	1	2022090909	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-03-31 22:59:46	2023-03-31 22:59:46
<input type="checkbox"/>	2	2022090908	Taylor Otwel	Limau Manis	1971-08-12	taylor@laravel.com	MI	Jl. M Hatta no.1 Padang	2023-03-31 16:10:23	2023-03-31 16:10:23
<input type="checkbox"/>	3	2022090908	Bill Gates	Payakumbuh	1963-05-01	bill@microsoft.com	MI	Jl. M Yamin no.1 Padang	2023-03-31 16:21:53	2023-03-31 16:21:53

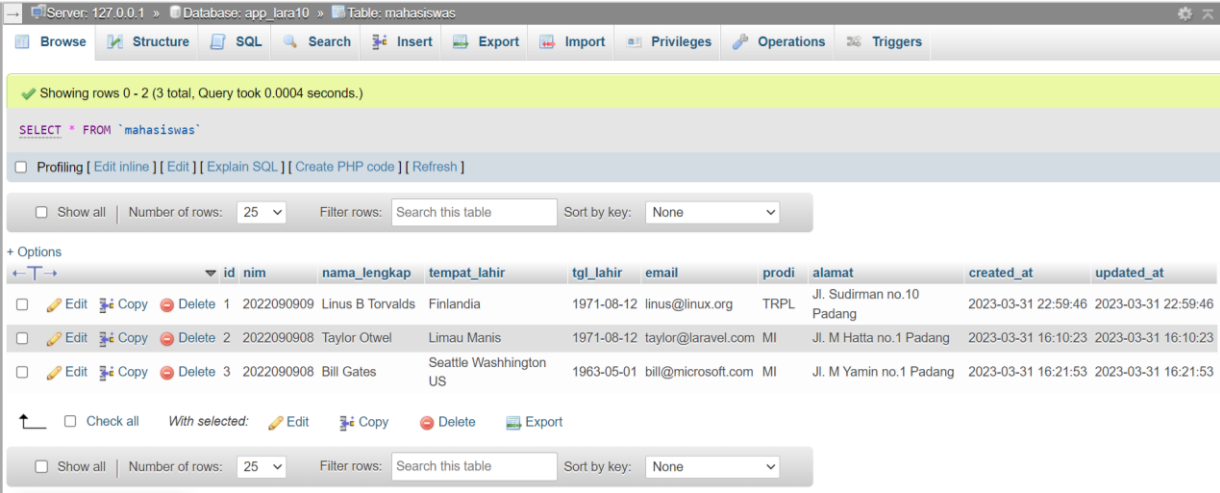
5. Update Data (DB::update)

Untuk proses update data, DB facade menyediakan method `DB::update()`. Cara penggunaan method ini sama seperti `DB::insert()`, dimana kita menulis query SQL sebagai

argument pertama, kemudian sebuah array nilai di argument kedua jika menggunakan prepared statement. Berikut contoh penggunaannya:

```
22         'nama_lengkap' => 'Bill Gates',
23         'tempat_lahir' => 'Payakumbuh',
24         'tgl_lahir' => '1963-05-1',
25         'email' => 'bill@microsoft.com',
26         'prodi' => 'MI',
27         'alamat' => 'Jl. M Yamin no.1 Padang',
28         'created_at' => now(),
29         'updated_at' => now()
30     ];
31 }
32 public function update()
33 {
34     $query = DB::update("UPDATE mahasiswa SET tempat_lahir = 'Seattle
35     Washington US' WHERE nama_lengkap=?", ['Bill Gates']);
36 }
```

Jalankan method update() dengan mengakses alamat: localhost:8000/update



Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

SELECT * FROM 'mahasiswa'

Number of rows: 25 Filter rows: Search this table Sort by key: None

				id	nim	nama_lengkap	tempat_lahir	tgl_lahir	email	prodi	alamat	created_at	updated_at
<input type="checkbox"/>	Edit	Copy	Delete	1	2022090909	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-03-31 22:59:46	2023-03-31 22:59:46
<input type="checkbox"/>	Edit	Copy	Delete	2	2022090908	Taylor Otwell	Limau Manis	1971-08-12	taylor@laravel.com	MI	Jl. M Hatta no.1 Padang	2023-03-31 16:10:23	2023-03-31 16:10:23
<input type="checkbox"/>	Edit	Copy	Delete	3	2022090908	Bill Gates	Seattle Washington US	1963-05-01	bill@microsoft.com	MI	Jl. M Yamin no.1 Padang	2023-03-31 16:21:53	2023-03-31 16:21:53

Number of rows: 25 Filter rows: Search this table Sort by key: None

6. Menghapus Data (DB::delete)

Untuk menghapus data tabel, DB facade menyediakan method DB::delete(). Penggunaannya juga sama seperti method DB::insert() dan DB::update(). Yang berbeda hanya di penulisan perintah SQL saja

```

30     });
31 }
32 public function update()
33 {
34     $query=DB::update("UPDATE mahasiswa SET tempat_lahir = 'Seattle
Washington US' WHERE nama_lengkap=?",['Bill Gates']);
35 }
36
37 public function delete()
38 {
39     $query=DB::delete("DELETE FROM mahasiswa WHERE nama_lengkap=?",['Bill
Gates']);
40 }
41

```

Setelah di jalankan, Hasil data mahasiswa tinggal 2

Showing rows 0 - 1 (2 total, Query took 0.0005 seconds.)

SELECT * FROM 'mahasiswa'

Number of rows: 25 Filter rows: Search this table Sort by key: None

		id	nim	nama_lengkap	tempat_lahir	tgl_lahir	email	prodi	alamat	created_at	updated_at
<input type="checkbox"/>	Edit	1	2022090909	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-03-31 22:59:46	2023-03-31 22:59:46
<input type="checkbox"/>	Edit	2	2022090908	Taylor Otwell	Limau Manis	1971-08-12	taylor@laravel.com	MI	Jl. M Hatta no.1 Padang	2023-03-31 16:10:23	2023-03-31 16:10:23

Number of rows: 25 Filter rows: Search this table Sort by key: None

7. Menampilkan Data (DB::select)

Buat method select pada MahasiswaController dengan query seperti di bawah ini :

```

36
37 public function delete()
38 {
39     $query=DB::delete("DELETE FROM mahasiswa WHERE nama_lengkap=?",['Bill
Gates']);
40 }
41
42 public function select()
43 {
44     $query=DB::select("SELECT * FROM mahasiswa");
45     dd($query);
46 }
47

```

Pada query 'SELECT * FROM mahasiswa' yang hasilnya disimpan ke dalam variabel \$query untuk kemudian di dd(), Untuk menjalankan method ini, silahkan akses localhost:8000/select:


```

array:2 [▼ // app\Http\Controllers\MahasiswaController.php:45
  0 => {#307 ▼
    +"id": 1
    +"nim": "2022090909"
    +"nama_lengkap": "Linus B Torvalds"
    +"tempat_lahir": "Finlandia"
    +"tgl_lahir": "1971-08-12"
    +"email": "linus@linux.org"
    +"prodi": "TRPL"
    +"alamat": "Jl. Sudirman no.10 Padang"
    +"created_at": "2023-03-31 22:59:46"
    +"updated_at": "2023-03-31 22:59:46"
  }
  1 => {#309 ▼
    +"id": 2
    +"nim": "2022090908"
    +"nama_lengkap": "Taylor Otwell"
    +"tempat_lahir": "Limau Manis"
    +"tgl_lahir": "1971-08-12"
    +"email": "taylor@laravel.com"
    +"prodi": "MI"
    +"alamat": "Jl. M Hatta no.1 Padang"
    +"created_at": "2023-03-31 16:10:23"
    +"updated_at": "2023-03-31 16:10:23"
  }
]

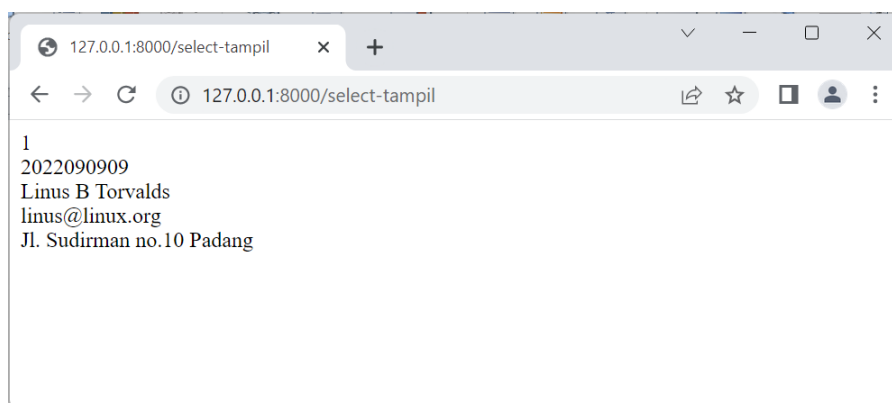
```

8. Menampilkan Data dari Object

```

41
42 public function select()
43 {
44     $query=DB::select("SELECT * FROM mahasiswa");
45     dd($query);
46 }
47 public function selectTampil()
48 {
49     $query=DB::select("SELECT * FROM mahasiswa");
50     echo ($query[0]->id) . "<br />";
51     echo ($query[0]->nim) . "<br />";
52     echo ($query[0]->nama_lengkap) . "<br />";
53     echo ($query[0]->email) . "<br />";
54     echo ($query[0]->alamat);
55 }
56

```



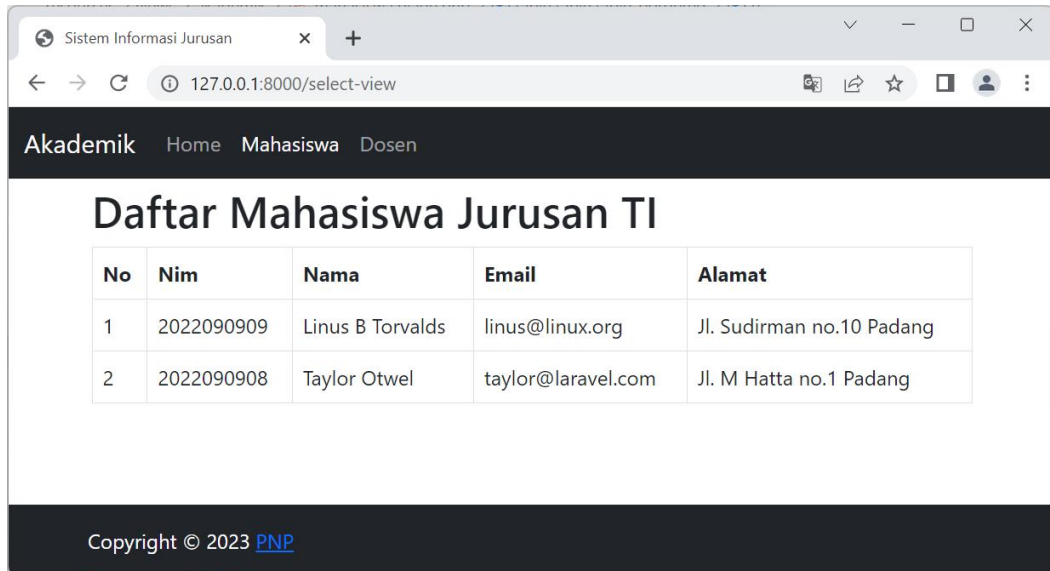
9. Menampilkan Data dengan View

```
app > Http > Controllers > MahasiswaController.php > MahasiswaController > selectTampil
48 {
49     $query=DB::select("SELECT * FROM mahasiswa");
50     echo ($query[0]->id) . "<br />";
51     echo ($query[0]->nim) . "<br />";
52     echo ($query[0]->nama_lengkap) . "<br />";
53     echo ($query[0]->email) . "<br />";
54     echo ($query[0]->alamat);
55 }
56
57 public function selectView()
58 {
59     $query=DB::select("SELECT * FROM mahasiswa");
60     return view('akademik.mahasiswa',['mahasiswas'=>$query]);
61 }
62
```

Perintah di baris 60 akan mengirim data ['mahasiswas' => \$query] ke dalam view mahasiswa.blade.php. Dengan demikian, di dalam view kita bisa mengakses semua array yang tersimpan di dalam \$query melalui variabel \$mahasiswas.

Berikutnya, silahkan buka file mahasiswa.blade.php di folder views\akademik. Kemudian ubah dengan kode berikut:

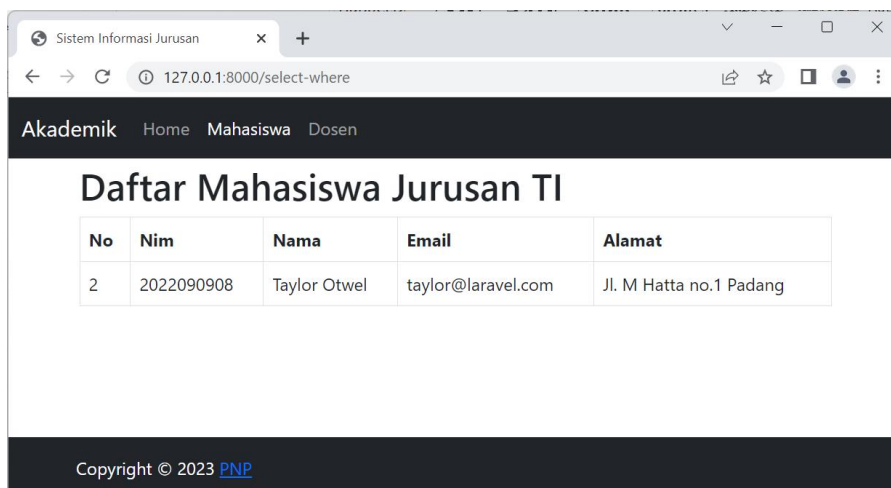
```
resources > views > akademik > mahasiswa.blade.php > table.table-bordered > tr
1 @extends('layouts.main')
2 @section('title')
3 @section('navMhs','active')
4
5 @section('content')
6 <h1>Daftar Mahasiswa Jurusan TI</h1>
7 <table class="table table-bordered">
8     <tr>
9         <th>No</th>
10        <th>Nim</th>
11        <th>Nama</th>
12        <th>Email</th>
13        <th>Alamat</th>
14    </tr>
15    @foreach ($mahasiswas as $mahasiswa)
16        <tr>
17            <td>{{ $mahasiswa->id }}</td>
18            <td>{{ $mahasiswa->nim }}</td>
19            <td>{{ $mahasiswa->nama_lengkap }}</td>
20            <td>{{ $mahasiswa->email }}</td>
21            <td>{{ $mahasiswa->alamat }}</td>
22        </tr>
23    @endforeach
24 </table>
25 @endsection
26
```

10. Select Where

Sebagai contoh, saya ingin membuat method baru untuk menampilkan data mahasiswa yang memiliki prodi MI dan di urutkan berdasarkan nama:

```
app > Http > Controllers > MahasiswaController.php > MahasiswaController
54         echo ($query[0]->alamat);
55     }
56
57     public function selectView()
58     {
59         $query=DB::select("SELECT * FROM mahasiswas");
60         return view('akademik.mahasiswa',['mahasiswas'=>$query]);
61     }
62
63     public function selectWhere()
64     {
65         $query=DB::select("SELECT * FROM mahasiswas WHERE prodi=? ORDER BY nim
66         ASC,['MI']");
67         return view('akademik.mahasiswa',['mahasiswas'=>$query]);
68     }
```

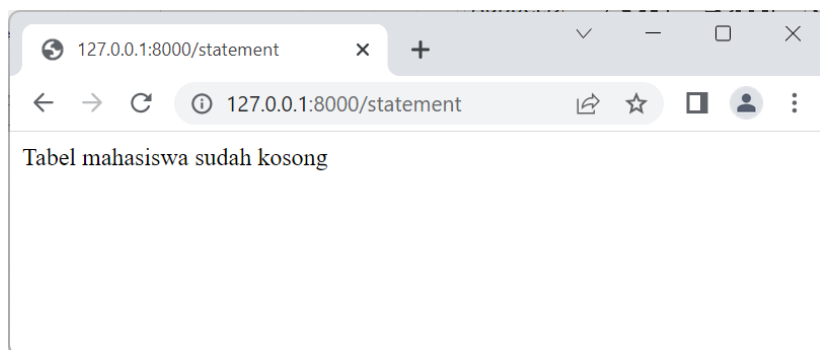


11. Menjalankan Query (DB::statement)

Method raw query yang kita bahas sejauh ini sudah mencakup semua keperluan **CRUD**, yakni `DB::insert()` untuk proses **create**, `DB::select()` untuk proses **read**, `DB::update()` untuk proses **update**, dan `DB::delete()` untuk proses **delete**.

Selain itu, DB facade juga menyediakan method `DB::statement()` sebagai method 'generik' untuk menjalankan query lain yang tidak termasuk 4 kategori CRUD tadi. Cara penggunaannya juga sama, kita tinggal menulis query SQL sebagai argument ke dalam method ini. Sebagai contoh, kita ingin menjalankan query `TRUNCATE mahasiswa` untuk menghapus semua data di tabel mahasiswa:

```
app > Http > Controllers > MahasiswaController.php > MahasiswaController > selectWhere
58 {
59     $query=DB::select("SELECT * FROM mahasiswa");
60     return view('akademik.mahasiswa',['mahasiswas'=>$query]);
61 }
62
63 public function selectWhere()
64 {
65     $query=DB::select("SELECT * FROM mahasiswa WHERE prodi=? ORDER BY nim
66     ASC,['MI']");
67     return view('akademik.mahasiswa',['mahasiswas'=>$query]);
68 }
69
70 public function statement()
71 {
72     $query=DB::statement("TRUNCATE mahasiswa");
73     return ('Tabel mahasiswa sudah kosong');
74 }
```



Hasilnya, tabel mahasiswa kembali kosong. Method `DB::statement()` ini juga bisa dipakai untuk menjalankan query-query lain.