Matakuliah Database dengan mySQL

Profile

Nama	**: Edy Nasri, S.Kom, M.Kom**
Panggilan	**: Anas**
HP	**: 0877-0998-8740**
WA	**: 0888-0120-0001**
BBM	**: 587EAF21**
Email	**: edynasri.unbaja@gmail.com**
Blog	**: bo-fsakti.blogspot.co.id**



Visi

Visi Program Studi Teknik Informatika disusun dengann mengacu pada Visi Universitas Banten Jaya (UNBAJA) dan Fakultas Ilmu Komputer (FILKOM). Adapun Visi Universitas Banten Jaya:

"Menjadi Universitas Terpercaya dan Unggul dalam Pengembangan IPTEK di Provinsi Banten Tahun 2018"

Sementara untuk Visi Fakultas Ilmu Komputer:

"Menjadi Fakultas Ilmu Komputer Terpercaya yang Memiliki Keunggulan Ilmu Pengetahuan dan Teknologi yang Terintegrasi di Provinsi Banten Tahun 2018"

Berdasarkan Visi Universitas Banten Jaya dan Fakultas Ilmu Komputer maka Visi Program Studi Teknik Informatika dirumuskan sebagai berikut:

"Menjadi Program Studi Teknik Informatika terpercaya yang memiliki keunggulan Ilmu Pengetahuan dan Teknologi yang Terintegrasi di Provinsi Banten tahun 2018"

Berdasarkan Visi Universitas Banten Jaya dan Fakultas Ilmu Komputer maka Visi Program Studi Sistem Informasi dirumuskan sebagai berikut:

"Menjadi Program Studi Sistem Informasi terpercaya yang memiliki keunggulan Ilmu Pengetahuan dan Teknologi yang Terintegrasi di Provinsi Banten tahun 2018"

1. Introduction

MySQL pertama kali dibuat dan dikembangkan oleh David Axmark, Allan Larsson dan Michael Widenius di Swedia, pada tahun 1980-an (web resmi : http://www.mysql.com).

MySQL merupakan sofware database open source, software ini sangat populer dengan kehandalan, kecepatan dan mudah dalam penggunaaanya, MySQL menjadi pilhan utama bagi pengembangan software dan aplikasi baik dipakai di plat form desktop maupun web. Dan banyak juga digunakan perushaan besar seperti : Google, Yahoo, youtube, wordpress, facebook dan lain2.

Penegnelan MySQL

- ✓ Pengenalan Database, DBMS, dan RDBMS
- ✓ Beberapa Istilah Database
- ✓ Hierarki Database
- ✓ Pengenalan Database MySQ

- **Pengenalan Database, DBMS, dan RDBMS**

  Database atau Basis data adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Database digunakan untuk menyimpan informasi atau data yang terintegrasi dengan baik di dalam komputer.

Dalam mengelola database diperlukan suatu perangkat lunak yang disebut **Database Management System (DBMS).** DBMS merupakan suatu sistem perangkat lunak yang memungkinkan user (pengguna) untuk membuat, memelihara, mengontrol, dan mengakses database secara praktis dan efisien. Dengan mengunakan DBMS, pengguna akan lebih mudah mengontrol dan memanipulasi data.

Sedangkan **Relationship Database Management System (RDBMS)** merupakan salah satu jenis DBMS yang mendukung adanya relationship atau hubungan antar tabel. Di samping RDBMS, terdapat jenis DBMS lain, misalnya Hierarchy DBMS, Object Oriented DBMS, dsb.

Beberapa software atau perangkat lunak DBMS yang sering digunakan dalam aplikasi program antara lain :

Beberapa software atau perangkat lunak DBMS yang sering digunakan dalam aplikasi program antara lain :

- MySQL – http://www.mysql.com
- PostgreSQL- http://www.postgresql.org
- DB2 - http://www-306.ibm.com/software/data/db2
- Firebird - http://www.firebirdsql.org
- Microsoft SQL Server - http://www.microsoft.com/sql
- Oracle - http://www.oracle.com
- Sybase - http://www.sybase.com
- Interbase - http://www.borland.com/interbase
- Teradata - http://www.teradata.com

Istilah Dalam Database

Istilah dalam database sangat penting untuk dipahami, adapun istilah yang umum dalam database antara lain :

- **Table**

Sebuah tabel merupakan kumpulan data (nilai) yang diorganisasikan ke dalam baris (record) dan kolom (field). Masing-masing kolom memiliki nama yang spesifik dan unik.

- **Field**

Field merupakan kolom dari sebuah table. *Field* memiliki ukuran *type* data tertentu yang menentukan bagaimana data nantinya tersimpan.

- **Record**

Field merupakan sebuah kumpulan nilai yang saling terkait.

- **Key**

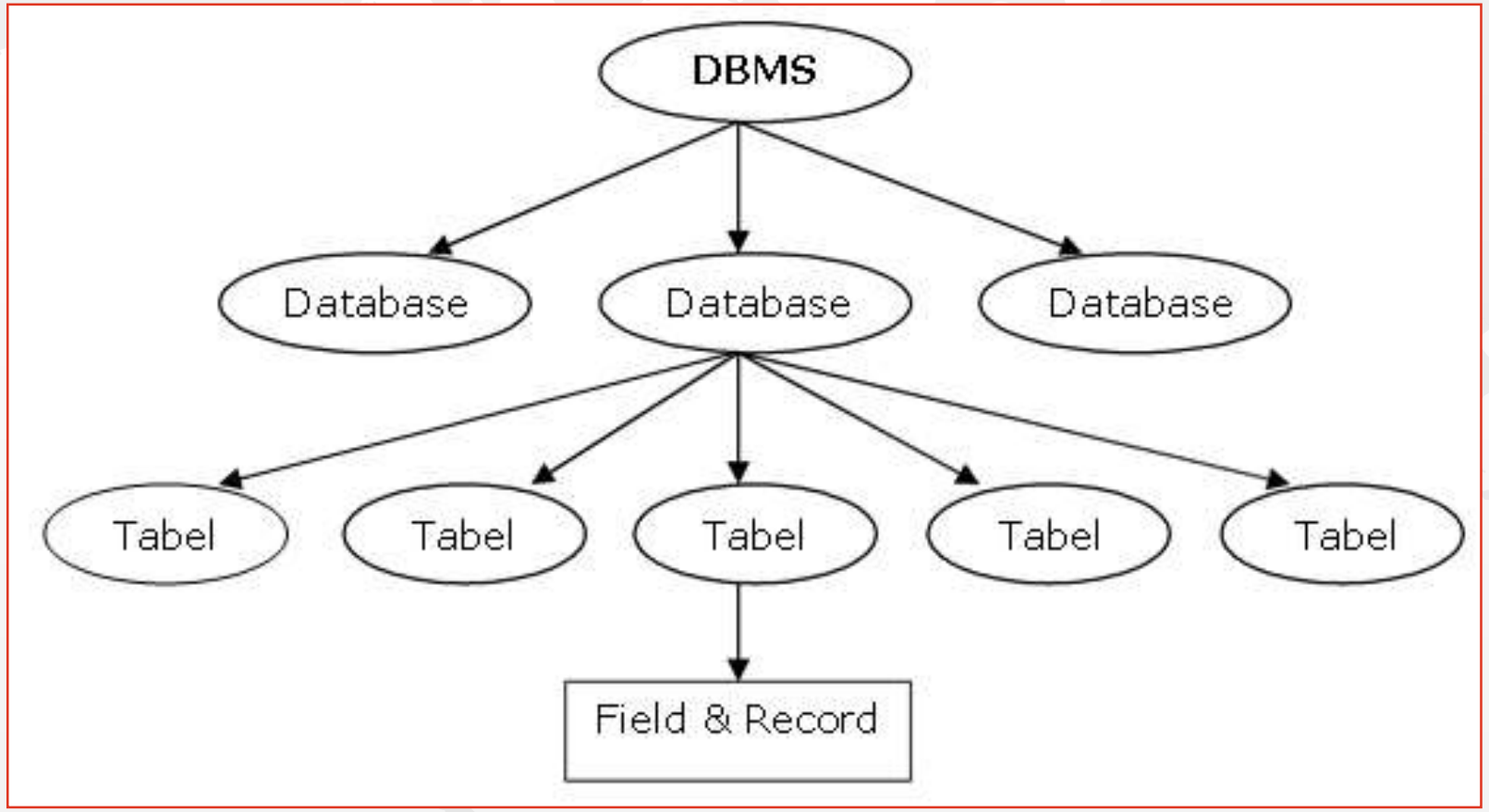
Key merupakan suatu field yang dapat dijadikan kunci dalam operasi tabel. Dalam konsep database, key memiliki banyak jenis diantaranya *Primary Key, Foreign Key, Composite Key*, dll.

- **SQL**

SQL atau Structured Query Language merupakan suatu bahasa (*language*) yang digunakan untuk mengakses database. SQL sering disebut juga sebagai query.

Hierarki Database

Hierarki atau urutan database digambarkan dalam gambar sbb



Database MySQL

Database MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL, berbasis *Database Management System (DBMS)* yang multithread, multi-user. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Fitur-fitur MySQL antara lain :

- **Relational Database System**. Seperti halnya software database lain yang ada di pasaran, MySQL termasuk RDBMS.
- **Arsitektur Client-Server**. MySQL memiliki arsitektur client-server dimana server database MySQL terinstal di server. Client MySQL dapat berada di komputer yang sama dengan server, dan dapat juga di komputer lain yang berkomunikasi dengan server melalui jaringan bahkan internet.

- Mengenal perintah SQL standar. SQL (Structured Query Language) merupakan suatu bahasa standar yang berlaku di hampir semua software database. MySQL mendukung SL versi SQL:2003.
- Dukungan MySQL : Mendukung **Sub Select**. Mulai versi 4.1 MySQL telah mendukung select dalam select (sub select); Mendukung **Views**. MySQL mendukung views sejak versi 5.0; Mendukung **Stored Prosedured (SP)**. MySQL mendukung SP sejak versi 5.0; Mendukung **Triggers**. MySQL mendukung trigger pada versi 5.0 namun masih terbatas. Pengembang MySQL berjanji akan meningkatkan kemampuan trigger pada versi 5.1.; Mendukung **replication**. Mendukung transaksi; Mendukung **foreign key**.; Tersedia fungsi GIS. ; Free (bebas didownload); Stabil dan tangguh; Fleksibel dengan berbagai pemrograman; Security yang baik; Dukungan dari banyak komunitas; Perkembangan software yang cukup cepat.

Tugas :

Coba lakukan Instalasi MySQL dengan menikuti tutorial pada blog: bo-fsakti.blogspot.co.id

2. Merancang Database

Bahasan :

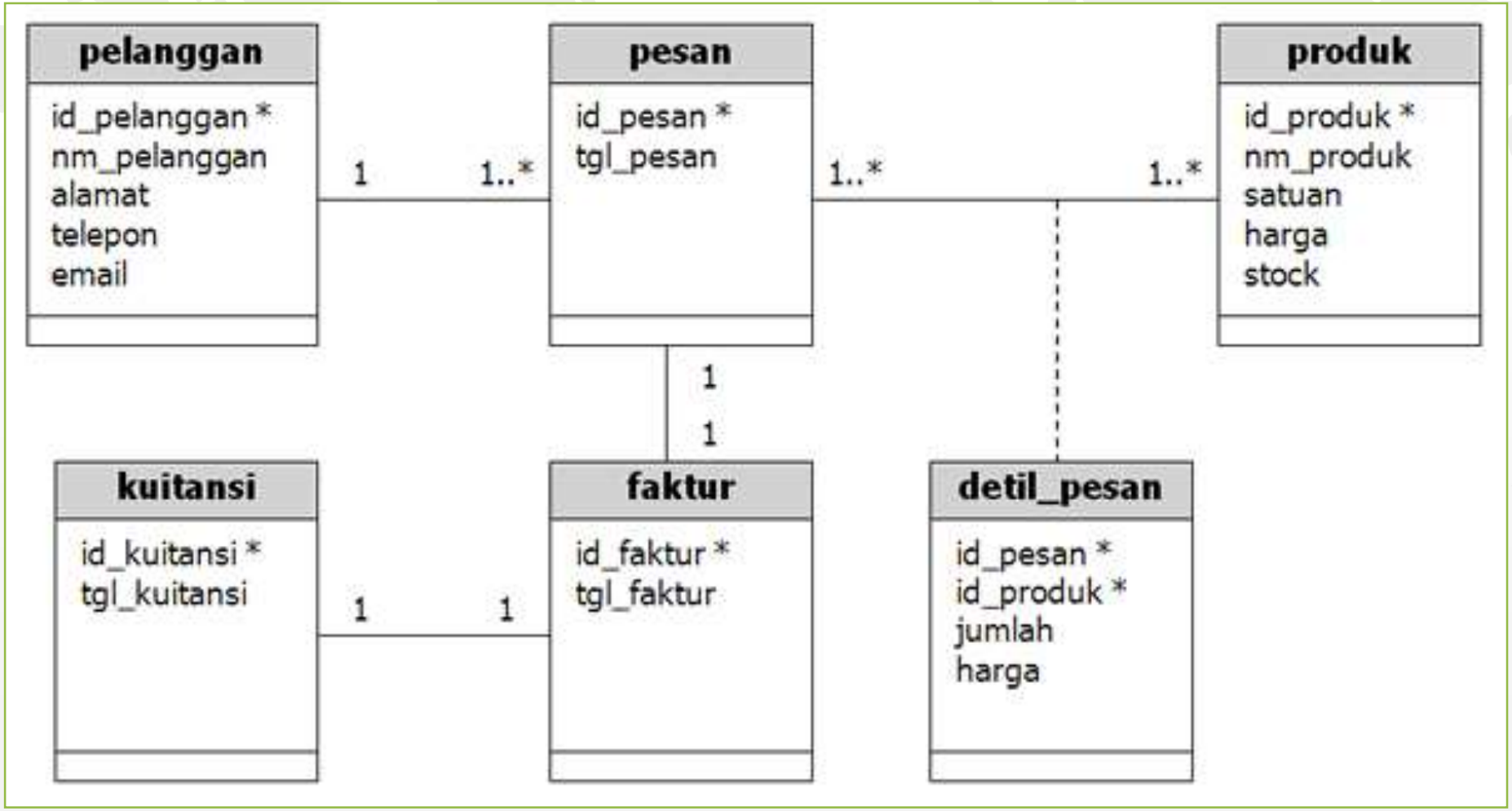
- Tipe Table MySQL
- Tipe Field MySQL
- Merancang suatu database yang baik.

Merancang database merupakan hal yang pertama kali harus dilakukan sebelum membuat suatu aplikasi berbasis database. Rancangan database yang baik akan menentukan seberapa baik sebuah aplikasi dibangun. Orang yang bertanggung jawab dalam merancang database biasanya disebut sebagai **seorang sistem analis**.

Berikut ini contoh sederhana sebuah rancangan database dalam pada Sistem Pemesanan Barang (ordering system). Rancangan database disajikan dalam bentuk ***class diagram***.

Contoh:

Class Diagram Sistem Pemesanan Barang



Jenis Tipe Pada Tabel MySQL

MySQL mendukung beberapa tipe tabel, tergantung konfigurasi saat proses instalasi MySQL. MySQL memiliki 3 (tiga) tipe tabel utama, yaitu *MyISAM, InnoDB dan HEAP.*

1. MyISAM

Tipe tabel MyISAM merupakan tipe tabel yang sederhana, stabil dan mudah digunakan. Jika kita akan menyimpan data sederhana yang tidak terlalu rumit, maka gunakanlah tipe tabel ini. Kelebihan utama MyISAM adalah kecepatan dan kestabilannya. Jika kita memilih tipe tabel MyISAM, maka MySQL secara otomatis akan menentukan salah satu dari tiga jenis tabel MyISAM, yaitu :

a. **MyISAM static**. Jenis ini digunakan ketika semua kolom dalam tabel didefinisikan dengan ukuran yang pasti (fixed). Dengan kata lain, tidak ada kolom yang memiliki tipe seperti VARCHAR, TEXT dan BLOB. Karena sifatnya yang fixed, maka jenis ini akan lebih cepat, aman dan stabil.

b. **MyISAM dymanic.** Jenis ini digunakan ketika terdapat kolom dengan tipe yang dinamis, seperti tipe kolom VARCHAR. Keuntungan utama dari jenis ini adalah ukuran yang dinamis. Jadi sifatnya lebih efektif karena ukuran data (file) menyesuaikan isi dari masing-masing kolom (field).

c. **MyISAM Compressed.** Kedua jenis MyISAM, static dan dynamic dapat dikompresi menjadi satu jenis yaitu MyISAM Compressed dengan perintah myisamchk. Tentunya hasilnya lebih kecil dari segi ukuran. Tabel yang terkompresi tidak dapat dikenakan operasi seperti INSERT, UPDATE dan DELETE.

2. InnoDB

Tipe tabel InnoDB merupakan tipe tabel MySQL yang mendukung proses transaksi. Tipe ini memiliki beberapa keunggulan, antara lain:

a. Mendukung transaksi antar tabel.
b. Mendukung row-level-locking.
c. Mendukung Foreign-Key Constraints.
d. Crash recovery.

3. HEAP

Tabel dengan tipe HEAP tidak menyimpan datanya di hardisk, tetapi menyimpan di RAM (memori). Tipe tabel ini biasanya digunakan sebagai tabel sementara (temporary). Tabel secara otomatis akan dihapus (hilang) dari MySQL saat koneksi ke server diputus atau server MySQL dimatikan.

Tipe Tabel yang Lain

Selain 3 (tiga) tipe tabel diatas, yaitu MyISAM, InnoDB dan HEAP, MySQL juga mendukung tipe tabel yang lain, yaitu:

a) **BDB**. Tipe tabel ini mirip tipe tabel InnoDB, namun penggunaannya belum maksimal.
b) **Archieve**. Tipe ini tersedia sejak MySQL versi 4.1. Tipe ini digunakan untuk menyimpan tabel yang terkompresi, dimana biasanya digunakan dalam proses backup.
c) **CSV**. Tipe ini digunakan untuk menyimpan data dalam bentuk file text yang dibatasi dengan koma (delimiter). Tipe ini tersedia sejak MySQL versi 4.1.
d) **NDB Table** (MySQL Cluster). Tersedia sejak MySQL versi 4.1.
e) **Federated** (External Tables). Tipe ini tersedia sejak MySQL versi 5.0.

Jenis Type Pada Field (Kolom) MySQL

MySQL memiliki cukup banyak tipe data untuk field (kolom) tabel. Tipe field (kolom) ini menentukan besar kecilnya ukuran suatu tabel. Tipe field di MySQL setidaknya terbagi menjadi beberapa kelompok, yaitu numerik, string, date and time, dan kelompok himpunan (set dan enum). Masing-masing tipe field memiliki batasan lebar dan ukurannya.

Tipe Numeric

Tipe data numerik digunakan untuk menyimpan data numeric (angka). Ciri utama data numeric adalah suatu data yang memungkinkan untuk dikenai operasi aritmatika seperti pertambahan, pengurangan, perkalian dan pembagian. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe numerik:

1. TINYINT

Penggunaan	: untuk menyimpan data bilangan bulat positif dan negatif.
Jangkauan	: -128 s/d 127
Ukuran	: 1 byte (8 bit).

2. SMALLINT
Penggunaan : untuk menyimpan data bilangan bulat positif dan negatif.
Jangkauan : -32.768 s/d 32.767
Ukuran : 2 byte (16 bit).

3. MEDIUMINT
Penggunaan : untuk menyimpan data bilangan bulat positif dan negatif.
Jangkauan : -8.388.608 s/d 8.388.607
Ukuran : 3 byte (24 bit).

4. INT
Penggunaan : untuk menyimpan data bilangan bulat positif dan negatif.
Jangkauan : -2.147.483.648 s/d 2.147.483.647
Ukuran : 4 byte (32 bit).

5. BIGINT

Penggunaan : untuk menyimpan data bilangan bulat positif dan negatif.
Jangkauan : ± 9,22 x 1018
Ukuran : 8 byte (64 bit).

6. FLOAT

Penggunaan : untuk menyimpan data bilangan pecahan positif dan negatif presisi tunggal.
Jangkauan : -3.402823466E+38 s/d -1.175494351E-38, 0, dan 1.175494351E-38 s/d 3.402823466E+38.
Ukuran : 4 byte (32 bit).

7. DOUBLE

Penggunaan : untuk menyimpan data bilangan pecahan positif dan negatif presisi ganda.

Jangkauan : -1.79...E+308 s/d -2.22...E-308, 0, dan 2.22...E-308 s/d 1.79...E+308.

Ukuran : 8 byte (64 bit).

8. REAL

Merupakan sinonim dari DOUBLE.

9. DECIMAL

Penggunaan : untuk menyimpan data bilangan pecahan positif dan negatif.

Jangkauan : -1.79...E+308 s/d -2.22...E-308, 0, dan 2.22...E-308 s/d 1.79...E+308.

Ukuran : 8 byte (64 bit).

10. NUMERIC

Merupakan sinonim dari DECIMAL.

Jenis Tipe Date dan Time

Tipe data *date* dan *time* digunakan untuk menyimpan data tanggal dan waktu. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe *date* dan *time*:

11. DATE
    Penggunaan : untuk menyimpan data tanggal.
    Jangkauan : 1000-01-01 s/d 9999-12-31 (YYYY-MM-DD)
    Ukuran : 3 byte.

12. TIME
    Penggunaan : untuk menyimpan data waktu.
    Jangkauan : -838:59:59 s/d +838:59:59 (HH:MM:SS)
    Ukuran : 3 byte.

13. DATETIME

Penggunaan : untuk menyimpan data tanggal dan waktu.
Jangkauan : '1000-01-01 00:00:00' s/d '9999-12-31 23:59:59'
Ukuran : 8 byte

14. YEAR

Penggunaan : untuk menyimpan data tahun dari tanggal.

Jangkauan : 1900 s/d 2155

Ukuran : 1 byte.

Jenis Tipe String (Text)

Tipe data string digunakan untuk menyimpan data string (text). Ciri utama data string adalah suatu data yang memungkinkan untuk dikenai operasi aritmatika seperti pertambahan, pengurangan, perkalian dan pembagian. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe string:

15. CHAR
Penggunaan : untuk menyimpan data string ukuran tetap.
Jangkauan : 0 s/d 255 karakter

16. VARCHAR
Penggunaan : untuk menyimpan data string ukuran dinamis.
Jangkauan : 0 s/d 255 karakter (versi 4.1), 0 s/d 65.535 (versi 5.0.3)

17. TINYTEXT
Penggunaan : untuk menyimpan data text.
Jangkauan : 0 s/d 255 karakter (versi 4.1), 0 s/d 65.535 (versi 5.0.3)

18. TEXT
Penggunaan : digunakan untuk menyimpan data text.
Jangkauan : 0 s/d 65.535 (216 - 1) karakter

19. MEDIUMTEXT
Penggunaan : untuk menyimpan data text.
Jangkauan : 0 s/d 224 - 1 karakter

20. LONGTEXT
Penggunaan : untuk menyimpan data text.
Jangkauan : 0 s/d 232 - 1 karakter

Jenis Tipe BLOB Biner)

Jenis Tipe data blob digunakan untuk menyimpan data biner. Tipe ini biasanya digunakan untuk menyimpan kode-kode biner dari suatu file atau object. BLOB merupakan singkatan dari Binary Large Object. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe blob:

21. BIT (sejak versi 5.0.3)
Penggunaan : untuk menyimpan data biner.
Jangkauan : 64 digit biner

22. TINYBLOB
Penggunaan : untuk menyimpan data biner.
Jangkauan : 255 byte

23. BLOB
Penggunaan : untuk menyimpan data biner.
Jangkauan : 216 - 1 byte

24. MEDIUMBLOB
Penggunaan : untuk menyimpan data biner.
Jangkauan : 224 - 1 byte

25. LONGBLOB
Penggunaan : digunakan untuk menyimpan data biner.
Jangkauan : 232 - 1 byte

Tipe Data yang Lain

Selain tipe data di atas, MySQL juga menyediakan tipe data yang lain. Tipe data di MySQL mungkin akan terus bertambah seiring dengan perkembangan versi MySQL. Berikut ini beberapa tipe data tambahan MySQL:

26. ENUM
Penggunaan : enumerasi (kumpulan data).
Jangkauan : sampai dengan 65535 string.

27. SET
Penggunaan : combination (himpunan data).
Jangkauan : sampai dengan 255 string anggotas.

Merancang Database yang Baik

Rancangan database menentukan suatu aplikasi efektif atau tidak, efisien atau tidak, baik atau tidak.

Beberapa Aturan Merancang Database yang Baik.

a) Tabel dalam database tidak boleh mengandung record (data) ganda, atau dengan kata lain tidak boleh ada redudancy data. Jika terdapat data yang sama, maka perlu dilihat kembali rancangan tabelnya.
b) Setiap tabel dalam database, harus memiliki field (kolom) yang unik. Field ini disebut sebagai Primary Key.
c) Tabel harus sudah normal.
d) Besar atau ukuran database hendaknya dibuat seminimal mungkin. Hal ini ditentukan oleh pemilihan tipe data yang tepat.
e) Merancang database hendaknya memperhatikan apakah rancangan dapat menampung data (record) sesuai yang dibutuhkan oleh aplikasi.

Penamaan Identifier.

a) Penamaan identifier (database, tabel, kolom) di MySQL bersifat case- sensitive. Penamaan identifier hendaknya konsisten untuk semua tabel dalam suatu database. Kita dapat menggunakan model lower-case, UPPER-CASE, camelCase dll.
b) Nama database, tabel dan kolom maksimal 64 karakter.
c) Hindari penggunaan karakter khusus, karena bisabermasalah dalam sistem operasi yang lain.
d) Pilih nama untuk field (kolom) yang mencerminkan isi dari data yang disimpan.

3. DASAR-DASAR SQL

- Pendahuluan
- Perintah DDL
- Perintah DML

Pendahuluan

SQL merupakan singkatan dari *Structured Query Language*. SQL atau juga sering disebut sebagai query merupakan suatu bahasa (*language*) yang digunakan untuk mengakses database.

Terdapat 3 (tiga) jenis perintah SQL, yaitu :

1. DDL (*Data Definition Language)*

DDL merupakan perintah SQL yang berhubungan dengan pendefinisian suatu struktur database, dalam hal ini *database* dan *table*. Beberapa perintah dasar yang termasuk DDL ini antara lain :

- CREATE
- ALTER
- RENAME
- DROP

2. DML (*Data Manipulation Language)*

DML merupakan perintah SQL yang berhubungan dengan manipulasi atau pengolahan data atau *record* dalam table. Perintah SQL yang termasuk dalam DML antara lain :

- SELECT
- INSERT
- UPDATE
- DELETE

3. DCL (*Data Control Language)*

DCL merupakan perintah SQL yang berhubungan dengan manipulasi user dan hak akses (priviledges). Perintah SQL yang termasuk dalam DCL antara lain :

- GRANT
- REVOKE

DDL (*Data Definition Language)*

Membuat, Menampilkan, Membuka dan Menghapus Database

Membuat Database (*CREATE DATABASE*)

Sintaks umum SQL untuk membuat suatu database adalah sebagai berikut :

CREATE DATABASE [IF NOT EXISTS] nama_database;

Contoh :

```
1 CREATE DATABASE PENJUALAN;
```

Menampilkan Database

```
1 SHOW DATABASES;
```

Menghapus database Database

Untuk menghapus suatu database, sintaks umumnya adalah sbb

```
DROP DATABASE [IF EXISTS] nama_database;
```

Contoh:

```
1 DROP DATABASE PENJUALAN;
```

Membuat Tabel Baru

SQL untuk membuat suatu *table*

```
CREATE TABLE nama_tabel ( field1 tipe(panjang), field2
tipe(panjang),
...
fieldn tipe(panjang),
PRIMARY KEY (field_key)
);
```

Sebagai contoh, kita akan membuat tabel baru dengan struktur SBB:

Nama tabel : pelanggan

No	Nama Field	Tipe	Panjang
1	id_pelanggan *	Varchar	5
2	nm_pelanggan	Varchar	30
3	alamat	Text	-
4	telepon	Varchar	20
5	email	Varchar	50

SQL :

```
1  CREATE TABLE pelanggan (
2      id_pelanggan varchar(5) not null,
3      nm_pelanggan varchar(30) not null,
4      alamat text,
5      telepon varchar (20),
6      emial varchar (50),
7      primary key(id_pelanggan)
8  );
9
```

Pada perintah di atas, beberapa hal yang perlu diperhatikan :

- CREATE TABLE merupakan perintah dasar dari pembuatan table.
- pelanggan merupakan nama tabel yang akan dibuat.
- id_pelanggan, nm_pelanggan, alamat, telepon dan email merupakan nama *field*.
- varchar dan text merupakan tipe data dari field
- NOT NULL merupakan option untuk menyatakan bahwa suatu field tidak boleh kosong.
- PRIMARY KEY merupakan perintah untuk menentukan field mana yang akan dijadikan primary key pada tabel.
- 5, 10, 30 dan 50 di belakang tipe data merupakan panjang maksimal dari suatu *field*.
- Untuk tipe data date dan text (juga date dan blob) panjang karakter maksimalnya tidak perlu ditentukan.
- diakhiri perintah dengan titik-koma (;)

Melihat Tabel

untuk melihat tabel mhs sudah benar-benar sudah ada maka dengan perintah

```
1 show TABLES
```

Untuk melihat struktur tabel "**mhs**" secara lebih detail, cobalah perintah atau query sebagai berikut :

```
1 DESCRIBE pelanggan;
```

Field	Type	Null	Key	Default	Extra
id_pelanggan	varchar(5)	NO	PRI	*NULL*	
nm_pelanggan	varchar(30)	NO		*NULL*	
alamat	text	YES		*NULL*	
telepon	varchar(20)	YES		*NULL*	
emial	varchar(50)	YES		*NULL*	

Mengubah Struktur Table dengan ALTER

perintah SQL-nya sebagai berikut :

```
ALTER TABLE nama_tabel alter_options;
```

- ALTER TABLE merupakan perintah dasar untuk mengubah tabel.
- nama_tabel merupakan nama tabel yang akan diubah strukturnya.
- alter_options merupakan pilihan perubahan tabel.

Option yang bisa digunakan dalam Alter

ADD definisi_field_baru

Option ini digunakan untuk menambahkan field baru dengan "**definisi_field_baru**" (nama field, tipe dan option lain).

ADD INDEX nama_index

Option ini digunakan untuk menambahkan index dengan nama"**nama_index"** pada tabel

ADD PRIMARY KEY (field_kunci)

Option untuk menambahkan primary key pada tabel

CHANGE field_yang_diubah definisi_field_baru

Option untuk mengubah field_yang_diubah menjadi definisi_field_baru

MODIFY definisi_field

Option untuk mengubah suatu field menjadi definisi_field

DROP nama_field

Option untuk menghapus field nama_field

RENAME TO nama_tabel_baru

Option untuk mengganti nama tabel

Contoh

Variasi perintah ALTER untuk mengubah struktur suatu tabel

1. Menambahkan field "**tgllahir**" ke tabel **pelanggan**

```
ALTER TABLE pelanggan ADD tgllahir date NOT NULL;
```

2. Menambahkan primary key pada suatu tabel

```
ALTER TABLE pelanggan ADD PRIMARY KEY(id_pelanggan);
```

3. Mengubah **tipe field tgllahir** menjadi varchar dalam tabel **pelanggan**

```
ALTER TABLE pelanggan MODIFY tgllahir varchar(8) NOT NULL;
```

4. Menghapus field **tgllahir** dari tabel **pelanggan**

```
ALTER TABLE pelanggan DROP tgllahir;
```

Mengubah Nama Tabel

Untuk mengubah nama suatu tabel :

```
RENAME TABLE pelanggan TO plg;
ALTER TABLE plg RENAME TO pelanggan;
```

Menghapus Tabel

Untuk menghapus sebuah tabel

```
DROP TABLE nama_tabel;
```

Contoh:

```
DROP TABLE pelanggan;
```

DML (Data Manipulation Language)

Menambah Record dengan INSERT

Perintah SQL untuk menambahkan *record* atau data ke dalam suatu tabel dengan cara :

Cara 1 :

```
INSERT INTO nama_tabel VALUES ('nilai1','nilai2',...);
```

Cara 2 :

```
INSERT INTO nama_tabel(field1,field2,...) VALUES
('nilai1','nilai2',...);
```

Cara 3 :

```
INSERT INTO nama_tabel SET field1='nilai1', field2='nilai2',...;
```

Contoh
SQL untuk Insert :

```
1  INSERT INTO pelanggan VALUES ('P0001','Faiz
   Sakti','Jl.  Raya Serang KM.1', '088811111000',
   'faizsakti@yahoo.com');
```

Tampilkan kotak kueri

✔ 1 baris ditambahkan. (Pencarian dilakukan dalam 0.0890 detik.)

Perintah SQL SELECT untuk melihat isi tabel pelanggan

```
SELECT * FROM pelanggan
```

Hasil



+ Opsi

	id_pelanggan	nm_pelanggan	alamat	telepon	emial
Ubah Salin Hapus	P0001	Faiz Sakti	Jl. Raya Serang KM.1	088811111000	faizsakti@yahoo.com

Buatlah untuk menambahkan data minimal 5; dengan 3 cara yang disebutkan diatas

Mengedit Record dengan UPDATE

Proses update dilakukan jika terdapat data atau record dalam suatu tabel yang perlu diperbaiki. Proses update hanya memerbaiki data tidak untuk menambahkan data (record) baru. Perubahan yang terjadi dalam proses update bersifat permanen, artinya setelah perintah dijalankan tidak dapat di-*cancel* (*undo*).

Perintah SQL untuk mengedit suatu *record* atau data dari suatu tabel adalah sebagai berikut:

```
UPDATE nama_tabel SET field1=’nilaibaru’ [WHERE kondisi];
```

Contoh :

```
UPDATE pelanggan SET alamat=‘Serang'
WHERE id_pelanggan='P0001';
```

Menghapus Record dengan DELETE

Proses delete dilakukan jika terdapat data atau record dalam suatu tabel yang perlu dihapus atau dihilangkan.

Perintah SQL untuk menghapus suatu *record* atau data dari tabel adalah sebagai berikut

```
DELETE FROM nama_tabel [WHERE kondisi];
```

Contoh :

```
DELETE FROM pelanggan WHERE
id_pelanggan='P0005';
```

Menampilkan Record dengan SELECT

Perintah SELECT digunakan untuk menampilkan, bisa berupa sejumlah data dari tabel dan bisa juga berupa suatu ekspresi. Dengan perintah SELECT bisa mengatur tampilan atau keluaran sesuai tampilan yang diinginkan.

Bentuk dasar perintah SELECT data dari tabel adalah sebagai berikut :

```
SELECT [field | *] FROM nama_tabel [WHERE kondisi];
```

Contoh :

1. Menampilkan seluruh data atau record (*) dari tabel **pelanggan**

```
SELECT * FROM pelanggan;
```

2. Menampilkan *field* **id_pelanggan** dan **nm_pelanggan** dari seluruh pelanggan dalam tabel **pelanggan**

```
SELECT id_pelanggan, nm_pelanggan FROM
pelanggan;
```

3. Menampilkan id, nama dan alamat dari data pelanggan yang mempunyai id **P0006**

```
SELECT id_pelanggan, nm_pelanggan, alamat FROM pelanggan
WHERE id_pelanggan = 'P0006';
```

4. Menampilkan id, nama dan email data semua pelanggan yang mempunyai email di **gmail**

```
SELECT id_pelanggan, nm_pelanggan, email
FROM pelanggan WHERE email LIKE '%gmail%';
```

5. Menampilkan data semua pelanggan yang beralamat di **Serang** dan mempunyai email di **gmail**.

```
SELECT id_pelanggan, nm_pelanggan, alamat, email FROM
pelanggan WHERE alamat = ‘Serang' && email LIKE '%gmail.com';
```

6. Menampilkan semua data pelanggan secara urut berdasarkan **nama pelanggan** dengan perintah **ORDER BY**

```
SELECT id_pelanggan, nm_pelanggan FROM pelanggan
ORDER BY nm_pelanggan;
```

7. Menampilkan semua data pelanggan secara urut berdasarkan nama pelanggan secara DESCENDING

```
SELECT id_pelanggan, nm_pelanggan FROM pelanggan
ORDER BY nm_pelanggan DESC;
```

8. Menampilkan 3 record (data) pertama dari tabel **pelanggan** secara urut berdasarkan **nama pelanggan** dengan **LIMIT**

```
SELECT id_pelanggan, nm_pelanggan FROM pelanggan ORDER BY nm_pelanggan LIMIT 0,3;
```

9. Menampilkan jumlah record yang ada di tabel **pelanggan**.

```
SELECT COUNT(*)FROM pelanggan;
```

4. Fungsi-Fungsi Dalam MySQL

- Fungsi String
- Fungsi Tanggal dan Waktu
- Fungsi Numerik
- Fungsi Lain-lainnya

Fungsi String

1) CONCAT

Digunakan untuk menggabungkan dua atau lebih string (kolom).

Misalnya akan menggabungkan kolom **alamat** dan **telepon** pada tabel **pelanggan** menjadi sebuah kolom:

CONCAT (str1, str2, ...)

Contoh:

```
SELECT nm_pelanggan, CONCAT(alamat,'',telepon) FROM pelanggan;
```

CONCAT_WS (separator, str1, str2, ...)

Digunakan untuk menggabungkan dua atau lebih string (kolom), dengan separator diantara masing-masing string

Contoh:

```
SELECT CONCAT_WS (',','Faiz','Fawwaz','Andi');
```

SUBSTR

Digunakan untuk mengambil atau memotong string dimulai dari karakter **awal** sebanyak **panjang** karakter.

Selain SUBSTR bisa juga digunakan fungsi sbb :

SUBSTRING (string, awal, panjang); SUBSTRING (string FROM awal FOR panjang); MID (string, awal, panjang)

```
SELECT SUBSTRING (‘Teknik Informatika’,1,6);
```

LENGTH

Digunakan untuk menghitung panjang suatu string.

Selain SUBSTR bisa juga digunakan fungsi sbb :

OCTET_LENGTH (string) CHAR_LENGTH (string) CHARACTER_LENGTH (string)

```
SELECT LENGTH (‘Serang Banten')
```

LEFT

Digunakan untuk memotong string dari sebelah kiri sebanyak **panjang** karakter

```
SELECT LEFT (‘Teknik Infomatika’, 6)
```

RIGHT

Digunakan untuk memotong string dari sebelah kanan sebanyak **panjang** karakter

```
SELECT RIGHT (‘Teknik Infomatika’, 6)
```

LTRIM

Digunakan untuk menghilangkan spasi di awal string (kiri)

```
SELECT LTRIM (‘  Teknik Infomatika’)
```

RTRIM

Digunakan untuk menghilangkan spasi di awal string (kanan)

```
SELECT LTRIM (‘Teknik Infomatika  ’)
```

TRIM

Digunakan untuk menghilangkan spasi di awal dan akhir string(kiri dan kanan)

```
SELECT TRIM (‘ Teknik Infomatika  ’)
```

REPLACE (string, from_str, to_str)

Digunakan untuk mengganti suatu string dengan string yang lain

```
SELECT REPLACE ('ww.mysql.com', 'w', 'x' );
```

REPEAT (string, jumlah)

Digunakan untuk menduplikasi suatu string sebanyak **jumlah**

```
SELECT REPEAT ('Mont', 3);
```

REVERSE (string)

Digunakan untuk membalik string

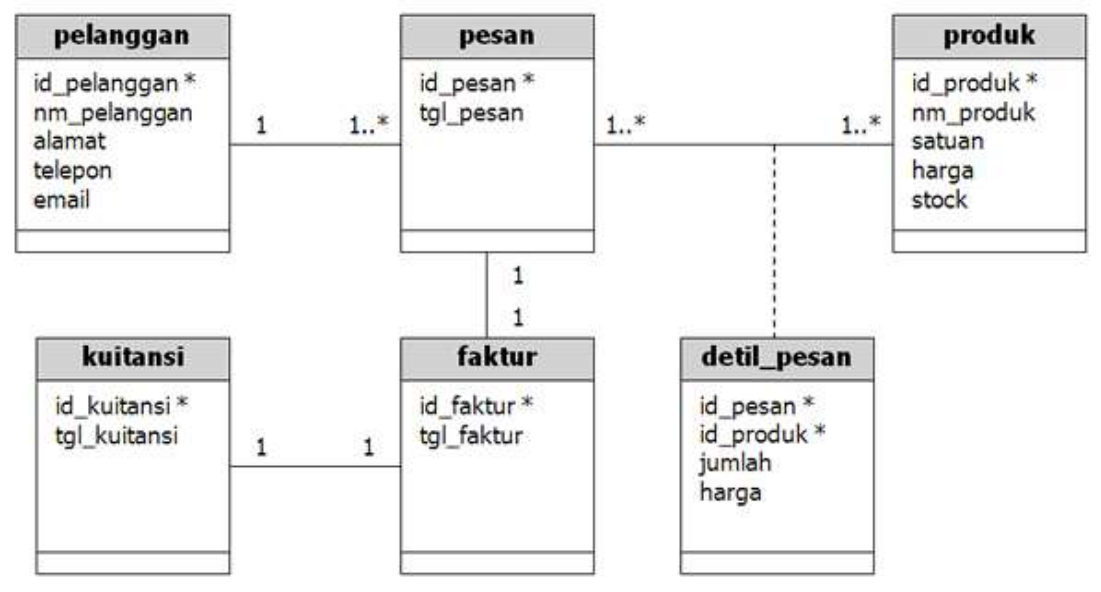
```
SELECT REVERSE ('mysql.com');
```

Perintah MySQL Lanjutan

- Perintah SELECT dari Banyak Tabel
- Pengelompokkan Hasil Query dengan GROUP BY
- HAVING
- SubSELECT
- Menampilkan Record secara Random
- Transaksi

perintah SQL lanjutan merupakan bahasab SQL kompleks seperti join antar tabel, grouping, sub select, random search dan transaksi (commit-and-roolback).

contoh yang disajikan mengacu pada **pemodelan data konseptual** Sistem Pemesanan (Penjualan) Barang sbb:



Untuk pembuatran tabel-tabel dari rancangan di atas, kita akan menggunakan tipe tabel **InnoDB**.

SQL dari pemodelan pemesanan diatas :

```
1  CREATE TABLE pelanggan (
2  id_pelanggan varchar(5) NOT NULL,
3  nm_pelanggan varchar(40) NOT NULL,
4  alamat text NOT NULL,
5  telepon varchar(20) NOT NULL,
6  email varchar(50) NOT NULL,
7  PRIMARY KEY  (id_pelanggan));
```

```
1  CREATE TABLE pesan (
2  id_pesan int(5) NOT NULL auto_increment,
3  d_pelanggan varchar(5) NOT NULL,
4  tgl_pesan date NOT NULL,
5  PRIMARY KEY  (id_pesan),
6  KEY id_pelanggan (id_pelanggan),
7  CONSTRAINT pesan_ibfk_1 FOREIGN KEY (id_pelanggan) REFERENCES pelanggan (id_pelanggan));
```

```
1  CREATE TABLE faktur (
2  id_faktur int(5) NOT NULL auto_increment,
3  id_pesan int(5) NOT NULL,
4  tgl_faktur date NOT NULL,
5  PRIMARY KEY  (id_faktur),
6  KEY id_pesan (id_pesan),
7  CONSTRAINT faktur ibfk 1 FOREIGN KEY (id pesan) REFERENCES pesan (id pesan));
```

```sql
CREATE TABLE kuitansi (
id_kuitansi int(5) NOT NULL auto_increment,
id_faktur int(5) NOT NULL,
tgl_kuitansi date NOT NULL,
PRIMARY KEY  (id_kuitansi),
KEY FK_kuitansi (id_faktur),
CONSTRAINT FK_kuitansi FOREIGN KEY (id_faktur) REFERENCES faktur (id_faktur));
```

```sql
CREATE TABLE produk (
id_produk varchar(5) NOT NULL,
nm_produk varchar(30) NOT NULL,
satuan varchar(10) NOT NULL,
harga decimal(10,0) NOT NULL default '0',
stock int(3) NOT NULL default '0',
PRIMARY KEY  (id_produk));
```

```sql
CREATE TABLE detil_pesan (
id_pesan int(5) NOT NULL,
id_produk varchar(5) NOT NULL,
jumlah int(5) NOT NULL default '0',
harga decimal(10,0) NOT NULL default '0',
PRIMARY KEY  (id_pesan,id_produk),
KEY FK_detil_pesan (id_produk),
KEY id_pesan (id_pesan),
CONSTRAINT FK_detil_pesan FOREIGN KEY (id_produk) REFERENCES produk (id_produk),
CONSTRAINT FK_detil_pesan2 FOREIGN KEY (id_pesan) REFERENCES pesan (id_pesan));
```

Perintah SELECT dari Banyak Tabel dengan JOIN

Masing-masing tabel akan berhubungan satu sama lain (memiliki relasi), relasi antar- tabel dapat berupa relasi 1-1, 1-M, atau M-N, sepertihalnya tabel diatas.

Untuk menggabungkan 2 (dua) atau lebih tabel, kita dapat menggunakan perintah JOIN.

1. **Inner Join**

   Tabel akan digabungkan dua arah, sehingga tidak ada data yang NULL di satu sisi.

   Sebagai contoh kita akan mengabungkan 2 tabel yaitu tabel pelanggan dan pesan dengan menampilkan : id_pelanggan, nm_pelanggan, id_pesan dan tgl_pesan.

Cara 1 Penggabungan dengan WHERE

```
SELECT tabel1.*, tabel2.* FROM tabel1, tabel2
WHERE tabel1.PK=tabel2.FK;
```

Tampilkan : id_pelanggan, nm_pelanggan, id_pesan dan tgl_pesan dari pelanggan dan pesan, dengan perintah SQL dibawah ini :

```
1   SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan, pesan.id_pesan, pesan.tgl_pesan
2   FROM pelanggan, pesan
3   WHERE pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Cara 2 Penggabungan dengan WHERE

```
SELECT tabel1.*, tabel2.*
FROM tabel1 INNER JOIN tabel2
ON tabel1.PK=tabel2.FK;
```

```
1   SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan, pesan.id_pesan, pesan.tgl_pesan
2   FROM pelanggan INNER JOIN pesan
3   ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

2. Outer Join

Tabel akan digabungkan satu arah, sehingga memungkinkan ada data yang **NULL** (kosong) di satu sisi. Outer Join terbagi menjadi 2 (dua) yaitu LEFT JOIN dan RIGHT.

LEFT JOIN

```
SELECT tabel1.*, tabel2.*
FROM tabel1 LEFT JOIN tabel2
ON tabel1.PK=tabel2.FK
```

```
1   SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan, pesan.id_pesan, pesan.tgl_pesan
2   FROM pelanggan LEFT JOIN pesan
3   ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

RIGHT JOIN

```
SELECT tabel1.*, tabel2.*
FROM tabel1 RIGHT JOIN tabel2
ON tabel1.PK=tabel2.FK;
```

```
1   SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan, pesan.id_pesan, pesan.tgl_pesan
2   FROM pelanggan RIGHT JOIN pesan
3   ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Analisa perbedaan tampilkan / output dari Left Join dan Right Join

Menggabungkan Tiga Tabel atau lebih

Untuk menggabungkan tiga tabel atau lebih, pada dasarnya sama dengan penggabungan 2 (dua) tabel. Sebagai contoh kita akan menampilkan barang-barang yang dipesan beserta nama barang dan harganya untuk pemesanan.

```
1   SELECT pesan.id_pesan, produk.id_produk, produk.nm_produk, detil_pesan.harga, detil_pesan.jumlah
2   FROM pesan, detil_pesan, produk
3   WHERE pesan.id_pesan=detil_pesan.id_pesan AND
4   detil_pesan.id_produk=produk.id_produk
5   AND pesan.id_pesan='1'
```

Pengelompokkan Hasil Query dengan GROUP BY

Pengelompokkan berdasarkan kriteria atau kondisi tertentu. Misalnya kita akan menampilkan jumlah barang yang dibeli untuk masing-masing transaksi (pemesanan).
Perhatikan perintah query berikut ini dan lihat hasilnya:

```
1   SELECT pesan.id_pesan, pesan.tgl_pesan, detil_pesan.jumlah
2   FROM pesan, detil_pesan
3   WHERE pesan.id_pesan=detil_pesan.id_pesan;
```

```
1   SELECT pesan.id_pesan, pesan.tgl_pesan, SUM(detil_pesan.jumlah) as jumlah
2   FROM pesan, detil_pesan
3   WHERE pesan.id_pesan=detil_pesan.id_pesan
4   GROUP BY id_pesan;
```

Analisa perbedaan tampilkan perintah diatas
Perbanyak latihan dirumah

Terima kasih
Bersambung ke materi selanjutnya...