MODUL V
SUB QUERY & VIEW

A. TUJUAN

- ✓ Memahami keterhubungan entitas di dalam basis data.
- ✓ Memahami operasi subquery dan jenis-jenisnya di dakam pengambilan data
- ✓ Mampu menyelesaikan kasus-kasus pengambilan data yang kompleks dengan pendekatan subquery
- ✓ 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. DASAR TEORI

1. Subquery

Subquery (disebut juga *subselect* atau *nested select* / *query* atau *inner- select*) adalah query yang ada di dalam perintah SQL lain misalnya **`SELECT`**, **`INSERT`**, **`UPDATE`**, atau **`DELETE`**. Keberadaan subquery secara nyata mampu menyederhanakan persoalan-persoalan rumit berkaitan query data. Sebagai contoh, misal terdapat pernyataan sebagai berikut:

> "Dapatkan data mahasiswa yang alamatnya sama dengan mahasiswa dengan NPM 52015002"

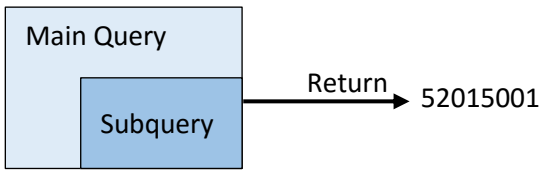
Secara normal, diperlukan dua tahapan untuk menyelesaikan kasus di atas. Pertama adalah mendapatkan alamat dari mahasiswa yang memiliki NPM 52015002. Langkah selanjutnya, baru bisa diketahui data mahasiswa yang alamatnya sama dengan mahasiswa dengan NPM 52015002. Adapun dengan memanfaatkan subquery, maka penyelesaian kasus di atas hanya memerlukan sebuah query (akan dijelaskan nanti). Pada hakekatnya, subquery sangat berguna ketika sebuah query didasarkan pada nilai-nilai yang tak diketahui. Sintaks formal subquery diperlihatkan sebagai berikut:

```
SELECT A1, A2, …, An
FROM r1, r2, r3, …, rm
WHERE P
      (SELECT A1, A2, …, An
       FROM r1, r2, r3, …, rm
       WHERE P)
```

Subquery dapat diklasifikasikan ke dalam tiga jenis : *scalar*, *multiple-row*, dan *multiple column*.

a. Scalar Subquery

Subquery baris tunggal (*scalar*) hanya mengembalikan hasil satu baris data. Bentuk subquery ini diperlihatkan seperti pada Gambar 1.

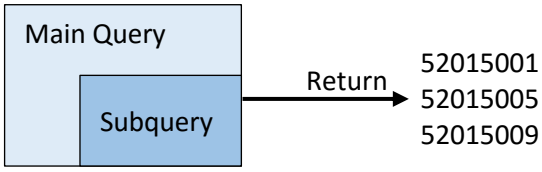


Gambar 1 *Scalar subquery*

Subquery baris tunggal dapat menggunakan operator baris tunggal **=**, **>**,**>=**, **<**, **<=**, atau **<>**.

b. Multiple-Row Subquery

Subquery baris ganda (*multiple-row*) mengembalikan lebih dari satu baris data. Bentuk subquery ini diperlihatkan seperti pada Gambar 2.

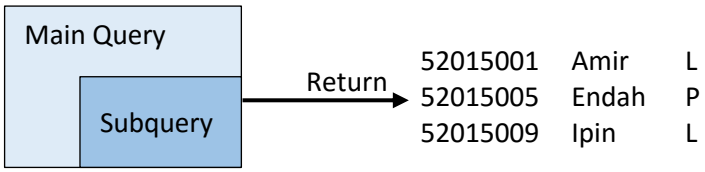


Gambar 2 *Multiple-row subquery*

Subquery baris ganda dapat menggunakan operator komparasi **`IN`**, **`ANY`** /**`SOME`**, atau **`ALL`**.

c. Multiple-Column Subquery

Subquery kolom ganda (*multiple-column*) mengembalikan lebih dari satu baris dan satu kolom data. Bentuk subquery ini diperlihatkan seperti pada Gambar 3.



Gambar 3 *Multiple-column subquery*

2. View

View dapat didefinisikan sebagai tabel maya *(virtual)* atau *logical* yang terdiri dari himpunan hasil *query*. Tidak seperti 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 pada dokumentasi MySQL, sintaks pendefinisian view diperlihatkan sebagai berikut:

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

3. 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 apapun, misalnya fungsi agregat.

Pada hakekatnya, 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 dan dapat diterapkan pada view.

C. LATIHAN

1. Himpunan Entitas

Sebelum memulai latihan, buatlah terlebih sebuah database bernama **akademik**, yang berisi tabel-tabel dengan struktur dan isi seperti berikut :

Tabel: mahasiswa.

npm	nama	jenis_kelamin	alamat
52015001	Arif	L	Jl. Mairo
52015002	Budi	L	Jl. Laiya
52015003	Christin	P	Jl. Butung
52015004	Dedi	P	Jl. Kasuari
52015005	Ekawati	L	Jl. Gagak
52015006	Firman	L	Jl. Garuda
52015007	Gina	P	Jl. Kenari

Tabel: ambil_mk

npm	kode_mk
52015001	13520015
52015003	13500016
52015004	13520013
52015004	13529006
52015011	13523006
52015023	13529003

Tabel: matakuliah

kode_mk	nama_mk	sks	semester	kode_dos
13520015	Praktikum Basis Data	1	3	911
13510008	Praktikum Basis Data	1	3	911
13520013	Sistem Basis Data	3	5	910
13500020	Jaringan Komputer	2	5	933
13500016	Sistem Operasi	3	5	910
13523006	Grafika Multimedia	3	5	912
13529006	Sistem Informasi	2	3	999

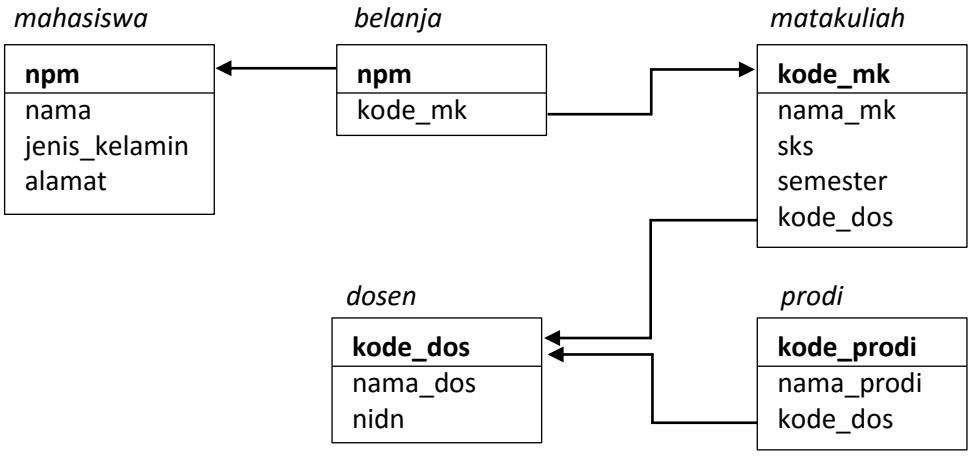
Tabel: dosen

kode_dos	nama_dos	alamat_dos
910	Sofyan	Jl. Cendrawasih
911	Marlina	Jl. Kakatua
912	Rahmawati	Jl. Pasar Maricaya
913	Hamdan	Jl. Tidung
914	Junaedy	Jl. Harimau

Tabel: prodi

kode_prodi	nama_prodi	kode_dos
SI	Sistem Informasi	910
IF	Informatika	913
MI	Manajemen Informatika	923

Himpunan entitas di atas dapat direpresentasikan ke dalam diagram skema (*schema diagram*) seperti Gambar 4



Gambar 4 Diagram Skema

2. **Scalar Subquery**

   Contoh subquery baris tunggal:

   Tampilkan data mahasiswa yang Jenis kelaminnya sama dengan mahasiswa dengan nama "Wati"!

```
MariaDB [akademik]> SELECT *
    -> FROM mahasiswa
    -> WHERE jenis_kelamin =
    ->                       (SELECT jenis_kelamin
    ->                        FROM mahasiswa
    ->                        WHERE nama = 'Christin');
+----------+----------+---------------+------------+
| npm      | nama     | jenis_kelamin | alamat     |
+----------+----------+---------------+------------+
| 52015003 | Christin | P             | Jl. Butung |
| 52015005 | Ekawati  | P             | Jl. Gagak  |
| 52015007 | Gina     | P             | Jl. Kenari |
+----------+----------+---------------+------------+
3 rows in set (0.00 sec)
```

   Bisa digambarkan, langkah pertama dari operasi di atas adalah mendapatkan jenis kelamin mahasiswa dengan nama **Wati** kemudian hasilnya yakni **P** akan digunakan oleh main query.

3. **Multiple-Row Subquery**

   Pada subquery ini, digunakan operator komparasi **`IN`**, **`ANY`** / **`SOME`**, atau **`ALL`**.

   a. **Operator `IN`**

   Operator memiliki arti: sama dengan member di dalam list. Sebagai contoh, operator ini dapat digunakan pada query untuk mendapatkan data dosen yang mengajar matakuliah

```
MariaDB [akademik]> SELECT d.kode_dos, d.nama_dos
    -> FROM dosen d
    -> WHERE d.kode_dos IN
    ->                     (SELECT kode_dos
    ->                     FROM matakuliah);
+----------+-----------+
| kode_dos | nama_dos  |
+----------+-----------+
|      910 | Sofyan    |
|      911 | Marlina   |
|      912 | Rahmawati |
+----------+-----------+
3 rows in set (0.01 sec)
```

   b. **Operator `ANY` / `SOME`**

   Operator memiliki arti: membandingkan suatu nilai dengan setiap nilai yang dikembalikan oleh subquery. Contohnya pada query untuk mendapatkan data matakuliah yang memiliki sks lebih besar dari sembarang sks matakuliah di semester 3.

```
MariaDB [akademik]> SELECT *
    -> FROM matakuliah
    -> WHERE sks > ANY
    ->                 (SELECT sks
    ->                  FROM matakuliah
    ->                  WHERE semester = 3);
+----------+--------------------+------+----------+----------+
| kode_mk  | nama_mk            | sks  | semester | kode_dos |
+----------+--------------------+------+----------+----------+
| 13520013 | Sistem Basis Data  |    3 |        5 |      910 |
| 13500020 | Jaringan Komputer  |    2 |        5 |      933 |
| 13500016 | Sistem Operasi     |    3 |        5 |      910 |
| 13523006 | Grafika Multimedia |    3 |        5 |      912 |
| 13529006 | Sistem Informasi   |    2 |        3 |      999 |
+----------+--------------------+------+----------+----------+
5 rows in set (0.00 sec)
```

Operator = **ANY** ekuivalen dengan **IN**.
Operator < **ANY** ekuivalen dengan **MAX** (kurang dari maks).
Operator > **ANY** ekuivalen dengan **MIN** (lebih dari min).

c. Operator `ALL`

Operator memiliki arti: membandingkan suatu nilai dengan semua nilai yang dikembalikan oleh subquery. Sebagai contoh operator ini dapat digunakan pada query untuk mendapatkan data matakuliah yang memiliki sks lebih besar dari semua sks matakuliah di semester 3.

```
MariaDB [akademik]> SELECT *
    -> FROM matakuliah
    -> WHERE sks > ALL
    ->                 (SELECT sks
    ->                  FROM matakuliah
    ->                  WHERE semester = 3);
+----------+--------------------+------+----------+----------+
| kode_mk  | nama_mk            | sks  | semester | kode_dos |
+----------+--------------------+------+----------+----------+
| 13520013 | Sistem Basis Data  |    3 |        5 |      910 |
| 13500016 | Sistem Operasi     |    3 |        5 |      910 |
| 13523006 | Grafika Multimedia |    3 |        5 |      912 |
+----------+--------------------+------+----------+----------+
3 rows in set (0.00 sec)
```

Operator < **ALL** ekuivalen dengan **MIN** (kurang dari min).
Operator > **ALL** ekuivalen dengan **MAX** (lebih dari maks).

4. Multiple-Column Subquery

Subquery kolom ganda (atau tabel) juga menggunakan operator komparasi **`IN`**, **`ANY`** / **`SOME`**, atau **`ALL`**. Pada query ini, nilai dari subquery dalam bentuk kolom ganda dikomparasi main query. Sebagai contoh, misalkan ingin ditampilkan data matakuliah yang semester dan sksnya sesuai dengan semester dan sks matakuliah dengan kode 13520015

```
MariaDB [akademik]> SELECT *
    -> FROM matakuliah
    -> WHERE (semester, sks) IN
    ->
    ->
    ->                          (SELECT semester, sks
                                 FROM matakuliah
                                 WHERE kode_mk = 13520015);
+----------+----------------------+------+----------+----------+
| kode_mk  | nama_mk              | sks  | semester | kode_dos |
+----------+----------------------+------+----------+----------+
| 13520015 | Praktikum Basis Data |    1 |        3 |      911 |
| 13510008 | Praktikum Basis Data |    1 |        3 |      911 |
+----------+----------------------+------+----------+----------+
2 rows in set (0.00 sec)
```

5. Operator `EXISTS` dan `NOT EXISTS`

Operator `EXISTS` dan Operator `NOT EXISTS` digunakan pada *correlated subquery* untuk memeriksa apakah subquery mengembalikan hasil atau tidak. Apabila subquery mengembalikan hasil, akan bernilai true dan sebaliknya bernilai false, jika tidak mengembalikan hasil.

Sebagai contoh, pernyataan berikut akan mendapatkan data matakuliah yang diambil oleh mahasiswa.

```
MariaDB [AKADEMIK]> SELECT * FROM matakuliah m
    -> WHERE EXISTS (SELECT * FROM ambil_mk a
    ->               WHERE m.kode_mk = a.kode_mk);
+----------+----------------------+------+----------+----------+
| kode_mk  | nama_mk              | sks  | semester | kode_dos |
+----------+----------------------+------+----------+----------+
| 13520015 | Praktikum Basis Data |    1 |        3 |      911 |
| 13520013 | Sistem Basis Data    |    3 |        5 |      910 |
| 13500016 | Sistem Operasi       |    3 |        5 |      910 |
| 13523006 | Grafika Multimedia   |    3 |        5 |      912 |
| 13529006 | Sistem Informasi     |    2 |        3 |      999 |
+----------+----------------------+------+----------+----------+
5 rows in set (0.00 sec)
```

Pernyataan berikut akan mendapatkan data matakuliah yang tidak diambil oleh mahasiswa.

```
MariaDB [AKADEMIK]> SELECT * FROM matakuliah m
    -> WHERE NOT EXISTS (SELECT * FROM ambil_mk a
    ->                   WHERE m.kode_mk = a.kode_mk);
+----------+----------------------+------+----------+----------+
| kode_mk  | nama_mk              | sks  | semester | kode_dos |
+----------+----------------------+------+----------+----------+
| 13510008 | Praktikum Basis Data |    1 |        3 |      911 |
| 13500020 | Jaringan Komputer    |    2 |        5 |      933 |
+----------+----------------------+------+----------+----------+
2 rows in set (0.03 sec)
```

6. Menggunakan View

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

a. Ketikkan pernyataan pembuatan *view* **vGetMhs** berikut di editor teks.

```
CREATE VIEW vGetMHS
AS
SELECT * FROM mahasiswa
```

b. Eksekusi file view di atas (sesuaikan path lokasi penyimpanan file).

c. Pemanggilan view tak ubahnya satu tabel.

```
MariaDB [AKADEMIK]> SELECT * FROM vGetMhs;
+----------+----------+---------------+-------------+
| npm      | nama     | jenis_kelamin | alamat      |
+----------+----------+---------------+-------------+
| 52015001 | Arif     | L             | Jl. Mairo   |
| 52015002 | Budi     | L             | Jl. Laiya   |
| 52015003 | Christin | P             | Jl. Butung  |
| 52015004 | Dedi     | L             | Jl. Kasuari |
| 52015005 | Ekawati  | P             | Jl. Gagak   |
| 52015006 | Firman   | L             | Jl. Garuda  |
| 52015007 | Gina     | P             | Jl. Kenari  |
+----------+----------+---------------+-------------+
7 rows in set (0.00 sec)
```

d. Apabila diperlukan, dimungkinkan melakukan penyaringan pada view.

```
MariaDB [AKADEMIK]> SELECT * FROM vGetMhs
    ->              WHERE jenis_kelamin = 'P';
+----------+----------+---------------+------------+
| npm      | nama     | jenis_kelamin | alamat     |
+----------+----------+---------------+------------+
| 52015003 | Christin | P             | Jl. Butung |
| 52015005 | Ekawati  | P             | Jl. Gagak  |
| 52015007 | Gina     | P             | Jl. Kenari |
+----------+----------+---------------+------------+
3 rows in set (0.00 sec)
```

Untuk mendapatkan informasi mengenai pendefinisian view, gunakan perintah **`SHOW CREATE VIEW nama_view`**

```
MariaDB [AKADEMIK]> SHOW CREATE VIEW vGetMhs;
+---------+-----------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------+--
--------------------+----------------------+
| View    | Create View

                                                                        | c
haracter_set_client | collation_connection |
+---------+-----------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------+--
--------------------+----------------------+
| vgetmhs | CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` S
QL SECURITY DEFINER VIEW `vgetmhs` AS select `mahasiswa`.`npm` AS `
npm`,`mahasiswa`.`nama` AS `nama`,`mahasiswa`.`jenis_kelamin` AS `j
enis_kelamin`,`mahasiswa`.`alamat` AS `alamat` from `mahasiswa` | u
tf8mb4              | utf8mb4_unicode_ci   |
+---------+-----------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------+--
--------------------+----------------------+
1 row in set (0.00 sec)
```

e. Sebagaimana objek-objek database lainnya view dapat dihapus menggunakan perintah **`DROP`**

```
MariaDB [AKADEMIK]> DROP VIEW vGetMhs;
Query OK, 0 rows affected (0.00 sec)
```

f. Apabila diperlukan view yang sudah terdefinisi juga dapat dimodifikasi dengan menggunakan perintah **`ALTER VIEW`**

7. View Kompleks

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

```
CREATE VIEW vJoin
AS
SELECT m.kode_mk, m.nama_mk, m.sks, m.semester
FROM matakuliah m
  LEFT JOIN
    (mahasiswa mhs LEFT JOIN ambil_mk a
     ON mhs.npm = a.npm)
  ON m.kode_mk = a.kode_mk
WHERE a.kode_mk IS NULL
```

Contoh pemanggilan *view* vJOIN

```
MariaDB [AKADEMIK]> SELECT * FROM vJoin;
+----------+----------------------+------+----------+
| kode_mk  | nama_mk              | sks  | semester |
+----------+----------------------+------+----------+
| 13510008 | Praktikum Basis Data |    1 |        3 |
| 13500020 | Jaringan Komputer    |    2 |        5 |
| 13523006 | Grafika Multimedia   |    3 |        5 |
+----------+----------------------+------+----------+
3 rows in set (0.00 sec)
```

Pada pendekatan subquery, *view* di atas dapat dituliskan sebagai berikut:

```
CREATE VIEW vSubQuery
AS
SELECT *
FROM matakuliah
WHERE kode_mk NOT IN
                    (SELECT kode_mk
                     FROM ambil_mk a, mahasiswa m
                     WHERE a.npm = m.npm)
```

8. Nested View

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

a. Ketikkan pernyataan pembuatan *view* berikut editor teks.

```
CREATE VIEW vMK
AS
SELECT *
FROM matakuliah
```

b. Eksekusi file *view*.

c. Buat view baru merujuk pada *view* vMK. Misalkan dengan tambahan predikat semster sama dengan 5.

```
CREATE VIEW vMK5
AS
SELECT *
FROM matakuliah
WHERE semester = 5;
```

d. Eksekusi file *view* vMK5. Hasil pemanggilan masing-masing *view* diperlihatkan sebagai berikut.

```
MariaDB [AKADEMIK]> SELECT * FROM vMK;
+----------+----------------------+------+----------+----------+
| kode_mk  | nama_mk              | sks  | semester | kode_dos |
+----------+----------------------+------+----------+----------+
| 13520015 | Praktikum Basis Data |    1 |        3 |      911 |
| 13510008 | Praktikum Basis Data |    1 |        3 |      911 |
| 13520013 | Sistem Basis Data    |    3 |        5 |      910 |
| 13500020 | Jaringan Komputer    |    2 |        5 |      933 |
| 13500016 | Sistem Operasi       |    3 |        5 |      910 |
| 13523006 | Grafika Multimedia   |    3 |        5 |      912 |
| 13529006 | Sistem Informasi     |    2 |        3 |      999 |
+----------+----------------------+------+----------+----------+
7 rows in set (0.00 sec)

MariaDB [AKADEMIK]> SELECT * FROM vMK5;
+----------+--------------------+------+----------+----------+
| kode_mk  | nama_mk            | sks  | semester | kode_dos |
+----------+--------------------+------+----------+----------+
| 13520013 | Sistem Basis Data  |    3 |        5 |      910 |
| 13500020 | Jaringan Komputer  |    2 |        5 |      933 |
| 13500016 | Sistem Operasi     |    3 |        5 |      910 |
| 13523006 | Grafika Multimedia |    3 |        5 |      912 |
+----------+--------------------+------+----------+----------+
4 rows in set (0.00 sec)
```

9. Updatable view

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

a. Ketikkan pernyataan *view* sederhana sebagai beikut.

```
CREATE VIEW vUpdate
AS
SELECT *
FROM mahasiswa
```

b. Periksa terlebih dahulu hasil pengambilan data.

```
MariaDB [AKADEMIK]> SELECT * FROM vUpdate;
+----------+----------+---------------+-------------+
| npm      | nama     | jenis_kelamin | alamat      |
+----------+----------+---------------+-------------+
| 52015001 | Arif     | L             | Jl. Mairo   |
| 52015002 | Budi     | L             | Jl. Laiya   |
| 52015003 | Christin | P             | Jl. Butung  |
| 52015004 | Dedi     | L             | Jl. Kasuari |
| 52015005 | Ekawati  | P             | Jl. Gagak   |
| 52015006 | Firman   | L             | Jl. Garuda  |
| 52015007 | Gina     | P             | Jl. Kenari  |
+----------+----------+---------------+-------------+
7 rows in set (0.00 sec)
```

c. Lakukan modifikasi pada *view* vUpdate (perhatikan, bukan di tabel).

```
MariaDB [AKADEMIK]> UPDATE vUpdate
    ->              SET alamat = 'Jl. Kakatua'
    ->              WHERE npm = 52015007;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

d. Periksa hasil modifikasi di *view*.

```
MariaDB [AKADEMIK]> SELECT * FROM vUpdate;
+----------+----------+---------------+-------------+
| npm      | nama     | jenis_kelamin | alamat      |
+----------+----------+---------------+-------------+
| 52015001 | Arif     | L             | Jl. Mairo   |
| 52015002 | Budi     | L             | Jl. Laiya   |
| 52015003 | Christin | P             | Jl. Butung  |
| 52015004 | Dedi     | L             | Jl. Kasuari |
| 52015005 | Ekawati  | P             | Jl. Gagak   |
| 52015006 | Firman   | L             | Jl. Garuda  |
| 52015007 | Gina     | P             | Jl. Kakatua |
+----------+----------+---------------+-------------+
7 rows in set (0.00 sec)
```

e. Langkah selanjutnya periksa data di tabel mahasiswa.

```
MariaDB [AKADEMIK]> SELECT * FROM mahasiswa;
+----------+----------+---------------+-------------+
| npm      | nama     | jenis_kelamin | alamat      |
+----------+----------+---------------+-------------+
| 52015001 | Arif     | L             | Jl. Mairo   |
| 52015002 | Budi     | L             | Jl. Laiya   |
| 52015003 | Christin | P             | Jl. Butung  |
| 52015004 | Dedi     | L             | Jl. Kasuari |
| 52015005 | Ekawati  | P             | Jl. Gagak   |
| 52015006 | Firman   | L             | Jl. Garuda  |
| 52015007 | Gina     | P             | Jl. Kakatua |
+----------+----------+---------------+-------------+
7 rows in set (0.00 sec)
```

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

10. 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 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 **`CASCADED`** 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:

```
CREATE VIEW vMkOption
AS
  SELECT *
  FROM matakuliah
  WHERE sks < 2
WITH CHECK OPTION
```

b. Definisikan *nested view* vMkLocal dengan opsi **LOCAL**

```
CREATE VIEW vMkLocal
AS
  SELECT *
  FROM vMkOption
  WHERE sks > 0
WITH LOCAL CHECK OPTION
```

c. Definisikan *nested view* vMkCascade dengan opsi **CASCADED**

```
CREATE VIEW vMkCascade
AS
  SELECT *
  FROM vMkOption
  WHERE sks > 0
WITH CASCADED CHECK OPTION
```

d. Berikan perintah untuk menambah data baru di *view* vMkLocal.

```
MariaDB [AKADEMIK]> INSERT INTO vMkLocal
    ->              VALUES (13521002, 'Bahasa Rakitan', 2, 7, 910);
Query OK, 1 row affected (0.02 sec)
```

e. Berikan perintah untuk menambah data baru di *view* vMkLocal.

```
MariaDB [AKADEMIK]> INSERT INTO vMkCascade
    ->              VALUES (1352103, 'Sistem Digital', 3, 5, 910);
ERROR 1369 (HY000): CHECK OPTION failed 'akademik.vmkcascade'
```

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

D. TUGAS PRATIKUM

1. Dapatkan data mahasiswa yang alamatnya sama dengan NPM 52015002, tidak termasuk mahasiswa tersebut.
2. Dapatkan matakuliah yang tidak diajar oleh dosen terdaftar.
3. Dapatkan data dosen yang yang mengajar matakuliah dengan sks lebih kecil dari sembarang sks.
4. Definisikan *updatable view* untuk mendapatkan npm dan nama mahasiswa yang mengambil matakuliah di semester 3
5. Definisikan *view* untuk mendapatkan nama dosen yang mengajar matakuliah dengan jumlah siswa terbanyak.

E. TUGAS RUMAH

1. Dapatkan data dosen yang mengajar matakuliah yang sksnya kurang dari sks matakuliah yang diajar dosen yang sekaligus menjadi ketua prodi Informatika, tidak termasuk ketua prodi Informatika.
2. Dapatkan npm, nama dan alamat mahasiswa yang tempat tinggalnya sama dengan dosen yang mengajar matakuliah dengan sks dibawah rata-rata.
3. Definisikan *updatable view* dengan *check option* untuk mendapatkan data matakuliah yang sksnya di antara 1 dan 4, dan semesternya lebih dari semester terkecil.
4. Definisikan *nested view* untuk mendapatkan data mahasiswa yang jenis kelaminnya

L, dengan *main view* berupa mahasiswa yang mengambil sembarang matakuliah yang sksnya lebih dari sks terkecil.