MODUL PRATIKUM
SISTEM BASIS DATA
(MYSQL)

DISUSUN OLEH: TIM DOSEN
PROGRAM STUDI TEKNIK INFORMATIKA

PROGRAM STUDI TEKNIK INFORMATIKA
UNIVERSITAS INDRAPRASTA PGRI

MODUL PRAKTIKUM SISTEM BASIS DATA (MYSQL)
Tim Dosen Program Studi Teknik Informatika

Cetakan I: Mei 2012

UNINDRA PRESS.
Jl. Nangka No.58C Tanjung Barat (TB Simatupang),
Jagakarsa, Jakarta Selatan 12530
Telp./Fax.: (021) 7818718 - 78835283
Homepage: www.unindra.ac.id/
Email : university@unindra.ac.id

Daftar Isi

1
PENGENALAN SQL

SEJARAH SQL

Sejak kapan SQL mulai diperkenalkan ke khalayak umum sebagai bahasa standar database? Pada tahun 1970, seiring dengan perkembangan database yang kian kompleks. E.F. Cold memperkenalkan database relasional dalam sebuah artikel yang berjudul "A Relational Model of data for large shared data bank". Dari sinilah lahir suatu konsep dasar untuk mengembangkan database atau basis data.

Jadi pada tahun 1979-lah awal mula lahirnya SQL, yaitu dimulai dari Codd yang memperkenalkan idenya tersebut dalam konsep yang lebih nyata, kemudian dikembangkan menjadi sebuah database relasional, yang salah satunya dari bahasa database relasional tersebut adalah SQL.

Kelahiran SQL juga tidak terlepas dengan sebuah proyek IBM yang dikenal dengan Sistem R. Proyek, yaitu sebuah proyek yang bertujuan untuk mengembangkan sebuah sistem database relasional yang dapat memenuhi segala jenis sistem pengoperasian database modern. Dengan memperkenalkan sebuah bahasa yang dinamakan sequel yang berkembang menjadi SQL.

PENGGUNAAN SQL

Menurut penggunaannya, perintah-perintah SQL dapat dikelompokkan menjadi 2 bagian, yaitu :

1. Secara Interpretasi (Interactive SQL), yaitu dengan cara memasukkan perintah-perintah SQL melalui console atau mikrokomputer dan secara langsung diproses sehingga dapat langsung dilihat.
2. Secara Sisip (Embedded SQL), yaitu dengan cara menyisipkan perintah-perintah SQL ke dalam bahasa pemrogram tertentu sehingga untuk melihatnya dibutuhkan media khusus yang dirancang oleh seorang programmer.

STATEMEN SQL

Yang dimaksud dengan statement SQL adalah sekumpulan perintah-perintah SQL yang memiliki peranan dalam pembentukan dan pengaturan suatu database.

Statemen SQL terbagi menjadi 3 bagian, yaitu :

1. DDL (Data Definition Language), yaitu sebuah perintah SQL yang berorientasi pada pembentukan atau penghapusan database, tabel dan index.

   Yang termasuk ke dalam kategori DDL :

   CREATE DATABASE DROP DATABASE
   CREATE TABLE DROP TABLE ALTER TABLE
   CREATE INDEX DROP INDEX
   CREATE VIEW

2. DML (Data Manipulation Language), yaitu perintah-perintah SQL yang berhubungan dengan data atau record, di antaranya menampilkan data, menghapus data, dan meng-update data.
   Yang termasuk ke dalam kategori DML :
   INSERT, SELECT, UPDATE, dan DELETE
3. DCL (Data Control Language), merupakan kumpulan perintah SQL yang berfungsi untuk melakukan pendefinisian pemakai yang boleh atau tidak mengakses database dan apa saja privilegenya.
   Yang termasuk dalam kategori DCL :
   COMMIT, ROLLBACK, GRANT, dan REVOKE.

MySQL Console dari DOS pada sistem operasi Windows penulisan perintah-perintahnya tidak membedakan huruf besar dan huruf kecil, tapi pada sistem operasi Unix/Linux huruf besar dan kecil harus dibedakan. Dan sintak SQL pada linux harus huruf kecil semua.

FUNGSI SQL

Fungsi-fungsi SQL ini penggunaannya harus bersamaan dengan perintah DML. Fungsi-fungsi SQL yang lazim digunakan, di antaranya adalah :

1. Fungsi Agregat, yaitu sebuah fungsi built-in yang hampir pasti ada dalam sistem database relasional. Dengan kata lain fungsi agregat merupakan fungsi standar dari SQL.
   Yang termasuk dalam fungsi agregat :
   AVG, SUM, COUNT, MAX, MIN, STD, dan STDDEV.
2. Fungsi Aritmatik, yaitu sebuah fungsi yang berguna dalam proses perhitungan atau manipulasi data numerik.
   Yang termasuk dalam fungsi aritmatik, di antaranya :
   + (penjumlahan), - (pengurangan), * (perkalian), / (pembagian), % (sisa hasil bagi), ABS (x), ACOS (x), ASIN (x), ATAN (x), COS (x), CEILING (x), ROUND (x), dan lain sebagainya.
3. Fungsi String, berfungsi untuk melakukan manipulasi data yang bertipe data datetime Format tanggal dan jam pada mySQL :
   Yyyy-mm-dd hh:ii:ss
   Yang termasuk dalam fungsi string, di antaranya :
   NOW ( ), HOUR ( ), MINUTE ( ), MONTH ( ), dan sebagainya.

* * *

2

DATA DEFINITION LANGUAGE

1. PENDAHULUAN

Sebelumnya, telah dijelaskan bagaimana cara melakukan pembuatan dan pengaturan database melalui browser, yaitu dengan phpmyadmin. Mungkin bagi yang terbiasa melakukan pembuatan dan pengaturan database lewat console atau dos-prompt akan terasa kurang nyaman jika harus melakukannya pada browser. Untuk itu pada bab ini, akan dibahas bagaimana cara melakukan pembuatan dan pengaturan database lewat console pada operating system Linux atau dos-prompt pada operating system Windows.

Cara pembuatan dan pengaturan database pada console Linux :

1. Pada console, masukkan perintah :
   su –
   Untuk masuk ke dalam super user.

2. Lalu masukkan perintah :
   /opt/lampp/lampp start
   Untuk mengaktifkan apache dan MySQL

3. Setelah mengaktifkan apapche dan MySQL, langkah selanjutnya yaitu menjalankan mysql. Dengan cara memasukkan perintah :
   /opt/lampp/bin/mysql
   Akan terlihat tampilan pada console.

Cara pembuatan dan pengaturan database pada dos-prompt Windows :

1. Setelah mengaktifkan command prompt atau dos-prompt, masukkan perintah seperti berikut :
   cd\Program Files\xampp\mysql\bin
2. Akan terlihat pada command prompt :
   C:\Program Files\xampp\mysql\bin> dan masukkan perintah seperti berikut :
   Mysql –u root

2. DATABASE

Terdapat perintah-perintah DDL dalam pembuatan, penghapusan, pengaktifan, dan menampilkan database.

- **Membuat Database**

Bentuk umum penulisannya :

→ CREATE DATABASE nama_database;

Ketentuan dalam membuat nama database, jangan menggunakan spasi jika memiliki nama database lebih dari satu kata. Dapat menggunakan underscore (_).
Contoh :
Mysql>CREATE DATABASE siswa_25;

- **Menampilkan Seluruh Database**

Bentuk umum penulisannya:
→ SHOW DATABASE;
Contoh :
Mysql>SHOW DATABASES;

- **Mengaktifkan Database**

Bentuk umum penulisannya :
Use name_database;
Contoh :
Mysql>USE siswa;

- **Menghapus Database**

Bentuk umum penulisannya :
DROP DATABASE nama_database;
Contoh :
Mysql>DROP DATABASE baru;

3. TABEL

Tabel merupakan media yang dapat melakukan proses relasional antartabel. Pada tabel terdapat field dan record.

Field merupakan judul kolom yang memiliki tipe data, size record, kunci relasi, dan sebagainya. Record adalah kumpulan data yang tersusun secara per baris.

Tipe Data

Tipe data dapat dikelompokkan menjadi 3 bagian, yaitu tipe data string, tipe data numerik, dan tipe data datetime.

Tipe Data String

Yang termasuk ke dalam kategori ini, adalah :

- Char

  Pendeklarasian Char (size)

  Char merupakan tipe data string yang menyediakan panjang karakter maksimal 255 karakter. Tipe data char juga memiliki panjang yang tetap untuk setiap data yang dimasukkan sesuai dengan panjang yang dideklarasikan. Contoh: Anda menentukan char (15), lalu terdapat record "Istimewa" pada field tersebut. Karena "Istimewa" memiliki 8 karakter maka mysql akan menambah 7 spasi untuk melengkapi sisa spasi menjadi 15 spasi sesuai dengan apa yang telh dideklarasikan.

- Varchar
  Pendeklarasian : Varchar (size)
  Pada dasarnya tipe data varchar memiliki kesamaan dengan tipe data char, yaitu memiliki panjang maksimal 255 karakter. Perbedaannya apabila ada pada char jika jumlah karakter pada suatu field kurang dari ukuran nilai yang telah dideklarasikan, maka sisanya akan ditambahkan oleh jumlah spasi yang tersisa. Tidak demikian halnya dengan varchard, jika jumlah karakter pada suatu field kurang dari ukuran nilai yang telah dideklarasikan, maka tidak akan ditambahkan spasi, melainkan hanya memasukkan jumlah karakter yang dimasukkan saja, sehingga tipe data varchar lebih hemat dari char.

- Tinytext
  Pendeklarasian: Tinytext
  Memiliki nilai size yang sama dengan varchar (255).

- Text dan Blob
  Pendeklarasian: Text atau Blob
  Tipe data text dan blob memikiki kesamaan dengan tipe data char dan varchar, yaitu memiliki kesamaan dan perbedaan. Persamaan antara tipe data text dan blob adalah dapat menampung teks atau string yang tidak terbatas jumlahnya. Perbedaan tipe data blob memungkinkan untuk menyimpan data gambar atau dokumen, sehingga antara dokumen dan gambar tidak lagi terpisah. Tipe data text hanya dapat menampung teks atau string saja.

- Mediumtext
  Pendeklarasian: Mediumtext
  Tipe data ini memiliki panjang maksimal 1.677.215 karakter.

- Longtext
  Pendeklarasian : Longtext
  Tipe data longtext memiliki size maksimal 4.294.967.295 karakter.

Tipe Data Numerik

Yang termasuk ke dalam kategori ini adalah :

- Integer/Int
  Pendeklarasian: Int(size)
  Nilai yang dapat disimpan antara -2.147.483.648 sampai 4.294.967.295.

- Tinyint
  Pendeklarasian: Tinyint(size)
  Nilai yang dapt disimpan antara -128 sampai 255.

- Mediumint
  Pendeklarasian: Mediumint(size)
  Nilai yang dapat disimpan antara -8.388.608 sampai 8.288.607.

- Bigint
  Pendeklarasian: Bigint(size)
  Nilai yang dapat disimpan antara -92.233.720.368.547.758.078 sampai 92.233.720.368.547.758.078.

- Float
  Pendeklarasian: Float
  Float menyimpan bilangan real dan tidak dapat bernilai negatif.

- Double
  Pendeklarasian: Double
  Kebalikan dari float yang hanya menerima bilangan real, double dapat menerima bilangan real atau desimal.

Tipe Data Date dan Time

Yang termasuk ke dalam kategori ini, adalah :

- Date
  Pendeklarasian: Date
  Date menyimpan nilai format YYYY-MM-DD. Nilai yang diizinkan antara 1000-01-01 sampai dengan 9999-12-31.

- Datetime
  Pendeklarasian: Datetime
  Datetime menyimpan nilai format YYYY-MM-DD HH:MM:SS. Nilai yang diizinkan antara 1000-01-01 00:00:00 sampai dengan 9999-12-31 23:59:59.

- Timestamp
  Pendeklarasian: Timestamp(size)
  Pada tipe data datetime ini, saat pendeklarasian harus disertakan dengan size. Contoh: timestamp(2) $\rightarrow$ YY atau timestamp(4) $\rightarrow$ YYMM.

- Time
  Pendeklarasian: Time
  Format time adalah HH:MM:SS (Hour:Minute:Second).

- Year
  Pendeklarasian: Year(digit)
  Untuk tipe data datetime year terdapat digit yang ditentukan dengan 2 atau 4. Jika dipilih 2 digit maka akan menghasilkan 00 (untuk 2000). Nilainya antara 1970-2069. Sedangkan untuk 4 digit nilainya antara 1901-2155.

Tipe Key (Kunci)

Penggunaan key pada umumnya digunakan pada relasi tabel, namun tidak menutup kemungkinan satu tabel juga membutuhkan key, yaitu primary key. Berikut ini merupakan tipe key yang digunakan dalam relasi antar tabel atau satu tabel :

- Super key
  Merupakan kumpulan atribute(field) atau satu atribut yang secara unik mengidentifikasi sebuah record pada suatu relasi.

- Candidate key
  Merupakan field unik yang umumnya dapat dijadikan sebagai relasi.

- Primary key
  Merupakan candidate key yang terpilih untuk mengidentifikasi record secara unik dalam suatu relasi. Jika terdapat auto_increment pada suatu field, maka field tersebut harus bersifat primary key.

- Alternative key
  Merupakan bagian dari candidate key yang tidak terpilih sebagai primary key.

- Foreign key
  Field yang menjadi penghubung suatu relasi, yaitu dari primary key.

Auto_Increment

Berfungsi untuk memberikan nilai dengan kelipatan satu dimulai dari 1 secara otomatis, sehingga user tidak perlu memasukkan nilai. Field yang dapat diberikan auto_increment harus yang bertipe data numerik. Dan umumnya digunakan dalam pembuatan nomor urut.

- **Membuat Tabel**

Yang perlu diperhatikan sebelum membuat sebuah tabel adalah melakukan pengaktifan database tertentu terlebih dahulu.

Bentuk umum penulisannya:
→ CREATE TABLE nama_table(field1 tipe(size),…);
Contoh:

```
Mysql>CREATE TABLE siswa(no int(5) auto_increment primary key,
    ->nis varchar(7), nama varchar(25));
```

- **Menghapus Tabel**

Bentuk umum penulisannya:
DROP TABLE nama_table;
Contoh:

```
Mysql>DROP TABLE kesiswaan;
```

- **Memodifikasi Tabel**

Macam modifikasi tabel, yaitu menambahkan field, mengganti size record suatu field, menghapus field, dan mengganti nama field.

- **Menambahkan Field**

  Bentuk umum penulisannya:

  ALTER TABLE nama_table ADD Column field tipe(size);

  Contoh:

  Mysql>ALTER TABLE tsiswa ADD column alamat varchar(30);

Catatan tambahan:

- Untuk menyisipkan atau menambahkan field pada awal field.

  Mysql>ALTER TABLE tsiswa ADD column phone varchar(30) first;
- Untuk menyisipkan atau menambahkan field setelah field tertentu.

  Mysql>ALTER TABLE tsiswa ADD column phone varchar(30)

  -> after alamat;

- **Mengganti Nama, Tipe Data, dan Size Field**

  Bentuk umum penulisannya:

  ALTER TABLE nama_table change old_field new_field tipe(size);

  Contoh:

  Mysql>ALTER TABLE tsiswa change phone

  -> telephone char(25);

  Phone varchar(30) → telephone char(25)

- **Mengganti Tipe Data Field**

  Bentuk umum penulisannya:

  ALTER TABLE nama_table modify field new_tipe(size);

  Contoh:

  Mysql>ALTER TABLE tsiswa modify telephone varchar(25);

  Telephone char(25) → telephone varchar(25)

- **Menghapus Nama Field**

  Bentuk umum penulisannya:

  ALTER TABLE nama_table DROP field;

  Contoh:

  Mysql>ALTER TABLE tsiswa DROP telephone;

- **Menampilkan Struktur Tabel**

Bentuk umum penulisannya:

DESC nama_table;

Contoh:

Mysql>DESC tsiswa;

- **Mengganti Nama Tabel**

Bentuk umum penulisannya:
ALTER TABLE old_table_name rename new_table_name;
Contoh :
Mysql>ALTER TABLE tsiswa rename kesiswaan;

- **Menghapus Tabel**

Bentuk umum penulisannya:
DROP TABLE nama_tabel;
Contoh:
Mysql>DROP TABLE kesiswaan;

- **Menampilkan Seluruh Tabel**

Bentuk umum penulisannya:
SHOW TABLES;
Contoh: Mysql>SHOW TABLES;

4. INDEX

Index berfungsi mempercepat proses pencarian data dalam suatu tabel. Adanya index pada suatu field tabel, menyebabkan proses pencarian otomatis akan dilakukan terlebih dahulu ke dalam index, apabila ditemukan baru akan diambilkan data yang sesungguhnya dari tabel. Apabila tidak ditemukan dalam index, sudah dapat dipastikan bahwa data tersebut memang tidak ada dalam tabel.

Index juga dapat dibuat untuk setiap kolom yang akan dijadikan kriteria tertentu untuk pencarian data, sehingga proses pencariannya akan lebih cepat.

Pada index terdapat perintah pembuatan dan penghapusan index, namun tidak terdapat perintah perubahan nama index.

- **Membuat Index**

Bentuk umum penulisannya:
CREATE INDEX nama_index ON nama nama_table(field);
Atau
ALTER TABLE nama_table ADD index nama_index(field);

Contoh:
Mysql>CREATE INDEX idxnomor ON kesiswaan(no);
Atau
Mysql>ALTER TABLE kesiswaan ADD index idxnis(nis);

- **Menghapus Index**

Penghapusan nama index tidak akan menghapus field table atau tabel, namun hanya memperlambat proses pencarian saja.

Bentuk umum penulisannya:
DROP INDEX nama_index ON nama_table;

Atau
ALTER TABLE nama_table DROP INDEX nama_index;
Contoh:
Mysql>DROP INDEX idxnomor ON kesiswaan;
Atau
Msyql>ALTER TABLE kesiswaan DROP INDEX idxnis;

5. MENGHAPUS PRIMARY KEY

Bentuk umum penulisannya :
ALTER TABLE nama_table DROP primary key, ADD primary key(no);
Contoh:
Mysql>ALTER TABLE kesiswaan DROP primary key;
->ADD primary key(no);

6. VIEW

Sebuah *view* adalah tabel yang dibangun dari satu atau beberapa tabel yang sudah ada. Secara fisik, *VIEW* tidak membuat penyimpanan data seperti tabel biasa, melainkan hanya menyimpan referensi/pointer ke *record* pada tabel-tabel yang berkaitan. *VIEW* biasa disebut juga *"virtual tabel"*. *View* dapat juga diciptakan dari beberapa tabel.

- **Membuat View**

  Bentuk umum penulisannya :

  CREATE VIEW view_name[(column1,column2,…)] AS SELECT statement FROM table_name [with check option];

  Contoh :

  Mysql>CREATE VIEW mhs AS SELECT * FROM mahasiswa;

Keterangan :
View_name → nama view yang akan dibuat.
Column → nama atribut untuk view.
Statement → atribut yang akan dipilih dari tabel basis data.
Table_name → nama tabel basis data.

- **Memperoleh Informasi pada View**

  View yang sudah dibuat dapat di akses seperti dalam mengakses tabel.

  Contoh :

  Mysql>SELECT * FROM mhs;

- **Mengubah View**

  Contoh :

  ALTER VIEW mhs AS
  SELECT npm, nama FROM mahasiswa;

- **Menghapus View**

  Bentuk umum penulisannya :

  DROP VIEW nama_view;

  Contoh :

  Mysql>DROP VIEW mhs;

3

DATA MANIPULATION LANGUAGE

1. INSERT

Insert merupakan perintah SQL yang berfungsi untuk menyisipkan nilai-nilai pada field-field tabel.

Bentuk umum penulisannya :
INSERT INTO nama_table VALUES (nil 1, nil 2, …);
Atau
INSERT INTO nama_table(field1, field2, …) VALUES (nil1, nil2, …);

Contoh:
Mysql>INSERT INTO kesiswaan(nis,nama,alamat)
->VALUES (“0123456”, “Sukamto”, “Jakarta”);
Atau
Mysql>INSERT INTO matakuliah
->VALUES (“M-01”,”Matematika”);

Mengapa dalam penulisan perintah insert terdapat perbedaan? Perbedaan itu disebabkan oleh beberapa sebab:

1. Rumusan bagian B: Adanya salah satu field yang memiliki sifat auto_increment, sehingga dalam melakukan penyisipan nilai secara otomatis langsung disisipi walau tanpa ada nilai yang disisipi secara langsung. Contoh rumus yang digunakan adalah:
   Mysql>INSERT INTO kesiswaan (nis,nama,alamat)
   ->VALUES (“0123456”, “Sukamto”, “Jakarta”)

   Perhatikan struktur tabel kesiswaan berikut ini :

Field	Type	Key	Default	Extra
no	Int(5)	PRI	NULL	auto_increment
nis	Varchar(7)	MUL	NULL	
nama	Varchar(25)		NULL	
alamat	Varchar(30)		NULL	

   Pada field “no” terdapat “auto_increment” pada bagian “Extra”. Karena sifat dari auto_increment, yaitu menyisipkan nilai secara otomatis, maka user tidak perlu menyisipi nilai. Oleh sebab itu hanya ada 3 field yang harus disisipi secara langsung, yaitu: nis, nama, dan alamat.

Yang dapat melakukan bentuk rumusan bagian "B", selain field yang memiliki auto_increment. Ada juga field yang harus dimasukkan dengan rumusan tertentu. Contoh, perhatikan tabel transaksi berikut ini :

Kode	Jenis	Harga	Jumlah
01	?	3500	5
03	?	25000	2

Untuk "Jenis" tidak dapat diinput secara manual melainkan harus menggunakan rumusan. Jika Kode 01 maka Elektronik, jika Kode 02 maka Komputer dan jika Kode 3 maka Game Station. Maka cara menyisipkan nilai untuk field Kode, Harga dan Jumlah adalah sebagai berikut :

```
Msyql>INSERT INTO transaksi(Kode, Harga, Jumlah)
    ->VALUES ("01",3500,5);
```

Untuk Jenis bila penyisipan nilainya secara interprestasi (langsung), maka hanya dapat dilakukan dengan perintah update. Jika penyisipannya dengan cara Embedded (penyisipan lewat program tertentu), dalam menyisipkan jenis dapat dilakukan dengan rumusan bagian B, atau dengan perintah update.

2. Rumusan bagian A: Apabila deretan field-field yang terdapat pada suatu tabel memiliki deretan nilai penyisipan yang sama. Perhatikan struktur tabel matakuliah, berikut ini:

Field	Type	Key	Default	Extra
Kode	Varchar(7)		NULL	
namamtk	Varchar(25)		NULL	

Karena field-field yang terdapat pada tabel matakuliah tidak mengandung sifat auto_increment atau rumusan tertentu, maka deretan field dan penyisipan nilai adalah sama. Contoh penyisipan adalah sebagai berikut :

```
Mysql>INSERT INTO matakuliah
    ->VALUES ("M-01", "Matematika");
```

2. SELECT

Select merupakan perintah untuk menampilkan record atau data. Dalam menampilkan record atau data dapat dilakukan dengan 2 cara, yaitu tanpa kondisi dan dengan kondisi.

Bentuk umum penulisan *tanpa kondisi*:

```
SELECT field1,… FROM nama_table;
Atau
SELECT * FROM nama_table;
```

Contoh:

```
Mysql>SELECT * FROM kesiswaan;
```

Artinya: Menampilkan seluruh record table kesiswaan.

Atau

```
Mysql>SELECT nis,nama,alamat FROM kesiswaan;
```

Artinya: Menampilkan data nis, nama, dan alamat saja.

Bentuk umum penulisan *dengan kondisi*:
SELECT field1, … FROM nama_table WHERE kondisi;
Atau
SELECT * FROM nama_table WHERE kondisi;

Contoh:
Mysql>SELECT * FROM transaksi WHERE kode=”01”;
Artinya: Menampilkan seluruh record yang berkode 01.
Atau
Mysql>SELECT kode,harga FROM transaksi WHERE kode=”03”;
Artinya: Menampilkan data kode dan harga yang berkode 03.

Yang perlu diperhatikan dalam penyeleksian data atau record dengan kondisi, terdapat operator-operator dan tanda-tanda khusus.

Operator Pembanding:

Operator	**Arti**
=	Sama Dengan
>	Lebih Besar
<	Lebih Kecil
>=	Lebih Besar Sama Dengan
<=	Lebih Kecil Sama Dengan
<>	Bukan atau Tidak Termasuk
Like	Sama halnya dengan Sama Dengan (=), namun pada Like pada diikuti oleh tanda-tanda khusus, yaitu : % (persen) dan _ (underscore). Penjelasan : % untuk menggantikan beberapa karakter._ untuk menggantikan satu karakter.

Operator Logika :

Operator	**Arti**
And	Dan atau Antara
Or	Atau

Untuk lebih mudah memahami perintah select dengan menggunakan kondisi, maka dibuatlah tabel kuliahan yang field dan recordnya seperti ini :

Nim	**Nama**	**Alamat**
21196353	Bintang	Bogor
41296525	Ningrum	Bogor
50096487	Pipit	Bekasi
10296832	Nurhayati	Jakarta
31296500	Budi	Depok
31292521	Lilik	Jakarta
41197522	Pratiwi	Jakarta

Bagaimana cara menampilkan seluruh data kuliahan bagi yang namanya mengandung huruf "i" ?

Penulisannya :
Mysql>SELECT * FROM kuliahan WHERE Nama Like '%i%';

Cara menampilkan data, namun hanya Nama dan Alamat saja bagi yang beralamat di Jakarta.

Penulisannya :
Mysql>SELECT Nama,Alamat FROM kuliahan WHERE
->Alamat='Jakarta';

Cara menampilkan seluruh data bagi yang beralamat di Jakarta dan Bogor.

Penulisannya:
Mysql>SELECT * FROM kuliahan WHERE
->Alamat='Jakarta' OR Alamat='Bogor';

Cara menampilkan seluruh data bagi yang namanya hanya terdiri dari 4 karakter.

Penulisannya:
Mysql>SELECT * FROM kuliahan WHERE Nama LIKE '___';
(4 x underscore)

Cara menampilkan seluruh data bagi yang namanya berawalan dari huruf 'P'.

Penulisannya:
Mysql>SELECT * FROM kuliahan WHERE Nama LIKE 'P%';

Cara menampilkan seluruh data pada tabel transaksi yang harganya antara Rp. 10.000 sampai dengan Rp. 35.000.

Penulisannya:
Mysql>SELECT * FROM transaksi WHERE
->Harga>=10000 AND Harga <=35000;

Cara menampilkan seluruh data mahasiswa secara urut dari Z ke A.

Penulisannya:
Mysql>SELECT * FROM kuliahan ORDER BY Nama DESC;

Cara menampilkan seluruh data mahasiswa secara urut dari A ke Z.

Penulisannya:
Mysql>SELECT * FROM kuliahan ORDER BY Nama ASC;

Cara menampilkan seluruh data mahasiswa yang tidak mengandung huruf 'U'.

Penulisannya:
Mysql>SELECT * FROM kuliahan WHERE Nama NOT LIKE '%u%';

Cara menampilkan seluruh data transaksi yang harganya bukan Rp. 20.000.

Penulisannya:
Mysql>SELECT * FROM transaksi WHERE harga<>20000;

---

3. UPDATE

Perintah update digunakan untuk melakukan penyimpanan hasil editing suatu data.

Sama halnya dengan perintah select, dalam proses update dapat dilakukan *tanpa kondisi* atau *dengan kondisi*.

Bentuk umum penulisan *tanpa kondisi* :
UPDATE nama_table SET field=nilai;

Contoh :
Mysql>UPDATE kuliahan SET Nama="Agus";

Penjelasan:
Seluruh nama mahasiswa akan berubah menjadi "Agus" semua.

Untuk itu diperlukan kondisi agr yang berubah hanya kondisi-kondisi tertentu.

Bentuk umum penulisan *dengan kondisi* :
UPDATE nama_table SET field=nilai WHERE kondisi;

Contoh :
Mysql>UPDATE kuliahan SET Nama="Budi Setiawan"
-> WHERE Nama="Budi";

Penjelasan :
Nama mahasiswa yang bernama “Budi” akan berubah menjadi “Budi Setiawan”.

Perhatikan tabel transaksi berikut ini :

Kode	Jenis	Harga	Jumlah
01	?	3500	5
03	?	25000	2
02	?	10000	2
03	?	27500	3

Jika Kode 01 maka Elektronik.
Jika Kode 02 maka Komputer.
Jika Kode 03 maka Game Station.
Cara menginput nilai untuk field “Jenis” menjadi Elektronik bagi yang berkode 01.

Penulisannya:
```
Mysql>UPDATE transaksi SET Jenis=”Elektronik” WHERE
    ->Kode=”01”;
```

Cara menginput nilai untuk field “Jenis” menjadi Komputer bagi yang berkode 02.

Penulisannya:
```
Mysql>UPDATE transaksi SET Jenis=”Komputer” WHERE
    ->Kode=”02”;
```

Cara menginput nilai untuk field “Jenis” menjadi Game Station bagi yang berkode 03.

Penulisannya:
```
Mysql>UPDATE transaksi SET Jenis=”Game Station” WHERE
    ->Kode=”03”;
```

4. DELETE

Delete memiliki fungsi untuk menghapus suatu data pada suatu tabel. Delete pun memiliki cara kerja yang terdiri dari 2 bagian, yaitu tanpa kondisi dan dengan kondisi.

Bentuk umum penulisan *tanpa kondisi* :
```
DELETE FROM nama_table;
```

Contoh:
```
Mysql>DELETE  FROM transaksi;
```

Penjelasan:
Seluruh data pada tabel kuliahan akan terhapus semua. Untuk itu harus berhati-hati dalam menggunakan perintah delete.

Bentuk umum penulisan *dengan kondisi* :
DELETE FROM nama_table WHERE kondisi;

Contoh :
Mysql>DELETE FROM transaksi WHERE Kode=”01”;

Penjelasan:
Seluruh data yang berkode 01 akan dihapus dari tabel transaksi.

Cara menghapus harga pada tabel transaksi yang memiliki range antara 10000 s/d 25000.

Penulisannya:
```
Mysql>DELETE FROM transaksi WHERE Harga>= 1000 AND
    ->Harga<=25000;
```

Latihan menerapkan perintah DDL dan DML pada kasus seperti berikut :

1. Buatlah database dengan nama “Mahasiswa”?
2. Buatlah tabel dengan nama “Penilaian” dengan struktur tabel seperti berikut :

Field	Type	Key	Default	Extra
Nim	Int(8)	PRI	NULL	
KdMk	Varchar(5)		NULL	
NamaMK	Varchar		NULL	
Mid	Float(5)		NULL	
Final	Float(5)		NULL	
NilRata	Float(5)		NULL	

3. Sisipkan nilai-nilainya seperti tabel berikut :

Nim	KdMK	NamaMK	Mid	Final	NilRata
10296832	KK021	?	75	82	?
10297732	KD132	?	77,5	73,5	?
20216832	KK021	?	83,5	81,7	?
30216832	KU122	?	65,4	77	?
10297732	KK021	?	79,5	82,5	?
20216832	KD132	?	80,5	83	?
30216832	KK021	?	67	78	?

4. Tampilkan seluruh record mahasiswa yang berkode matakuliah atauu KdMK=KK021!

5. Tampilkan record field Nim,KdMK dan Mid saja bagi yang nilai mid-nya antara 75 dan 90!

6. Sisipkan nilai untuk field nama matakuliah (NamaMK), dengan ketentuan sebagai berikut :

KdMK	NamaMK
KK021	Sistem Basis Data
KD132	SIM
KU122	Pancasila

7. Sisipkan nilai untuk field nilai rata-rata (NilRata), dengan ketentuan sebagai berikut :
   Mid + Final / 2

8. Hapuslah record yang memiliki Nim = 10296832!

* * *

4
FUNGSI AGREGAT

1. FUNGSI AGREGAT

- **SUM (ekspresi)**

Berfungsi untuk mencari total nilai keseluruhan yang terdapat pada suatu field yang bersifat numerik. Untuk lebih mudahnya dalam membuat suatu ilustrasi, perhatikan tabel nilai di bawah ini :

Nim	KdMK	Mid	Final
10296832	KK021	75	82
10297732	KD132	75	73,5
20216832	KK021	83,5	81,7
30216832	KU122	65,4	77
10297732	KK021	79,5	82,5

Contoh:

```
Mysql>SELECT SUM(Mid)Total_Mid FROM nilai;
```

Hasilnya:

Total_Mid
378,4

```
Mysql>SELECT SUM(DISTINCT Mid)Total_Mid FROM nilai;
```

Hasilnya :

Total_Mid
303,4

- **COUNT (x)**

Memiliki fungsi untuk mencari berapa jumlah total baris yang terdapat pada suatu tabel.

Contoh:

```
Mysql>SELECT COUNT(*)Jumlah_Record FROM nilai;
```

Hasilnya:

Jumlah_Record
5

- **AVG (ekspresi)**

  Berfungsi untuk mencari nilai rata-rata pada suatu pada field bersifat numerik.

Contoh :

Mysql>SELECT AVG(Final) Rata-rata_Final FROM nilai;

Hasilnya :

Rata-rata_Final
79.34

- **MAX (ekspresi)**

  Berfungsi untuk mencari nilai tertinggi dari suatu field yang bersifat numerik.

Contoh:

Mysql>SELECT MAX(Final)Nilai_Final_Tertinggi FROM nilai;

Hasilnya:

Nilai_Final_Tertinggi
83

- **MIN (ekspresi)**

  Berfungsi untuk mencari nilai terendah dari suatu field yang bersifat numerik.

Contoh:

Mysql>SELECT MIN(Mid)Nilai_Mid_Terendah FROM nilai;

Hasilnya:

Nilai_Mid_Terendah
65,4

* * *

5
FUNGSI TANGGAL

- **ADDATE (x, Interval nilai_interval)**

  Berfungsi untuk mendapatkan tanggal baru karena proses dari penjumlahan nilai interval :

Daftar Tipe Nilai Interval :

Tipe Nilai	Keterangan
Second	Satuan detik
Minute	Satuan menit
Hour	Satuan jam
Day	Satuan hari
Month	Satuan bulan
Year	Satuan tahun

Minute_Second	"Menit:Detik"
Hour_Minute	"Jam:Menit"
Day_Hour	"Hari:Jam"
Year_Month	"Tahun:Bulan"
Hour_Second	"Jam:Detik"

Contoh:

```
Mysql>SELECT ADDDATE("2005-05-31",Interval 10 day)
    ->Hasilnya_Setelah_Ditambah_10hari;
```

Hasilnya:

Hasilnya_Setelah_Ditambah_10hari
2005-06-10

- **CURDATE ( )**

  Menghasilkan tanggal saat ini, namun tidak seperti halnya fungsi now ( ) yang disertai dengan waktu.

Contoh :

```
Mysql>SELECT curtime( )Tanggal_Hari_Ini;
```

Hasilnya:

Tanggal_Hari_Ini
2006-05-07

- **CURTIME ( )**

Menghasilkan waktu terkini.

Contoh :

Mysql>SELECT curtime( )Waktu_Saat_Ini;

Hasilnya:

Waktu_Saat_Ini
10:25:05

- **CURRENT_TIMESTAMP( )**

Menampilkan tanggal saat ini berikut dengan jam, menit, dan detik.

Contoh:

Mysql>SELECT current_timestamp( )
->Waktu_Saat_Ini_Berikut_Waktu_Terkini;

Hasilnya:

Waktu_Saat_Ini_Berikut_Waktu_Terkini
2006-05-07 17:50:20

- **DAYNAME (penanggalan)**

Berfungsi untuk menampilkan nama hari sesuai dengan tanggal saat itu.

Contoh :

Mysql>SELECT dayname(“2006-05-07”) Tanggal_Tersebut_Hari;

Hasilnya:

Tanggal_Tersebut_Hari
Sunday

- **DAYOFMONTH (penanggalan)**

Berfungsi untuk menampilkan tanggal pada suatu format penanggalan dari sebulan.

Contoh:

Mysql>SELECT dayofmonth(“2006-05-07”)Tanggalnya_adalah;

Hasilnya:

Tanggalnya_adalah
7

- **DAYOFWEEK (penanggalan)**

Berfungsi untuk menampilkan hari dari seminggu dengan menggunakan kode angka.

Berikut ini daftar nama hari dan kode angkanya:

Nama Hari	Kode Angka
Sunday	1
Monday	2
Tuesday	3
Wednesday	4
Thursday	5
Friday	6
Saturday	7

Contoh:

Mysql>SELECT dayofweek(“2006-05-07”)Kode_Tanggalnya:

Hasilnya:

Kode_Tanggalnya
1

- **DAYOFYEAR (penanggalan)**

Berfungsi untuk menampilkan hari ke berapa dalam setahun.

Contoh:

Mysql>SELECT dayofyear(“2006-05-07”)Saat_Ini_Hari_Ke;

Hasilnya:

Kode_Tanggalnya
147

- **EXTRACT(nilai FROM penanggalan/waktu)**

Fungsi extract ini dapat mengambil bagian dari tanggal, bulan, atau tahun saja dari suatu penanggalan. Juga dapat mengambil bagian dari jam, menit, atau detik dari suatu pengaturan waktu.

Contoh extract dari penanggalan:

Mysql>SELECT Extract(Day FROM”2006-05-07”)Nilai_Extractnya;

Hasilnya:

Nilai_Extractnya
7

Contoh extract dari pengaturan waktu:

Hasilnya:

Nilai_Extractnya
15

- **FROM_DAYS(tanggal dalam nilai)**

Fungsi ini berguna dalam mengubah nilai menjadi bentuk penanggalan. Namun nilai tersebut harus hanya terdiri dari 6 angka. Contoh: 814516, 901234.

Contoh:

Mysql>SELECT FROM_DAYS(901510)Penanggalannya_adalah;

Hasilnya:

Hasilnya:

Penanggalannya_adalah
2468-04-01

- **HOUR (pengaturan waktu)**

Berfungsi untuk mengambil nilai jam dari suatu pengaturan waktu.

Contoh:

Mysql>SELECT HOUR("12:35:01")Jamnya_adalah;

Hasilnya:

Jamnya_adalah
12

- **MINUTE (pengaturan waktu)**

Berfungsi untuk mengambil nilai menit dari suatu pengaturan waktu

Contoh:

Mysql>SELECT minute ("12:35:01")Jamnya_adalah;

Hasilnya:

Hasilnya:

Menitnya adalah
35

- **SECOND (pengaturan waktu)**

Berfungsi untuk mengambil nilai detik dari suatu pengaturan waktu.

Contoh:

Mysql>SELECT second("12:35:01")Jamnya_adalah;

Hasilnya:

Hasilnya:

Detiknya_adalah
1

- **MONTH (penanggalan)**

  Berfungsi untuk mengambil nilai bulan dari suatu pengaturan tanggal.

Contoh:
Mysql>SELECT month"(1978-11-21")Bulan_ke;

Hasilnya:

Bulan_ke
11

- **MONTHNAME (penanggalan)**

  Berfungsi untuk mengambil nilai bulan dari suatu pengaturan tanggal namun diubah menjadi nama bulan.

Contoh:
Mysql>SELECT monthname("1978-11-21")Ini_Bulan;

Hasilnya:

Ini_Bulan
November

- **NOW ( )**

  Menampilkan penanggalan saat ini berikut dengan waktu saaat ini. Fungsi now ( ) sama dengan fungsi current_timestamp( ).

Contoh:
Mysql>SELECT now( )Sekarang;

Hasilnya:

Sekarang
2006-07-05 19:35:10

- **YEAR(penanggalan)**

  Menampilkan tahun dalam suatu penanggalan.

Contoh:
Mysql>SELECT year("2006-05-07")Tahun;

Hasilnya:

Tahun
2006

- **TO_DAYS (penanggalan)**

Menampilkan nilai peninggalan dari suatu penanggalan.

Contoh:
Mysql>SELECT to_days(“2006-05-07”)Nilainya_adalah;

Hasilnya:

Nilainya_adalah
732803

- **DATE_FORMAT(penanggalan/waktu, simbol format)**

Berfungsi untuk merubah nilai penanggalan atau pengaturan waktu dengan format yang diinginkan.

Berikut ini merupakan daftar simbol format:

Simbol Format	Keterangan
%M	Menampilkan nama bulan bukan dalam bentuk singkatan
%m	Menampilkan bulan dengan angka.
%b	Menampilkan nama bulan dalam bentuk singkatan.
%W	Menampilkan nama hari bukan dalam bentuk singkatan.
%D	Menampilkan nilai hari ke dalam sebulan.
%Y	Menampilkan tahun dengan format 4 digit.
%y	Menampilkan tahun dengan format 2 digit.
%j	Menampilkan nomor hari dalam setahun.
%a	Menampilkan nama hari dalam bentuk singkatan.
%d	Menampilkan nomor hari dalam sebulan.
%r	Menampilkan jam dalam format 12 jam.
%T	Menampilkan jam dalam format 24 jam.
%H	Menampilkan jam dalam format 24 jam : 00-23.
%h	Menampilkan jam dalam format 12 jam : 00-12.
%S	Menampilkan detik

Contoh :
Mysql>SELECT date_format(“2006-08-17”, ‘%W, %M, %D, %Y)
->Formatnya;

Hasilnya:

Formatnya
Thursday, August 17th 2006

* * *

6
FUNGSI STRING

- **Ascii (x)**

Berfungsi untuk mencari nilai ascii dari suatu string.
Contoh:
Mysql>SELECT ascii("%")Nilai_Asciinya;

Hasilnya:

Nilai_Asciinya
37

- **Char (x1, x2, …)**

Fungsi ini merupakan kebalikan dari fungsi ascii(x), jika pada fungsi ascii (x) mengembalikan string menjadi nilai ascii. Fungsi char(x1,…) adalah mengubah fungsi ascii menjadi karakter.
Contoh:
Mysql>SELECT char(37, 65, 43, 37, 66)Karakternya_adalah;

Hasilnya:

Karakternya_adalah
%A+%B

- **Char_Length(string) atau Length(string)**

Char_length(string) memiliki fungsi yng sama dengan length(string), yaitu untuk menghitung jumlah karakter pada sebuah string.
Contoh:
Mysql>SELECT char_length("Aku Ingin Pulang")
->Jumlah_Karakternya;

Hasilnya:

Jumlah_Karakternya
732803

- **Encode (string,string_enkripsi)**

Berfungsi untuk merubah nilai string menjadi kode-kode tertentu. Dalam merubah string menjadi kode harus terdapat string dan string_enkripsi. Jika hanya terdapat string saja maka proses tidak dapat dilakukan.

Contoh:

Mysql>SELECT encode(“AuTo”,”CAD”)Kodenya_adalah;

- **Left(string,nilai pengambilan dari kiri)**

Berfungsi untuk mengambil karakter terkiri dari suatu string dengan jumlah pengambilan karakter dari kiri.

Contoh:

Mysql>SELECT left(“Indonesia,3”)3 Digit_Dari_Kiri;

Hasilnya:

3Digit_Dari_Kiri
Ind

- **Mid(string,posisi,nilai pengambilan dari posisi)**

Berfungsi untuk mengambil karakter dari suatu string, sebelumnya harus menentukan terlebih dahulu posisi pengambilan. Setelah itu baru menentukan jumlah karakter yang akan diambil dari posisi yang telah ditentukan.

Contoh:

Mysql>SELECT mid(“Indonesia,4,3”)3Digit_Dari_Huruf_O;

Hasilnya:

3Digit_Dari_Huruf_O
One

- **Right (string,nilai pengambilan dari kanan)**

Berfungsi untuk mengambil karakter terkanan dari suatu string dengan jumlah pengambilan karakter dari kanan.

Contoh:

Mysql>SELECT right(“Indonesia,3”)3Digit_Dari_Kanan;

Hasilnya:

3Digit_Dari_Kanan
Sia

- **Lcase(string) Atau Lower(string)**

Berfungsi untuk merubah tipe karakter dari suatu string dari huruf kapital menjadi huruf kecil.

Contoh:

```
Mysql>SELECT lcase("KUMENANTI DALAM KESUNYIAN")
    ->Perubahannya_adalah;
```

Hasilnya:

Perubahannya_adalah
Kumenanti dalam kesunyian

- **Ucase(string) Atau Upper(string)**

Fungsi ucase(string) atau upper(string) adalah merupakan kebalikan dari fungsi lcase(string), yaitu untuk merubah huruf kecil menjadi huruf kapital atau besar.

Contoh:

```
Mysql>SELECT upper("kumenanti dalam kesunyian")
    ->Perubahannya_adalah;
```

Hasilnya:

Perubahannya_adalah
KUMENANTI DALAM KESUNYIAN

- **Ltrim(string)**

Berfungsi untuk menghilangkan atau menghapus spasi dari bagian kiri suatu string.

Contoh:

```
Mysql>SELECT ltrim ("          Kugapai harapan")
    ->Setelah_di_ltrim;
```

Hasilnya:

Setelah_di_ltrim
Kugapai harapan

- **Rtrim(string)**

Berfungsi untuk menghilangkan atau menghapus spasi dari bagian kanan suatu string.

Contoh:

```
Mysql>SELECT rtrim("Kugapai harapan      ")
    ->Setelah_di_rtrim;
```

Hasilnya:

Setelah_di_rtrim
Kugapai harapan

- **Trim(string)**

Berfungsi untuk menghilangkan atau menghapus spasi dari bagian kiri dan kanan suatu string.

Contoh:

```
Mysql>SELECT trim("          Kugapai harapan          ")
    ->Setelah_di_trim;
```

Hasilnya:

Setelah_di_trim
Kugapai harapan

- **Password(string)**

Berfungsi untuk merubah suatu string menjadi kode sandi yang telah dienkripsi.

Contoh:

```
Mysql>SELECT password("rahasia")Di_password_menjadi;
```

Hasilnya:

Di_password_menjadi
747fc82325405198

* * *

7

DATABASE RELATION

1. PENDAHULUAN

Yang dimaksud dengan database relational atau normalisasi adalah suatu proses untuk merelasikan field dari tabel yang satu dengan tabel lainnya, di mana dalam salah satu tabel terdapat field yang bersifat primary key atau foreign key. Normalisasi ditemukan pada tahun 1970 oleh E.F.. Codd (seorang peneliti IBM) sebagai ketidakpuasannya dengan metode penyimpanan data saat itu. Lewat bukunya yang berjudul "*A relational model of data for large shared databanks*" maka tercetuslah database relasional.

Pada dasarnya model relasional digunakan untuk mengatasi kesulitan dalam pengelolaan dan pengaksesan data. Alasan yang melandasi mengapa dibutuhkan relasi tabel, yaitu karena terdapatnya anomali-anomali (error atau inkonsistensi data) yang harus dihindari, agar keutuhan data dan kepastian data terjamin. Anomali-anomali itu meliputi anomali update, insert, dan delete.

Untuk memahami konsep relasi, perhatikan daftar mahasiswa dan matakuliah berikut ini:

Nomor	Nim	Kode-mtk	Sks
1	99130012	101	2
2	99145021	105	4
3	99276521	103	4
4	99130012	102	2
5	98254123	103	4
6	99145021	104	2
7	99671204	106	4
8	99130012	105	4
9	99276521	102	2
10	99145021	103	4
11	98254123	101	2

1. Anomali Insert

Kesalahan yang terjadi di saat proses penyisipan record baru.

Contoh:

Jika kampus akan mengadakan matakuliah baru dengan kode matakuliah 107, maka proses penyisipan untuk kode matakuliah 107 tidak dapat dilakukan sampai ada mahasiswa yang mengambil matakuliah tersebut.

2. Anomali Delete

Kesalahan yang terjadi di saat proses penghapusan record atau tuple.

Contoh:

Mahasiswa yang memiliki Nim 99671204 memutuskan untuk membatalkan mengambil matakuliah 106, maka dengan demikian jika didelete record tersebut, akan berakibat hilangnya informasi tentang kode matakuliah 106.

3. Anomali Update

Anomali atau kesalahan yang terjadi pada saat proses suatu record.

Contoh:

Jika Anda perhatikan pada tabel kuliahan, terdapat banyak kode matakuliah yang diambil oleh mahasiswa dengan NIM yang berbeda. Misalnya akan dilakukan perubahan SKS untuk kode matakuliah 103, maka akan dilakukan proses update beberapa kali sesuai dengan banyaknya jumlah yang mengambil kode matakuliah tersebut.

Dari anomali-anomali yang terjadi di atas maka daftar mahasiswa dan matakuliah bukan termasuk Well-Structure relation (sebuah relasi dengan jumlah kerangkapan datanya sedikit).

Dengan demikian daftar mahasiswa & matakuliah harus dilakukan proses normalisasi, yaitu dengan membagi 2 tabel yang terpisah. Tabel-tabel tersebut adalah sebagai berikut ini :

Tabel matakuliah:

Kode-mtk	Sks
101	2
102	4
103	4
104	2
105	4
106	2

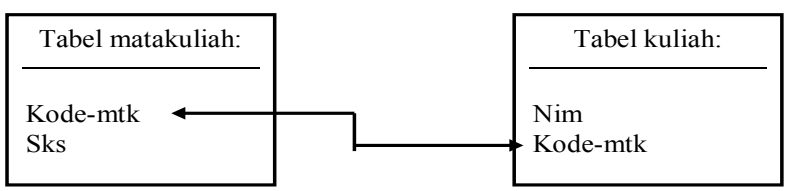
Tabel kuliah:

Nim	Kode-mtk
99130012	101
99145021	105
99276521	103
99130012	102
98254123	103
99145021	104

Dari kedua tabel tersebut akan dibentuk relasi yang saling terkait antara satu dengan yang lainnya. Untuk melakukan prosoes relasi dibutuhkan field yang memiliki primary key atau foreign key.

Sifat field primary key, yaitu tidak boleh terdapat data sama dalam field tersebut. Contoh: pada tabel kuliah field ‘Kode-mtk’ terdapat banyak data yang sama, seperti kode matakuliah 103. Berarti field ‘Kode-mtk’ pada tabel kuliah tidak dapat dijadikan primary key. Jika Anda amati field ‘Kode-mtk’ pada tabel matakuliah tidak terdapat data yang sama, berarti field ‘Kode-mtk’ pada tabel matakuliah dapat dijadikan primary key.

Kesimpulannya field ‘Kode-mtk’ pada tabel matakuliah sebagai primary key, sedangkan field ‘Kode-mtk’ pada tabel kuliah sebagai foreign key. Untuk lebih jelasnya perhatikan skema di sini :



Dari relasi kedua tabel ini, yaitu tabel matakuliah dan kuliah dapat terbentuk daftar mahasiswa & matakuliah seperti yang nampak seperti berikut:

Daftar mahasiswa & matakuliah:

Nomor	Nim	Kode-mtk	Sks
1	99130012	101	2
2	99145021	105	4
-	-	-	-

2. PEMBENTUKAN NORMALISASI

Perubahan daftar mahasiswa & matakuliah menjadi 2 tabel, yaitu tabel matakuliah dan kuliah adalah merupakan bentuk normalisasi. Dengan demikian terjadi suatu kondisi di mana proses penginputan record baru untuk kode matakuliah dilakukan pada tabel matakuliah, maka secara otomatis akan terlihat perubahan SKS atau kode matakuliah, maka secara otomatis akan terlihat pula dalam daftar mahasiswa & matakuliah.

Berikut ini akan dijelaskan seputar pembentukan normalisasi dari bentuk tidak normal (Unnormalized Form).

a) Bentuk tidak normal (Unnormalized Form)

Pada bagian ini setiap record yang diinput tidak mengalami tahap penyeleksian, sehingga akan terjadi kerangkapan data atau duplikasi data. Jadi data yang diinput terlihat apa adanya sesuai dengan yang diinput.

Contoh :

Kobuk	Judul	Pengarang	RK1	RK2	RK3
101-K	Belajar Sendiri VB. 6	M. Bakri	001	003	001
103-C	Kisah Cinta Sang Penyair	Minolsta	003	001	
110-S	Belajar Cepat Berhitung	Cokro S.	002	002	001
106-A	Bertauhid Yang Benar	Ust. Soleh	003		003

b) Bentuk 1NF (First Normal Form)

Pada bentuk form jenis pertama ini atau yang disebut 1NF harus bersifat atomik. Atom yaitu zat terkecil yang masih dapat dipecah lagi.

Ciri-ciri 1NF:

- Tiap field harus bernilai “atomik, yaitu setiap recordnya hanya terdiri dari satu nilai.
- Setiap field harus hanya memiliki satu pengertian dan memiliki nama yang unik.
- Tidak terdapat record yang sama atau bernilai ganda.

Hasil perubahan dari Unnormalized Form menjadi 1NF adalah sebagai berikut :

Kobuk	Judul	Pengarang	Kode_RK
101-K	Belajar Sendiri VB. 6	M. Bakri	001
101-K	Belajar Sendiri VB. 6	M. Bakri	003
103-C	Kisah Cinta Sang Penyair	Minolsta	001
103-C	Kisah Cinta Sang Penyair	Minolsta	003
110-S	Belajar Cepat Berhitung	Cokro S.	001
110-S	Belajar Cepat Berhitung	Cokro S.	002
106-A	Cara Cepat Belajar Database	Lina Marlina	003

c) Bentuk 2NF (Second Normal Form)

Apabila bentuk normal pertama atau 1NF terpenuhi, maka dapat menuju ke bentuk normal kedua atau 2NF. Pada tahap ini tabel yang terdapat pada bentuk normal pertama atau 1NF akan dibagi menjadi 2 bagian, yaitu :

Tabel Buku

Kobuk	Judul	Pengarang
101-K	Belajar Sendiri VB. 6	M. Bakri
103-C	Kisah Cinta Sang Penyair	Minolsta
110-S	Belajar Cepat Berhitung	Cokro S.
106-A	Cara Cepat Belajar Database	Lina Marlina

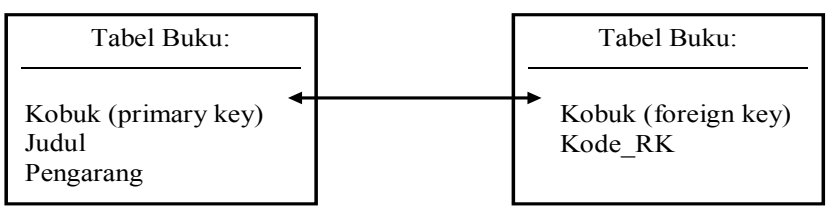
Tabel Rak Buku

Kobuk	Kode_RK
101-K	001
101-K	003
103-C	001
103-C	003
110-S	001
110-S	002
106-A	003

d) Bentuk 3NF (Third Normal Form)

Pada tahap bentuk normal ketiga, terjadi suatu relasi yang terdapat dependency, yaitu field yang bergantung dengan field yang lain pada tabel yang berbeda. Seperti pada contoh bentuk normal kedua di mana field Kobuk (foreign key) pada tabel rak buku bergantung kepada field Kobuk (primary key) pada tabel Buku.

Dengan demikian contoh bentuk normal kedua telah memenuhi syarat untuk dijadikan bentuk normal ketiga.



3. JENIS RELASI TABEL

Terdapat 3 jenis relasi tabel, yaitu One to One, One to Many, dan Many to Many.

- **Relasi One to One**

Pada jenis relasi one to one, setiap record pada tabel induk hanya memiliki satu relasi dengan tabel anak.

Contoh :

Tabel Wali Kelas	
TD_WK	**Nama Wali Kelas**
01	Suhendar
02	Samsusi
03	Zainab

Tabel Kelas		
Ko Kelas	**Kelas**	**ID_WK**
A	1A	01
B	1B	02
C	1C	03

- **Relasi One to Many**

Relasi one to many, yaitu suatu jenis relasi tabel yang memberikan hak untuk berrelasi dengan lebih dari satu baris.

Contoh :

Tabel Hewan	
KdHwn	**Jenis_Hewan**
01	Kambing
02	Sapi
03	Kerbau

ID_Pemilik	Nama_Pemilik	Jumlah	Jenis_Hewan
A001	H. Bajuri	2	Kambing
A003	Zainal Sambri	1	Kambing
A002	Andi Solahudin	1	Kerbau
A003	Zainal Sambri	1	Sapi
A002	Andi Solahudin	2	Kambing
A004	Syamsudin	3	Kambing
A005	Diah Armintai	2	Kambing
A001	H. Bajuri	1	Sapi
A001	H. Bajuri	1	Kerbau

Contoh relasi di atas menunjukkan bahwa setiap pengqurban dapat berqurban dengan lebih dari satu jenis hewan yang berbeda. Contoh seperti H. Bajuri yang berqurban kambing, sapi, dan kerbau.

- **Relasi Many To Many**

Pembentukan relasi many to many akan terjadi apabila beberapa baris pada sebuah tabel berelasi ke beberapa jenis pada tabel yang lain. Untuk merepresentasi relasi ini tidak cukup hanya menggunakan dua table, melainkan harus memerlukan tabel perantara lain.

Contoh :

Tabel Penerbit		
ID_Penerbit	**Nama Penerbit**	**Kota**
01	CV. Pustaka Jaya	Jakarta
02	Setia Kawan	Jakarta
03	Alexander	Bandung
04	Surya Kencana	Padang

Tabel_Pengarang		
ID_Pengarang	**Nama_Pengarang**	**Alamat**
A01	Dede Sunarya	Jakarta
A02	Drs. Amelia Anggoro	Jakarta
A03	Mohamad Suharto, S.H.	Subang
A04	Ust. Yudo Kuncoro	Bandung
A05	M. Zaidan Alwi	Medan
A06	Apriastuti Nur Komalasari	Jakarta
A07	Retno Yuniar	Subang

Tabel_Toko		
Nama_Pengarang	**ID_Toko**	**Nama Penerbit**
Ust. Yudo Kuncoro	1	CV. Pustaka Jaya
Mohamad Suharto, S.H.	2	Setia Kawan
Dede Sunarya	3	CV. Pustaka Jaya
Ust. Yudo Kuncoro	4	Surya Kencana
Apriastuti Nur Komalasari	5	Setia Kawan
Mohamad Suharto, S.H.	6	CV. Pustaka Jaya
Retno Yuniar	7	Surya Kencana
Drs. Amelia Anggoro	8	Alexander

Pada Tabel Toko terlihat bahwa penerbit dapat berhubungan dengan lebih dari satu pengarang, begitu juga halnya dengan pengarang yang dapat berelasi dengan banyak penerbit.

4. MASALAH YANG TIMBUL AKIBAT NORMALISASI

Walaupun normalisasi telah membantu dalam menghilangkan anomali-anomali, baik itu anomali insert, update, atau delete. Ada hal yang ditimbulkan setelah proses normalisasi itu berhasil, yaitu permasalahan tampilan tabel dan permasalahan integritas referensial.

Untuk memudahkan memahami kedua permasalahan tersebut, perhatikan tabel Buku dan Rak Buku.

- **Permasalahan Tampilan Tabel**

Jika pada sebelumnya hanya terdapat satu table, maka untuk menampilkan semua record, cukup hanya dengan menggunakan perintah sql: SELECT * FROM nama_tabel. Sedangkan untuk menampilkan beberapa record saja dapat menggunakan sintaks LIMIT dan OFFSET.

Contoh :

- Untuk menampilkan 5 record pertama dari tabel buku.
  Mysql>SELECT * FROM buku LIMIT 5;
- Untuk menampilkan 3 baris pertama yang diurutkan berdasarkan pengarang.
  Mysql>SELECT * FROM buku ORDER BY pengarang LIMIT 3;
- Menampilkan 2 baris setelah melewati 2 baris pertama yang diurutkan berdasarkan kode buku.

Mysql>SELECT * FROM buku ORDER BY kobuk LIMIT 2 OFFSET 2;

Setelah melalui proses normalisasi, maka telah terbentuk dua tabel, yaitu tabel Buku dan tabel Rak Buku. **Pernyataan SQL untuk menampilkan record tidak sama dengan sebelum proses normalisasi**. Perintah SQL-nya yaitu :
SELECT nama_tabel.nama_field,... FROM nama_tabel.nama_field

Contoh :

Menampilkan judul buku pengarang, dan rak buku yang kode rak bukunya 001.

```
Mysql>SELECT Buku.Judul,Buku.Pengarang, Rakbuku.Kode_RK
    ->FROM Buku, Rak buku WHERE Rakbuku.Kode_RK='001';
```

- **Permasalahan Integritas Referensial**

Berupa Maintenance Consistency of Reference antara 2 buah tabel relasi yang saling terkait.
Contoh :

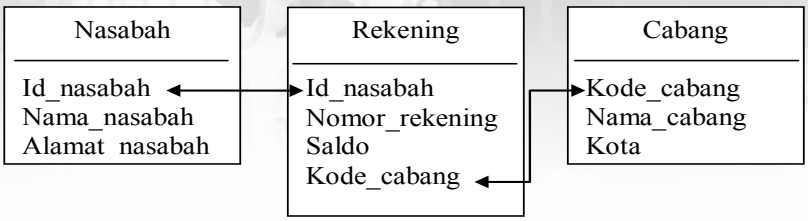
- Sebelum terjadinya proses normalisasi, setiap penambahan record dapat dilakukan pada tabel Relasi. Setelah normalisaasi, untuk menambahkan rak buku harus melalui proses pengecekan terlebih dahulu, apakah buku yang akan ditampilkan pada suatu rak apakah sudah ada dalam tabel Buku atau belum. Jika belum terdapat kode buku tersebut pada tabel Buku, maka proses penginputan record pada tabel Rak Buku tidak dapat dilakukan.
- Jika ingin melakukan penghapusan data pada tabel Buku harus berhati-hati, karena akan berpengaruh pada tabel Rak Buku. Jika Anda menghapus pada bagian tabel Rak Buku saja, maka tidak akan berpengaruh pada tabel Buku. Karena sifat field Kobuk pada tabel Buku bersifat primary key, sedangkan field Kobuk pada tabel Rak Buku sebagai penghubung saja.

* * *

8

RELASI ANTAR 2 TABEL (WHERE)

Diberikan sampel 3 buah tabel yang saling berelasi satu sama lain beserta sampel datanya.



Nasabah

Rekening

Id_nasabah	Nomor_rekening	Saldo	Kode_cabang
AA001	001000123	20.000.000	A123
AB002	002000123	5.000.000	B123
CC003	003000123	75.000.000	C123
CD004	004000123	20.000.000	D123
DD005	005000123	1.500.000	E123

Id_nasabah	Nama_nasabah	Alamat_nasabah
AA001	Rudi	Jl. Sepat
AB002	Yuni	Jl. Baung
CC003	Hani	Jl. Gurame
CD004	Kiki	Jl. Nila
DD005	Lala	Jl. Mujair

Cabang

Kode_cabang	Nama_cabang	kota
A123	JKT01	Jakarta Pusat
B123	JKT02	Jakarta Selatan
C123	BDO01	Bandung
D123	SBY01	Surabaya
E123	DPK01	Depok

Buat database dengan nama bank, kemudian buat 3 tabel di atas..!

Isikan data sesuai tabel di atas..!

1. Tampilkan id_nasabah, nama_nasabah, dan saldo..!
   - Tentukan tabel mana saja yang dibutuhkan.
     Untuk menampilkan id_nasabah dan nama_nasabah kita membutuhkan tabel nasabah. Sedangkan untuk menampilkan saldo kita membutuhkan tabel rekening. Jadi kita akan menggunakan 2 tabel tersebut.
   - Perhatikan relasi dari tabel tersebut. Tabel nasabah dan rekening dihubungkan oleh atribut id_nasabah.
   - Buat query SQL.
     SELECT id_nasabah.nasabah, nama_nasabah.nasabah, saldo.rekening
     FROM nasabah, rekening
     WHERE nasabah.id_nasabah = rekening.id_nasabah;

2. Tampilkan nomor_rekening dan nama_cabang..!
   - SELECT rekening.nomor_rekening, cabang.nama_cabang
     FROM rekening, cabang
     WHERE rekening.kode_cabang = cabang.kode_cabang;

3. Tampilkan nama_nasabah, alamat_nasabah, dan nomor_rekening..!

4. Tampilkan nomor_rekening dan kota..!

Mahasiswa

npm	nama	kelas
2009123001	Abigail	4B
2009123002	Reddi	4C
2009123003	Reva	4A
2009123004	Chyntia	4E
2009123005	Steven	4B

Nilai

npm	no_mk	mid	final
2009123001	110011	78	80
2009123002	120011	80	87
2009123003	130011	69	75
2009123004	110011	80	60
2009123005	140011	85	90

Matakuliah

no_mk	nama_mk	sks
110011	Basis data	4
120011	Komunikasi data	2
130011	Algoritma	3
140011	Jaringan Komputer	3
150011	Logika matematika	3

Buatlah database dengan nama mhs dan tabel di atas kemudian kerjakan latihan..!

1. Tampilkan npm dan no_mk..!
2. Tampilkan npm, nama, dan mid..!
3. Tampilkan tampilkan nama mahasiswa yang nilai midnya di atas 80..!
4. Tampilkan nama, kelas yang nilai mid dibawah 80 atau nilai final di atas 80..!
5. Tampilkan nama matakuliah yang nilai mid di atas 80..!

* * *

9

RELASI ANTAR 3 TABEL (..WHERE..)

Gunakan database bank..!

1. Tampilkan nama nasabah yang saldonya di atas 20 juta dan rekeningnya tercatat di kota bandung..!
    a. Tentukan tabel yang akan digunakan. (nasabah, rekening, cabang)
    b. Cari atribut penghubung antar tabel. (nasabah dengan rekening dihubungkan oleh atribut id_nasabah, sedangkan rekening dan cabang dihubungkan oleh atribut kode_cabang).
    c. Buat query SQL.

```
SELECT nasabah.nama_nasabah
FROM cabang, rekening, nasabah
WHERE nasabah.id_nasabah = rekening.id_nasabah
        rekening.kode_cabang = cabang.kode_cabang
        AND saldo  > ‘ 20000000’
        AND kota = ‘bandung’;
```

2. Tampilkan nama_nasabah, saldo, dan kota..!
3. Tampilkan id_nasabah dan nomor_rekening yang cabangnya berada di Surabaya..!
4. Tampilkan tampilkan nama nasabah yang saldonya di atas 5.000.000 dan berada di cabang Jakarta Pusat..!
5. Tampilkan nama nasabah yang tinggal di Jl. Gurame atau yang cabangnya di kota Depok..!

Gunakan database mhs..!

1. Tampilkan nama mahasiswa dan nama matakuliah yang nilai midnya *antara* 70 sampai 80..!

```
SELECT  mahasiswa.nama, matakuliah.nama_mk
FROM mahasiswa, matakuliah
WHERE mid BETWEEN 70 AND 80;
```

2. Tampilkan nama mahasiswa yang mengambil mata kuliah komunikasi data..!
3. Tampilkan nama mahasiswa, nilai final, dan nama mata kuliahnya..!
4. Tampilkan nama mahasiswa yang nilainya kurang dari 80 dan mengambil mata kuliah Algoritma..!
5. Tampilkan kelas yang nilai finalnya di atas 80 atau yang jumlah sksnya 3..!

* * *

10

RELASI ANTAR TABEL (JOIN)

JOIN adalah penggabungan data yang berasal dari beberapa tabel. Operator yang biasa digunakan adalah sama dengan (=), maka sering disebut dengan *equality join* atau *equijoin*.

Equijoin dikelompokkan ke dalam dua bagian yaitu inner equijoin (inner join) dan outer equijoin (outer join). Yang termasuk dalam Outer Join adalah Left Join dan Right Join.

Bentuk penggabungan data yang terakhir yang akan dibahas adalah perkalian Cartesian (Cartesian product) atau disebut juga dengan cross join atau full join.

1. JOIN / INNER JOIN

   Akan menghasilkan baris-baris yang cocok antar kedua tabel paling tidak satu baris.

   Sintaks :

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name = table_name2.column_name;
```

2. LEFT JOIN

   Akan menampilkan seluruh baris dari tabel di sebelah kiri (tabel 1), walaupun tidak ada yang cocok dengan tabel di sebelah kanan (tabel 2).

   Sintaks :

```
SELECT column_name(s)
FROM table_name1
LEFT JOIN table_name2
ON table_name1.column_name = table_name2.column_name
```

3. RIGHT JOIN

   Akan menampilkan seluruh baris dari tabel di sebelah kanan (tabel 2), walaupun tidak ada yang cocok dengan tabel di sebelah kiri (tabel 1).

   Sintaks :

```
SELECT column_name(s)
FROM table_name1
RIGHT JOIN table_name2
ON table_name1.column_name = table_name2.column_name
```

4. FULL JOIN
   Akan menampilkan baris-baris yang cocok dari salah satu tabel.
   Sintaks :

```
SELECT column_name(s)
FROM table_name1
FULL JOIN table_name2
ON table_name1.column_name = table_name2.column_name
```

Contoh :

Buat database penjualan..!

Pelanggan

P_Id	Nama	Alamat	Kota
1	Hani	Jl. Bunga	Jakarta Timur
2	Stefan	Jl. Ikan	Jakarta Barat
3	Pipit	Jl. Buah	Jakarta Selatan

Order

O_Id	NoOrder	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

1. JOIN/INNER JOIN
   Menampilkan nama dan nomor order yang diurutkan berdasarkan nama.
   Sintaks :

```
SELECT Pelanggan.nama, Order.NoOrder
FROM Pelanggan
INNER JOIN Order
ON Pelanggan.P_Id=Order.P_Id
ORDER BY Perlanggan.nama;
```

   Hasilnya :

Nama	NoOrder
Hani	22456
Hani	24562
Pipit	77895
Pipit	44678

2. LEFT JOIN
   Menampilkan nama dan nomor order yang di urutkan berdasarkan nama.

   Sintaks :

```
SELECT Pelanggan.Nama, Order.NoOrder
FROM Pelanggan
LEFT JOIN Order
ON Pelanggan.P_Id=Order.P_Id
ORDER BY Pelanggan.Nama;
```

   Hasilnya :

Nama	NoOrder
Hani	22456
Hani	24562
Pipit	77895
Pipit	44678
Stefan	Null

3. RIGHT JOIN
   Menampilkan nama dan nomor order yang di urutkan berdasarkan nama.

   Sintaks :

```
SELECT Pelanggan.Nama, Order.NoOrder
FROM Pelanggan
RIGHT JOIN Order
ON Pelanggan.P_Id=Order.P_Id
ORDER BY Pelanggan.Nama;
```

   Hasilnya :

Nama	OrderNo
Hani	22456
Hani	24562
Pipit	77895
Pipit	44678
Null	34764

4. FULL JOIN/CROSS JOIN
   Menampilkan nama dan nomor order yang di urutkan berdasarkan nama.

   Sintaks :

```
SELECT Pelanggan.Nama, Order.NoOrder
FROM Pelanggan
FULL JOIN Order
ON Pelanggan.P_Id = Order.P_Id
ORDER BY Pelanggan.Nama;
```

   Hasilnya :

Nama	OrderNo
Hani	22456
Hani	24562
Pipit	77895
Pipit	44678
Stefan	Null
Null	34764

Penggunaan cross join tanpa menggunakan ON dan USING.
Contoh :

```
SELECT Pelanggan.Nama, Order.NoOrder
FROM Pelanggan
CROSS JOIN Order;
```

Hasilnya :

nama	no_order
Hani	77895
Hani	44678
Hani	22456
Hani	24562
Hani	34764
Stefan	77895
Stefan	44678
Stefan	22456
Stefan	24562
Stefan	34764
Pipit	77895
Pipit	44678
Pipit	22456
Pipit	24562
Pipit	34764

Buat database animal..!
Buat tabel-tabel berikut dan cari hasil dari JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN..!

Animal

id	animal
1	Cat
2	Dog
3	Cow

food

id	Food
1	Milk
2	Bone
3	Grass

* * *

11

UNION, INTERSECT, EXCEPT

- UNION berguna untuk menampilkan hasil gabungan dari dua tabel.
  Sintaks :

```
SELECT column_name (s) FROM table_name1
UNION
SELECT column_name(s) FROM table_name2;
```

- INTERSECT berguna untuk menampilkan irisan dari dua tabel.
  Sintaks :

```
SELECT column_name (s) FROM table_name1
INTERSECT
SELECT column_name (s) FROM table_name2;
```

- EXCEPT berguna untuk menampilkan perkecualian dari dua tabel.
  Sintaks :

```
SELECT column_name (s) FROM table_name1
EXCEPT
SELECT column_name (s) FROM table_name2;
```

Buat database dengan nama union kemudian buat tabel-tabel berikut :

cabang_a

p_id	p_nama
1	Hani
2	Pipit A
3	Pipit B
4	Stefan

cabang_b

p_id	p_nama
1	Lona
2	Marni
3	Pipit B
4	Jon

Contoh :

1. SELECT p_nama FROM cabang_a
   UNION
   SELECT p_nama FROM cabang_b;

   Hasilnya :

p_nama
Hani
Jon
Lona
Marni
Pipit A
Pipit B
Stefan

2. SELECT p_nama FROM cabang_a
   UNION ALL
   SELECT p_nama FROM cabang_b;

   Hasilnya :

p_nama
Hani
Pipit A
Pipit B
Stefan
Lona
Marni
Pipit B
Jon

3. SELECT p_nama FROM cabang_a
   INTERSECT
   SELECT p_nama FROM cabang_b;

p_nama
Pipit B

4. SELECT p_nama FROM cabang_a
   EXCEPT
   SELECT p_nama FROM cabang_b;

p_nama
Hani
Pipit A
Stefan

Buat database angka..!

Buat tabel-tabel di bawah ini..!

Tampilkan hasil dari UNION, UNION ALL, EXCEPT, INTERSECT..!

Angka1

1
2
3
4
5

Angka2

3
4
5
6
7

* * *