

MODUL 9

VIEW

A. TUJUAN

- Memahami konsep dasar view di dalam basis data
- Memahami implementasi view, termasuk algoritma dan jenis-jenisnya yang tersedia
- Mampu menyelesaikan kasus-kasus pengambilan data dengan menggunakan pendekatan view

B. PETUNJUK

- Awali setiap aktivitas dengan do'a, semoga berkah dan mendapat kemudahan.
- Pahami tujuan, dasar teori, dan latihan-latihan praktikum dengan baik dan benar.
- Kerjakan tugas-tugas praktikum dengan baik, sabar, dan jujur.
- Tanyakan kepada asisten/dosen apabila ada hal-hal yang kurang jelas

C. DASAR TEORI

1. View

View dapat didefinisikan sebagai tabel maya (*virtual*) atau *logical* yang terdiri dari himpunan hasil *query*. Tidak seperti pada umumnya tabel di dalam basis data relasional, *view* bukanlah bagian dari skema fisik. *View* bersifat dinamis, ia mengandung data dari tabel yang direpresentasikannya. Dengan demikian, ketika tabel yang menjadi sumber datanya berubah, data di *view* juga akan berubah.

Merujuk pda dokumentasi MySQL, sintaks pendefinisian view diperlihatkan sebagai berikut :

```
CREATE
  [OR REPLACE]
  [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
  [DEFINER = { user | CURRENT USER }]
  [SQL SECURITY { DEFINER | INVOKER }]
  VIEW view_name [(column_list)]
  AS select statement
  [WITH [CASCADED | LOCAL] CHECK OPTION]
```

2. Updatable View

View dapat berisi *read-only* atau *updatable*. Kondisi ini sangat dipengaruhi oleh adanya pendefinisian *view* itu sendiri. Bagaimanapun, untuk menciptakan *updatable view*, pernyataan `SELECT` yang didefinisikan di *view* harus mengikuti aturan-aturan sebagai berikut :

- Pernyataan `SELECT` tidak boleh merujuk ke lebih dari satu tabel.
- Pernyataan `SELECT` tidak boleh menggunakan klausa `GROUP BY` atau `HAVING`.
- Pernyataan `SELECT` harus tidak menggunakan `DISTINCT`.
- Pernyataan `SELECT` harus tidak merujuk ke *view* lain yang tidak *updatable*.
- Pernyataan `SELECT` tidak boleh mengandung ekspresi apa pun, misalnya fungsi agregat.

Pada hakikatnya, jika sistem *database* mampu menentukan pemetaan balik dari skema *view* ke skema tabel dasar, maka *view* memungkinkan untuk di *update*. Dalam kondisi ini, operasi-operasi `INSERT`, `UPDATE` dan `DELETE` dapat diterapkan pada *view*.

D. LATIHAN

1. Himpunan Entitas

Dalam latihan ini, digunakan 5 buah tabel yaitu tabel mahasiswa, matakuliah, `ambil_mk`, dosen, dan jurusan. Untuk itu, ciptakan terlebih dahulu tabel-tabel tersebut apabila belum ada. Dibawah ini adalah data yang digunakan oleh masing-masing tabel untuk praktikum ini:

Tabel Mahasiswa

Nim	nama	Jenis_kelamin	Alamat
101	Arif	L	Jl. Kenangan
102	Budi	L	Jl. Jombang
103	Wati	P	Jl. Surabaya
104	Ika	P	Jl. Jombang
105	Tono	L	Jl. Jakarta
106	Iwan	L	Jl. Bandung
107	Sari	P	Jl. Malang

Praktikum Basis Data 2017 – TE UM

Tabel Dosen

Kode_dos	Nama_dos	Alamat_dos
10	Suharto	Jl. Jombang
11	Martono	Jl. Kalpataru
12	Rahmawati	Jl. Jakarta
13	Bambang	Jl. Bandung
14	Nurul	Jl. Raya Tidar

Tabel Matakuliah

Kode_mk	Nama_mk	Sks	Semester	Kode_dos
PTI447	Praktikum Basis Data	1	3	11
TIK342	Praktikum Basis Data	1	3	11
PTI333	Basis Data Terdistribusi	3	5	10
TIK123	Jaringan Komputer	2	5	33
TIK333	Sistem Operasi	3	5	10
PTI123	Grafika Komputer	3	5	12
PTI777	Sistem Informasi	2	3	99

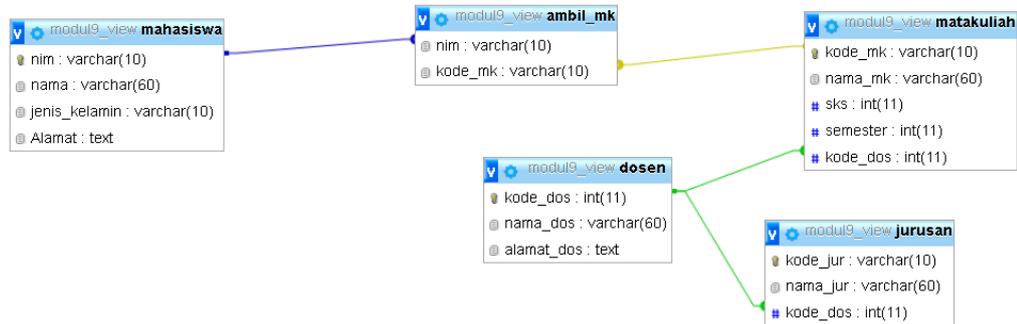
Tabel Ambil_mk

Nim	Kode_mk
101	PTI447
103	TIK333
104	PTI333
104	PTI777
111	PTI123
123	PTI999

Tabel Jurusan

Kode_jur	Nama_jur	Kode_dos
TE	Teknik Elektro	10
TM	Teknik Mesin	13
TS	Teknik Sipil	23

Himpunan entitas di atas dapat direpresentasikan ke dalam diagram skema (schema diagram) seperti Gambar 4.1 berikut ini:

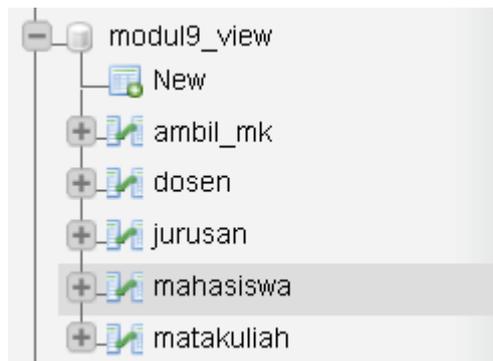


Gambar 4.1 Diagram Skema Database

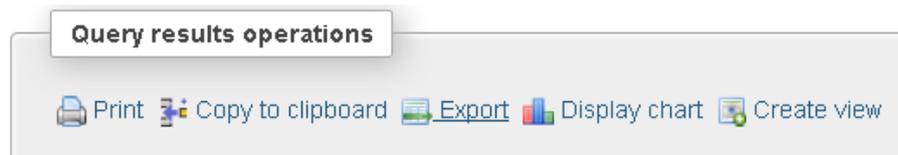
2. Pembuatan View

Secara umum, pembuatan *view* tidak berbeda dengan objek-objek *database* lainnya.

- Masuk ke *Localhost/PHPMyAdmin/*
- Masuk ke tabel mahasiswa di database yang sudah dibuat sebelumnya.



- Pilih *Create View*



d. Isikan seperti di bawah ini

e. Pilih “GO”

f. Buka *view* yang sudah dibuat sebelumnya

nim	nama	jenis_kelamin	Alamat
101	Arif	L	Jl.Kenangan
102	Budi	L	Jl.Jombang
103	Wati	P	Jl. Surabaya
104	Ika	P	Jl. Jombang
105	Tono	L	Jl. Jakarta
106	Iwan	L	Jl. Bandung
107	Sari	P	Jl. Malang

g. Menampilkan *Query View*

```
1 SHOW CREATE VIEW vgetmhs;
```

h. Memodifikasi *View*

Masuk pada tab “*Structure*” dari *view* vGetMhs

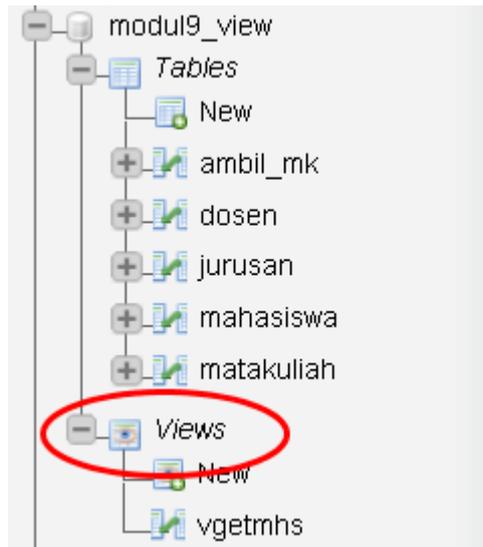


Pilih “*Edit view*”



i. Menghapus *View*

Masuk pada tab Views



Pilih “Drop” pada view yang akan dihapus



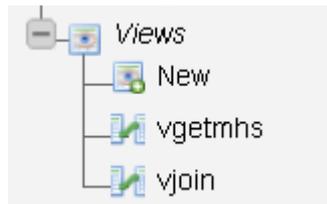
3. View Kompleks

View dapat mendefinisikan suatu pernyataan yang kompleks, misalnya melibatkan fungsi-fungsi agregat, join atau bahkan subquery. Sebagai ilustrasi view berikut melibatkan join untuk mendapatkan matakuliah yang tidak diambil oleh mahasiswa terdaftar.

- a. Buat *view* “vJOIN” dengan *query* sebagai berikut:

```
1 SELECT m.kode_mk, m.nama_mk, m.sks, m.semester,  
2     m.kode_dos FROM matakuliah m  
3 LEFT JOIN  
4     ambil_mk a ON m.kode_mk = a.kode_mk  
5 WHERE a.kode_mk IS NULL
```

- b. Eksekusi pembuatan *view*
c. Buka *view* vjoin



- d. Tampilan vjoin

kode_mk	nama_mk	sks	semester	kode_dos
PTI447	Praktikum Basis Data	1	3	11
TIK123	Jaringan Komputer	2	5	14

4. Nested View

Umumnya *view* diciptakan dengan mengacu pada tabel (seperti contoh-contoh sebelumnya). Namun juga tak menutup kemungkinan bagi kita untuk menciptakan *view* yang mengacu pada *view*. Pendekatan inilah yang dikenal sebagai *view* bersarang (*nested view*).

- a. Buatlah *view* vMK seperti pada gambar di bawah ini:

VIEW name	<input type="text" value="vMK"/>
Column names	<input type="text"/>
	1 SELECT * FROM matakuliah

- b. Eksekusi pembuatan *view*

- c. Buatlah *view* vMK5 seperti pada gambar di bawah ini:

VIEW name	vMK5
Column names	
	<pre> 1 SELECT * FROM vMK 2 WHERE semester = 5 </pre>

- d. Eksekusi pembuatan *view*
 e. Lakukan *Browse* pada vMK5

kode_mk	nama_mk	sks	semester	kode_dos
	Click the drop-down arrow to toggle column's visibility.	3	5	12
PTI333	Basis Data Terdistribusi	3	5	10
PTI447	Praktikum Basis Data	1	3	11
PTI777	Sistem Informasi	2	3	13
TIK123	Jaringan Komputer	2	5	14
TIK333	Sistem Operasi	3	5	10
TIK342	Praktikum Basis Data	1	3	11

- f. Lakukan *Browse* pada vMK5

kode_mk	nama_mk	sks	semester	kode_dos
PTI123	Grafika Komputer	3	5	12
PTI333	Basis Data Terdistribusi	3	5	10
TIK123	Jaringan Komputer	2	5	14
TIK333	Sistem Operasi	3	5	10

5. Updatable View

Sebagaimana disinggung di awal, *view* dapat bersifat *updatable* . Untuk mengetahui lebih jelasnya, perhatikan dan ikuti langkah-langkah berikut:

- a. Buatlah *view* sederhana sebagai berikut:

VIEW name	vUpdate
Column names	
	<pre> 1 SELECT * FROM mahasiswa </pre>

Praktikum Basis Data 2017 – TE UM

b. Periksa hasilnya

nim	nama	jenis_kelamin	Alamat
101	Arif	L	Jl.Kenangan
102	Budi	L	Jl.Jombang
103	Wati	P	Jl. Surabaya
104	Ika	P	Jl. Jombang
105	Tono	L	Jl. Jakarta
106	Iwan	L	Jl. Bandung
107	Sari	P	Jl. Malang

c. Lakukan perintah update pada *view* vUpdate

```
Run SQL query/queries on table modul9_view.vupdate: ⓘ  
  
1 UPDATE `vupdate` SET alamat = "Jl. Mangga"  
2 WHERE nim = "107"
```

d. Periksa hasilnya pada *view* vUpdate

nim	nama	jenis_kelamin	Alamat
101	Arif	L	Jl.Kenangan
102	Budi	L	Jl.Jombang
103	Wati	P	Jl. Surabaya
104	Ika	P	Jl. Jombang
105	Tono	L	Jl. Jakarta
106	Iwan	L	Jl. Bandung
107	Sari	P	Jl. Mangga

e. Periksa *record* dari tabel mahasiswa

nim	nama	jenis_kelamin	Alamat
101	Arif	L	Jl.Kenangan
102	Budi	L	Jl.Jombang
103	Wati	P	Jl. Surabaya
104	Ika	P	Jl. Jombang
105	Tono	L	Jl. Jakarta
106	Iwan	L	Jl. Bandung
107	Sari	P	Jl. Mangga

f. Terlihat bahwa modifikasi di view vUpdate akan memengaruhi data di tabel mahasiswa.

6. Check Option

Pada saat menciptakan *updatable view*, MySQL mengizinkan kita untuk menspesifikasikan bagaimana *parser* akan bekerja. Langkah ini dilakukan dengan mengaktifkan **CHECK OPTION**. Sederhananya, opsi ini mengakibatkan *parser me-review* klausa **WHERE** ketika memproses pernyataan *update* di *view*.

Ada dua jenis keyword yang bisa digunakan saat aktivasi *check option*: **LOCAL** dan **CASCADED**. Keyword **LOCAL** membatasi pemeriksaan hanya sebatas pada *view* yang didefinisikan, sedangkan mencakup semua *view* yang terkait misalkan dalam kasus *nested view*.

Untuk mengetahui penggunaan *check option*, perhatikan langkah-langkah berikut:

a. Definisikan *updatable view* sebagai berikut:

VIEW name	vmkoption
Column names	
	1 SELECT * FROM matakuliah WHERE sks < 2

b. Definisikan *nested view* dengan *Local Check Option* sebagai berikut:

VIEW name: vmklocal

Column names:

```
1 SELECT * FROM vmkoption
2 WHERE sks > 0
3
```

AS

WITH CHECK OPTION: LOCAL

c. Definisikan *nested view* dengan *Cascaded Check Option* sebagai berikut:

VIEW name: vmkcascade

Column names:

```
1 SELECT * FROM vmkoption
2 WHERE sks > 0
```

AS

WITH CHECK OPTION: CASCADED

d. Eksekusi perintah *Insert* pada vmklocal

Server: 127.0.0.1 >> Database: modul9_view >> View: vmklocal

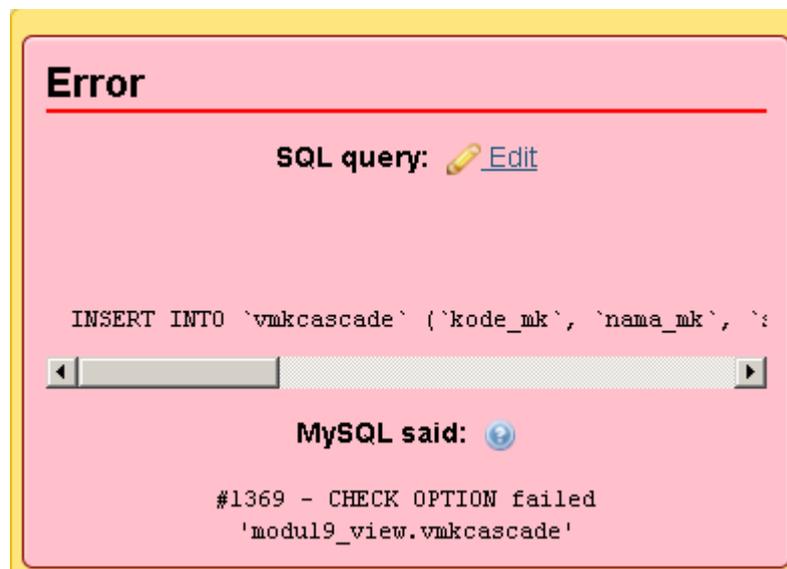
Browse Structure SQL Search Insert Export Privileges Operations

Column	Type	Function	Null	Value
kode_mk	varchar(10)			PTI999
nama_mk	varchar(60)			Statistka
sks	int(11)			2
semester	int(11)			4
kode_dos	int(11)			12

Go

e. Eksekusi perintah *Insert* pada vmkcascade

Column	Type	Function	Null	Value
kode_mk	varchar(10)			PTI998
nama_mk	varchar(60)			Workshop Jaringan Komputer
sks	int(11)			3
semester	int(11)			5
kode_dos	int(11)			14



f. Penambahan pada *view* vMkCascade gagal dilaksanakan karena terhambat oleh rule opsi **CASCADDED** dimana *view* induk (vMkOption) menyatakan bahwa sks harus kurang dari 2.

E. TUGAS PRAKTIKUM

1. Definisikan *view* untuk mendapatkan data mahasiswa yang hanya mengambil sks lebih dari 2 sks!
2. Definisikan *view* dosen yang mengajar mahasiswa beserta jumlah mahasiswa yang diajar!
3. Definisikan *nested view* dari tugas praktikum nomor 2 hanya dosen yang mengajar mahasiswa terbanyak!

F. TUGAS RUMAH

1. Definisikan *nested view* untuk mendapatkan data mahasiswa yang jenis kelaminnya laki-laki dengan *main view* berupa mahasiswa yang mengambil sembarang matakuliah yang sksnya lebih dari sks terkecil !
2. Definisikan *updatable view* dengan *local check option* untuk mendapatkan data matakuliah dengan sks diantara 1 dan 4.