PRAKTIKUM SUB QUERY

2. Dasar Teori

2.1 Select Into Statement

SELECT INTO adalah sebuah perintah yang ada pada MySQL yang digunakan untuk memindahkan data dari satu tabel ke tabel yang lain atau dari satu database ke database yang lain. SELECT INTO sebenarnya juga dapat memilah field yang mana sajakah yang ingin kita pindahkan dengan cara menyebutkan nama fieldnya. Struktur SQL yang digunakan adalah **select * into tabelBaru from tabelLama;** dan bila hanya digunakan untuk memindah kolom-kolom tertentu saja maka Syntax yang digunakan adalah **select namaKolom1, namaKolom2, namaKolom3 into tabelBaru from tabelLama;** Berikut adalah beberapa contoh penggunaan SELECT INTO :

2.1.1 Membuat backup sebuah tabel

SELECT * INTO backup_transaksi FROM transaksi;

2.1.2 Membuat backup sebuah tabel ke dalam database yang baru

SELECT * INTO backup_trsndskdi IN ‘backup_database.mdb’ FROM transaksi;

2.1.3 Membuat backup beberapa kolom sebuah tabel

SELECT ID,NamaPelanggan INTO backup_transaksi FROM transaksi;

2.2 Sub Query

Subquery (disebut juga *subselect* atau *nested select / query* atau *inner-select*) adalah query `SELECT` yang ada di dalam perinta SQL lain misalnya `SELECT, INSERT, UPDATE,` atau `DELETE.` Keberadaan subquery secara nyata mampu menyederhanakan persoalan persoalan rumit berkaitan query data. Sintaks formal subquery diperlukan sebaga berikut :

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

2.2.1 Manfaat Subquery

2.2.1.1 Subquery digunakan untuk menyelesaikan persoalan dimana terdapat suatu nilai yang tidak diketahui (unknown values).

2.2.1.2 Meng-copy data dari satu tabel ke tabel lain

2.2.1.3 Menerima data dari inline view

2.2.1.4 Mengambil data dari tabel lain untuk kemudian di update ke tabel yang dituju

2.2.1.5 Menghapus baris dari satu tabel berdasarkan baris dari tabel lain

Perintah ini hanya bisa menerima satu buah hasil dari subquery, jika hasil dari subquery ada lebih adari satu maka akan terjadi error.

2.2.2 Ada beberapa bahasan terkait penggunaah Subquery ini antara lain :

2.2.2.1 **Sub query dengan IN**

Jika operator '=' hanya digunakan untuk hasil yang tepat satu, maka jika ingin menampilkan yang memiliki hasil lebih dari satu maka menggunakan perintah IN. Dan struktur Query yang digunakan dalam hal ini adalah :

SELECT namakolom FROM namatabel WHERE kondisi opeatorperbandingan IN (subquery);

2.2.2.2 **Subquery dengan ALL**

Command ALL diikuti dengan operator perbandingan digunakan memiliki arti menampilkan nilai jika perbandingan bernilai benar untuk semua data. Operator perbandingan tersebut berupa (<, >, =, !=). Berikut adalah query dasar dari subquery all

SELECT namakolom FROM namatabel WHERE kondisi opeatorperbandingan ALL (subquery);

2.2.2.3 **Subquery dengan ANY**

Command ANY diikuti dengan operator perbandingan memiliki arti menampilkan nilai yang sesuai dengan apapun yang dihasilkan oleh sub query. ANY berbeda dengan IN, jika IN itu semua data, sedangkan ANY hanya beberapa data. Contoh query dasar :

SELECT namakolom FROM namatabel WHERE kondisi opeatorperbandingan ANY (subquery);

2.2.2.4 **Subquery dengan EXISTS**

Perintah EXISTS disini berguna untuk mengartur penampilan hasil query, Query Utama akan dijalankan jika Subquery bernilai TRUE (ada hasilnya) jika hasilnya kosong maka Query utama tidak akan dijalankan. Lawan dari statement EXISTS adalah NOT EXISTS. Query yang digunakan adalah :

SELECT namakolom FROM namatabel WHERE EXIST / NOT EXIST (subquery);

2.2.2.5 **Insert Into Statement**

Untuk meng-copy data dari suatu tabel ke tabel lain kita bisa menggunakan perintah INSERT INTO. Tetapi sebelu kita menggunakan perintah INTO, INTO ini kita harus membuat tabel baru yang jumlah field dan urutannya sama dengan tabel field sebelumnya, nama dari field tidak harus sama. Query untuk perintah INSERT INTO adalah sebagai berikut :

INSERT INTO namatabel2 SELECT namakolom FROM namatabel1 ;

Kita juga bisa memberikan kondisi sesuai dengan kondisi yang kita inginkan dengan penambahan perintah WHERE.

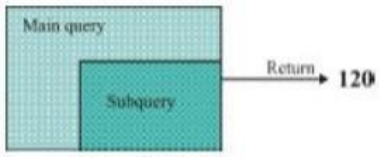
2.2.3 Klarifikasi Subquery

Sebagai contoh, misalnya terdapat pernyataan sebagai berikut : “dapatkan data mahasiswa yang alamatnya sama dengan mahasiswa dengan nim 104”

Secara normal, diperluakan dua tahapan untuk menyelesaikan kasus di atas. Pertama adalah mendapatakan alamat dari mahasiswa yang memiliki nim 104. Langkah selanjutnya, baru kita bisa mengetahui data mahasiswa yang alamatnya sama dengan mahasiswa nim 104.

2.2.3.1 **Scalar Subquery**

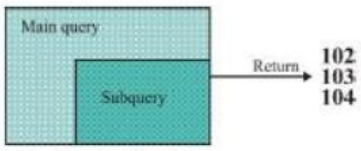
Subquery baris tunggal (*scalar*) hanya mengembalikan hasil satu baris data. Bentuk subquery ini diperhatikan sepertu gambar 1. Subquery baris tunggal dapat menggunakan operator baris tunggal =, >, >=, <. <=, atau <>.



Gambar 1 Scalar subquery

2.2.3.2 **Multiple-Row Subquery**

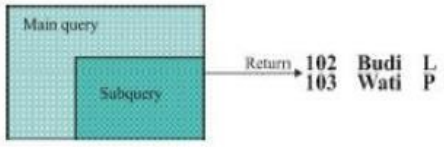
Subquery baris anda (*multiple-row*) mengembalikan lebih dari satu baris data. Bentuk subquery ini diperhatikan seperti gambar 2. Subquery kolom ganda dapat menggunakan operator komparasi **IN, ANY/SOME,** atau **ALL.**



Gambar 2 Multiple-row subquery

2.2.3.3 **Multiple-Column Subquery**

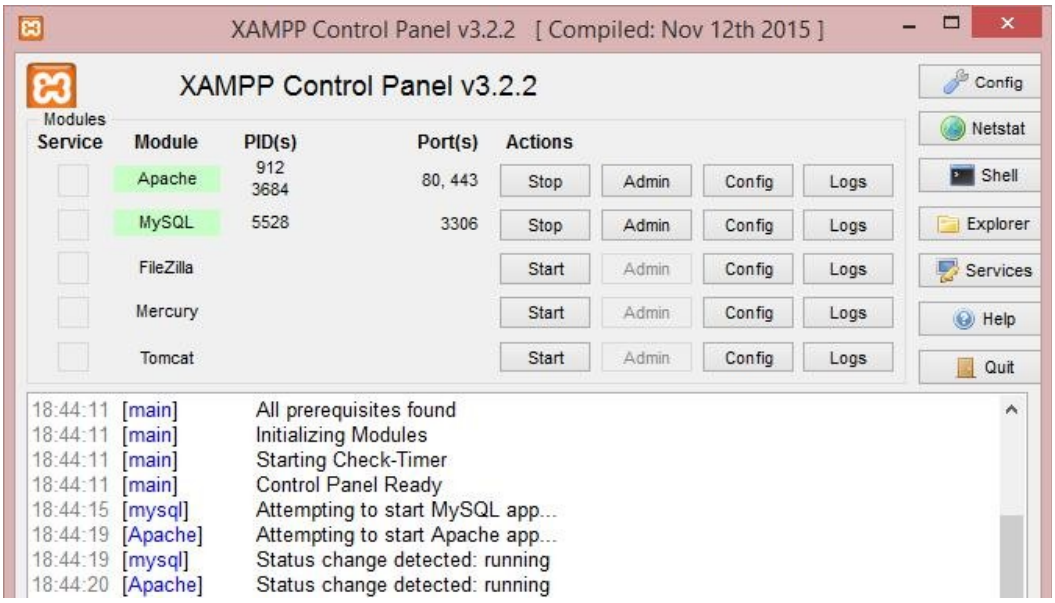
Subquery kolom ganda (*multipe-column*) mengembalikan lebih dari satu bari dan satu kolam data. Bentuk subquery ini diperhatikan seperti gambar 3.



Gambar 3 Multiple-column subquery

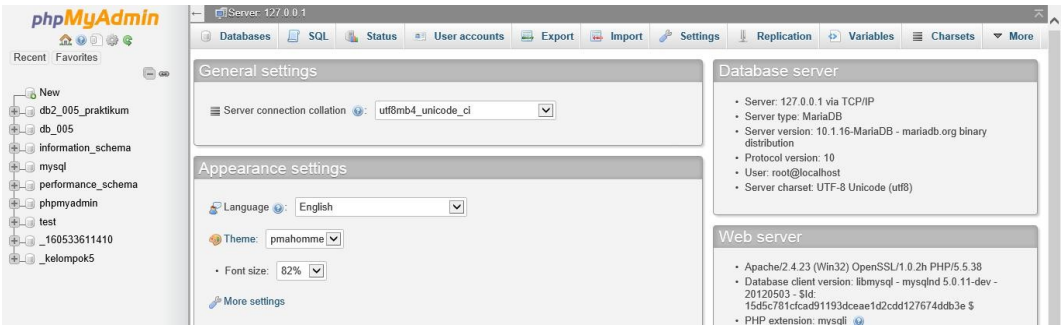
3. Latihan

Sebelum melakukan tugas latihan, praktikan membuka XAMPP CONTROL PANEL dan memilih *Start* pada *Module* MySQL dan *Apache* kemudian klik Admin atau mengakses `localhost/phpmyadmin` pada browser.



Gambar 1. Start MySQL dan Apache

Maka akan tampil seperti pada gambar dibawah ini :



Gambar 2. Tampilan phpmyadmin

3.1 Himpunan Entitas

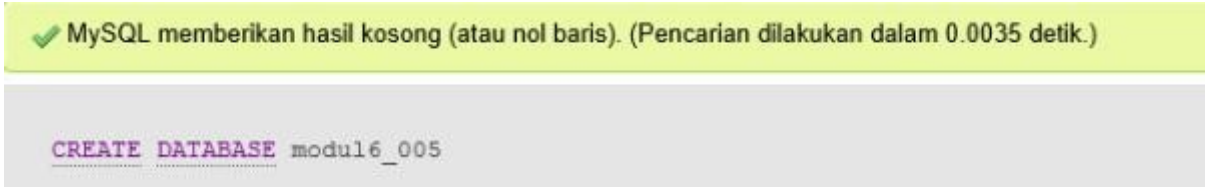
3.1.1 Membuat Tabel

Untuk memulai latihan kali ini praktikkan menciptakan objek query, pada halaman awal phpMyAdmin praktikan memilih menu SQL, setelah itu muncul halaman editor SQL, dalam editor tersebut tempat untuk praktikkan mengetikkan pernyataan SQL, untuk membuat *database* maka diketikkan pernyataan CREATE DATABASE ‘nama_database’; nama_database bisa disesuaikan dengan keinginan pengguna, dalam praktikum ini praktikkan mengetikkan `CREATE DATABASE modul6_005;` sehingga nama database yang dibuat adalah modul6_005.

```
1 CREATE DATABASE modul6_005;
```

Gambar 3. CREATE DATABASE modul6_005; pada SQL

Untuk mengeksekusi pernyataan SQL tersebut dengan klik tombol Go. Setelah itu *database* modul6_005 akan berhasil dibuat.



Gambar 4. Database modul6_005 berhasil dibuat

Setelah menciptakan sebuah database, praktikkan menciptakan lima buah tabel secara bergantian dengan klik SQL pada awal tampilan phpMyAdmin, kemudian mengetikkan pernyataan-pernyataan yang dibutuhkan pada praktikum kali ini. Struktur untuk pernyataan pembuatan tabel adalah sebagai berikut :

CREATE TABLE ‘nama_tabel’ (

‘nama_field1’ TIPEDATA(*length*),

‘nama_field2’ TIPEDATA(*length*),

PRIMARY KEY(nama_field));

Dalam praktikum kali ini praktikkan membuat tabel yang pertama bernama MAHASISWA, dengan memiliki 4 *field* yaitu NIM yang bertipe data CHAR dengan *length* 3. *Field* kedua yaitu NAMA_MHS yang bertipe data VARCHAR yang memiliki *length* 30. *Field* ketiga yaitu JK_MHS yang bertipe data CHAR yang memiliki *length* 1. *Field* keempat yaitu ALAMAT_MHS yang bertipe data VARCHAR yang memiliki *length* 50. Praktikkan memilih tipe data VARCHAR dan CHAR karena tipe data tersebut lebih *general* sebenarnya penggunaan tipe data TEXT juga diterima, namun kurang standar. NIM di set sebagai *primary key* karena bersifat unik. Kemudian eksekusi pernyataan penciptaan tabel dengan klik Go.

```
1 CREATE TABLE MAHASISWA (
2     NIM CHAR (3),
3     NAMA_MHS VARCHAR(30),
4     JK_MHS CHAR (1),
5     ALAMAT_MHS VARCHAR (50),
6     PRIMARY KEY (NIM)
7     );
```

Gambar 5. CREATE TABLE MAHASISWA; pada SQL

Gambar 6. Tabel MAHASISWA telah berhasil dibuat

Untuk mengecek struktur dari tabel MAHASISWA dengan klik MAHASISWA pada modul6_005 kemudan pilih *Structure*.

#	Nama	Jenis	Penyortiran	Atribut	Kosong	Bawaan	Ekstra	Tindakan
1	NIM	char(3)			Tidak	*Tidak ada*		Ubah Hapus Ku
2	NAMA_MHS	varchar(30)			Ya	*NULL*		Ubah Hapus Ku
3	JK_MHS	char(1)			Ya	*NULL*		Ubah Hapus Ku
4	ALAMAT_MHS	varchar(50)			Ya	*NULL*		Ubah Hapus Ku

Gambar 7. Struktur tabel MAHASISWA pada database modul6_005

Tabel yang kedua bernama MATAKULIAH, dengan memiliki 5 *field* yaitu KD_MK yang bertipe data VARCHAR dengan *length* 6. *Field* kedua yaitu NAMA_MK yang bertipe data VARCHAR yang memiliki *length* 30. *Field* ketiga yaitu SKS yang bertipe data INT yang memiliki *length* 1. *Field* keempat yaitu SEMESTER yang bertipe data INT yang memiliki *length* 1. *Field* kelima yaitu KODE_DOS yang bertipe data CHAR yang memiliki *length* 2. KD_MK di set sebagai *primary key* karena bersifat unik. Kemudian eksekusi pernyataan penciptaan tabel dengan klik Go.

```
CREATE TABLE MATAKULIAH (
    KD_MK VARCHAR (6),
    NAMA_MK VARCHAR(30),
    SKS INT (1),
    SEMESTER INT (1),
    KODE_DOS CHAR (2),
    PRIMARY KEY (KD_MK)
    );
```

Gambar 8. CREATE TABLE MATAKULIAH; pada SQL

```
CREATE TABLE MATAKULIAH ( KD_MK VARCHAR (6), NAMA_MK VARCHAR(30), SKS INT (1), SEMESTER INT (1), KODE_DOS CHAR (2), PRIMARY KEY (KD_MK) )
```

Gambar 9. Tabel MATAKULIAH telah berhasil dibuat

Untuk mengecek struktur dari tabel MATAKULIAH dengan klik MATAKULIAH pada modul6_005 kemudan pilih *Structure*.

#	Nama	Jenis	Penyortiran	Atribut	Kosong	Bawaan	Ekstra	Tindakan
1	KD_MK	varchar(6)			Tidak	*Tidak ada*		Ubah Hapus Kun
2	NAMA_MK	varchar(30)			Ya	*NULL*		Ubah Hapus Kun
3	SKS	int(1)			Ya	*NULL*		Ubah Hapus Kun
4	SEMESTER	int(1)			Ya	*NULL*		Ubah Hapus Kun
5	KODE_DOS	char(2)			Ya	*NULL*		Ubah Hapus Kun

Gambar 10. Struktur tabel MATAKULIAH pada database modul6_005

Tabel yang ketiga bernama JURUSAN, dengan memiliki 3 *field* yaitu KODE_JUR yang bertipe data CHAR dengan *length* 2. *Field* kedua yaitu NAMA_JUR yang bertipe data VARCHAR yang memiliki *length* 30. *Field* ketiga yaitu KODE_DOS yang bertipe data

CHAR yang memiliki *length* 2. KODE_JUR di set sebagai *primary key* karena bersifat unik. Kemudian eksekusi pernyataan penciptaan tabel dengan klik Go.

```
CREATE TABLE JURUSAN (
    KODE_JUR CHAR (2),
    NAMA_JUR VARCHAR(30),
    KODE_DOS CHAR (2),
    PRIMARY KEY (KODE_JUR)
    );
```

Gambar 11. CREATE TABLE JURUSAN; pada SQL

```
CREATE TABLE JURUSAN ( KODE_JUR CHAR (2), NAMA_JUR VARCHAR(30), KODE_DOS CHAR (2), PRIMARY KEY (KODE_JUR) )
```

Gambar 12. Tabel JURUSAN telah berhasil dibuat

Untuk mengecek struktur dari tabel JURUSAN dengan klik JURUSAN pada modul6_005 kemudan pilih *Structure*.

	#	Nama	Jenis	Penyortiran	Atribut	Kosong	Bawaan	Ekstra	Tindakan
☐	1	KODE_JUR	char(2)			Tidak	*Tidak ada*		Ubah Hapus Kur
☐	2	NAMA_JUR	varchar(30)			Ya	*NULL*		Ubah Hapus Kur
☐	3	KODE_DOS	char(2)			Ya	*NULL*		Ubah Hapus Kur

Gambar 13. Struktur tabel JURUSAN pada database modul6_005

Tabel yang keempat bernama DOSEN, dengan memiliki 3 *field* yaitu KODE_DOS yang bertipe data CHAR dengan *length* 2. *Field* kedua yaitu NAMA_DOS yang bertipe data VARCHAR yang memiliki *length* 30. *Field* ketiga yaitu ALAMAT_DOS yang bertipe data VARCHAR yang memiliki *length* 50. KODE_DOS di set sebagai *primary key* karena bersifat unik. Kemudian eksekusi pernyataan penciptaan tabel dengan klik Go.

```
CREATE TABLE DOSEN (
    KODE_DOS CHAR(2),
    NAMA_DOS VARCHAR(30),
    ALAMAT_DOS VARCHAR(50),
    PRIMARY KEY (KODE_DOS)
    );
```

Gambar 14. CREATE TABLE DOSEN; pada SQL

```
CREATE TABLE DOSEN ( KODE_DOS CHAR(2), NAMA_DOS VARCHAR(30), ALAMAT_DOS VARCHAR(50), PRIMARY KEY (KODE_DOS) )
```

Gambar 15. Tabel DOSEN telah berhasil dibuat

Untuk mengecek struktur dari tabel DOSEN dengan klik DOSEN pada modul6_005 kemudan pilih *Structure*.

	#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
☐	1	KODE_DOS	char(2)			No	*None*		Change Drop Prim
☐	2	NAMA_DOS	varchar(30)			Yes	*NULL*		Change Drop Prim
☐	3	ALAMAT_DOS	varchar(50)			Yes	*NULL*		Change Drop Prim

Gambar 16. Struktur tabel DOSEN pada database modul6_005

Tabel yang kelima bernama AMBIL_MK, dengan memiliki 2 *field* yaitu NIM yang bertipe data CHAR dengan *length* 3. *Field* kedua yaitu KD_MK yang bertipe data VARCHAR yang memiliki *length* 6. *Primary key* tidak di set. Kemudian eksekusi pernyataan penciptaan tabel dengan klik Go.

```
CREATE TABLE AMBIL_MK (
    NIM CHAR(3),
    KD_MK VARCHAR(6)
    );
```

Gambar 14. CREATE TABLE AMBIL_MK; pada SQL

```
CREATE TABLE AMBIL_MK ( NIM CHAR(3), KD_MK VARCHAR(6) )
```

Gambar 15. Tabel AMBIL_MK telah berhasil dibuat

Untuk mengecek struktur dari tabel AMBIL_MK dengan klik AMBIL_MK pada modul6_005 kemudan pilih *Structure*.

	#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
☐	1	NIM	char(3)			Yes	*NULL*		Change Drop
☐	2	KD_MK	varchar(6)			Yes	*NULL*		Change Drop

Gambar 16. Struktur tabel AMBIL_MK pada database modul6_005

Setelah kelima tabel berhasil dibuat kemudian praktikkan menambahkan data pada kelima tabel tersebut melalui pernyataan SQL. Struktur untuk pernyataan menambahkan data pada tabel adalah sebagai berikut :

INSERT INTO ‘nama_tabel’ VALUES

(‘data_field1, ‘data_field2’, ‘data_fiedlke-n’),

(‘data_field1, ‘data_field2’, ‘data_fiedlke-n’);

Praktikkan menambahkan data pada 3 tabel MAHASISWA, MATAKULIAH, JURUSAN sesuai dengan yang diperintahkan pada modul, dengan data seperti pada gambar dibawah ini.

```
INSERT INTO mahasiswa VALUES
("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");
```

```
INSERT INTO matakuliah VALUES
("PTI123","GRAFIKA MULTIMEDIA","3","5","12"),
("PTI333","BASIS DATA TERDISTRIBUSI","3","5","10"),
("PTI447","PRAKTIKUM BASIS DATA","1","3","11"),
("PTI777","SISTEM INFORMASI","2","3","99"),
("TIK123","JARINGAN KOMPUTER","2","5","33"),
("TIK333","SISTEM OPERASI","3","5","10"),
("TIK342","PRAKTIKUM BASIS DATA","1","3","11");
```

```
INSERT INTO jurusan VALUES
("TE","TEKNIK ELEKTRO","10"),
("TM","TEKNIK MESIN","13"),
("TS","TEKNIK SIPIL","23");
```

Gambar 17. INSERT INTO mahasiswa, matakuliah, jurusan;;

Setelah praktikkan mengeksekusi dengan perintah Go. Data-data tersebut akan berhasil ditambahkan pada masing-masing tabel. Untuk melihat data yang telah ditambahkan pada tabel MAHASISWA dengan pernyataan sebagai berikut :

```
SELECT * FROM `mahasiswa`
```

Gambar 18. Melihat data pada tabel MAHASISWA;

Kemudian akan menampilkan data dari tabel MAHASISWA

NIM	NAMA_MHS	JK_MHS	ALAMAT_MHS
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

Gambar 19. Data pada tabel MAHASISWA;

Untuk melihat data yang telah ditambahkan pada tabel MATAKULIAH dengan pernyataan sebagai berikut :

```
SELECT * FROM `matakuliah`
```

Gambar 20. Melihat data pada tabel MATAKULIAH;

Kemudian akan menampilkan data dari tabel MATAKULIAH

KD_MK	NAMA_MK	SKS	SEMESTER	KODE_DOS
PTI123	GRAFIKA MULTIMEDIA	3	5	12
PTI333	BASIS DATA TERDISTRIBUSI	3	5	10
PTI447	PRAKTIKUM BASIS DATA	1	3	11
PTI777	SISTEM INFORMASI	2	3	99
TIK123	JARINGAN KOMPUTER	2	5	33
TIK333	SISTEM OPERASI	3	5	10
TIK342	PRAKTIKUM BASIS DATA	1	3	11

Gambar 21. Data pada tabel MATAKULIAH;

Untuk melihat data yang telah ditambahkan pada tabel JURUSAN dengan pernyataan sebagai berikut :

```
SELECT * FROM `jurusan`
```

Gambar 22. Melihat data pada tabel JURUSAN;

Kemudian akan menampilkan data dari tabel JURUSAN

KODE_JUR	NAMA_JUR	KODE_DOS
TE	TEKNIK ELEKTRO	10
TM	TEKNIK MESIN	13
TS	TEKNIK SIPIL	23

Gambar 23. Data pada tabel JURUSAN;

Kemudian praktikkan menambahkan data pada tabel DOSEN sesuai dengan yang diperintahkan pada modul, dengan data seperti pada gambar dibawah ini.

```
INSERT INTO dosen VALUES
("10","SUHARTO","JL. JOMBANG"),
("11","MARTONO","JL. KALPATARU"),
("12","RAHMAWATI","JL. JAKARTA"),
("13","BAMBANG","JL. BANDUNG"),
("14","NURUL","JL. RAYA TIDAR");
```

Gambar 24. INSERT INTO DOSEN;

Untuk melihat data yang telah ditambahkan pada tabel DOSEN dengan pernyataan sebagai berikut :

```
SELECT * FROM `dosen`
```

Gambar 25. Melihat data pada tabel DOSEN

Kemudian akan menampilkan data dari 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

Gambar 26. Data pada tabel DOSEN

Praktikkan menambahkan data pada tabel AMBIL_MK sesuai dengan yang diperintahkan pada modul, dengan data seperti pada gambar dibawah ini.

```
INSERT INTO ambil_mk VALUES
("101","PTI447"),
("103","TIK333"),
("104","PTI333"),
("104","PTI777"),
("105","PTI123"),
("107","PTI777");
```

Gambar 27. INSERT INTO AMBIL_MK;

Untuk melihat data yang telah ditambahkan pada tabel AMBIL_MK dengan pernyataan sebagai berikut :

```
SELECT * FROM `ambil_mk`
```

Gambar 28. Melihat data pada tabel AMBIL_MK

Kemudian akan menampilkan data dari tabel AMBIL_MK

NIM	KD_MK
101	PTI447
103	TIK333
104	PTI333
104	PTI777
105	PTI123
107	PTI777

Gambar 29. Data pada tabel AMBIL_MK

Setelah semua tabel berhasil dibuat, praktikkan akan memulai percobaan latihan.

3.1.2 Relasi

Sebelum membuat relasi kelima tabel tersebut, praktikkan set beberapa *field* sebagai *foreign key*. Dengan cara klik tabel yang bersangkutan dan pilih index pada *field* yang bersangkutan. Yang dibuat sebagai *foreign key* antara lain *field* NIM, KD_MK pada tabel ambil_mk, *field* KODE_DOS pada tabel matakuliah, dan KODE_DOS pada tabel jurusan.

Gambar 30. Set INDEX pada field NIM tabel ambil_mk

Gambar 31. Set INDEX pada field KD_MK tabel ambil_mk

Setelah NIM dan KD_MK pada tabel ambil_mk di set index maka akan muncul kunci berwarna abu-abu seperti pada gambar dibawah ini.

Gambar 32. Hasil dari set INDEX pada field NIM, KD_MK tabel ambil_mk

Gambar 33. Set INDEX pada field KODE_DOS tabel matakuliah

Setelah KODE_DOS pada tabel matakuliah di set index maka akan muncul kunci berwarna abu-abu seperti pada gambar dibawah ini.

Gambar 34. Hasil dari set INDEX pada field KODE_DOS tabel matakuliah

Gambar 35. Set INDEX pada field KODE_DOS tabel jurusan

Setelah KODE_DOS pada tabel jurusan di set index maka akan muncul kunci berwarna abu-abu seperti pada gambar dibawah ini.

Gambar 36. Hasil dari set INDEX pada field KODE_DOS tabel jurusan

Setelah set *field* yang perlu di index kemudian membuat relasi dari kelimat tabel tersebut dengan klik *database* modul6_005 kemudian pilih *More* kemudian klik *designer*.

Gambar 30. Tabel pada database db2_005_praktikum

Kemudian praktikkan relasikan antara kelima tabel dengan pilih buat relasi/*create relation*.

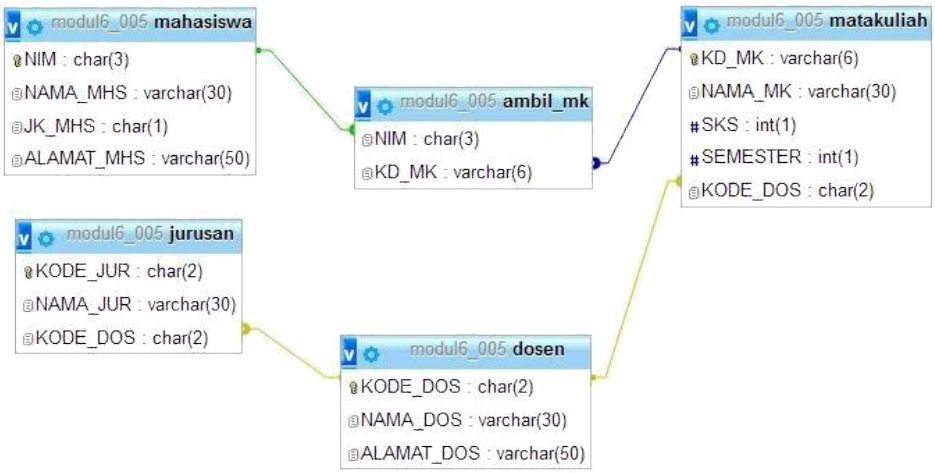
Gambar 37. Create relation pada database modul6_005

Kemudian pilih *primary key* atau kunci rujukan dan hubungkan dengan *foreign key*. Seperti gambar dibawah ini.

Gambar 38. Pilih kunci rujukan

Gambar 39. Pilih foreign key

Sehingga kelima tabel tersebut saling terhubung seperti gambar dibawah ini.



Gambar 40. Relationship pada database modul6_005

Pada tabel mahasiswa NIM sebagai *primary key* direlasikan dengan NIM pada tabel ambil_mk yang merupakan *foreign key*, kemudian KD_MK pada tabel matakuliah sebagai *primary key* direlasikan dengan KD_MK pada tabel ambil_mk yang merupakan *foreign key*, relasi selanjutnya adalah tabel dosen dengan KODE_DOS sebagai *primary key* direlasikan dengan KODE_DOS pada tabel jurusan dan KODE_DOS pada tabel matakuliah yang merupakan *foreign key*.

Kelima tabel tersebut telah direlasikan, sebelumnya praktikkan mendapatkan kesulitan untuk relasi pada KODE_DOS jurusan yang dihubungkan dengan KODE_DOS dosen dan KODE_DOS matakuliah, hal tersebut karena pada tabel jurusan dan matakuliah terdapat data KODE_DOS yang tidak tercatat dalam tabel dosen. Sehingga data pada tabel jurusan yang memiliki KODE_DOS 23 diubah sementara menjadi data KODE_DOS yang terdapat pada tabel dosen, praktikkan mengubahnya menjadi 11, dan KODE_DOS 13 pada tabel jurusan juga praktikkan ubah menjadi 12, sedangkan pada tabel matakuliah juga praktikkan mengubah data KODE_DOS yang memiliki KODE_DOS 99 dan 33, praktikkan mengubahnya menjadi 11 dan 12. Setelah itu praktikkan berhasil merelasikan kelima tabel tersebut, praktikkan mengembalikan data yang diubah tadi ke data yang sebenarnya seperti pada modul.

3.2 Scalar Subquery

Subquery baris tunggal (*scalar*) hanya mengembalikan hasil satu baris data. Contoh subquery baris tunggal adalah mendapatkan data mahasiswa yang jenis kelaminnya sama dengan mahasiswa dengan nama “WATI”. Dapat mengeksekusi pernyataan SQL sebagai berikut :

```
SELECT * FROM mahasiswa
WHERE JK_MHS =
(SELECT JK_MHS
FROM mahasiswa
WHERE NAMA_MHS = "WATI");
```

Gambar 41. Eksekusi scalar subquery

Sebagai hasilnya, didapatkan jenis kelamin mahasiswa dengan nama “WATI”, yakni “P” yang selanjutnya digunakan oleh *main query* sehingga menghasilkan sebagai berikut :

NIM	NAMA_MHS	JK_MHS	ALAMAT_MHS
103	WATI	P	JL.SURABAYA
104	IKA	P	JL.JOMBANG
107	SARI	P	JL.MALANG

Gambar 42. Hasil dari eksekusi scalar subquery

Karena WATI memiliki jenis kelamin P sehingga pernyataan diatas menampilkan data mahasiswa yang berjenis kelamin P.

3.3 Multiple-Row Subquery

3.3.1 Operator IN

Operator IN memiliki arti : sama dengan member di dalam list. Sebagai contoh, kita bisa menggunakan operator ini untuk mendapatkan data dosen yang mengajar matakuliah, dengan mengeksekusi pernyataan SQL dibawah ini :

```
SELECT d.KODE_DOS, d.NAMA_DOS
FROM dosen d
WHERE d.KODE_DOS IN
(SELECT KODE_DOS
FROM matakuliah);
```

Gambar 43. Eksekusi subquery dengan operator IN

Pada latihan ini menggunakan operator IN untuk menampilkan data dosen yang mengajar matakuliah dengan menampilkan KODE_DOS, NAMA_DOS dari tabel DOSEN kemudian menggunakan klausa WHERE KODE_DOS untuk spesifikasi data pada tabel dosen, operator IN dan seleksi untuk mendapatkan KODE_DOS yang mengajar pada tabel matakuliah. Pada latihan kali ini praktikkan menggunakan fitur *derived table* agar efisiensi dalam penulisan pernyataan dan mempermudah membedakan *field-field* tabel yang berbeda.

KODE_DOS	NAMA_DOS
10	SUHARTO
11	MARTONO
12	RAHMAWATI

Gambar 44. Hasil dari eksekusi subquery dengan operator IN

Sehingga dari eksekusi pernyataan diatas menampilkan *field* KODE_DOS, NAMA_DOS dengan 3 *record* antara lain “10, SUHARTO”, “11,MARTONO”, “12,RAHMAWATI” data dosen dengan KODE_DOS 13 dan 14 tidak ditampilkan karena tidak tercatat sebagai pengajar matakuliah pada tabel matakuliah, sehingga data tersebut tidak ditampilkan.

3.3.2 Operator ANY/SOME

Operator ANY / SOME memiliki arti : membandingkan suatu nilai dengan setiap nilai yang dikembalikan oleh subquery. Misalkan kita ingin mendapatkan data matakuliah yang memiliki sks lebih besar dari sembarang sks matakuliah di semester 3, dengan mengeksekusi pernyataan SQL dibawah ini :

```
SELECT * FROM matakuliah
WHERE SKS > ANY
(SELECT SKS
FROM matakuliah
WHERE SEMESTER = 3);
```

Gambar 45. Eksekusi subquery dengan operator ANY

Pada latihan ini menggunakan operator ANY untuk mendapatkan data matakuliah yang memiliki sks lebih besar dari sembarang sks matakuliah di semester 3 dengan menampilkan semua *field* pada tabel MATAKULIAH kemudian menggunakan klausa WHERE SKS>ANY untuk spesifikasi data yang berarti SKS lebih dari nilai minimal dan seleksi SKS pada tabel matakuliah dengan klausa WHERE yang memiliki SEMESTER 3. Perlu diketahui operator ANY memiliki arti dibawah ini :

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

KD_MK	NAMA_MK	SKS	SEMESTER	KODE_DOS
PTI123	GRAFIKA MULTIMEDIA	3	5	12
PTI333	BASIS DATA TERDISTRIBUSI	3	5	10
PTI777	SISTEM INFORMASI	2	3	99
TIK123	JARINGAN KOMPUTER	2	5	33
TIK333	SISTEM OPERASI	3	5	10

Gambar 46. Hasil dari eksekusi subquery dengan operator ANY

Sehingga dari eksekusi pernyataan diatas menampilkan *field* yang terdapat pada tabel matakuliah dengan 5 *record* antara lain “PTI123, GRAFIKA MULTIMEDIA, 3, 5, 12”, “PTI333, BASIS DATA TERDISTRIBUSI, 3, 5, 10”, “PTI777, SISTEM INFORMASI, 2, 3, 99”, “TIK123, JARINGAN KOMPUTER, 2, 5, 33”, “TIK333, SISTEM OPERASI,

3, 5, 10” . Karena didapatkan pada tabel matakuliah yang berstatus SEMESTER 3 memiliki SKS 1, dan SKS 2. Maka nilai minimal yang digunakan adalah 1 karena 1 lebih kecil daripada 2, sehingga SKS 1 digunakan *main query*. Maka dari itu matakuliah dengan NAMA_MK PRAKTIKUM BASIS DATA tidak ditampilkan dari hasil pernyataan diatas, karena tidak memenuhi pernyataan.

3.3.3 Operator ALL

Operator ALL memiliki arti: membandingkan suatu nilai dengan semua nilai yang dikembalikan oleh subquery. Misal, kita ingin mendapatkan data matakuliah yang memiliki sks lebih besar dari semua sks matakuliah di semester 3, dengan mengeksekusi pernyataan SQL dibawah ini :

```
MariaDB [modul6_005]> SELECT * FROM matakuliah WHERE SKS>ALL (SELECT SKS FROM ma
takuliah WHERE SEMESTER=3);
+--------+-----------------------+------+----------+----------+
| KD_MK  | NAMA_MK               | SKS  | SEMESTER | KODE_DOS |
+--------+-----------------------+------+----------+----------+
| PTI123 | GRAFIKA MULTIMEDIA    |    3 |        5 | 12       |
| PTI333 | BASIS DATA TERDISTRIBUSI |    3 |        5 | 10       |
| TIK333 | SISTEM OPERASI        |    3 |        5 | 10       |
+--------+-----------------------+------+----------+----------+
3 rows in set (0.02 sec)
```

Gambar 47. Pernyataan beserta hasil eksekusi subquery dengan operator ANY

Pada latihan ini praktikkan menggunakan *command prompt* karena praktikkan menggunakan phpMyAdmin versi 4.5.1 yang belum *support* dengan pernyataan yang akan dieksekusi. Praktikkan menggunakan operator ALL untuk mendapatkan data matakuliah yang memiliki sks lebih besar dari semua sks matakuliah di semester 3 dengan menampilkan semua *field* pada tabel MATAKULIAH kemudian menggunakan klausa WHERE SKS>ALL untuk spesifikasi data yang berarti SKS lebih dari nilai maksimal dan seleksi SKS pada tabel matakuliah dengan klausa WHERE yang memiliki SEMESTER 3. Perlu diketahui operator ALL memiliki arti dibawah ini :

Operator < **ALL** ekuivalen dengan **MIN** (kurang dari min).

Operator > **ALL** ekuivalen dengan **MAX** (lebih dari maks)

Sehingga dari eksekusi pernyataan diatas menampilkan *field* yang terdapat pada tabel matakuliah dengan 3 *record* antara lain “PTI123, GRAFIKA MULTIMEDIA, 3, 5, 12”, “PTI333, BASIS DATA TERDISTRIBUSI, 3, 5, 10”, “TIK333, SISTEM OPERASI, 3, 5, 10” . Karena didapatkan pada tabel matakuliah yang berstatus SEMESTER 3 memiliki SKS 1, dan SKS 2. Maka nilai maksimal yang digunakan adalah 2 karena 2 lebih besar daripada 1, sehingga SKS 2 digunakan *main query*. Maka dari itu matakuliah dengan NAMA_MK PRAKTIKUM BASIS DATA, SISTEM INFORMASI, JARINGAN

KOMPUTER tidak ditampilkan dari hasil pernyataan diatas, karena tidak memenuhi pernyataan.

3.4 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 kita ingin menampilkan data matakuliah yang semester dan sksnya sesuai dengan semester dan sks matakuliah dengan kode “PTI447”, dengan mengeksekusi pernyataan SQL dibawah ini :

```
SELECT * FROM matakuliah
WHERE (SEMESTER, SKS) IN
(SELECT SEMESTER, SKS
FROM matakuliah
WHERE KD_MK = "PTI447");
```

Gambar 48. Eksekusi multiple column subquery

Pada latihan ini menggunakan operator IN untuk mendapatkan data matakuliah yang memiliki semester dan sksnya sesuai semester dan sks matakuliah kode “PTI447” dengan menampilkan semua *field* pada tabel MATAKULIAH kemudian menggunakan klausa WHERE (SEMESTER, SKS) untuk spesifikasi data SEMESTER dan SKS pada tabel MATAKULIAH, operator IN dan seleksi untuk mendapatkan SEMESTER, SKS dari tabel matakuliah dengan klausa WHERE yang memiliki KD_MK “PTI447”.

KD_MK	NAMA_MK	SKS	SEMESTER	KODE_DOS
PTI447	PRAKTIKUM BASIS DATA	1	3	11
TIK342	PRAKTIKUM BASIS DATA	1	3	11

Gambar 49. Hasil dari eksekusi multiple column subquery

Sehingga dari eksekusi pernyataan diatas menampilkan *field* yang terdapat pada tabel matakuliah dengan 2 *record* antara lain “PTI447, PRAKTIKUM BASIS DATA, 1, 3, 11”, “TIK342, PRAKTIKUM BASIS DATA, 1, 3, 11”. Karena didapatkan pada tabel matakuliah dengan kode PTI447 memiliki SEMESTER 3 memiliki SKS 1. Sehingga SEMESTER 3 dengan SKS 1 digunakan *main query*. Maka dari itu matakuliah dengan NAMA_MK PRAKTIKUM BASIS DATA ditampilkan dari hasil pernyataan diatas, karena memenuhi pernyataan.

3.5 Operator EXIST dan NOT EXIST

Operator EXISTS dan NOT EXISTS digunakan pada *correlated subquery* untuk memeriksa apakah subquery mengembalikan hasil atau tidak. Apabila subquery mengembalikan hasil, EXIST akan mengembalikan nilai TRUE. Begitu pula sebaliknya, jika tidak

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

3.5.1 EXIST

```
SELECT * FROM matakuliah k
WHERE EXISTS (SELECT * FROM ambil_mk a
WHERE k.KD_MK = a.KD_MK);
```

Gambar 50. Eksekusi subquery dengan operator EXIST

Pada latihan ini menggunakan operator EXIST untuk mendapatkan data matakuliah yang diambil oleh mahasiswa dengan menampilkan semua *field* pada tabel MATAKULIAH kemudian menggunakan klausa WHERE EXIST untuk mengembalikan hasil yang bernilai TRUE dan seleksi semua data pada tabel ambil_mk dengan klausa WHERE yang memiliki KD_MK pada tabel matakuliah sama dengan KD_MK pada tabel ambil_mk. Pada latihan kali ini praktikkan menggunakan fitur *derived table* agar efisiensi dalam penulisan pernyataan dan mempermudah membedakan *field-field* tabel yang berbeda.

KD_MK	NAMA_MK	SKS	SEMESTER	KODE_DOS
PTI123	GRAFIKA MULTIMEDIA	3	5	12
PTI333	BASIS DATA TERDISTRIBUSI	3	5	10
PTI447	PRAKTIKUM BASIS DATA	1	3	11
PTI777	SISTEM INFORMASI	2	3	99
TIK333	SISTEM OPERASI	3	5	10

Gambar 51. Hasil dari eksekusi subquery dengan operator EXIST

Sehingga dari eksekusi pernyataan diatas menampilkan *field* yang terdapat pada tabel matakuliah dengan 5 *record* antara lain “PTI123, GRAFIKA MULTIMEDIA, 3, 5, 12”, “PTI333, BASIS DATA TERDISTRIBUSI, 3, 5, 10”, “PTI447, PRAKTIKUM BASIS DATA, 1, 3, 11”, “PTI777, SISTEM INFORMASI, 2, 3, 99”, “TIK333, SISTEM OPERASI, 3, 5, 10”. Data matakuliah dengan KD_MK TIK123 dan TIK342 tidak ditampilkan karena tidak tercatat sebagai matakuliah yang diambil mahasiswa pada tabel ambil_mk, sehingga data tersebut tidak ditampilkan.

3.5.2 NOT EXIST

```
MariaDB [modul6_005]> SELECT * FROM matakuliah k WHERE NOT EXISTS (SELECT * FROM
 ambil_mk a WHERE k.KD_MK = a.KD_MK);
+--------+-----------------+------+----------+----------+
| KD_MK  | NAMA_MK         | SKS  | SEMESTER | KODE_DOS |
+--------+-----------------+------+----------+----------+
| TIK123 | JARINGAN KOMPUTER |    2 |        5 | 33       |
| TIK342 | PRAKTIKUM BASIS DATA |    1 |        3 | 11       |
+--------+-----------------+------+----------+----------+
2 rows in set (0.00 sec)
```

Gambar 52. Pernyataan beserta hasil eksekusi subquery dengan operator NOT EXIST

Pada latihan ini praktikkan menggunakan *command prompt* karena praktikkan menggunakan phpMyAdmin versi 4.5.1 yang belum *support* dengan pernyataan yang akan dieksekusi. Praktikkan menggunakan operator NOT EXIST untuk mendapatkan data matakuliah yang tidak diambil oleh mahasiswa dengan menampilkan semua *field* pada tabel MATAKULIAH kemudian menggunakan klausa WHERE NOT EXIST untuk mengembalikan hasil yang bernilai FALSE dan seleksi semua data pada tabel ambil_mk dengan klausa WHERE yang memiliki KD_MK pada tabel matakuliah sama dengan KD_MK pada tabel ambil_mk. Pada latihan kali ini praktikkan menggunakan fitur *derived table* agar efisiensi dalam penulisan pernyataan dan mempermudah membedakan *field-field* tabel yang berbeda.

Sehingga dari eksekusi pernyataan diatas menampilkan *field* yang terdapat pada tabel matakuliah dengan 2 *record* antara lain “TIK123, JARINGAN KOMPUTER, 2, 5, 33”, “TIK342, PRAKTIKUM BASIS DATA, 1, 3, 11”. Data matakuliah dengan KD_MK PTI123, PTI333, PTI447, PTI777, dan TIK333 tidak ditampilkan karena tercatat sebagai matakuliah yang diambil mahasiswa pada tabel ambil_mk, sehingga data tersebut tidak ditampilkan.

4. Tugas Praktikum

4.1 Dapatkan kode dan nama matakuliah dosen yang menjadi Ketua Jurusan Teknik Elektro.

```
1 SELECT KD_MK, NAMA_MK FROM matakuliah k
2 WHERE k.KODE_DOS =
3 (SELECT j.KODE_DOS FROM jurusan j
4  WHERE KODE_JUR = "TE");
```

Gambar 54. Eksekusi scalar subquery

Pada latihan ini menggunakan subquery baris tunggal (*scalar*) hanya mengembalikan hasil satu baris data. Untuk mendapatkan kode dan nama matakuliah dosen yang menjadi ketua jurusan teknik elektro dengan menampilkan KD_MK, NAMA_MK dari tabel MATAKULIAH kemudian menggunakan klausa WHERE KODE_DOS untuk spesifikasi data pada tabel matakuliah, operator = (sama dengan) dan seleksi untuk mendapatkan KODE_DOS pada tabel jurusan, lalu klausa WHERE yang menjadi spesifikasi KODE_JUR = TE pada tabel jurusan. Pada latihan kali ini praktikkan menggunakan fitur *derived table* agar efisiensi dalam penulisan pernyataan dan mempermudah membedakan *field-field* tabel yang berbeda.

KD_MK	NAMA_MK
PTI333	BASIS DATA TERDISTRIBUSI
TIK333	SISTEM OPERASI

Gambar 55. Hasil eksekusi scalar subquery

Sehingga dari eksekusi pernyataan diatas menampilkan *field* yang terdapat KD_MK dan NAMA_MK dengan 2 *record* antara lain “PTI333, BASIS DATA TERDISTRIBUSI”, “TIK333, SISTEM OPERASI”. Data matakuliah dengan KD_MK PTI123, PTI447, PTI777, TIK123 dan TIK342 tidak ditampilkan karena matakuliah tersebut tercatat sebagai data matakuliah yang tidak diajarkan oleh dosen yang menjadi ketua jurusan elektro, sehingga data tersebut tidak ditampilkan.

4.2 Dapatkan data mahasiswa yang tidak mengambil matakuliah.

```
1 SELECT * FROM mahasiswa WHERE NIM NOT IN
2 (SELECT NIM FROM ambil_mk);
```

Gambar 56. Eksekusi subquery dengan operator NOT IN

Pada latihan ini menggunakan operator NOT IN untuk menampilkan data mahasiswa yang tidak mengambil matakuliah dengan menampilkan semua data dari tabel MAHASISWA kemudian menggunakan klausa WHERE NIM untuk spesifikasi data pada tabel mahasiswa, operator NOT IN dan seleksi untuk mendapatkan NIM yang tidak ada pada tabel ambil_mk.

NIM	NAMA_MHS	JK_MHS	ALAMAT_MHS
102	BUDI	L	JL.JOMBANG
106	IWAN	L	JL.BANDUNG

Gambar 57. Hasil dari eksekusi subquery dengan operator NOT IN

Sehingga dari eksekusi pernyataan diatas menampilkan *field* yang ada pada tabel mahasiswa dengan 2 *record* antara lain “102, BUDI, L, JL.JOMBANG”, “106, IWAN, L, JL.BANDUNG” data mahasiswa dengan NIM 101, 103, 104, 105, dan 107 tidak ditampilkan karena tercatat sebagai mahasiswa yang mengambil matakuliah pada tabel ambil_mk, sehingga data tersebut tidak ditampilkan.

Selain menggunakan subquery dengan operator NOT IN juga dapat menggunakan subquery dengan operator NOT EXIST, dengan pernyataan dibawah ini.

```
MariaDB [modul6_005]> SELECT * FROM mahasiswa m WHERE NOT EXISTS (SELECT * FROM
ambil_mk a WHERE m.NIM = a.NIM) ;
+-----+----------+--------+-------------+
| NIM | NAMA_MHS | JK_MHS | ALAMAT_MHS |
+-----+----------+--------+-------------+
| 102 | BUDI     | L      | JL.JOMBANG |
| 106 | IWAN     | L      | JL.BANDUNG |
+-----+----------+--------+-------------+
2 rows in set (0.00 sec)
```

Gambar 58. Pernyataan beserta hasil eksekusi subquery dengan operator NOT EXIST

Pada latihan ini praktikkan menggunakan *command prompt* karena praktikkan menggunakan phpMyAdmin versi 4.5.1 yang belum *support* dengan pernyataan yang akan dieksekusi. Praktikkan menggunakan operator NOT EXIST untuk mendapatkan data mahasiswa yang tidak mengambil matakuliah dengan menampilkan semua *field* pada tabel mahasiswa. Terlihat hasil eksekusi pernyataan subquery dengan operator NOT IN dan operator NOT EXIST menampilkan data yang sama.

4.3 Dapatkan data dosen yang mengajar matakuliah diatas semester 3.

```
1 SELECT * FROM dosen d WHERE KODE_DOS IN
2 (SELECT KODE_DOS FROM matakuliah m WHERE SEMESTER>3);
```

Gambar 59. Eksekusi subquery dengan operator IN

Pada latihan ini menggunakan operator IN untuk menampilkan data dosen yang mengajar matakuliah diatas semester 3 dengan menampilkan semua *field* dari tabel DOSEN kemudian menggunakan klausa WHERE KODE_DOS untuk spesifikasi data pada tabel dosen, operator IN dan seleksi untuk mendapatkan KODE_DOS yang mengajar pada tabel matakuliah dan menggunakan klausa WHERE SEMESTER>3 untuk spesifikasi dosen yang mengajar matakuliah diatas semester 3.

KODE_DOS	NAMA_DOS	ALAMAT_DOS
10	SUHARTO	JL. JOMBANG
12	RAHMAWATI	JL. JAKARTA

Gambar 60. Hasil dari eksekusi subquery dengan operator IN

Sehingga dari eksekusi pernyataan diatas menampilkan *field* KODE_DOS, NAMA_DOS, ALAMAT_DOS dengan 2 *record* antara lain “10, SUHARTO, JL.JOMBANG”, “12, RAHMAWATI, JL.JAKARTA” data dosen dengan KODE_DOS 11, 13 dan 14 tidak ditampilkan karena tidak tercatat sebagai pengajar matakuliah diatas semester 3 pada tabel matakuliah, sehingga data tersebut tidak ditampilkan.

4.4 Dapatkan data matakuliah dosen yang bukan merupakan Ketua Jurusan Teknik Elektro.

```
1 SELECT * FROM matakuliah WHERE KODE_DOS NOT IN (SELECT KODE_DOS FROM jurusan WHERE KODE_JUR="TE")
2 AND KODE_DOS IN (SELECT KODE_DOS FROM dosen);
```

Gambar 61. Eksekusi subquery dengan operator NOT IN dan IN

Pada latihan ini menggunakan operator NOT IN untuk menampilkan data matakuliah dosen yang bukan merupakan ketua jurusan teknik elektro dengan menampilkan semua data dari tabel MATAKULIAH kemudian menggunakan klausa WHERE KODE_DOS untuk spesifikasi data pada tabel matakuliah, operator NOT IN dan seleksi untuk mendapatkan KODE_DOS yang bukan merupakan dosen ketua jurusan teknik elektro pada tabel jurusan

dengan klausa WHERE KODE_JUR=”TE”, kemudian menggunakan operator AND untuk lebih menspesifikasi data dimana KODE_DOS tabel matakuliah berada pada KODE _DOS tabel dosen.

KD_MK	NAMA_MK	SKS	SEMESTER	KODE_DOS
PTI123	GRAFIKA MULTIMEDIA	3	5	12
PTI447	PRAKTIKUM BASIS DATA	1	3	11
TIK342	PRAKTIKUM BASIS DATA	1	3	11

Gambar 62. Hasil dari eksekusi subquery dengan operator NOT IN dan IN

Sehingga dari eksekusi pernyataan diatas menampilkan *field* yang ada pada tabel matakuliah dengan 3 *record* antara lain “PTI123, GRAFIKA MULTIMEDIA, 3, 5, 12”, “PTI447, PRAKTIKUM BASIS DATA, 1, 3, 11”, “TIK342, PRAKTIKUM BASIS DATA, 1, 3, 11” data matakuliah dengan KD_MK PTI333 dan TIK333 tidak ditampilkan karena tercatat sebagai matakuliah yang diajarkan oleh dosen ketua juruan teknik elektro pada tabel ambil_mk, sedangkan KD_MK PTI777, TIK123 tidak ditampilkan karena KODE_DOS pada matakuliah tersebut tidak tercatat pada KODE_DOS yang ada pada tabel dosen sehingga data tersebut tidak ditampilkan.

4.5 Dapatkan data dosen pengajar matakuliah yang tidak diambil oleh mahasiswa.

```
1 SELECT * FROM dosen WHERE KODE_DOS IN (SELECT KODE_DOS FROM matakuliah WHERE KD_MK NOT IN
2                                        (SELECT KD_MK FROM ambil_mk));
```

Gambar 63. Eksekusi subquery dengan operator IN dan NOT IN

Pada latihan ini menggunakan operator IN untuk menampilkan data dosen pengajar matakuliah yang tidak diambil oleh mahasiswa dengan menampilkan semua *field* dari tabel DOSEN kemudian menggunakan klausa WHERE KODE_DOS untuk spesifikasi data pada tabel dosen, operator IN dan seleksi untuk mendapatkan KODE_DOS yang mengajar pada tabel matakuliah dan menggunakan klausa WHERE KD_MK NOT IN untuk spesifikasi KD_MK yang tidak ada pada seleksi KD_MK di tabel ambil_mk.

KODE_DOS	NAMA_DOS	ALAMAT_DOS
11	MARTONO	JL. KALPATARU

Gambar 64. Hasil dari eksekusi subquery dengan operator IN dan NOT IN

Sehingga dari eksekusi pernyataan diatas menampilkan *field* KODE_DOS, NAMA_DOS, ALAMAT_DOS dengan 1 *record* antara lain “11, MARTONO, JL.KALPATARU” data dosen tersebut muncul karena KODE_DOS 11 mengajarkan pada KD_MK TIK342 dengan NAMA_MK PRAKTIKUM BASIS DATA yang dimana matakuliah tersebut tidak diambil oleh mahasiswa yang tercatat pada tabel ambil_mk.

5. Tugas Rumah

5.1 Dapatkan data dosen yang mengajar matakuliah dengan sks lebih besar dari sembarang sks.

```
SELECT * FROM dosen  WHERE KODE_DOS IN (SELECT KODE_DOS FROM matakuliah
                                        WHERE SKS>ANY (SELECT SKS FROM matakuliah));
```

Gambar 65. Eksekusi subquery dengan operator IN dan ANY

Pada latihan ini menggunakan operator IN untuk menampilkan data dosen yang mengajar matakuliah dengan sks lebih besar sembarang sks dengan menampilkan semua *field* dari tabel DOSEN kemudian menggunakan klausa WHERE KODE_DOS untuk spesifikasi data pada tabel dosen, operator IN dan seleksi untuk mendapatkan KODE_DOS yang mengajar pada tabel matakuliah dan menggunakan klausa WHERE SKS>ANY untuk spesifikasi SKS yang lebih besar sembarang sks dengan seleksi SKS pada tabel matakuliah.

KODE_DOS	NAMA_DOS	ALAMAT_DOS
10	SUHARTO	JL. JOMBANG
12	RAHMAWATI	JL. JAKARTA

Gambar 66. Hasil dari eksekusi subquery dengan operator IN dan ANY

Sehingga dari eksekusi pernyataan diatas menampilkan *field* KODE_DOS, NAMA_DOS, ALAMAT_DOS dengan 2 *record* antara lain “10, SUHARTO, JL.JOMBANG”, “12, RAHMAWATI, JL.JAKARTA” data dosen dengan KODE_DOS 11, 13 dan 14 tidak ditampilkan karena tidak tercatat sebagai pengajar matakuliah dengan sks lebih besar sembarang sks, sehingga data tersebut tidak ditampilkan.

5.2 Dapatkan data mahasiswa yang tinggal atau satu wilayah dengan dosen yang bukan merupakan Ketua Jurusan Teknik Elektro.

```
SELECT * FROM mahasiswa
WHERE ALAMAT_MHS IN
(SELECT ALAMAT_DOS FROM dosen
 WHERE KODE_DOS NOT IN
(SELECT KODE_DOS FROM jurusan
 WHERE KODE_JUR="TE"))
```

Gambar 67. Eksekusi subquery dengan operator IN dan NOT IN

Pada latihan ini menggunakan operator IN untuk menampilkan data mahasiswa yang tinggal dengan dosen yang bukan merupakan ketua jurusan teknik elektro dengan menampilkan semua *field* dari tabel MAHASISWA kemudian menggunakan klausa WHERE ALAMAT_MHS untuk spesifikasi data pada tabel mahasiswa, operator IN dan seleksi untuk mendapatkan ALAMAT_DOS pada tabel dosen yang sama dengan alamat mahasiswa dan menggunakan klausa WHERE KODE_DOS NOT IN untuk spesifikasi KODE_DOS di tabel jurusan dengan klausa WHERE KODE_JUR=”TE” untuk spesifikasi

KODE_DOS yang selain KODE_DOS yang berstatus sebagai kepala jurusan teknik elektro.

NIM	NAMA_MHS	JK_MHS	ALAMAT_MHS
105	TONO	L	JL.JAKARTA
106	IWAN	L	JL.BANDUNG

Gambar 68. Hasil dari eksekusi subquery dengan operator IN dan NOT IN

Sehingga dari eksekusi pernyataan diatas menampilkan *field* pada tabel mahasiswa dengan 2 *record* antara lain “105, TONO, L, JL.JAKARTA”, “106, IWAN, L, JL.BANDUNG” data mahasiswa tersebut muncul karena KODE_DOS 12 memiliki ALAMAT di JL.JAKARTA dan KODE_DOS 13 memiliki ALAMAT di JL.BANDUNG yang dimana data dosen tersebut tercatat pada tabel dosen dan tidak tercatat sebagi ketua jurusan teknik elektro.

5.3 Dapatkan data mahasiswa yang diajar oleh Ketua Jurusan Teknik Elektro.

```
1 SELECT * FROM mahasiswa WHERE NIM IN (SELECT NIM FROM ambil_mk WHERE KD_MK IN
2                                       (SELECT KD_MK FROM matakuliah WHERE KODE_DOS IN
3                                        (SELECT KODE_DOS FROM jurusan WHERE KODE_JUR="TE")));
```

Gambar 69. Eksekusi subquery dengan operator IN

Pada latihan ini menggunakan operator IN untuk menampilkan data mahasiswa yang diajar oleh ketua jurusan teknik elektro dengan menampilkan semua *field* dari tabel MAHASISWA kemudian menggunakan klausa WHERE NIM untuk spesifikasi data pada tabel mahasiswa, operator IN dan seleksi untuk mendapatkan NIM yang ada pada tabel ambil_mk dan menggunakan klausa WHERE KD_MK IN dimana KD_MK terdapat pada seleksi KD_MK pada tabel matakuliah dan menggunakan klausa WHERE KODE_DOS IN dimana KODE_DOS terdapat pada seleksi KODE_DOS pada tabel jurusan dengan klausa WHERE KODE_JUR=”TE” untuk spesifikasi kode jurusan sama dengan TE pada tabel jurusan.

NIM	NAMA_MHS	JK_MHS	ALAMAT_MHS
103	WATI	P	JL.SURABAYA
104	IKA	P	JL.JOMBANG

Gambar 70. Hasil dari eksekusi subquery dengan operator IN

Sehingga dari eksekusi pernyataan diatas menampilkan *field* dari tabel mahasiswa dengan 2 *record* antara lain “103, WATI, P, JL.SURABAYA”, “104, IKA, P, JL.JOMBANG” data mahasiswa dengan NIM 101, 102, 105, 106 dan 107 tidak ditampilkan karena tidak tercatat sebagai mahasiswa yang mengambil matakuliah dengan dosen pengajar ketua jurusan teknik elektro, sehingga data tersebut tidak ditampilkan.

6. Kesimpulan

Dalam praktikum modul 6 Subquery dapat disimpulkan bahwa subquery (disebut juga *subselect* atau *nested select / query* atau *inner-select*) adalah query `SELECT` yang ada di dalam perintah SQL lain misalnya `SELECT, INSERT, UPDATE,` atau `DELETE.` Subquery memiliki 3 jenis yaitu subquery baris tunggal (*scalar*) yang hanya mengembalikan hasil satu baris data, subquery baris ganda (*multiple-row*) yang mengembalikan lebih dari satu baris data. Subquery dapat menggunakan operator komparasi **IN, ANY/SOME,** atau **ALL,** dan yang terakhir adalah subquery kolom ganda (*multipe-column*) mengembalikan lebih dari satu bari dan satu kolam data.

7. Daftar Pustaka

Jurusan Teknik Elektro. 2017. Modul 6 : Subquery. Malang: Universitas Negeri Malang.

Fikri, Ahmad. Subquery. 2014. https://mbahsantri.wordpress.com/2014/11/19/sub-query-indeks/, diakses pada pada tanggal 29 Maret 2017, pukul 02:58 WIB.