



## MATERI PERTEMUAN 12

### Konsep Dasar Basis data Dan Tipe Data



#### TAHUKAH KAMU...?

“pengetian database?”  
“file, field, & record?”  
“jenis-jenis database?”  
“database relational?”  
“jenis dan type data?”

#### A. Definisi basis data

**Basis** dapat diartikan sebagai markas atau gudang, tempat bersarang atau berkumpul. **Data** representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan peristiwa, konsep, keadaan, dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya.



Nah, dari kedua pengertian tersebut, maka dapat ditarik kesimpulan bahwa pengertian dari **Basis Data** adalah Kumpulan file / table yang saling berelasi (berhubungan) yang disimpan dalam media penyimpanan elektronik. Dapat dikatakan pengertian lain dari **basis data** adalah koleksi terpadu dari data yang saling berkaitan yang dirancang untuk memenuhi kebutuhan informasi suatu enterprise (dunia usaha). Dari pengertian tersebut dapat diambil kesimpulan pada masing-masing table / file didalam database berfungsi untuk menampung / menyimpan data-data, dimana masing-masing data yang ada pada table / file tersebut saling berhubungan dengan satusama lainnya.

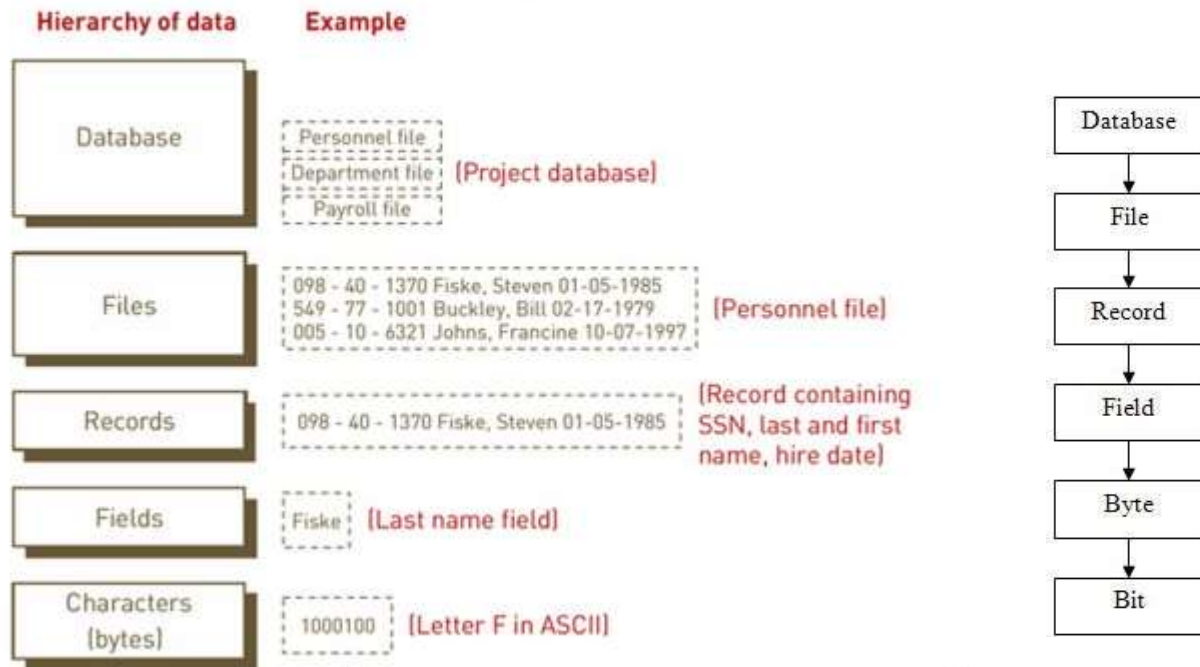
Pemakai (Programmer, User mahir, user umum, user khusus)

- Pengguna akhir / end user : dapat dibagi menjadi 2, yaitu pengguna aplikasi dan pengguna interaktif.
- Pengguna aplikasi adalah orang yang mengoperasikan program aplikasi yang dibuat oleh pemrogram aplikasi sedangkan pengguna interaktif adalah orang yang dapat memberikan perintah-perintah pada antar muka basis data, misalnya SELECT, INSERT, dll.
- Pemrogram aplikasi : Orang yang membuat program aplikasi yang menggunakan basis data.
- Administrator (*database administrator*) / Operator : orang yang bertanggung jawab terhadap pengelolaan basis data.

#### B. Hirarki basis data

Database adalah suatu sistem informasi yang menginterasikan kumpulan dari data yang saling berhubungan satu dengan yang lainnya. Tujuan database adalah untuk menentukan data yang dibutuhkan dalam sistem, sehingga informasinya yang dihasilkan dapat dipenuhi dengan baik. Adapun bentuk dari hirarki sebuah database dapat dilihat dari gambar 1 adalah sebagai berikut.

# The Hierarchy of Data

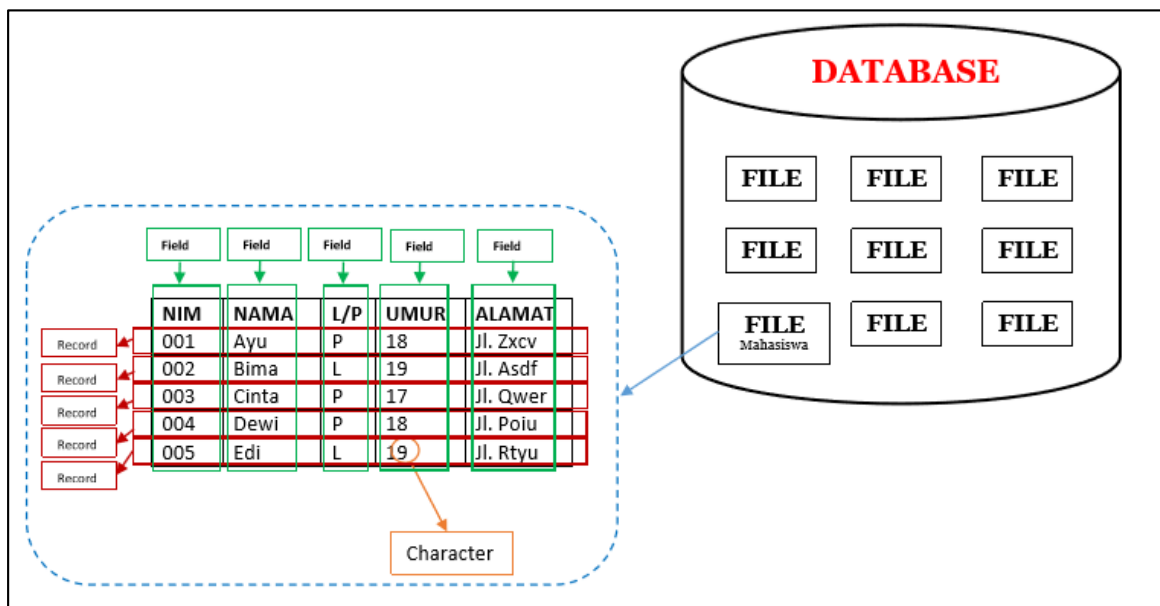
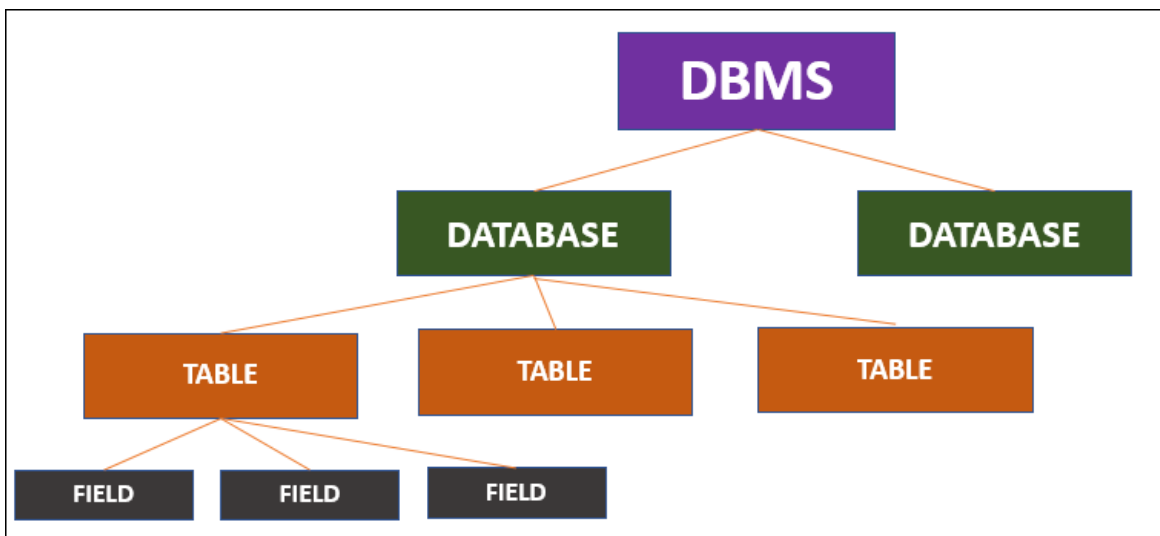


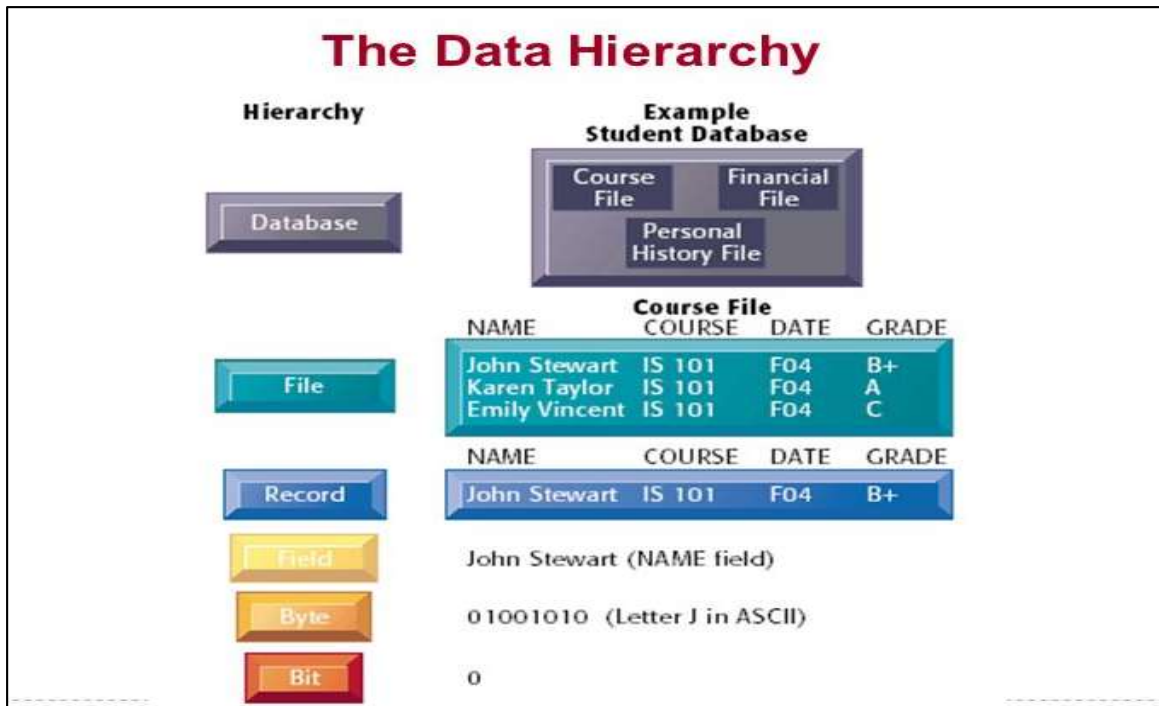
2

Gambar 1. Hirarki data dalam database

Hirarki data dalam database mulai dari yang terbesar ke yang terkecil yaitu :

1. **Database**  
Suatu *database* menggambarkan data yang saling berhubungan antara satu dengan yang lainnya.
2. **File**  
Suatu *file* menggambarkan suatu kesatuan data yang sejenis, dimana kumpulan dari *file* membentuk suatu database.
3. **Record**  
Suatu tuple/*record* menggambarkan suatu unit data individu yang tertentu dimana kumpulandari *record* membentuk suatu file.
4. **Field**  
Suatu *field* menggambarkan suatu *attribute* dari *record*, dimana kumpulan *field* membentuk suatu *record*.
5. **Byte**  
*Attribute* dari *field* berupa huruf yang membentuk nilai dari sebuah field.
6. **Bit**  
Merupakan bagian terkecil dari data secara keseluruhan yaitu berupa karakter ASCII (*American Standar Code Form InformationIntercharge*). 0 (*nol*) adalah satu yang merupakan komponen pembentuk *byte*.





### C. Tipe Database

Dalam dunia IT internasional, tipe atau model *database* memiliki banyak cabang. Hal tersebut dikarenakan bedanya fungsi dan tujuan sistem yang diadopsi pada *database*. Sedangkan saat ini, secara umum database dibagi menjadi 12 tipe. Tipe yang paling terkenal adalah **basis data relasional**. Berikut 12 tipe *database* beserta pengertiannya:

#### 1. Operational Database (NoSQL)

*Operational Database* atau sistem manajemen basis data operasional biasa digunakan untuk memperbarui data secara *real-time*. Tak hanya sekedar melihat datanya saja, tipe basis data ini memungkinkan pengguna untuk memodifikasi data seperti menambah, mengubah, dan menghapus data secara *real-time*. Sejak tahun 1990-an, pasar *software operational database* sebagian besar dikuasai oleh sistem dari SQL (Structured Query Language). Saat ini pasar DBMS operasional berkembang secara drastis dengan banyaknya pendatang baru dan lebih inovatif. Selain itu mereka juga mendukung pertumbuhan penggunaan data tidak terstruktur (NoSQL) dan mesin DBMS NoSQL.

*Database NoSQL* biasanya berfokus pada skalabilitas data. Selain itu, NoSQL meninggalkan “konsistensi data” dengan tidak menyediakan transaksi seperti yang dilakukan pada sistem sebelumnya. Saat ini basis data operasional sangat mendukung tipe lain, seperti arsitektur *distributed database*. Distributed database dapat meningkatkan distribusi dalam menyediakan ketersediaan data yang tinggi dan toleransi kesalahan melalui kemampuan replikasi dan skala. Gartner, Inc telah menerbitkan “*Magic Quadrant*” pada sistem ini di tahun 2013. Dengan itu, ia menjawab permintaan dunia web dalam mengatasi tantangan mengelola data secara “*Big Data*”.

#### 2. Analytical Database

Tipe *analytical database* merupakan *database* yang dapat menyimpan dan mengelola big data, termasuk bisnis, pasar, dan data pelanggan untuk analisis business intelligence (BI). Terdiri dari data dan informasi yang diringkas agar mudah untuk dibaca. Selain itu ia dioptimalkan secara khusus untuk skalabilitas dan *query* yang lebih cepat. *Database* tipe ini sangat dibutuhkan oleh organisasi manajemen dan pengguna-pengguna lainnya.

#### 3. Data Warehouse

Sesuai dengan namanya, *data warehouse (DW)* memiliki tujuan untuk menyimpan data dari waktu ke waktu. Data yang tersimpan merupakan database operasional. Basis data pada *warehouse* bisa menjadi sumber utama dalam mencari informasi yang telah diperiksa, diubah, dan terintegrasi. Basis data ini biasanya digunakan oleh para manajer dan *end-user* lainnya yang memiliki izin untuk mengaksesnya. Melalui perkembangannya, *data warehouse* memberikan fasilitas berupa membagikan data tanpa arsitektur dalam memfasilitasi skala ekstrim.

#### 4. Distributed Database

Tipe selanjutnya adalah *distributed database*, terdiri dari dua berkas atau lebih serta terletak di situs yang berbeda, baik di jaringan yang sama maupun di jaringan yang berbeda sama sekali. Ia memiliki fungsi untuk mengelompokkan setiap departemen melalui fungsi divisi. Pada penerapannya, sistem ini digunakan untuk mendistribusikan *database* melalui *workgroup locale* di kantor regional, kantor cabang, serta lokasi kerja lainnya yang berkaitan. *Database* ini bisa mencakup segmen operasional dan *user database*, dan data yang dihasilkan hanya digunakan oleh pengguna situs itu sendiri.

#### 5. End-User Database

*End-User Database* merupakan basis data yang dikembangkan oleh *end-user* itu sendiri melalui *workstation* mereka. Berbagai jenis berkas data dibuat sendiri dengan menggunakan prosedur tersendiri. Contoh model dari *end-user database* adalah *word processing*, *spreadsheet* hingga *download file*.

#### 6. Real-Time Database

*Real-time database* adalah model data yang sistem pengolahannya dibuat guna menangani beban kerja suatu lembaga besar seperti negara. Pengolahan data tersebut bisa berubah-ubah sesuai dengan permintaan dan bersifat terus-menerus. Seperti berubahnya harga mata uang dollar yang berubah setiap menit. Di balik itu semua, terdapat basis data *real-time* yang bekerja dengan cepat. *Real-time database* biasa digunakan oleh lembaga akuntansi, hukum, perbankan, multimedia, analisis data ilmiah, serta catatan medis.

#### 7. Relational Database

Sistem Manajemen Basis Data Relasional (RDBMS) adalah basisdata yang tabel-tabel didalamnya berisi data yang hubungannya sudah ditentukan sebelumnya. Hubungan setiap data memiliki sifat yang relasional serta tersimpan dalam bentuk kolom dan baris. Kolom dihasilkan dari tabel yang memiliki atribut sedangkan baris dalam tabel mewakili record/catatan.

Dalam penerapannya DBMS menggunakan *Structured Query Language* (SQL) yang memiliki fitur untuk memberi perintah CRUD, yakni Create (mengisi) Read (membaca/mencari) Update (mengubah) Delete (menghapus) beberapa informasi data yang ada di database. *Database* relasional dapat bekerja pada setiap tabel yang memiliki *unique field key* dengan menunjukkan setiap baris. *Field key* tersebut digunakan untuk menghubungkan satu tabel data ke tabel data lainnya. *Database* relasional merupakan tipe *database* yang paling populer dan sering digunakan. Beberapa produk yang biasa digunakan adalah Oracle, SQL Server, MySQL, IBM DB2, dan SQLite.

#### 8. External Database

*External database* adalah basis data yang menyediakan akses untuk pihak eksternal. Seperti sebuah layanan komersial yang meminta *feedback* berupa biaya untuk mengakses sebuah produk jasa. Akses tersebut bertujuan untuk memberikan penghargaan kepada sumber informasi karena telah membuat informasi penting bagi *end-user*.

#### 9. Navigational Database

Melalui basis data navigasi, pengguna bisa menemukan informasi yang dituju dengan memberikan sebuah kata kunci yang sesuai. Biasanya, informasi tersebut berupa objek yang ditemukan melalui beberapa referensi.

#### 10. Hypermedia Database

Basis data tipe ini memungkinkan berbagai laman multimedia yang saling berhubungan dan bisa diakses dengan fitur *hyperlink*. Melalui "*home page*", laman web lainnya dapat dihubungkan dengan cara menambah kode alamat URL sumber lainnya. Media ini bisa berupa teks, gambar, video, musik, grafik, dan sebagainya.

#### 11. In Memory Database

Basis data selanjutnya merupakan basis data memori yang terdapat pada perangkat keras. Basis data ini sangat bergantung pada memori utama dan sistem kerjanya berbeda jika menggunakan *disk* yang berbasis manajemen penyimpanan. Menggunakan memori utama, basis data akan bekerja lebih cepat melalui optimasi algoritma internal sederhana. Hal tersebut dikarenakan penggunaan CPU yang sangat ringan untuk mengeksekusi instruksi. Dalam

implementasinya, basis data ini digunakan untuk jaringan telekomunikasi yang memerlukan pengoperasian cepat secara darurat. Respon memori akan bekerja sesegera mungkin karena dianggap sangat penting.

## 12. Document Oriented Database

Basis data yang berorientasi pada dokumen merupakan bahasa komputer yang dibuat untuk dokumen. Sistem ini tidak menyimpan data dalam format tabel dengan ukuran seragam di setiap rekamannya seperti *relational database*. *Document oriented database* menyimpan setiap catatan sebagai dokumen yang memiliki keunikan khusus. Sejumlah *field* apapun dapat ditambahkan melalui dokumen. *Field* tersebut juga dapat diisi dengan beberapa bagian data. Sekian untuk penjelasan mengenai 12 tipe *database* yang perlu diketahui. Segala bentuk basis data memang perlu diketahui kegunaannya untuk memudahkan pengguna. 12 tipe tersebut juga sangat umum digunakan pada jaringan komputer maupun internet untuk memberikan berbagai macam informasi secara cepat dan mudah.

### D. Relational Database

Relational database adalah kelompok item dalam data dengan hubungan yang sudah ditentukan sebelumnya. Umumnya item ini disusun menjadi tabel yang terdiri dari kolom dan baris. Tabel dipakai untuk menyimpan informasi objek yang direpresentasikan dalam database. Kolom dalam tabel memuat data tertentu dan nilai atribut. Baris pada tabel menunjukkan kumpulan nilai dari satu objek. Tiap baris pada tabel dapat ditandai secara unik atau dalam bentuk ID. Ini kerap disebut kunci utama. Relational database adalah penyajian data yang digunakan untuk membuat kesimpulan atau analisis. Data ini dapat diakses langsung tanpa harus menyusun ulang menjadi tabel baru dalam database.

Dilansir dari Educba, kelebihan ini membuat database relational ini tetap digunakan dan terus dikembangkan dari waktu ke waktu. Berikut adalah beberapa di antaranya:

#### 1. Sederhana

Database ini cukup sederhana dan tidak membutuhkan hierarki data yang sangat kompleks untuk membuatnya. Bahkan, data yang ada bisa dengan mudah ditangani oleh SQL query.

#### 2. Mudah diakses

Tidak ada aturan khusus yang dibuat untuk bisa mengakses semua data yang sudah dibuat dalam bentuk tabel. Dalam relational database siapa pun bisa mengakses dengan mudah dan cepat. Bahkan, mereka bisa melakukan modifikasi seperti menggabungkan tabel atau data terkait dengan lebih cepat.

#### 3. Akurasi data tinggi

Saat menggunakan relational database, kita akan menggunakan kunci utama dan kunci asing. Hadirnya dua kunci untuk melakukan seleksi ini membuat dua data yang berhubungan satu sama lain tidak mengalami duplikasi. Karena dalam relational database apa pun yang sama akan ditampilkan sekali saja. Jadi, akurasi datanya menjadi lebih tinggi.

#### 4. Fleksibilitas

Relational database cenderung lebih fleksibel dalam banyak hal khususnya berhubungan dengan penambahan data. Apabila ada tambahan data yang jumlahnya besar, semua bisa ditampung dan diolah tanpa membebani. Artinya pemegang database bisa dengan mudah melakukan modifikasi pada data baik menambah, mengurangi, atau mengganti.

#### 5. Sistem keamanan tinggi

Biasanya setiap orang yang mengakses database akan memiliki hak akses yang berbeda-beda. Jadi, meski siapa pun bisa masuk, hanya beberapa saja yang bisa melakukan modifikasi. Dengan sistem ini, keamanan database akan selalu terjaga. Karena jika ada perubahan akan diketahui siapa yang melakukan itu.

Tidak semua database memiliki fungsi relational. Untuk mengetahui jenis database apa saja yang memiliki fungsi itu, simak ulasan berikut ini.

- a) MySQL, Salah satu aplikasi relational database open source terbaik yang ada saat ini. Selain itu aplikasi ini juga beberapa lisensi tambahan yang bisa diambil sesuai dengan kebutuhan. MySQL memberikan kemudahan saat input

data dan memiliki performa tinggi. Dikutip dari Phoenix Nap, aplikasi ini cocok untuk kebutuhan pengembangan website dan aplikasi.

- b) MariaDB, Aplikasi ini sebenarnya dibuat dengan base MySQL dengan beberapa penambahan. MariaDB menambahkan ruangan untuk engine dan mengatasi adanya limitasi. Dengan kemampuan ini database menjadi lebih cepat. Salah satu hasil dari relational database yang dibuat dengan MariaDB adalah Google, Mozilla, hingga Wikimedia.
- c) PostgreSQL, Hampir sama dengan MariaDB, aplikasi ini juga merupakan pengembangan meski masih dalam naungan open source. Kelebihan dari database ini adalah performa dan fleksibilitas yang meningkat saat menangani database. PostgreSQL juga bisa melakukan pembacaan data yang cepat untuk melakukan analisis. Produk dari database ini adalah Skype dan Instagram.

## E. Macam-Macam Tipe Data Database

Pada artikel sebelumnya admin sudah membahas tentang pengertian database dan pengenalan mysql dan kali ini kita akan membahas tentang macam-macam type data pada database mysql. Apa itu type data ? Type data adalah jenis nilai yang ditampung pada variabel, nilai ini dapat berupa numerik, (angka), teks, ataupun gambar. Tipe data dalam database digunakan untuk mendefinisikan suatu kolom atau field. Jenis-jenis tipe data bermacam-macam dan secara umum tipe data pada mysql ada empat kelompok yaitu Numeric, String, Date dan Tipe Data Blob.

### 1. Tipe Data Numeric

Type data numerik (Angka) dapat kita gunakan untuk variabel konstanta yang menyimpan nilai berupa angka. berikut ini adalah kelompok tipe data numerik.

Tipe Data	Keterangan	Ukuran
Integer atau Int [(m)]	Bilangan bulat ( Positif / Negatif )	4 byte
	Signed value : -2147683648 to 2147683647	
	Unsigned value : 0 to 4294967295	
Decimal atau Dec (M,D)	Bilangan pecahan ( Positif / Negatif )	M byte
	Bilangan desimal dengan nilai tergantung besaran M dan D	
Tinyint [(m)]	Bilangan bulat ( Positif / Negatif )	1 byte
	Signed value : -128 to 127	
	Unsigned value : 0 to 255	
Smallint [(m)]	Bilangan bulat ( Positif / Negatif )	2 byte
	Signed value : -32768 to 32767	
	Unsigned value : 0 to 65535	
Mediumint [(m)]	Bilangan bulat ( Positif / Negatif )	3 byte
	Signed value : -8388608 to 838860	
	Unsigned value : 0 to 16777215	
Bigint [(m)]	Bilangan bulat ( Positif / Negatif )	8 byte
	Signed value : - 922337203685477808 to 9223372036854775807	
	Unsigned value : 0 to 18446744073709551615	
Float (m,d)	Bilangan pecahan presisi tunggal	4 byte
Double [(m,d)]	Bilangan pecahan presisi ganda	8 byte

### 2 Tipe Data String

Tipe data string dapat kita gunakan untuk menyimpan data yang berupa string ( Text ) atau karakter. berikut kelompok tipe data string.

<b>Type Data</b>	<b>Keterangan</b>
Char atau character	A Fixed-length character string : menyatakan deretan karakter (string) yang lebarnya tetap yaitu maksimum adalah 255 karakter
Varchar	A variable-length character string : Data string dengan lebar data yang bervariasi (M), Maksimum lebar adalah 255 karakter
Text	Teks Dengan Panjang Maksimal 65535
Tinyblob	A very small BLOB (binary large object)
BLOB	A small BLOB
Mediumblob	A medium-sized BLOB
Longblob	A large BLOB

### 3. Tipe Data Date and Time ( Tanggal )

Tipe data date and time digunakan untuk menyimpan data yang berupa tanggal atau waktu. berikut kelompok tipe data Date and Time.

<b>Type Data</b>	<b>Keterangan</b>	<b>Ukuran</b>
Date	Digunakan untuk tanggal dengan format "YYYY-MM-DD" Range nilai : "1000-01-01" s.d "9999-12-31"	3 byte
Time	Digunakan untuk waktu dengan format "hh:mm:ss" Range nilai : - 838:59:59" s.d "838:59:59"	3 byte
Datetime	Digunakan untuk tanggal dan waktu dengan format "YYYY-MM-DD hh:mm:ss" Range nilai : "1000-01-01 00:00:00" s.d "9999-12-31 23:59:59"	8 byte
Time stamp	Digunakan untuk penulisan tanggal dan waktu dengan format "YYYYMMDDhhmmss"	4 byte
Year	Digunakan untuk penulisan tahun dengan format "YYYY" Range nilai : 1901 s.d 2155	1 byte

### 4. Tipe Data Blob

Tipe data blob merupakan tipe data yang dapat digunakan untuk menyimpan data biner yang mampu menampung gambar, video, musik, dan lain-lain. Berikut kelompok dari tipe data blob.

<b>Type Data</b>	<b>Keterangan</b>	<b>Jangkauan</b>
BIT	Menyimpan data biner.	64 digit biner
TINYBLOB	Gambar ukuran kecil	255 byte
BLOB	Gambar	4
MEDIUMBLOB	Gambar ukuran sedang	224-1 byte
LOB	Gambar ukuran besar	232- 1 byte

### 5. Tipe Data Lainnya

- Date: Tanggal
- Datetime: Waktu (Tanggal Dan Jam)
- Time : Jam
- Enum('Nilai1', 'Nilai2', ...): Nilai Enumerasi
- Boolean: Tipe Benar Atau Salah



## ▪ LATIHAN 12

**Kerjakan soal-soal berikut dengan baik dan benar!**

- 1) Jelaskan pengertian basis dan data..!
- 2) Sebutkan dan jelaskan Hirarki data dalam database mulai dari yang terbesar sampai yang terkecil.
- 3) Gambarkan hirarki basis data beserta contohnya..!
- 4) Apa yang dimaksud dengan Relational Database ?
- 5) Sebutkan dan Jelaskan kelebihan menggunakan database relational !
- 6) Sebutkan 3 jenis database yang memiliki fungsi yang bersifat relasional.
- 7) Jelaskan pengertian type data.!
- 8) Jelaskan pengertian Type data numerik ! dan sebutkan contohnya (minimal 3) !
- 9) Jelaskan pengertian Type data string ! dan sebutkan contohnya (minimal 3) !
- 10) Jelaskan pengertian Type data date & time ! dan sebutkan contohnya (minimal 3) !
- 11) Jelaskan pengertian Type data blob ! dan sebutkan contohnya (minimal 3) !
- 12) Jelaskan pengertian Type data lainnya ! dan sebutkan contohnya (minimal 3) !



## MATERI PERTEMUAN 13

### Perancangan Basisdata



#### TAHUKAH KAMU...?

"Cara Instalasi Visio di Windows?"

"Bagaimana menggambarkan ERD dengan Visio?"

#### A. ERD Dan Simbolnya

ERD merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi. ERD berfungsi untuk memodelkan struktur data dan hubungan antar data, untuk menggambarannya digunakan beberapa notasi dan simbol. Fungsi dari penggambaran ERD adalah:

- Untuk memodelkan struktur data dan hubungan antar data
- Model dapat diuji dengan mengabaikan proses yang dilakukan
- Menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi
- Mendokumentasikan data-data yang ada dengan cara mengidentifikasi tiap jenis entitas dan hubungannya.

#### 1. Entitas

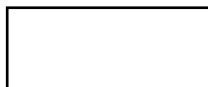
Dalam Sistem Basis Data Entitas ini berupa sekumpulan data yang memiliki suatu informasi yang bermanfaat bagi pungggunanya. Entitas dalam Sistem Basis Data memiliki peran sendiri-sendiri dalam menyampaikan informasi. Entitas memiliki peranan penting dalam Sistem Basis Data, karena jika tidak ada sekumpulan entitas Sistem Basis Data tidak akan terbentuk. Karena Sistem Basis Data terbentuk dari satu data dan dijadikan satu kemudian dihubungkan agar menghasilkan informasi yang jelas bagi pengguna Sistem Basis Data tersebut.

Entitas (entity) adalah sebuah objek yang keberadaannya dapat dibedakan terhadap objek lain. Entitas dapat berupa orang, benda, tempat, kejadian, konsep.

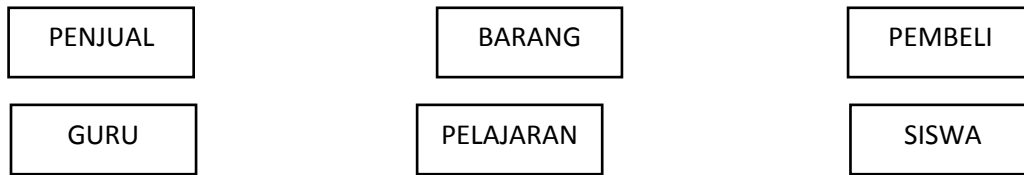
- Orang: Mahasiswa, Dosen, Pemasok, Penjual
- Benda: Mobil, Mesin, Ruangan
- Tempat: Negara, Desa, Kampung
- Kejadian: Penjualan, Registrasi
- Konsep: Rekening, Kursus

Entitas adalah sebuah objek yang keberadaannya dapat dibedakan terhadap objek lain. Entitas dapat berupa orang, benda, tempat, kejadian, konsep. Sebuah entitas memiliki sejumlah atribut. Contoh: mahasiswa memiliki nama dan alamat. Himpunan entitas adalah sekumpulan entitas yang berbagi atribut yang sama. Contoh: sekumpulan mahasiswa, dosen, atau perusahaan.

Lambang :



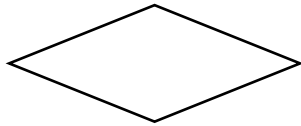
Contoh penerapan :



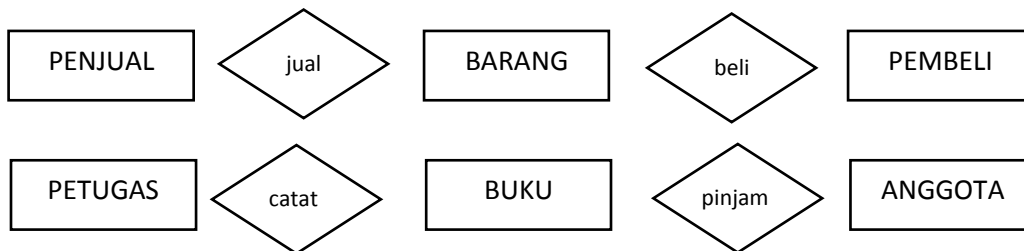
## 2. Relasi

Relasi digunakan untuk menghubungkan beberapa tabel, sehingga data-data yang disimpan dalam tabel tetap normal. Relation atau relasi adalah hubungan antar entitas, berupa kata kerja aktif maupun kata kerja pasif dan dapat dibaca bolak balik. Contoh : Dipinjam atau meminjam, mencatat atau dicatat

Lambang :



Contoh penerapan :

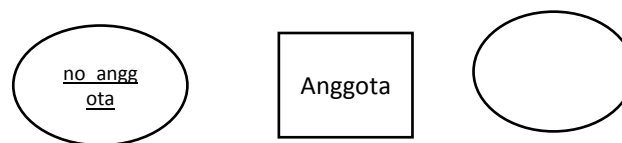


## 3. Atribut

Pada dasarnya Atribut merupakan karakteristik dari Entity atau relationship, yang menyediakan penjelasan detail tentang entity atau relationship tersebut. Atribut merupakan karakteristik dari Entity atau relationship, yang menyediakan penjelasan detail tentang entity atau relation tersebut. Atribut dari sebuah kelas mempresentasikan properti-properti yang dimiliki oleh kelas tersebut.

Setiap Entitas pasti memiliki Atribut yang mendeskripsikan karakteristik dari Entitas tersebut. Penentuan/pemilihan atribut-atribut yang relevan bagi sebuah entitas merupakan hal penting lainnya dalam pembentukan model data. Penetapan atribut bagi sebuah entitas umumnya memang didasarkan pada fakta yang ada, tetapi tidak selalu seperti itu. Istilah atribut sebenarnya identik dengan pemakaian kolom data. Atribut terdiri dari beberapa jenis. Diantaranya adalah:

1. Atribut Key ; Adalah atribut yang digunakan untuk menentukan suatu Entity secara unik dan berbeda

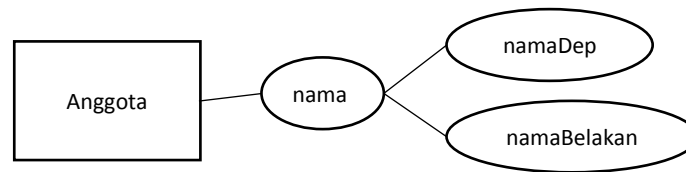


2. Atribut Simple ; Atribut yang hanya memiliki nilai tunggal

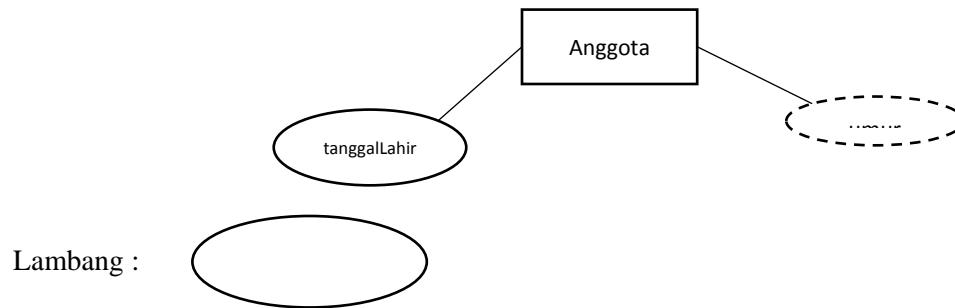


3. Atribut Multivalue ; Atribut yang memiliki sekelompok nilai untuk setiap instant Entity

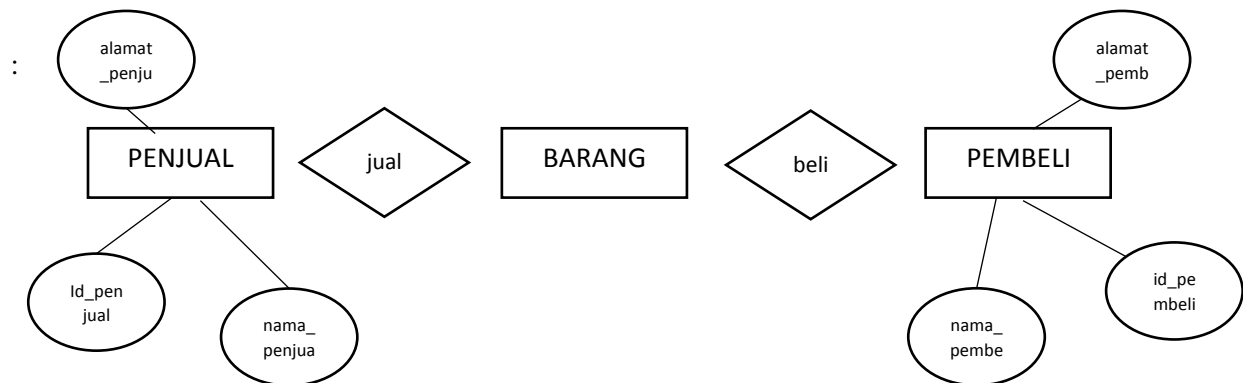
4. Atribut composite ; Suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu. Contoh:



5. Atribut Derivatif ; Merupakan suatu atribut yang berasal atau dihasilkan dari atribut yang lain. Contoh:



Contoh penerapan :



### TAHU KAH KAMU...!

Dalam sebuah ERD terdapat konsep tabel *parent* dan *child* atau **tabel induk** dan **anak**. Sebaiknya dalam menggambar ERD tentukan dahulu tabel utama/parent (induk) yang terdiri atas objek dan subjek. Database yang baik biasanya memiliki 2 entitas subjek utama dan 1 objek utama. Berikut adalah tabel contohnya:

Database	Subjek 1	Subjek 2	Objek
Perpustakaan	Peminjam/anggota	Petugas/admin	Buku
Penjualan/POS	Pembeli/Costumer	Petugas/admin	Barang
Pembelajaran	Siswa	Guru	Pelajaran
Absensi pegawai	Pegawai	Petugas/admin	Absensi

## B. Derajat Relationship

Kardinalitas pemetaan atau rasio kardinalitas menunjukkan jumlah entitas yang dapat dihubungkan ke satu entitas lain dengan suatu relasi. Kardinalitas pemetaan meliputi :

### 1. Hubungan satu ke satu (one to one)

yaitu satu entitas dalam A dihubungkan dengan maksimum satu entitas dalam B (**Simbol 1 : 1**).

*Contoh* : mahasiswa dengan kelas, satu mahasiswa hanya boleh mempunyai satu kelas.

### 2. Hubungan satu ke banyak (one to many)

Yaitu satu entitas dalam A dihubungkan dengan sejumlah entitas dalam B. Satu entitas dalam B dihubungkan dengan maksimum satu entitas dalam A (**Simbol 1 : m**).

*Contoh* : orang tua dengan anak, satu orang tua boleh memiliki banyak anak.

### 3. Hubungan banyak ke satu (many to one)

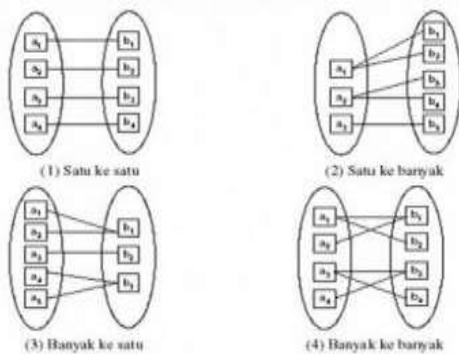
Yaitu satu entitas dalam A dihubungkan dengan maksimum satu entitas dalam B. Satu entitas dalam B dapat dihubungkan dengan sejumlah entitas dalam A (**Simbol m : 1**).

*Contoh* : anak dengan orang tua, sejumlah anak hanya bisa dimiliki satu orang tua.

### 4. Hubungan banyak ke banyak (many to many)

Satu entitas dalam A dihubungkan dengan sejumlah entitas dalam B, dan satu entitas dalam B dihubungkan dengan sejumlah entitas dalam A (**Simbol m : n**).

*Contoh* : matakuliah dengan mahasiswa, banyak mata kuliah dapat dimiliki banyak mahasiswa.



## C. Cara Menggambarkan ERD

Beberapa metode untuk membuat ERD, diantaranya:

### 1. Menentukan entitas

Entitas : objek fisik/non-fisik

Contoh:

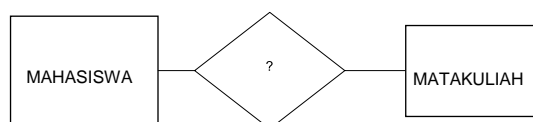
Akademik, mahasiswa, dosen, matakuliah, dll

Perpustakaan: anggota, buku, petugas, dll

Klinik: pasien, dokter, obat, catatan\_medis, dll

### 2. Menentukan relasi

Identifikasi relasi antar entitas



### CONTOH RELASI...!

Database	Subjek 1	Relasi	Objek
Perpustakaan	Peminjam/anggota	Pinjam	Buku
Penjualan/POS	Pembeli/Costumer	Pesan	Barang
Pembelajaran	Siswa	Dapat	Pelajaran
Absensi pegawai	Pegawai	Lakukan	Absensi

#### 3. Gambar ERD sementara

Gambarkan ERD dari relasi-relasi yang telah teridentifikasi



#### 4. Tentukan kardinalitas

Tentukan rasio kardinalitas

Satu ke satu / one to one

Satu ke banyak / one to many

Banyak ke satu / many to one

Banyak ke banyak / many to many

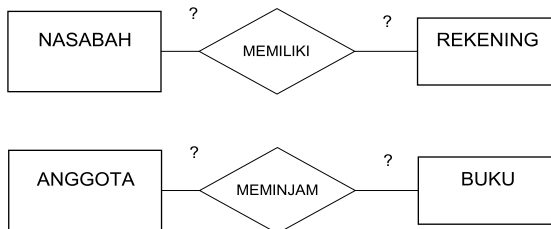
Jenis:

(1:1)

(1:M)

(M:1)

(M:N)



#### 5. Tentukan kunci utama

Identifikasi kunci utama (primary key) dari setiap entitas

Mahasiswa

☐ ?

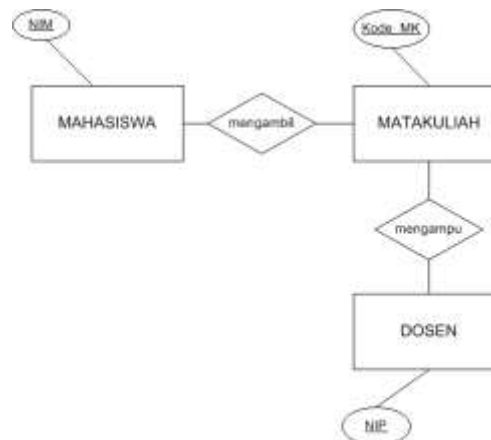
Matakuliah

☐ ?

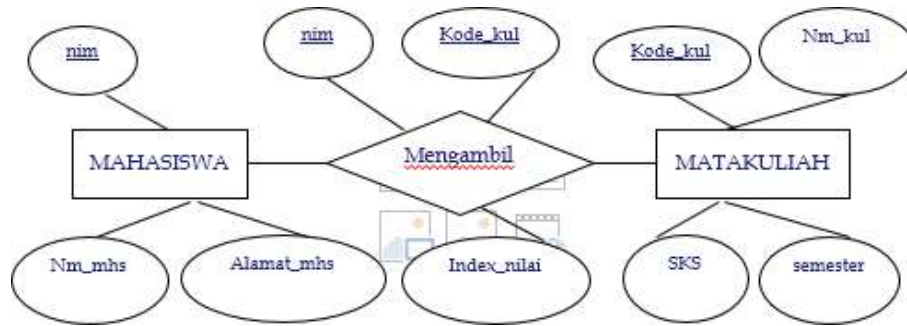
Buku

☐ ?

#### 6. Gambar ERD berdasar kunci



7. Tentukan atribut



Contoh atribut:

- Siswa : NISN, nama, alamat, dll
- Buku : idbuku, judul, harga, halaman, dll
- Admin: id, nama, email, username, password, dll

8. Pemetaan atribut

- Atribut Sederhana dan Komposit.
- Atribut Bernilai Tunggal dan Bernilai Banyak.
- Atribut Tersimpan/Tercatat dan Derivat.
- Atribut Identitas/Pengenal

9. Gambar ERD dengan atributLengkapi atribut

- Atribut Sederhana dan Komposit.
- Atribut Bernilai Tunggal dan Bernilai Banyak.Atribut Tersimpan/Tercatat dan Derivat.
- Atribut Identitas/Pengenal

10. Periksa hasil

Evaluasi hasil pembuatan ER diagram

**CATATAN PENTING**

- Biasakan gunakan **Kata Dasar** untuk relasi. Hal ini karena relasi bersifat *reverse*, yaitu dapat dibaca bolak balik.
- Hindari menggunakan bahasa yang terlalu umum dan tidak spesifik, misal memakai kata “input” untuk relasi bendahara dengan gaji. Tentunya semua data akan diinput ketika kita sudah mempunyai aplikasi. Sehingga kata menginput/diinput sangat tidak tepat untuk dijadikan sebagai nama sebuah relasi.
- Hindari menggunakan kata yang ambigu, misalnya memakai kata “punyai” untuk relasi siswa dan kelas. Hal ini karena semua entitas yang dimasukkan tentunya akan mempunyai/memiliki atribut. Sehingga kata mempunya/memiliki **sangat tidak tepat** untuk dijadikan sebagai nama sebuah relasi.
- Hindari menggunakan nama relasi yang sama pada beberapa relasi. Hal ini akan sangat beresiko jika relasi-relasi tersebut memenuhi syarat untuk dijadikan tabel, karena akan terjadi duplikat nama tabel pada DBMS jika itu dilakukan.

## ■ CONTOH STUDY KASUS

### Studi Kasus : Membuat ER Diagram Perpustakaan

Sistem ini berfungsi untuk mengelola informasi dalam perpustakaan yang meliputi data buku, anggota, petugas dan proses peminjaman di perpustakaan. Berikut adalah aturan-aturan yang perlu diperhatikan saat membuat rancangan *entity relationship diagram*.

- Untuk dapat melakukan peminjaman seorang pengunjung harus menjadi anggota
- Seorang anggota dapat meminjam lebih dari satu buku dalam satu waktu
- Seorang anggota dapat memiliki lebih dari satu nomor telepon
- Sebuah buku bisa lebih dari satu pengarang
- Semua proses pendaftaran buku, anggota dan peminjaman hanya dapat di lakukan/dilayani oleh petugas
- Data buku, anggota dan petugas di simpan dengan id yang unik, sehingga tidak ada id yang sama.

### Menentukan Entitas

Berdasarkan aturan yang telah di definisikan di atas, maka dapat di tentukan bahwa jumlah entitas ada sebanyak 4 yaitu:

1. Buku
2. Anggota
3. Petugas
4. Peminjaman

### Menentukan Atribut

#### 1. Buku

- id\_buku : *integer* (PK)
- judul : *string*
- pengarang : *string*
- kategori : *integer*
- penerbit : *string*
- tahun : *date*

#### 2. Anggota

- id\_anggota : *integer* (PK)
- nama : *string*
- jenis\_kelamin : *integer*
- no\_telp : *string*
- alamat : *string*

#### 3. Petugas

- username : *string* (PK)
- password : *string*
- nama : *string*
- level : *integer*
- no\_petugas : *integer*

#### 4. Peminjaman

- id\_peminjaman : *integer* (PK)
- tgl\_pinjam : *date*
- tgl\_kembali : *date*

### Menentukan Kardinalitas Relasi

#### 1. Dipinjam dalam peminjaman

- Merupakan relasi antar entitas buku dan peminjaman, yang mana memiliki arti bahwa buku dapat di pinjam oleh anggota dan di simpan di dalam pada entitas peminjaman.



- Kardinalitas antar entitas buku dan peminjaman adalah *one to many* karena sebuah buku dapat di pinjam oleh banyak anggota.

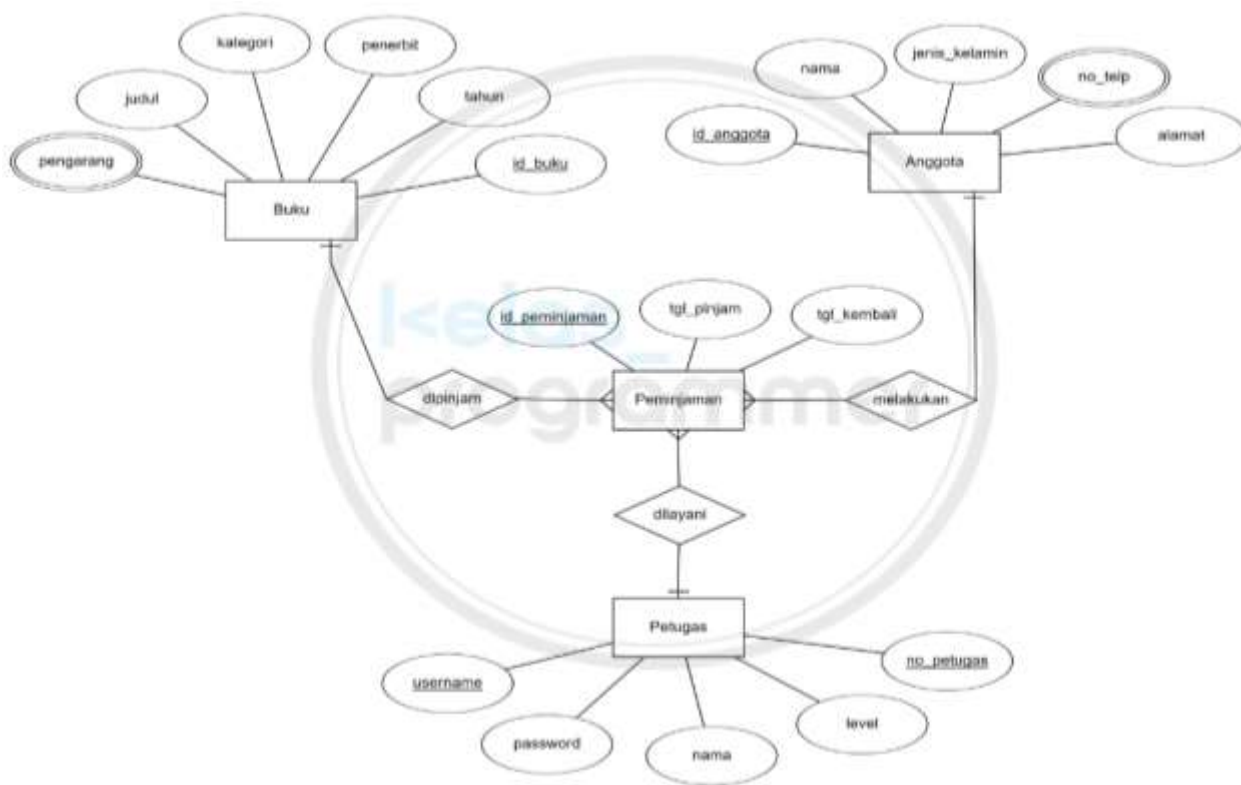
## 2. Melakukan Peminjaman

- Merupakan relasi antar entitas anggota dan peminjaman, di mana anggota yang melakukan peminjaman buku akan di simpan pada entitas peminjaman.
- Kardinalitas antar entitas anggota dan peminjaman adalah *one to many* yang mana satu anggota dapat melakukan peminjaman dalam waktu yang sama pada lebih dari satu buku.

## 3. Peminjaman dilayani

- Merupakan relasi antar entitas petugas dan peminjaman. di mana semua proses peminjaman akan di layani oleh petugas.
- Kardinalitas relasi antar entitas petugas dan peminjaman adalah *one to many* karena satu petugas dapat melayani banyak peminjaman buku.

## Hasil Akhir ER Diagram Perpustakaan



## Studi Kasus : ERD Tentang Penjualan Barang Online

Penjualan merupakan aktifitas menjual produk bisa berupa barang ataupun jasa. Aktifitas ini di lakukan oleh dua pihak yakni penjual dan pembeli. Pada studi kasus kali ini kita akan coba membuat perancangan basis data menggunakan pemodelan ERD dengan aturan-aturan berikut:

- Seorang penjual dapat menjual banyak barang
- Seorang pembeli dapat membeli lebih dari satu barang yang dijual oleh penjual
- Stok barang akan berkurang sesuai jumlah barang yang dibeli
- Nomor telepon penjual boleh lebih dari satu
- Penjual, pembeli dan barang dapat di identifikasi dengan id yang berbeda (unik)

## Langkah-langkah Membuat ERD

1. Menentukan entitas

2. Menentukan atribut termasuk atribut kunci
3. Identifikasi relasi
4. Menentukan kardinalitas

### Menentukan Entitas

Berdasarkan aturan-aturan yang di definisikan di atas dapat kita tentukan jumlah entitas ada sebanyak 3 yakni:

1. Penjual
2. Pembeli
3. Barang

### Menentukan Atribut

Selanjutnya dari ketiga entitas tersebut kita jabarkan atribut-atribut yang melekat pada masing-masing entitas. Atribut yang bersifat unik akan di jadikan sebagai atribut kunci (*primary key*).

#### 1. Penjual

- id\_penjual : integer (PK)
- nama\_penjual : string
- email : string
- no\_telp : string
- alamat : string

#### 2. Pembeli

- id\_pembeli : integer (PK)
- nama\_pembeli : string
- alamat : string
- email : string

#### 3. Barang

- id\_barang : integer (PK)
- nama\_barang : string
- harga : integer
- stok : integer

Atribut dengan kode (PK) akan menjadi atribut kunci (*primary key*) pada masing-masing entitas.

### Menentukan Relasi & Kardinalitasnya

#### 1. Menjual

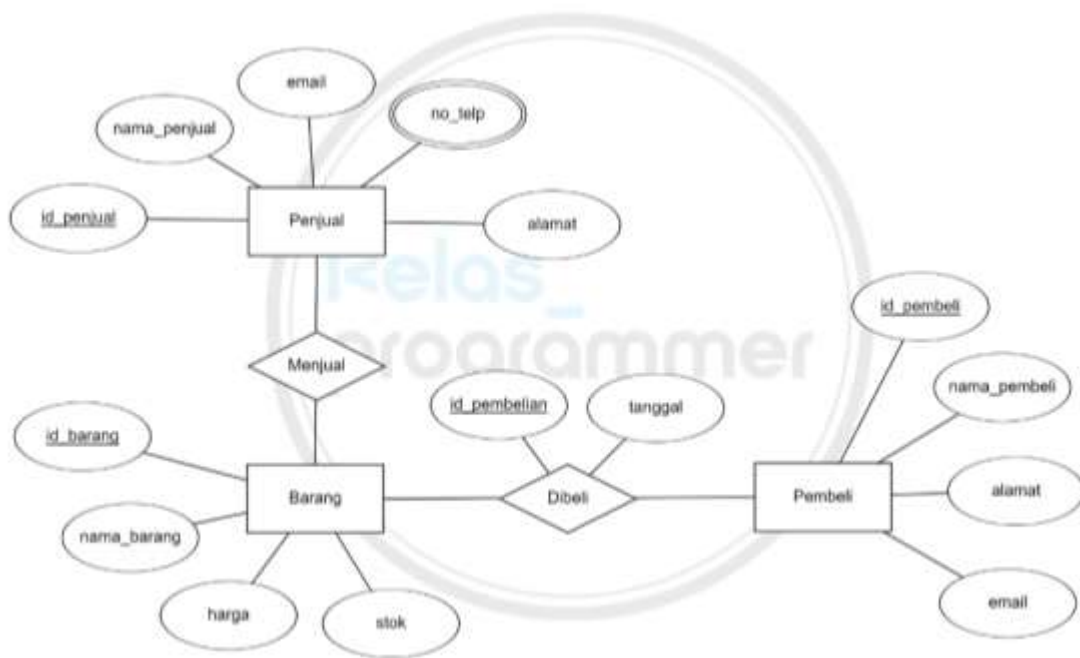
- Merupakan relasi antara entitas penjual dan barang yang berarti setiap penjual boleh menjual barang apapun yang ingin dijual.
- Kardinalitas relasi antara penjual dan barang adalah *one to many* karena seorang penjual dapat menjual banyak barang atau banyak barang dapat dijual oleh seorang penjual.

#### 2. Membeli

- Merupakan relasi antara entitas pembeli dan barang yang berarti pembeli dapat melakukan pembelian barang yang dijual oleh petugas.
- Kardinalitas relasi antara pembeli dan barang adalah *one to many* karena seorang pembeli dapat membeli banyak barang atau dengan kata lain banyak barang boleh di beli oleh seorang pembeli.

### Hasil ERD Penjualan Barang Online

Dari tahap-tahap di atas maka dapat di buat rancangan ERD penjualan barang online adalah sebagai berikut:



Pada entitas Penjualan atribut `no_telp` menggunakan atribut multi valued karena bisa jadi penjual memiliki lebih dari satu nomor telepon.

### Studi Kasus : Sistem Informasi Penggajian Karyawan

Sistem informasi penggajian karyawan adalah sebuah sistem informasi yang dapat mengelola data karyawan/pegawai, berdasarkan golongan dan gaji yang di terima baik untuk gaji pokok dan tunjangan. Aturan penggajian karyawan yang akan di modelkan adalah sebagai berikut:

- Pegawai mendapatkan gaji pokok dan tunjangan
- Tunjangan pegawai di hitung berdasarkan golongan
- Setiap karyawan hanya memiliki satu golongan

### Langkah-langkah membuat ERD

Dari aturan-aturan yang di jabarkan di atas selanjutnya kita boleh membuat diagram ER melalui tahapan-tahapan berikut:

1. Tentukan entitas
2. Menentukan atribut termasuk atribut kunci
3. Identifikasi Relasi
4. Menentukan Kardinalitas

### Menentukan Entitas

Jumlah entitas dalam pemodelan ERD berdasarkan aturan di atas ada sebanyak 3 entitas yaitu:

1. Pegawai
2. Gaji
3. Golongan

### Menentukan Atribut

#### 1. Pegawai

- `id_pegawai` : integer (PK)
- `nama` : string
- `jenis_kelamin` : string
- `telepon` : string

- alamat : string

## 2. Gaji

- id\_gaji : integer (PK)
- gaji pokok : integer
- tunjangan : integer

## 3. Golongan

- id\_golongan : integer (PK)
- jenis\_golongan : string

Atribut dengan kode (PK) akan menjadi atribut kunci pada setiap entitas

## Menentukan Relasi dan Kardinalitasnya

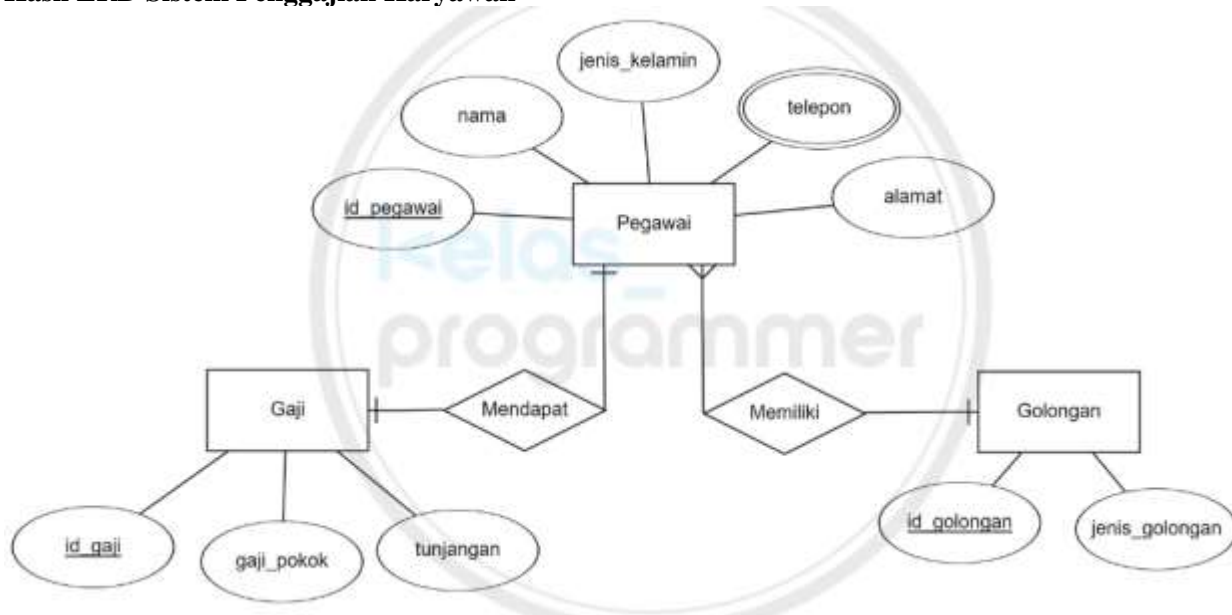
### 1. Pegawai Mendapat Gaji

- Merupakan relasi antar entitas pegawai dan gaji, yang memiliki makna bahwa pegawai mendapat gaji baik gaji pokok maupun tunjangan.
- Kardinalitas relasi antara entitas pegawai dan gaji adalah *one to one* karena seorang karyawan mendapat satu gaji (pokok dan tunjangan) di dalam satu periode.

### 2. Pegawai Memiliki Golongan

- Merupakan relasi antara entitas pegawai dan golongan, dimana setiap pegawai terbagi ke dalam golongan-golongan tertentu.
- Kardinalitas relasi antara entitas pegawai dan golongan adalah *many to one* karena banyak karyawan berada dalam satu golongan atau satu golongan terdiri dari banyak karyawan.

## Hasil ERD Sistem Penggajian Karyawan



## Studi Kasus : ERD Sistem Informasi Pemesanan/reservasi kamar Hotel

Sistem Informasi Pemesanan atau reservasi kamar hotel adalah sebuah sistem informasi yang dapat mengelola data pemesanan kamar hotel yang akan mencatat aktifitas penyewaan di satu hotel misalnya tamu yang memesan kamar, kamar yang tersedia, tipe dan harga kamar dan pencatatan transaksi penyewaan seperti tanggal masuk (*CheckIn*) dan tanggal keluar (*CheckOut*). Berikut ini adalah batasan-batasan yang harus diatasi saat perancangan ERD sistem informasi pemesanan kamar hotel:

1. Tamu dapat memilih tipe kamar dengan harga sesuai tipe tersebut
2. Tamu harus melakukan pemesanan (*booking*) di bagian reservasi

3. Reservasi akan mencatat tanggal masuk (*CheckIn*) dan tanggal keluar (*CheckOut*)
4. Pihak Reservasi akan menyiapkan kamar yang disewa oleh tamu tersebut.

### **Langkah-langkah Membuat ERD**

1. Menentukan entitas yang terlibat
2. Menentukan atribut termasuk atribut kunci
3. Identifikasi relasi
4. Identifikasi kardinalitas relasi

#### **A. Menentukan Entitas**

Berdasarkan aturan di atas, entitas yang dapat diidentifikasi ada sebanyak 3 entitas yaitu:

1. Tamu
2. Kamar
3. Reservasi

#### **B. Menentukan Atribut**

##### **1. Entitas Tamu**

- id\_tamu : integer (PK)
- nama\_tamu : varchar
- alamat : varchar
- no\_telp : varchar
- email : varchar

##### **2. Kamar**

- id\_kamar : integer (PK)
- nama : varchar
- tipe : char
- harga integer

##### **3. Reservasi**

- kode\_reservasi : integer (PK)
- tanggal\_masuk : date
- tanggal\_keluar : date

#### **C. Identifikasi Relasi & Kardinalitas**

##### **1. Memilih**

- Relasi memilih merupakan relasi antara entitas tamu dan kamar yang memiliki arti bahwa tamu dapat memilih tipe/jenis kamar yang tersedia di dalam Hotel tersebut. Kardinalitas relasi antara entitas tamu dan kamar adalah *one to many* karena seorang tamu dapat memilih lebih dari satu kamar sesuai kebutuhan tamu tersebut.

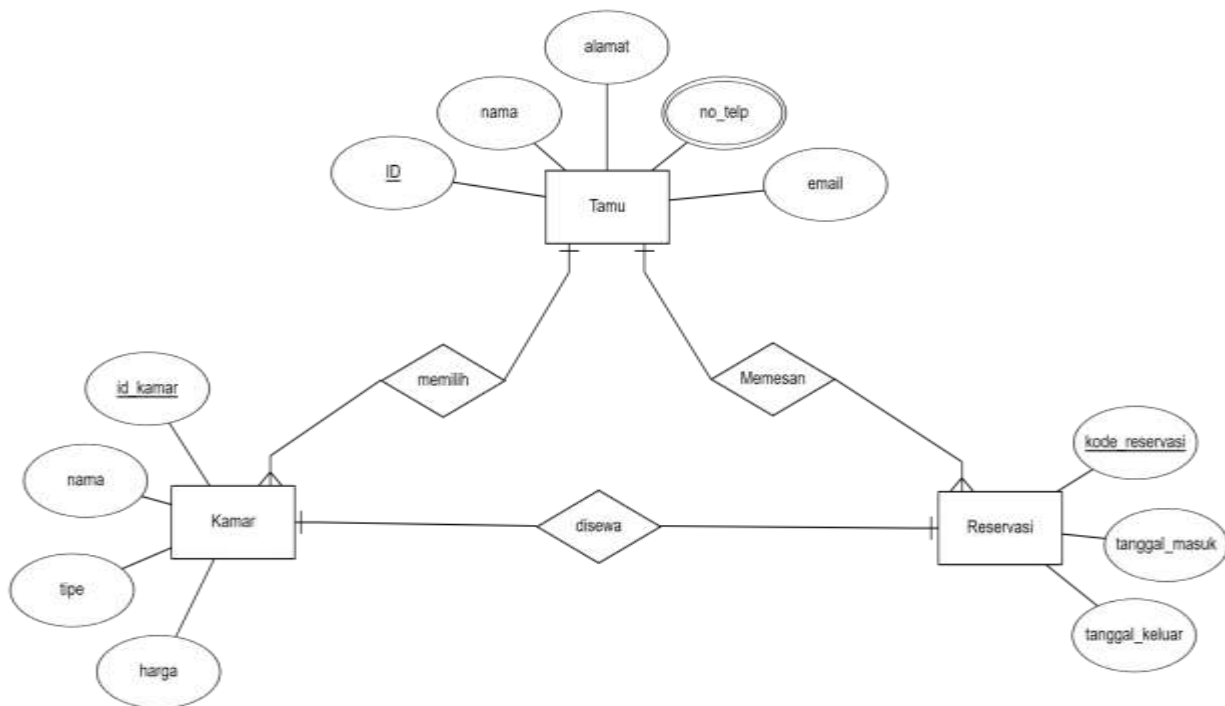
##### **2. Memesan**

- Relasi memesan merupakan relasi antara entitas tamu dan reservasi dimana tamu perlu melakukan pemesanan/*booking* dibagian reservasi.
- Kardinalitas relasi antara entitas tamu dan reservasi adalah *one to many* karena seorang tamu dapat memesan (*booking*) lebih dari satu pesanan kamar hotel.

##### **3. Disewa**

- Relasi Disewa adalah relasi antara entitas reservasi dan kamar dimana kamar yang disewa oleh tamu akan tercatat dibagian reservasi. Bagian reservasi nantinya akan mencatat tanggal masuk (*CheckIn*) dan tanggal keluar (*CheckOut*).
- Kardinalitas relasi antara entitas kamar dan reservasi adalah *one to one* karena satu kamar dalam satu waktu hanya dapat disewa oleh satu orang tamu. Ketika kamar telah disewa maka tamu yang lain tidak dapat memesan kamar tersebut.

## Hasil Akhir ERD Pemesanan Kamar Hotel



### TAHU KAH KAMU...!

Setiap ERD memiliki interaksi antar subjek dan objek yang ada didalamnya. Prosesnya dilakukan setelah kita menentukan judul database, menentukan subjek 1 dan subjek 2, menentukan objek, kemudian menentukan interaksi antara objek dan subjek. Interaksi ini kita buat secara logis sesuai jenis database dan objek/subjek yang ada didalamnya.

Dalam sebuah ERD terdapat konsep tabel *parent* dan *child* atau **tabel induk** dan **anak**. Sebaiknya dalam menggambar ERD tentukan dahulu tabel utama/parent (induk) yang terdiri atas objek dan subjek. Database yang baik biasanya memiliki 2 entitas subjek utama dan 1 objek utama. Berikut adalah tabel contoh database yang memiliki 2 subjek 1 objek.

Database	Subjek 1	Subjek 2	Objek	Interaksi
Perpustakaan	Peminjam	Petugas	Buku	Pinjam
Penjualan/POS	Pembeli	Petugas	Barang	Transaksi
Pembelajaran	Siswa	Guru	Pelajaran	Belajar
Absensi pegawai	Pegawai	Petugas	Absensi	Kehadiran

## D. Normalisasi

Normalisasi adalah proses pembentukan struktur basis data yang lebih sederhana sehingga *redudancy* dan *ambiguity* bisa dihilangkan. Normalisasi merupakan sebuah teknik logical desain dalam sebuah basis data yang mengelompokkan atribut dari berbagai entitas dalam suatu relasi sehingga membentuk struktur relasi yang baik tanpa pengulangan data dan ambiguity. Secara singkat, Normalisasi Database merupakan suatu proses pengelompokan berbagai atribut data yang ada serta membentuknya menjadi suatu entitas yang fleksibel, sederhana, dan mudah beradaptasi.

Tujuan utama normalisasi adalah mengelompokkan atribut ke dalam relasi sehingga data gandanya minimal. Tahap Normalisasi dimulai dari tahap paling ringan (1NF) hingga paling ketat (5NF). Biasanya hanya sampai pada tingkat 3NF atau BCNF karena sudah cukup memadai untuk menghasilkan tabel-tabel yang berkualitas baik. Sebuah tabel dikatakan baik (efisien) atau normal jika memenuhi 3 kriteria sbb:

- Jika ada dekomposisi (penguraian) tabel, maka dekomposisinya harus dijamin aman (*Lossless-Join Decomposition*).
- Terpeliharanya ketergantungan fungsional pada saat perubahan data (Dependency Preservation).
- Tidak melanggar Boyce-Codd Normal Form (BCNF)

Relasi yang memiliki data ganda menimbulkan banyak masalah. Selain boros ruang penyimpanan, ada tiga masalah utama yang muncul, yaitu ketika: (1) menambahkan data baru, (2) mengubah dan (3) menghapus data yang ada. Ketiga masalah ini, biasa disebut anomali *update*, diatasi dengan normalisasi. *Sebagian pengguna basis data, praktisi, penulis, dsb. memaknai kata relasi sebagai hubungan. Secara semantik mungkin benar, namun kurang tepat kalau berkaitan dengan istilah dalam basis data relasional.* Permasalahan-Permasalahan Umum Database Tanpa Normalisasi:

Sebagaimana telah disinggung sebelumnya, selain boros ruang penyimpanan, adanya data ganda menimbulkan tiga anomali *update* yaitu (1) anomali penambahan, (2) anomali penghapusan, dan (3) anomali modifikasi.

### a) Anomali Penambahan

Anomali penambahan terjadi ketika ada data baru ditambahkan ke dalam sebuah relasi. (a). Bila ada pegawai baru yang ditempatkan di Cabang C01, maka ke dalam relasi *PegawaiCabang*, selain data pegawai (NIP, Nama, Alamat), data detail cabang C01 (NomorCabang, AlamatCabang, Telepon Cabang) juga harus dimasukkan dengan risiko terjadi ketidakkonsistenan data. Sebaliknya, pada relasi *Pegawai* pada Gambar 1 (b), cukup dimasukkan data pegawai dan nomor cabang C01 tempat ia bekerja tanpa harus memasukkan detail cabang.

Bila ada cabang baru yang belum ada pegawainya, maka ke dalam relasi *PegawaiCabang*, data detail cabang baru (NomorCabang, AlamatCabang, TeleponCabang) dimasukkan dengan atribut pegawai (NIP, Nama, Alamat) dikosongkan. Hal ini tak mungkin dilakukan karena NIP sebagai *primary key* tidak boleh kosong. Berarti data cabang baru tak dapat ditambahkan. Sebaliknya, pada relasi Cabang pada Gambar 1(c), detail cabang baru dapat ditambahkan tanpa masalah.

### b) Anomali Penghapusan

Anomali penghapusan terjadi ketika ada data yang dihapus. Lihat relasi *PegawaiCabang* pada Gambar 1(a). Bila pegawai dengan NIP 19009 dihapus, maka data detail cabang C03 pada relasi *PegawaiCabang* juga ikut terhapus. Sebaliknya, penghapusan pegawai dengan NIP 19009 pada relasi *Pegawai* pada Gambar 1(b) tidak mengakibatkan hilangnya data detail cabang C03.

### c) Anomali Modifikasi

Anomali modifikasi terjadi ketika ada data yang diubah. Lihat relasi *PegawaiCabang* pada Gambar 1(a). Misalkan cabang C03 pindah alamat dan nomor teleponnya pun berubah. Pada relasi *PegawaiCabang*, modifikasi data harus dilakukan secara berulang ke seluruh data cabang C03 yang ada. Hal ini dapat menimbulkan ketidakkonsistenan data. Sebaliknya, modifikasi data pada relasi *Cabang* pada Gambar 1(c) cukup dilakukan sekali.

Dalam siklus basis data proses desain dilakukan setelah tahap analisis kebutuhan data selesai dilakukan. Ada dua pendekatan desain basis data, yaitu:

1. Desain *top-down* menggunakan model *entity-relationship* (ER). Desain dimulai dengan mengidentifikasi entitas, dilanjutkan dengan *relationship* (hubungan) atau interaksi antar entitas dan kardinalitas atau multiplisitas. Setiap entitas — boleh jadi juga *relationship* (atau sebaliknya) — dilengkapi dengan atribut, *primary key* dan *foreign key* (bila ada).
2. Desain *bottom-up* dengan melakukan proses normalisasi. Desain dimulai dengan mengidentifikasi atribut, kemudian mengelompokkannya ke dalam kumpulan data untuk membentuk relasi.

## 1. Proses Normalisasi Bottom-Up

### a) Tabel Universal

Tabel Universal (Universal / Star Table) adalah sebuah tabel yang merangkum semua kelompok data yang saling berhubungan, bukan merupakan tabel yang baik.

	nrp	mhs.nama	mhs.alamat	kodekul	namakul	sks	kodesem	nihuruf	dsn.nama	dsn.alamat
▶	11020001	Abdullah Machrus	Jl. Sinoman 1/ 11 Mojokerto	SP	Software Perkantoran	2 1		A	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020002	Achmad Fajril	Jl. Panglima Sudirman XII / 30	SP	Software Perkantoran	2 1		A	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020003	Achmad Ridho	Geluran RT 13 / 03 Sepanjang S	SP	Software Perkantoran	2 1		E	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020004	Adi Christanto	Jl. Wonorejo IV / 45 Surabaya	SP	Software Perkantoran	2 1		AB	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020005	Aloysius Rendy	Pucangan VII / 9 Surabaya	SP	Software Perkantoran	2 1		D	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020006	Anita Rachmawati	Perum Canda Bhirawa Asri N - 1	SP	Software Perkantoran	2 1		E	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020007	Arif Fachrudin	Jl. Gubernur Suryo No. 15	SP	Software Perkantoran	2 1		E	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020008	Arohman Agung	Kupang Gunung Timur IV / 24 A	SP	Software Perkantoran	2 1		C	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby

Kelemahan:

- ✓ Pengulangan informasi
- ✓ Potensi inkonsistensi data pada operasi pengubahan
- ✓ Tersembunyinya informasi tertentu
- ✓ Tidak normal

Tujuan **normalisasi** adalah menyempurnakan struktur tabel menjadi lebih baik. Bentuk normalisasi yang sering digunakan adalah 1<sup>st</sup> NF, 2<sup>nd</sup> NF, 3<sup>rd</sup> NF, dan BCNF.

### b) Tahapan Normalisasi

Sebagai contoh kasus proses normalisasi, lihat contoh faktur belanja pada Gambar 2. Tabel pada Gambar 3 yang berisi data faktur belanja merupakan tabel yang belum normal (UNF, *unnormalized form*) karena belum memenuhi karakteristik sebuah relasi. Salah satu karakteristiknya menyatakan bahwa setiap sel — yakni perpotongan baris dan kolom harus bernilai tunggal. Sel pada kolom *JumlahBarang*, *Satuan*, *KodeBarang*, *NamaBarang*, *HargaSatuan*, *JumlahHarga* memiliki dua nilai.

TeraSoft Jl. Gajah 29 Palembang				30 Juni 2021 Kepada Fulan Jl. ABC 7 No. HP 0812345	
No. Faktur: 035					
Jumlah Barang	Satuan	Kode Barang	Nama Barang	Harga Satuan	Jumlah Harga
10	keping	DVD17	DVD blank	2000	20000
2	buah	KBD09	Keyboard	35000	70000
				Total	90000

Gambar 2 Faktur belanja



Tgl	Nama	Alamat	NoHP	No Faktur	Jumlah Barang	Satuan	Kode Barang	Nama Barang	Harga Satuan	Jumlah Harga	Total
1/5/21	Fulan	Jl.ABC 7	0812345	035	10 2	keping buah	DVD17 KBD09	DVD blank Keyboard	2000 35000	20000 70000	90000

**Gambar 3** Tabel UNF (*unnormalized form* ‘bentuk tidak normal’)

Tgl	Nama	Alamat	NoHP	No Faktur	Jumlah Barang	Satuan	Kode Barang	Nama Barang	Harga Satuan	Jumlah Harga	Total
1/5/21	Fulan	Jl.ABC 7	0812345	035	10	keping	DVD17	DVD blank	2000	20000	90000
1/5/21	Fulan	Jl.ABC 7	0812345	035	2	buah	KBD09	Keyboard	35000	70000	90000

**Gambar 4** Relasi 1NF (*first normal form* ‘bentuk normal pertama’)

### 1. 1NF (First Normal Form ‘Bentuk Normal Pertama’)

Relasi yang memenuhi 1NF adalah relasi yang setiap perpotongan baris dan kolomnya berisi satu dan hanya satu nilai. Supaya bernilai tunggal, tabel UNF pada Gambar 3 dimodifikasi dengan memisahkan nilai sel yang tidak tunggal ke baris baru sehingga diperoleh relasi seperti pada Gambar 4. Perhatikan bahwa setiap perpotongan baris dan kolomnya berisi satu dan hanya satu nilai. Ini berarti relasi tersebut telah memenuhi syarat 1NF. Dengan demikian, dari faktur belanja diperoleh sebuah relasi 1NF yang skema relasinya sebagai berikut:

**FakturBelanja** (Tgl, Nama, Alamat, NoHP, NoFaktur, Jumlah, Satuan, KodeBarang, NamaBarang, HargaSatuan, Jumlah Harga, Total)

Dua atribut terakhir adalah atribut turunan dan boleh diabaikan (boleh tidak disimpan dalam basis data). Skema relasinya menjadi sebagai berikut:

**FakturBelanja** (Tgl, Nama, Alamat, NoTelepon, NoFaktur, Jumlah, Satuan, KodeBarang, NamaBarang, HargaSatuan)

Mengubah bentuk unnormalized ke 1NF tidak diperbolehkan adanya :

- ✓ Atribut yang bernilai banyak (*multivalue*).
- ✓ Atribut komposit atau kombinasi dari keduanya.
- ✓ Harga domain atribut harus merupakan harga atomik

Contoh :

- ✓ Data Mahasiswa sbb :

NIM	Nama	Hobi1	Hobi2	Hobi3
12020001	Heri Susanto	Sepak bola	membaca komik	berenang
12020013	Siti Zulaiha	Memasak	menyanyi	
12020015	Dini Susanti	Menjahit	membuat roti	

- ✓ Tabel-tabel di atas tidak memenuhi syarat 1NF
- ✓ Dekomposisi menjadi :

- ✓ Tabel Mahasiswa

NIM	Nama
12020001	Heri Susanto
12020013	Siti Zulaiha
12020015	Dini Susanti

- ✓ Tabel Hobi

NIM	Hobi
12020001	Sepak bola
12020001	membaca komik
12020001	berenang
12020013	Memasak
12020013	menyanyi
12020015	Menjahit
12020015	membuat roti

## 2. 2NF (Second Normal Form ‘Bentuk Normal Kedua’)

Relasi 2NF adalah relasi yang memenuhi 1NF dan setiap atribut bukan *primary key* memiliki ketergantungan fungsional penuh pada *primary key*. Jadi, ada dua hal yang berkaitan dengan relasi 2NF, yaitu *primary key* dan ketergantungan fungsional.

### Ketergantungan Fungsional

Jika A dan B adalah atribut relasi R, maka B dinyatakan memiliki ketergantungan fungsional pada A (ditulis  $A \rightarrow B$ ), jika setiap nilai A berasosiasi tepat dengan satu nilai B. A dan B boleh jadi terdiri atas satu atau beberapa atribut. Determinan ketergantungan fungsional adalah atribut atau sekumpulan atribut yang ada pada sisi kiri anak panah.

Nama mahasiswa memiliki ketergantungan fungsional pada NIM,  $NIM \rightarrow \text{Nama}$ , karena untuk satu nilai NIM hanya diperoleh satu nama. Tetapi tidak sebaliknya. NIM tidak memiliki ketergantungan fungsional pada nama karena untuk satu nama boleh jadi dimiliki oleh beberapa orang mahasiswa sehingga diperoleh beberapa NIM.

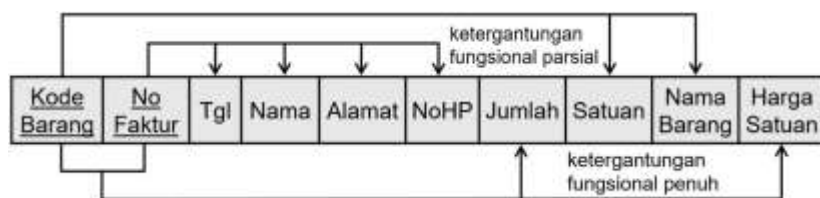
### Ketergantungan Fungsional Penuh

Bila A dan B adalah atribut sebuah relasi, maka B dinyatakan memiliki ketergantungan fungsional penuh pada A jika B bergantung fungsional pada A secara keseluruhan dan bukan pada sebagian dari A. A adalah *primary key* atribut tunggal atau *primary key* komposit yang merupakan gabungan dari beberapa atribut. Karena persyaratan relasi 2NF berkaitan dengan *primary key*, maka harus ditentukan terlebih dahulu *primary key* relasi 1NF. Tidak satu pun atribut tunggal memenuhi syarat sebagai *primary key*. Oleh karena itu, diambil gabungan dua atribut yaitu *NoFaktur* + *KodeBarang* sebagai *primary key* komposit sehingga skema relasinya menjadi:

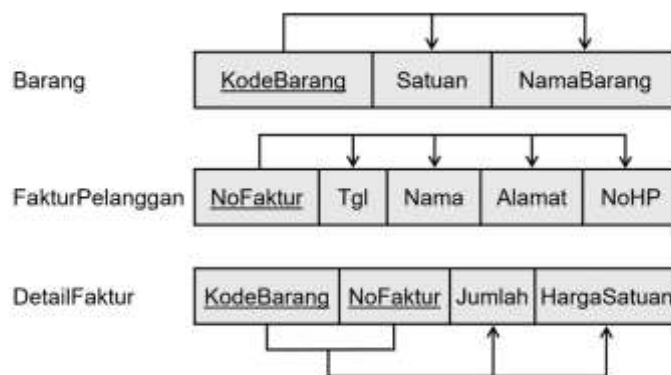
**FakturBelanja** (NoFaktur, KodeBarang, Tgl, Nama, Alamat, NoHP, Jumlah, Satuan, NamaBarang, HargaSatuan)

Selanjutnya, setiap atribut bukan *primary key* dicek apakah memiliki ketergantungan fungsional penuh pada keseluruhan atribut *primary key* atau hanya pada sebagian *primary key* (parsial).

Lihat Gambar 5. Ada dua kelompok atribut yang memiliki ketergantungan fungsional pada sebagian *primary key* (ketergantungan parsial). Pertama, atribut *Satuan*, *NamaBarang*, memiliki ketergantungan fungsional hanya pada *KodeBarang*; kedua, atribut *Tgl*, *Nama*, *Alamat*, dan *NoHP* memiliki ketergantungan fungsional hanya pada *NoFaktur*. Ini berarti relasi *FakturBelanja* memang belum memenuhi 2NF.



Gambar 5 Diagram ketergantungan fungsional pada relasi 1NF FakturBelanja



**Gambar 6** Relasi 2NF

Kelompok atribut yang memiliki ketergantungan fungsional parsial dipisahkan ke dalam relasi berbeda sehingga dari relasi 1NF *FakturBelanja* diperoleh tiga buah relasi 2NF yaitu *Barang*, *FakturPelanggan* dan *DetailFaktur*. Lihat Gambar 6. Semua atribut yang bukan *primary key* pada ketiga relasi memiliki ketergantungan fungsional penuh pada *primary key*.

Bentuk normal 2NF terpenuhi dalam sebuah tabel jika telah memenuhi bentuk 1NF, dan semua atribut selain *primary key*, secara utuh memiliki FD pada *primary key*. Sebuah tabel tidak memenuhi 2NF, jika ada atribut yang ketergantungannya (FD) hanya bersifat parsial saja (hanya tergantung pada sebagian dari *primary key*). Jika terdapat atribut yang tidak memiliki ketergantungan terhadap *primary key*, maka atribut tersebut harus dipindah atau dihilangkan. Ketergantungan fungsional  $X \twoheadrightarrow Y$  dikatakan **penuh** jika menghapus suatu atribut A dari X berarti Y tidak lagi bergantung fungsional. Ketergantungan fungsional  $X \twoheadrightarrow Y$  dikatakan **partial** jika menghapus suatu atribut A dari X berarti Y masih bergantung fungsional.

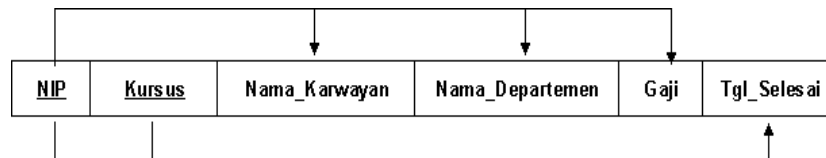
Contoh :

- Tabel berikut ini memenuhi 1NF, tetapi tidak termasuk 2NF

NIM	NamaMhs	Alamat	KodeMk	Matakuliah	sks	NilaiHuruf
980001	Ali Akbar	Jl Ahmad Yani	SD	Struktur Data	2	A
980004	Indah Susanti	Jl Tanjungpura	SD	Struktur Data	2	B
980001	Ali Akbar	Jl Ahmad Yani	BD	Basis Data	3	
980002	Budi Haryanto	Jl Purnama	BD	Basis Data	3	
980004	Indah Susanti	Jl Tanjungpura	BD	Basis Data	3	
980001	Ali Akbar	Jl Ahmad Yani	BI	Bahasa Indonesia	2	B
980003	Ali Akbar	Jl Imam Bonjol	BI	Bahasa Indonesia	2	B
980002	Budi Haryanto	Jl Ahmad Yani	M1	Matematika 1	2	C
980003	Ali Akbar	Jl Imam Bonjol	M1	Matematika 1	2	A

- Tidak memenuhi 2NF, karena {NIM, KodeMk} yang dianggap sebagai *primary key* sedangkan:
  - {NIM, KodeMk}  $\twoheadrightarrow$  NamaMhs
  - {NIM, KodeMk}  $\twoheadrightarrow$  Alamat
  - {NIM, KodeMk}  $\twoheadrightarrow$  Matakuliah
  - {NIM, KodeMk}  $\twoheadrightarrow$  Sks
  - {NIM, KodeMk}  $\twoheadrightarrow$  NilaiHuruf
- Tabel tersebut perlu didekomposisi menjadi beberapa tabel yang memenuhi syarat 2NF
- Functional dependency-nya sbb:
  - {NIM, KodeMk}  $\twoheadrightarrow$  NilaiHuruf (fd1)
  - NIM  $\twoheadrightarrow$  {NamaMhs, Alamat} (fd2)
  - KodeMk  $\twoheadrightarrow$  {Matakuliah, Sks} (fd3)

- Sehingga :  
 fd1 (NIM, KodeMk, NilaiHuruf) □ Tabel Nilai  
 fd2 (NIM, NamaMhs, Alamat) □ Tabel Mahasiswa fd3 (KodeMk, Matakuliah, Sks) □ Tabel Matakuliah
- Suatu format tabel Normal I (1NF) : (menghilangkan Redundansi)



✓ Bentuk Normal II (2NF) : (Decompose)



### 3. 3NF (Third Normal Form ‘Bentuk Normal Ketiga’)

Relasi 3NF adalah relasi yang memenuhi 1NF, 2NF dan atribut yang bukan *primary key* tidak memiliki ketergantungan transitif pada *primary key*. *Ketergantungan Transitif* Misal  $A, B, C$  adalah atribut sebuah relasi. Jika  $A \rightarrow B$  dan  $B \rightarrow C$ , maka dikatakan  $C$  memiliki ketergantungan transitif pada  $A$  melalui  $B$ . Dengan kata lain, ketergantungan transitif adalah ketergantungan fungsional di antara atribut bukan *primary key*. Atribut bukan *primary key* setiap relasi 2NF harus dicek apakah memiliki ketergantungan transitif. Lihat Gambar 6. Skema ketiga relasi adalah sebagai berikut.

- **Barang** (KodeBarang, Satuan, NamaBarang)
- **DetailFaktur** (KodeBarang, NoFaktur, Jumlah, HargaSatuan)
- **FakturPelanggan** (NoFaktur, Tgl, Nama, Alamat, NoHP)

Pada relasi *Barang*, tidak ada ketergantungan antara atribut *Satuan* dengan *NamaBarang*. Begitu pula pada relasi *DetailFaktur*, tidak ada ketergantungan antara atribut *Jumlah* dengan *HargaSatuan*. Ini berarti keduanya sudah memenuhi 3NF karena tidak memiliki ketergantungan transitif. Pada relasi *FakturPelanggan*, atribut *NoHP* bernilai unik, ada ketergantungan transitif *Nama* dan *Alamat* pada *NoFaktur* melalui *NoHP* sebagai berikut:

$NoFaktur \rightarrow NoHP$   $NoHP \rightarrow Nama, Alamat$ . Dengan demikian, relasi *FakturPelanggan* belum memenuhi 3NF. Lihat Gambar 7.

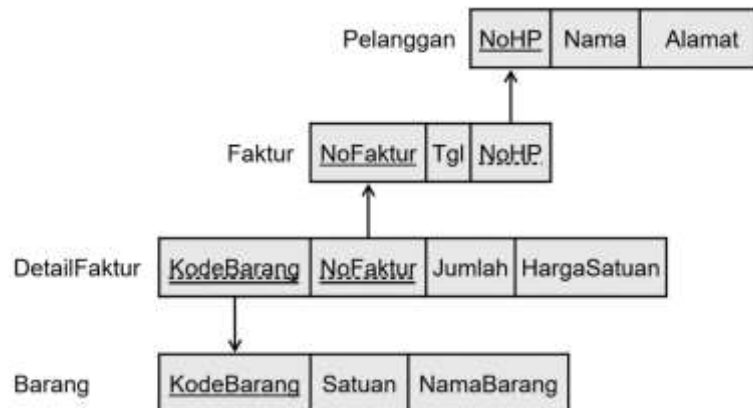


Gambar 7 Ketergantungan transitif



**Gambar 8** Relasi 3NF

Selanjutnya, kelompok atribut yang memiliki ketergantungan transitif dipisahkan ke dalam relasi berbeda sehingga dari relasi 2NF *FakturPelanggan* diperoleh dua buah relasi 3NF yaitu relasi *Faktur* dan relasi *Pelanggan*. Lihat Gambar 8.



**Gambar 9** Hasil normalisasi sampai dengan 3NF

Jadi, hasil akhir proses normalisasi data faktur belanja adalah empat buah relasi 3NF yaitu relasi *Pelanggan*, *Faktur*, *DetailFaktur* dan *Barang*. Lihat Gambar 9. Keterkaitan antar relasi dapat dilihat dari atribut *foreign key* yang digarisbawahi dengan garis terputus-putus merujuk ke atribut *primary key* yang digarisbawahi dengan garis menerus.

Bentuk normal 3NF terpenuhi jika telah memenuhi bentuk 2NF, dan jika tidak ada atribut *non primary key* yang memiliki ketergantungan terhadap atribut *non primarykey* yang lainnya (ketergantungan transitif).

Contoh :

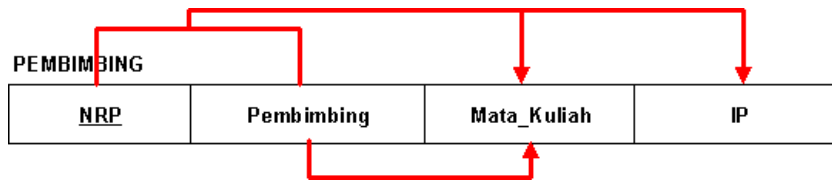
- ✓ Tabel mahasiswa berikut ini memenuhi syarat 2NF, tetapi tidak memenuhi 3NF
- ✓ Karena masih terdapat atribut *non primary key* (yakni Kota dan Provinsi) yang memiliki ketergantungan terhadap atribut *non primary key* yang lain (yakni KodePos) :  
KodePos □ {Kota, Provinsi}
- ✓ Sehingga tabel tersebut perlu didekomposisi menjadi :
  - Mahasiswa (NIM, NamaMhs, Jalan, KodePos)
  - KodePos (KodePos, Provinsi, Kota)

#### 4. BCNF Boyce-Code Normal Form

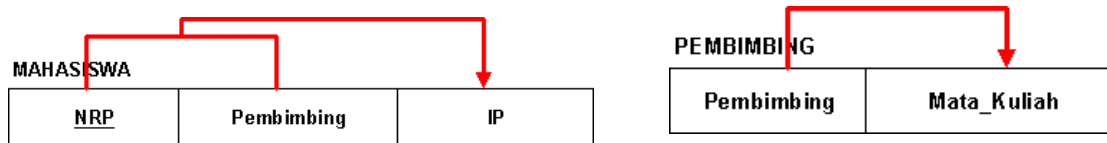
Secara praktis, tujuan rancangan database adalah cukup sampai pada 3NF. Akan tetapi untuk kasus-kasus tertentu kita bisa mendapatkan rancangan yang lebih baik lagi apabila bisa mencapai ke BCNF. Suatu relasi R dikatakan dalam bentuk BCNF: jika dan hanya jika setiap AtributKunci (Key) pada suatu relasi adalah KunciKandidat (CandidateKey). KunciKandidat (CandidateKey) adalah atribut-atribut dari entitas yang mungkin dapat digunakan sebagai kunci (key) atribut. BCNF hampir sama dengan 3NF, dengankata lain setiap BCNF adalah 3NF.

Contoh :

✓ Normal II (2NF) :



✓ Normal III (3NF) atau BCNF



## Contoh Penerapan

### #1 Bentuk Tidak Normal (*unnormalize*)

Bentuk tidak normal (*unnormalized*) merupakan kumpulan data yang direkam tidak ada keharusan dengan mengikuti suatu format tertentu. Pada bentuk tidak normal terdapat *repeating group* (Pengulangan Group), sehingga pada kondisi ini data menjadi permasalahan dalam melakukan manipulasi data (*insert, update, dan delete*) atau biasa disebut anomali.

Contoh:

No. Faktur	Tgl	Kode Pelanggan	Nama	Kode Barang	Nama Barang	Harga	Qty
F-001	12/12/2016	P-001	M Fikri Setiadi	B-001	Sampo	12.000,-	1
				B-002	Kopi	15.000,-	1
F-002	13/12/2016	P-002	Jack	B-002	Kopi	15.000,-	1
				B-003	Teh	7.000,-	2

### #2. Normal Pertama (1 NF)

Dalam *relational database* tidak diperkenankan adanya repeating group karena dapat berdampak terjadinya anomali. Oleh karena itu tahap unnorml akan menghasilkan bentuk normal tahap pertama (1 NF) yang dapat di definisikan sebagai berikut:

- Normal pertama (1 NF), suatu relasi atau tabel memenuhi normal pertama jika dan hanya jika setiap setiap atribut dari relasi tersebut hanya memiliki nilai tunggal dalam satu baris (*record*).
- Tiap *field* hanya satu pengertian, bukan merupakan kumpulan kata yang mempunyai arti ganda dan tidak ada set atribut yang berulang-ulang atau atribut bernilai ganda.
- Pada data tabel sebelumnya data belum normal sehingga harus diubah kedalam bentuk normal pertama dengan cara membuat baris berisi kolom jumlah yang sama dan setiap kolom hanya mengandung satu nilai. Berikut perubahannya:

No. Faktur	Tgl	Kode Pelanggan	Nama	Kode Barang	Nama Barang	Harga	Qty
F-001	12/12/2016	P-001	M Fikri Setiadi	B-001	Sampo	12.000,-	1
F-001	12/12/2016	P-001	M Fikri Setiadi	B-002	Kopi	15.000,-	1
F-002	13/12/2016	P-002	Jack	B-002	Kopi	15.000,-	1
F-002	13/12/2016	P-002	Jack	B-003	Teh	7.000,-	2

Bentuk normalisasi pertama (1 NF) ini mempunyai ciri yaitu setiap data dibentuk file datar atau rata (*flat file*), data dibentuk dalam satu record demi satu record dan nilai-nilai dari field-field berupa nilai yang tidak dapat dibagi-bagi lagi.

### #3. Normal Kedua (2 NF)

Dalam perancangan database relational tidak diperkenankan adalah partial functional dependency kepada primary key, karena dapat berdampak terjadinya anomali. Oleh karena itu tahap normalisasi pertama akan menghasilkan bentuk normal kedua (2 NF) yang dapat didefinisikan sebagai berikut:

Normalisasi kedua (2 NF), suatu relasi memenuhi relasi kedua jika dan hanya jika relasi tersebut memenuhi normal pertama dan setiap atribut yang bukan kunci (*non key*) bergantung secara fungsional terhadap kunci utama (*Primary key*). Berikut perubahannya:

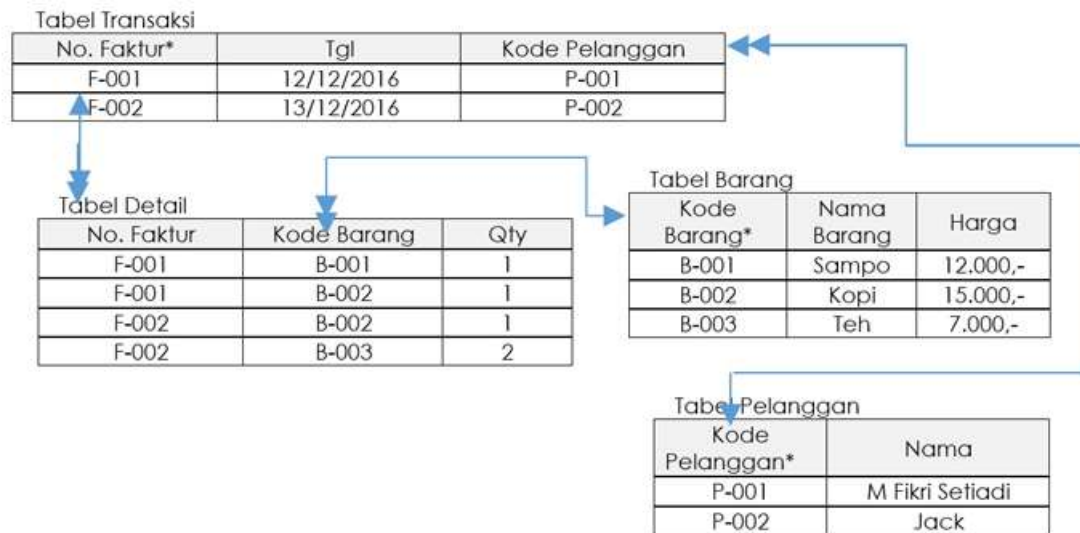


Bentuk normal kedua ini mempunyai syarat yaitu bentuk data yang telah memenuhi kriteria bentuk normal pertama. Atribut bukan kunci haruslah bergantung secara fungsional pada kunci utama (*primary key*), sehingga untuk membentuk normal kedua haruslah sudah ditentukan kunci-kunci field.

### #4. Normal Ketiga (3 NF)

Dalam perancangan database relational tidak diperkenankan adanya transitive dependency karena dapat berdampak terjadinya anomali. Oleh karena itu harus dilakukan normalisasi tahap ketiga (3 NF) yang dapat didefinisikan sebagai berikut:

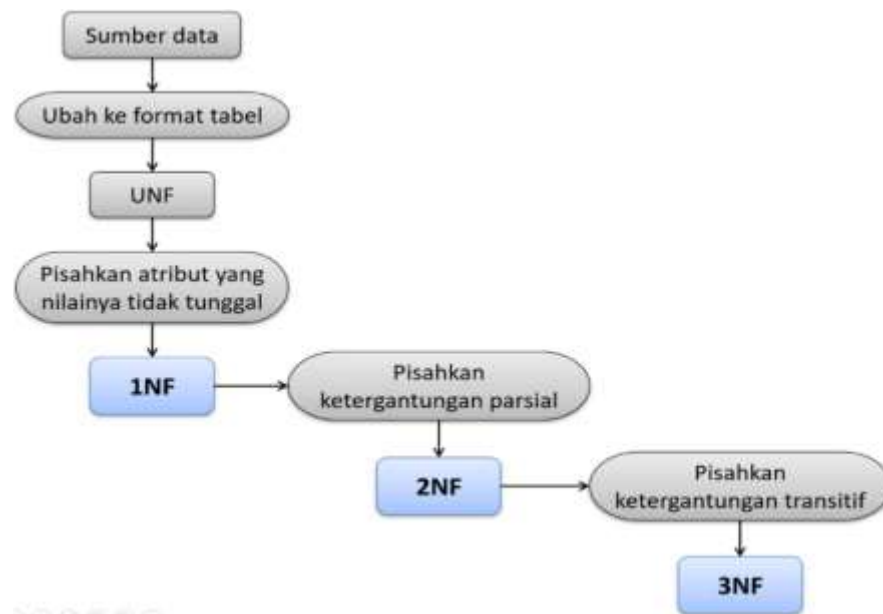
Normalisasi ketiga (3 NF), suatu relasi memenuhi normal ketiga jika dan hanya jika relasi tersebut memenuhi normal kedua dan setiap atribut bukan kunci (*non key*) tidak mempunyai transitive functional dependency kepada kunci utama (*primary key*). Berikut perubahannya:



Bentuk normal ketiga (3 NF) ini relasi haruslah dalam bentuk normal kedua dan semua atribut bukan kunci utama tidak punya hubungan transitif. Artinya setiap atribut bukan kunci harus bergantung hanya pada primary key secara keseluruhan, dan bentuk normalisasi ketiga sudah didapat tabel yang optimal.

## Penutup

Selain boros ruang penyimpanan, relasi yang memiliki data ganda menimbulkan tiga masalah utama, yaitu ketika: (1) menambahkan data baru, (2) mengubah dan (3) menghapus data yang ada. Ketiga masalah ini, biasa disebut anomali *update*.



**Gambar 10** Proses normalisasi

Untuk mengatasi anomali *update*, telah dibahas bagaimana melakukan normalisasi sekurang-kurangnya sampai dengan 3NF. Secara keseluruhan, Gambar 10 menunjukkan urutan prosesnya.



## LAPORAN PRAKTIKUM : NORMALISASI BOTTOM UP

FormCetekPinjam

SAP CRYSTAL REPORTS

Main Report

**PERPUSTAKAAN SMK KOSGOROI KOTA SOLOK**  
**Tanda Bukti Peminjaman Buku**

<b>Kode Pinjam</b> : PJ006	<b>Judul Buku</b> : Akuntasnsi 3B
<b>Kode Anggota</b> : A008	<b>Tgl Pinjam</b> : 5/18/2015
<b>Nama Anggota</b> : Rahma Dewita	<b>Jatuh Tempo</b> : 5/25/2015
<b>Kode Buku</b> : B010	

TTD

\*/ Harap Mengembalikan Tepat Waktu

**ADMINISTRASI**

Current Page No.: 1 Total Page No.: 1 Zoom Factor: 100%

### 1. Ekstraksi/konversi Tabel (unf)

kodeanggota	namaanggota	kodebuku	judulbuku	tglpinjam	jatuhtempo	admin

## 2. Normalisasi Tabel

1nf

**bukti peminjaman buku**

kodepinjam	kodeanggota	namaanggota	kodebuku	judulbuku	tglpinjam	jatuhtempo	admin
pj006	a008	Rahma Dewita	B010	Akuntansi	5/18/2015	5/25/2015	pt01

2nf

**anggota**

kodeanggota	namaanggota
a008	rahma dewita

**pinjam**

kodepinjam	tglpinjam	jatuhtempo	admin
pj006	5/18/2015	5/25/2015	pt01

**buku**

kodebuku	judulbuku
B010	Akuntansi

**admin**

kodepetugas	namapetugas

3nf

**anggota**

kodeanggota	namaanggota
a008	rahma dewita

**pinjam**

kodepinjam	kodepetugas	kodeanggota	tglpinjam	jatuhtempo
pj006		a008	5/18/2015	5/25/2015

**buku**

kodebuku	judulbuku
B010	Akuntansi

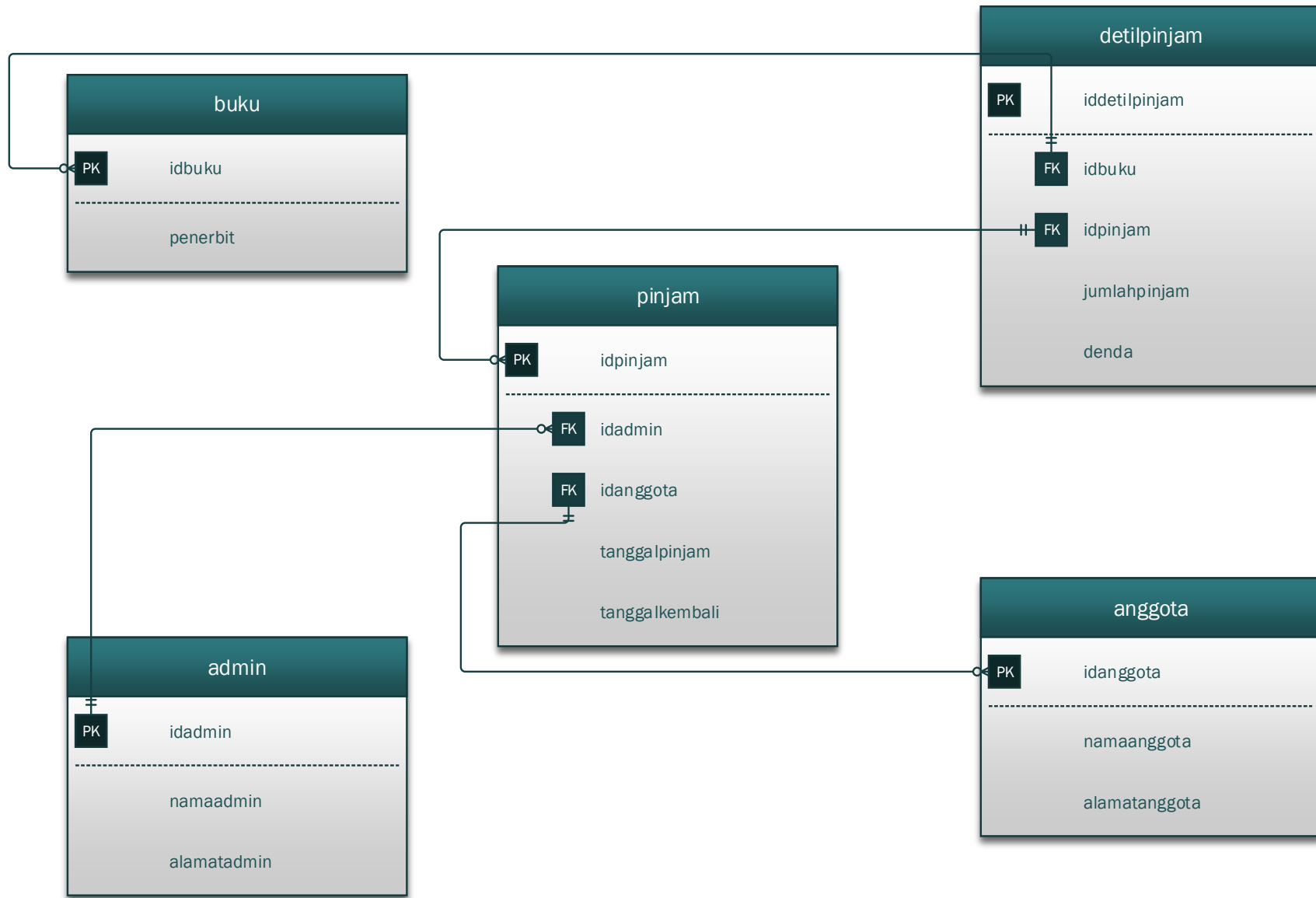
**detil pinjam**

kodepinjam	kodepinjam	kodebuku	jumlahpinjam	terlambat	denda
pj006	pj006	B010			

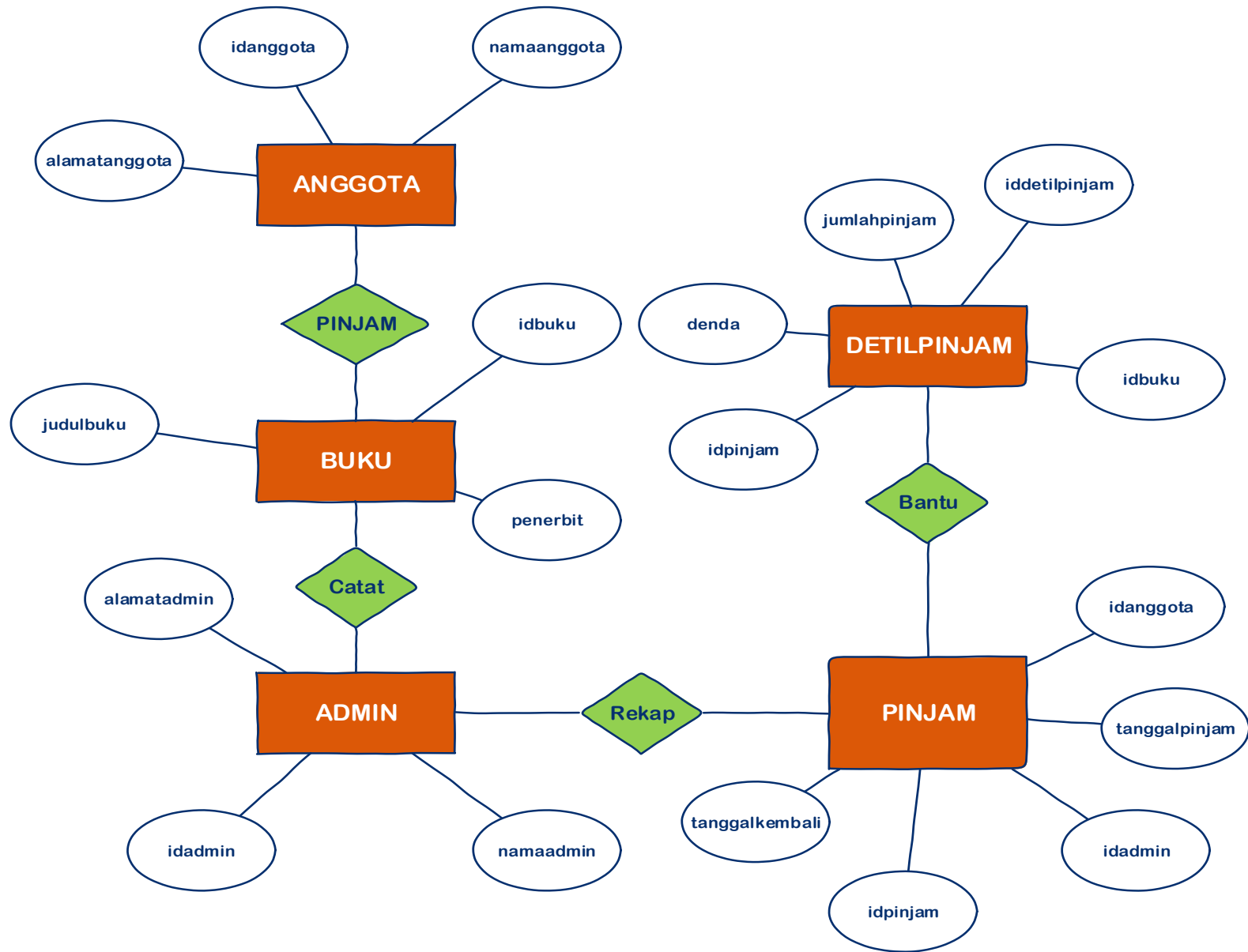
**admin**

kodepetugas	namapetugas

### 3. Membuat ER-Table (ERT)



## 4. Membuat ER-Diagram



## 2. Proses Normalisasi Top-Down

Desain top-down menggunakan model entity-relationship (ER). Desain dimulai dengan mengidentifikasi entitas, dilanjutkan dengan relationship (hubungan) atau interaksi antar entitas dan kardinalitas atau multiplisitas. Setiap entitas boleh jadi juga relationship (atau sebaliknya) dilengkapi dengan atribut, primary key dan foreign key (bila ada). Proses normalisasi dapat dilakukan saat pembuatan ERD. Prosesnya dilakukan setelah kita menentukan judul database, menentukan subjek 1 dan subjek 2, menentukan objek, menentukan interaksi antara objek dan subjek, kemudian kita bisa menentukan pemecahan atau normalisasi dari interaksi yang telah dibuat.

Database	Subjek 1	Subjek 2	Objek	Interaksi	Normalisasi
Perpustakaan	Peminjam	Petugas	Buku	Pinjam	detilPinjam
Penjualan/POS	Pembeli	Petugas	Barang	Transaksi	detilTransaksi
Pembelajaran	Siswa	Guru	Pelajaran	Belajar	detilBelajar
Absensi pegawai	Pegawai	Petugas	Absensi	Kehadiran	detilKehadiran

### ▪ CONTOH KASUS :

## PEMBUATAN SISTEM INFORMASI (APLIKASI) BENGKEL KOMPUTER

### A. Deskripsi

Pada studi kasus kali ini saya akan coba membuat perancangan basis data menggunakan pemodelan ERD dengan aturan-aturan berikut:

- Seorang petugas dapat menjual banyak layanan
- Seorang pembeli dapat membeli lebih dari satu layanan yang dijual oleh petugas
- Seorang petugas dapat merekap banyak transaksi
- Transaksi memiliki detiltransaksi
- Stok layanan akan berkurang sesuai jumlah layanan yang dibeli
- Nomor telepon petugas boleh lebih dari satu
- Petugas, pembeli, layanan dan transaksi dapat diidentifikasi dengan id yang berbeda (unik)

### B. Menentukan Entitas

Berdasarkan aturan-aturan yang di definisikan di atas dapat kita tentukan jumlah entitas ada sebanyak 3 yakni:

1. Petugas
2. Pembeli
3. Layanan
4. Transaksi / penjualan

### C. Menentukan Atribut

Selanjutnya dari ketiga entitas tersebut kita jabarkan atribut-atribut yang melekat pada masing-masing entitas. Atribut yang bersifat unik akan di jadikan sebagai atribut kunci (*primary key*).

## 1. Petugas

- id\_pelanggan int 11
- nama\_pelanggan varchar30
- desa\_pelanggan varchar50
- kec\_pelanggan varchar30
- hp\_pelanggan varchar30

## 2. Pembeli

- id\_petugas int 11 not null primarykey auto increment
- nama\_petugas varchar 30 not null
- desa\_petugas varchar 50 not null
- kec\_petugas varchar 30 not null
- hp\_petugas varchar 30 not null
- jabatan varchar 30 not null

## 3. Layanan

- id\_layanan int 11 not null primarykey auto increment
- nama\_layanan varchar 50 not null
- harga int 11 not null

## 4. Transaksi

- id\_transaksi int 11 not null primarykey auto increment
- id\_pelanggan int 11 not null foreignkey
- id\_petugas int 11 not null foreignkey
- tanggal date not null

## 5. Detiltransaksi

- id\_transaksi int 11 not null foreignkey
- id\_layanan int 11 not null foreignkey
- jumlah int 20 not null
- subtotal int 20 not null

Atribut dengan kode (PK) akan menjadi atribut kunci (primary key) pada masing-masing entitas.

## D. Menentukan Relasi & Kardinalitasnya

### 1. Jual

- Merupakan relasi antara entitas petugas dan layanan yang berarti setiap petugas boleh menjual layanan apapun yang ingin dijual.
- Kardinalitas relasi antara petugas dan layanan adalah *one to many* karena seorang petugas dapat menjual banyak layanan atau banyak layanan dapat dijual oleh seorang petugas.

### 2. Beli

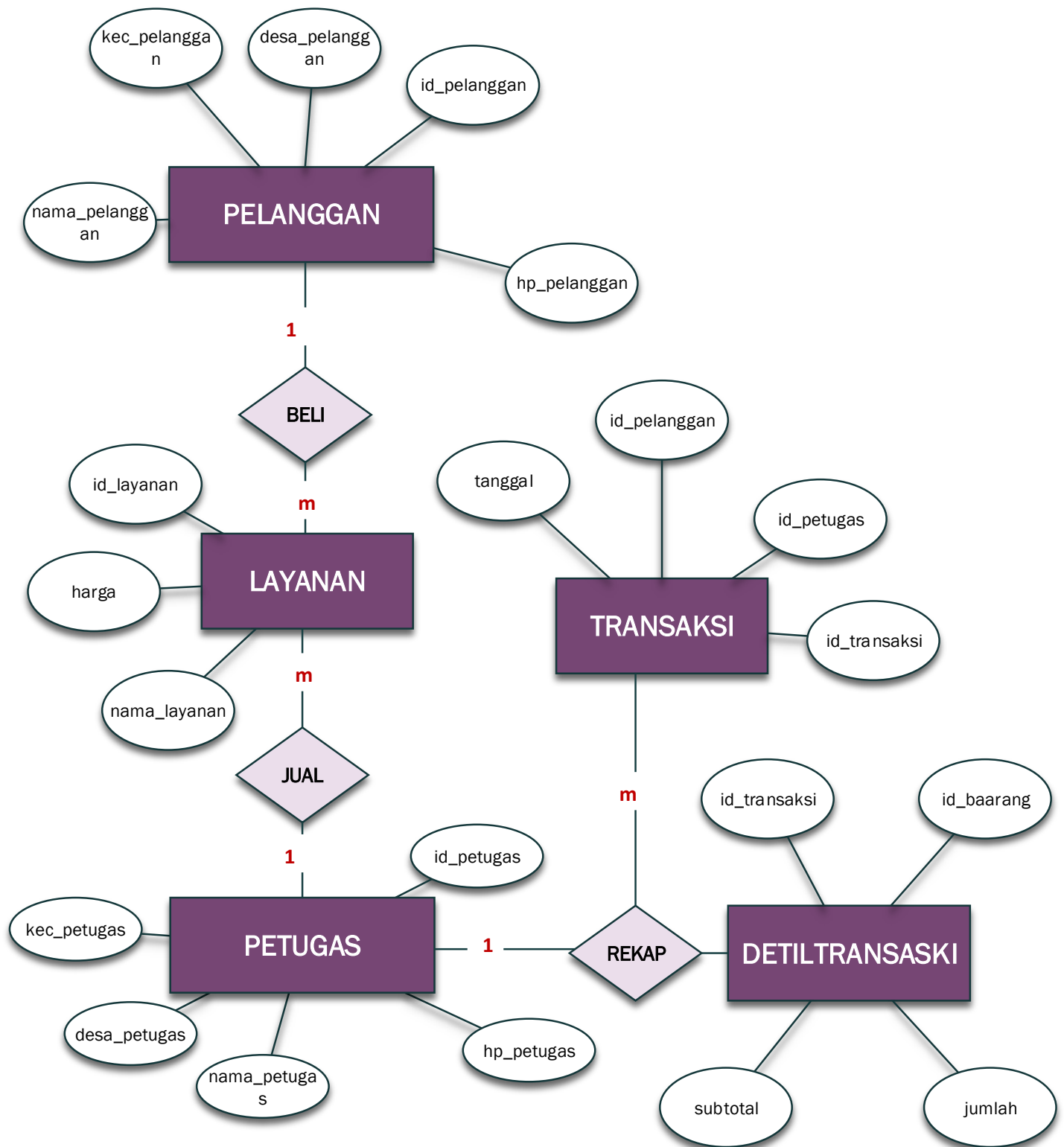
- Merupakan relasi antara entitas pembeli dan layanan yang berarti pembeli dapat melakukan pembelian layanan yang dijual oleh petugas.
- Kardinalitas relasi antara pembeli dan layanan adalah *one to many* karena seorang pembeli dapat membeli banyak layanan atau dengan kata lain banyak layanan boleh di beli oleh seorang pembeli.

### 3. Rekap

- Merupakan relasi antara entitas petugas dengan transaksi dan detil transaksi yang berarti petugas dapat melakukan rekap transaksi yang dibuat oleh petugas.
- Kardinalitas relasi antara petugas dengan transaksi dan detil transaksi adalah *one to many* karena seorang petugas dapat merekap banyak transaksi dan detil transaksi atau dengan kata lain banyak transaksi dan detil transaksi boleh di rekap oleh seorang petugas.

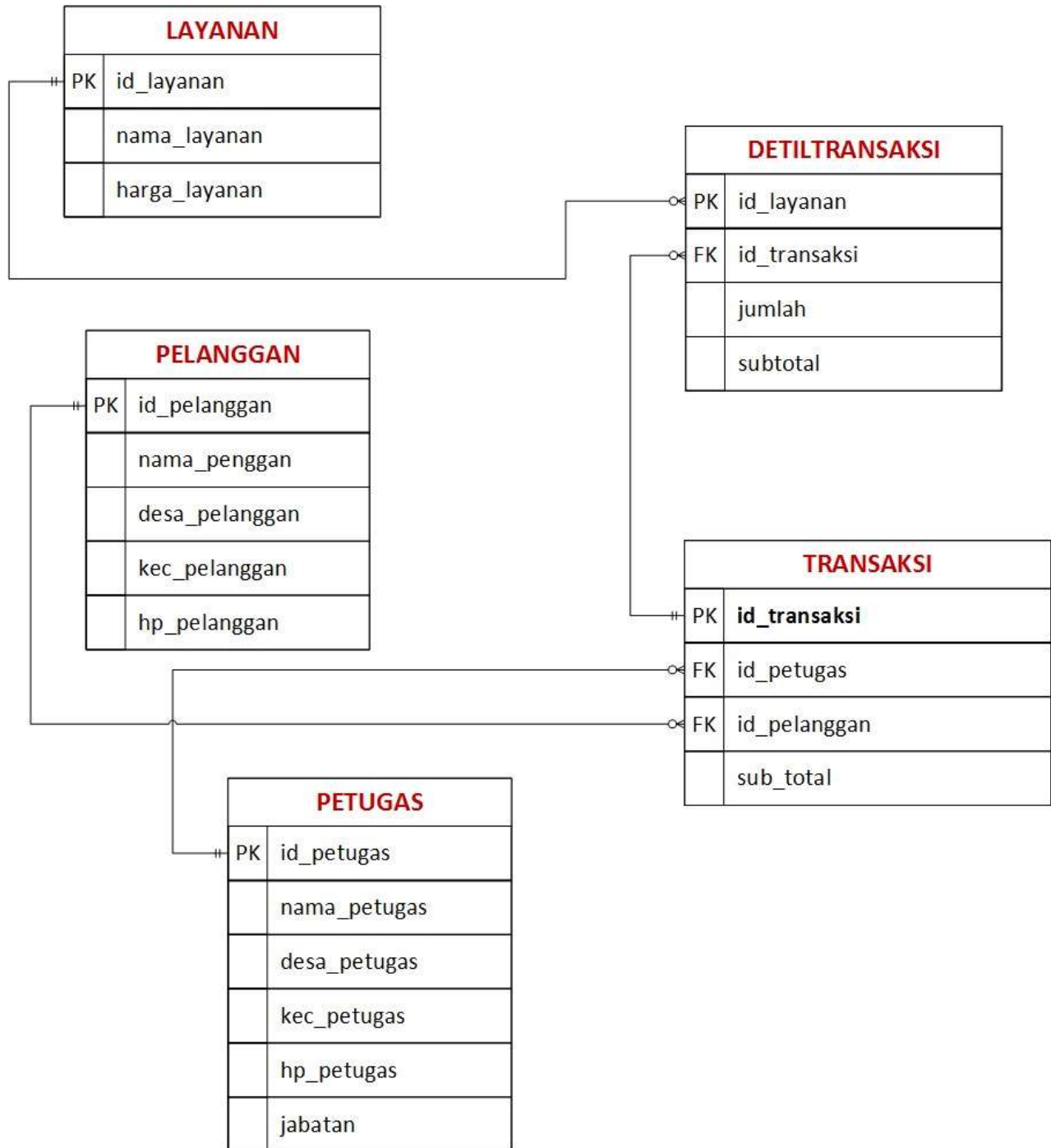
### E. Hasil ERD Petugasan Layanan Online

Dari tahap-tahap di atas maka dapat di buat rancangan ERD petugasan layanan online adalah sebagai berikut:



## F. Desain Logikal

Desain logikal yaitu proses pembuatan model dari informasi yang digunakan perusahaan berdasarkan model dan data spesifik. Deskripsi implementasi *database* berdasarkan hasil desain logikal dengan *Entity Relationship Diagram* (ERD) pada *Database Management System* (DBMS) menghasilkan **ER-Table** sebagai berikut





## G.Desain Fisikal & Source SQL

### Menggambarkan Rancangan Entitas Pada Basisdata Secara Fisikal (Physical Data Disain) serta Membuat Source SQL pembuatan Masing-Masing Tabel/Entitas

#### Tabel pelanggan

No	Nama kolom	Tipe data	Lebar	Null	kunci	keterangan
1	<b>id_pelanggan</b>	int	11	not null	primarykey	auto increment
2	nama_peanggan	varchar	30			
3	desa_pelanggan	varchar	50			
4	kec_pelanggan	varchar	30			
5	hp_pelanggan	varchar	30			

```
create table pelanggan (  
id_pelanggan int(11)not null auto_increment,  
nama_pelanggan varchar (30) not null,  
desa_pelanggan varchar (50),  
kec_pelanggan varchar (30),  
hp_pelanggan varchar (14),  
primary key (id_pelanggan)  
);
```

#### Tabel petugas

No	Nama kolom	Tipe data	Lebar	Null	kunci	keterangan
1	<b>id_petugas</b>	int	11	not null	primarykey	auto increment
2	nama_petugas	varchar	30	not null		
3	desa_petugas	varchar	50	not null		
4	kec_petugas	varchar	30	not null		
5	hp_petugas	varchar	30	not null		
6	jabatan	varchar	30	not null		

```
create table petugas (  
id_petugas int(11)not null auto_increment,  
nama_petugas varchar (30)not null,  
desa_petugas varchar (50)not null,  
kec_petugas varchar (30)not null,  
hp_petugas varchar (14)not null,  
jabatan varchar (50)not null,  
primary key (id_petugas)  
);
```

#### Tabel layanan

No	Nama kolom	Tipe data	Lebar	Null	kunci	keterangan
1	<b>id_layanan</b>	int	11	not null	primarykey	auto increment
2	nama_layanan	varchar	50	not null		
3	harga	int	20	not null		

```
create table layanan (  

```

```
id_layanan int(11)not null primary key auto_increment,
nama_layanan varchar (50) not null,
harga int(20) not null
);
```

#### Tabel transaksi

No	Nama kolom	Tipe data	Lebar	Null	kunci	keterangan
1	<b>id_transaksi</b>	int	11	not null	primarykey	auto increment
2	<i>id_pelanggan</i>	int	11	not null	foreignkey	
3	<i>id_petugas</i>	int	11	not null	foreignkey	
4	tanggal	date		not null		

```
create table transaksi (
id_transaksi int(11)not null auto_increment,
id_petugas int(11)not null,
id_pelanggan int(11)not null,
tanggal date not null,
primary key(id_transaksi),
constraint id_petugas foreign key (id_petugas) references petugas (id_petugas),
constraint id_pelanggan foreign key (id_pelanggan) references pelanggan (id_pelanggan)
);
```

#### Tabel detailtransaksi

No	Nama kolom	Tipe data	Lebar	Null	kunci	keterangan
1	<i>id_transaksi</i>	int	11	not null	foreignkey	
2	<i>id_layanan</i>	int	11	not null	foreignkey	
3	jumlah	int	20	not null		
4	subtotal	int	20	not null		

```
create table detailtransaksi (
id_transaksi int(11)not null,
id_layanan int(11)not null,
jumlah int (20) not null,
subtotal int(20) not null,
constraint id_transaksi foreign key (id_transaksi) references transaksi (id_transaksi),
constraint id_layanan foreign key (id_layanan) references layanan (id_layanan)
);
```

## H.Implementasi Syntax Sql Database Melalui CMD

### Membuat Basisdata

```
C:\Users\BANG MUSLIM>cd\
C:\>cd/xampp/mysql/bin
C:\xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 1448
Server version: 10.1.38-MariaDB mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> create database ulanganahmadi;
Query OK, 1 row affected (0.01 sec)
MariaDB [(none)]> use ulanganahmadi
```

### Membuat Tabel

```
MariaDB [ulanganahmadi]> create table pelanggan (
-> id_pelanggan int(11)not null auto_increment,
-> nama_pelanggan varchar (30) not null,
-> desa_pelanggan varchar (50),
-> kec_pelanggan varchar (30),
-> hp_pelanggan varchar (14),
-> primary key (id_pelanggan)
-> );
Query OK, 0 rows affected (0.11 sec)
```

```
MariaDB [ulanganahmadi]> create table petugas (
-> id_petugas int(11)not null auto_increment,
-> nama_petugas varchar (30)not null,
-> desa_petugas varchar (50)not null,
-> kec_petugas varchar (30)not null,
-> hp_petugas varchar (14)not null,
-> jabatan varchar (50)not null,
-> primary key (id_petugas)
-> );
Query OK, 0 rows affected (0.04 sec)
```

```
MariaDB [ulanganahmadi]> create table layanan (
-> id_layanan int(11)not null primary key auto_increment,
-> nama_layanan varchar (50) not null,
-> harga int(20) not null
-> );
Query OK, 0 rows affected (0.04 sec)
```

```
MariaDB [ulanganahmadi]> create table transaksi (
-> id_transaksi int(11)not null auto_increment,
-> id_petugas int(11)not null,
-> id_pelanggan int(11)not null,
-> tanggal date not null,
-> primary key(id_transaksi),
-> constraint id_petugas foreign key (id_petugas) references petugas (id_petugas),
-> constraint id_pelanggan foreign key (id_pelanggan) references pelanggan (id_pelanggan)
-> );
Query OK, 0 rows affected (0.05 sec)
```

```

MariaDB [ulanganahmadil] > create table detiltransaksi (
  -> id_transaksi int(11) not null,
  -> id_layanan int(11) not null,
  -> jumlah int(20) not null,
  -> subtotal int(20) not null,
  -> constraint id_transaksi foreign key (id_transaksi) references transaksi (id_transaksi),
  -> constraint id_layanan foreign key (id_layanan) references layanan (id_layanan)
  -> );
Query OK, 0 rows affected (0.04 sec)

```

Medeskripsikan tabel

```

MariaDB [ulanganahmadil] > desc layanan;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_layanan | int(11)   | NO   | PRI | NULL    | auto_increment |
| nama_layanan | varchar(50) | NO   |     | NULL    |               |
| harga      | int(20)   | NO   |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.02 sec)

MariaDB [ulanganahmadil] > desc petugas;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_petugas | int(11)   | NO   | PRI | NULL    | auto_increment |
| nama_petugas | varchar(30) | NO   |     | NULL    |               |
| desa_petugas | varchar(50) | NO   |     | NULL    |               |
| kec_petugas | varchar(30) | NO   |     | NULL    |               |
| hp_petugas | varchar(14) | NO   |     | NULL    |               |
| jabatan    | varchar(50) | NO   |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.02 sec)

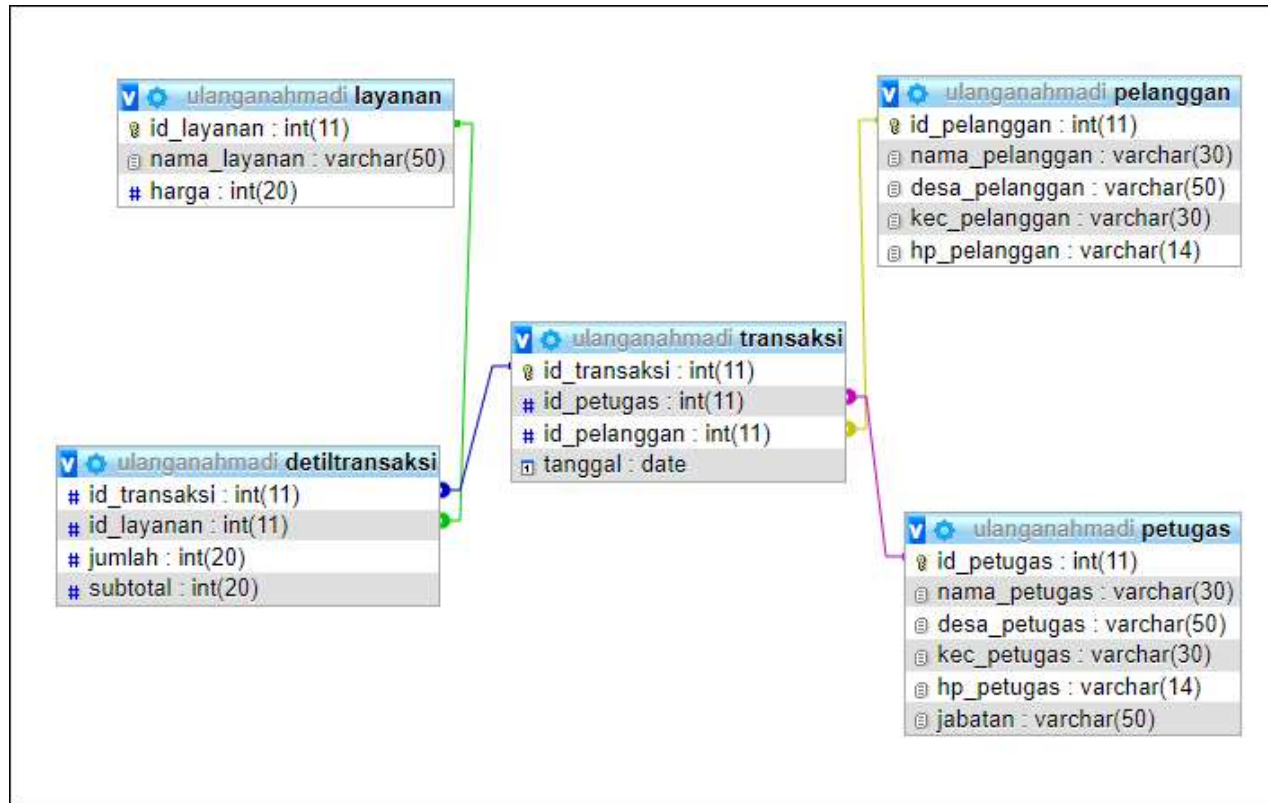
MariaDB [ulanganahmadil] > desc pelanggan;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_pelanggan | int(11)   | NO   | PRI | NULL    | auto_increment |
| nama_pelanggan | varchar(30) | NO   |     | NULL    |               |
| desa_pelanggan | varchar(50) | YES  |     | NULL    |               |
| kec_pelanggan | varchar(30) | YES  |     | NULL    |               |
| hp_pelanggan | varchar(14) | YES  |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)

MariaDB [ulanganahmadil] > desc transaksi;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_transaksi | int(11)   | NO   | PRI | NULL    | auto_increment |
| id_petugas   | int(11)   | NO   | MUL | NULL    |               |
| id_pelanggan | int(11)   | NO   | MUL | NULL    |               |
| tanggal     | date      | NO   |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)

MariaDB [ulanganahmadil] > desc detiltransaksi;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_transaksi | int(11)   | NO   | MUL | NULL    |               |
| id_layanan   | int(11)   | NO   | MUL | NULL    |               |
| jumlah       | int(20)   | NO   |     | NULL    |               |
| subtotal     | int(20)   | NO   |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+

```

## I. Hasil Desain Konseptual Skema Relasi Database Di PhpMyadmin



### ▪ LATIHAN 13

**Kerjakan soal berikut dengan baik dan benar!**

- 1) Jelaskan pengertian ERD, dan gambarkan 1 buah contoh ERD Penjualan..!
- 2) Jelaskan pengertian ERT (entity relationship table), dan gambarkan 1 buah contoh ERT Penjualan..!
- 3) Jelaskan pengertian entitas dan contohnya..!
- 4) Jelaskan pengertian atribut dan contohnya..!
- 5) Atribut ada 5 jenis, sebutkan dan jelaskan dengan gambar simbolnya..!
- 6) Jelaskan pengertian kardinalitas dan sebutkan 4 jenis kardinalitas beserta simbol penulisannya..!
- 7) Sebutkan 9 tahapan dalam menyusun sebuah ERD secara berurutan..!
- 8) Jelaskan pengertian normalisasi..!
- 9) Jelaskan 3 jenis anomali jika data di database tidak dinormalisasi..!
- 10) Jelaskan pengertian top-down dan bottom-up dalam siklus pembentukan database..!

### ▪ PROJECT PERTEMUAN 13

**Kerjakan project berikut dengan baik dan benar!**

- 1) Pilih sebuah judul project ..!
- 2) Buatlah rancangan awal dengan menentukan entitas, atribut, dan relasi.
- 3) Gambarkan ERD project yang kamu buat dengan menggunakan visio..!
- 4) Lakukan normalisasi melalui ERT beserta relasi dan kardinalitasnya!



## MATERI PERTEMUAN 15

### SQL ; Command Line Interface (DDL)

---



#### TAHUKAH KAMU...?

PHP memiliki beberapa fungsi untuk mencetak teks ke layar:

- fungsi `echo()`;
- fungsi `print()`;
- fungsi `printf()`.

Apa saja perbedaan fungsinya...?

#### A. SQL Dalam Basisdata

SQL adalah singkatan dari Structured Query Language. Sedangkan pengertian SQL adalah suatu bahasa (language) yang digunakan untuk mengakses data di dalam sebuah database relasional. SQL sering juga disebut dengan istilah query, dan bahasa SQL secara praktiknya digunakan sebagai bahasa standar untuk manajemen database relasional. Hingga saat ini hampir seluruh server database atau software database mengenal dan mengerti bahasa SQL.

Sejarah SQL Awal mula lahirnya bahasa SQL yaitu pada bulan Juni 1970, dimana saat Jhonny Oracle yang merupakan seorang peneliti dari perusahaan IBM memiliki gagasan pembuatan basis data relasional, ide tersebut dituangkan dalam sebuah artikel. Di dalam artikel tersebut juga dibahas mengenai kemungkinan membuat sebuah bahasa standar untuk mengakses data dalam database tersebut. Bahasa standar tersebut diberinama SEQUEL (Structured English Query Language). Setelah kemunculan artikel tersebut lalu IBM memutuskan untuk mengembangkan pembuatan bahasa SEQUEL. Namun penamaan SEQUEL dalam bahasa standar tersebut bermasalah dengan hukum sehingga diubahlah menjadi SQL.

Jenis Perintah SQL Dalam penggunaan SQL terdapat beberapa perintah yang berguna untuk mengakses dan memanajemen data yang terdapat dalam database. Jenis perintah SQL secara umum dibagi kepada tiga sub perintah, yaitu DDL (Data Definition Language), DML (Data Manipulation Language), dan DCL (Data Control Language). Ketiga sub perintah tersebut sangat perlu untuk dipahami bagi anda yang ingin menguasai bahasa sql dan mahir dalam pembuatan database.

MySQL adalah suatu perangkat lunak database relasi (*Relational Database Management System* atau RDBMS), seperti halnya ORACLE, Postgresql, MS SQL, dan sebagainya. MySQL dikembangkan sekitar tahun 1994 oleh sebuah perusahaan pengembang software dan konsultandatabase bernama MYSQL AB yang berada di Swedia. Waktu itu perusahaan tersebut masih bernama TcX DataKonsult AB, dan tujuan awal dikembangkannya MySQL adalah untuk mengembangkan aplikasi berbasis web pada client. MySQL menyebut produknya sebagai database open source terpopuler di dunia. Berdasarkan riset dinyatakan bahwa bahwa diplatform Web, dan baik untuk kategori open source maupun umum, MySQL adalah database yang paling banyak dipakai. Menurut perusahaan pengembangnya, MySQL telah terpasang di sekitar 3 juta komputer. Puluhan hingga ratusan ribu situs mengandalkan MySQL bekerja siangmalam memompa data bagi para pengunjungnya.

SQL yang menjadi salah satu dari macam-macam bahasa pemrograman ini, menjadi bahasa pemrograman yang paling banyak di gunakan. Dalam prakteknya SQL memiliki beberapa perintah yang banyak macamnya, yang berfungsi untuk membantu proses manajemen basis data tersebut. Dengan adanya perintah – perintah tersebut, proses manajemen basis data bisa dilakukan oleh administrator tanpa harus mengutak-atik komputer server.

Perintah – perintah dalam struktur bahasa SQL tersebut terbagi menjadi 3 jenis bahasa, yaitu bahasa DDL (Data Definition Language), DML (Data Management Language), dan juga DCL (Data Control Language). Namun untuk

dapat mengakses basisdata dengan SQL sebelumnya kita harus memahami konsep CLI terlebih dahulu.

## **B. Pengertian & Perkembangan CLI**

CLI dapat diperluas sebagai ‘Command Line Interface’, yang tidak lain adalah antarmuka pengguna yang memungkinkan melakukan operasi fungsional apa pun pada sistem dengan menerima input dalam bentuk perintah, alih-alih mengklik dan mengetik bentuk input. Hal ini umum dalam Sistem Operasi Disk tradisional (DOS), di mana perintah dimasukkan dalam antarmuka prompt perintah DOS. Akhir-akhir ini, CLI ini diterapkan oleh perangkat lunak atau administrator sistem untuk mengkonfigurasi sistem komputer, karena tindakan serupa dalam antarmuka pengguna grafis (GUI) dapat memakan banyak waktu dan tindakan manual.

Command Line Interface (CLI) adalah antarmuka pengguna berbasis teks ( UI ) yang digunakan untuk menjalankan program, mengelola file komputer, dan berinteraksi dengan komputer. Command Line Interface juga disebut antarmuka pengguna baris perintah , antarmuka pengguna konsol , dan antarmuka pengguna karakter . CLI menerima sebagai perintah input yang dimasukkan oleh keyboard; perintah yang dipanggil pada prompt perintah kemudian dijalankan oleh komputer.

Saat ini, sebagian besar vendor menawarkan antarmuka pengguna grafis ( GUI ) sebagai default untuk sistem operasi (OS) seperti Windows, Linux dan macOS. Sebagian besar sistem berbasis Unix saat ini menawarkan Command Line Interface dan antarmuka pengguna grafis. Sistem operasi MS-DOS dan shell perintah di sistem operasi Windows adalah contoh Command Line Interface. Selain itu, platform pengembangan bahasa pemrograman seperti Python dapat mendukung Command Line Interface. Baris perintah menurun popularitasnya setelah pengenalan OS komputer pribadi berbasis GUI seperti Microsoft Windows dan Mac OS “klasik” Apple pada 1980-an. Baris perintah tetap menjadi alat penting bagi profesional TI, pengembang perangkat lunak, admin sistem , administrator jaringan, dan banyak lainnya yang lebih memilih antarmuka yang lebih presisi dan dapat direproduksi ke sistem mereka.

Kembali di tahun 1960-an, tidak ada antarmuka visual grafis seperti itu, hanya teks murni. Ketika komputer berevolusi sepanjang tahun 1970-an dan 80-an, CLI berbasis teks masih berkuasa. Keuntungan dari CLI adalah bahwa antarmuka berbasis teks tidak memerlukan banyak sumber daya. Perintah cenderung sederhana, meskipun mungkin ada ribuan perintah yang berbeda, tergantung pada perangkat lunak atau sistem. CLI biasanya memungkinkan otomatisasi tugas berulang sampai batas tertentu, berkat fitur seperti menyimpan perintah yang sering digunakan atau pengeditan cepat.

## **Mengapa Menggunakan CLI dan Bukan GUI?**

Seperti yang sudah dijelaskan, GUI dikembangkan setelah mouse ditemukan dan menjadi perangkat input baru untuk mengoperasikan komputer. GUI memiliki tampilan yang menarik serta mudah dipahami. Sayangnya, untuk menjalankan beberapa task penting, CLI masih lebih baik. Berikut beberapa alasan CLI dinilai lebih efektif dibanding GUI. Namun, sekali lagi, gunakan mana pun yang dirasa lebih sesuai dengan pekerjaan Anda.

### **1. Menggunakan Lebih Sedikit Resource**

Sudah jadi rahasia umum bahwa program berbasis teks hanya menggunakan sedikit resource di komputer. Jadi, dengan command line interface, Anda bisa menjalankan banyak task serupa dengan resource minimum.

### **2. Tepat dan Akurat**

Anda bisa menggunakan perintah tertentu untuk menuju file atau folder yang lebih spesifik. Tidak akan ada kendala asalkan perintah yang diketik benar. Jika sudah memahami dasar-dasarnya, Anda pasti bisa menulis sintaksis dengan lebih mudah.

### **3. Mudah untuk Menjalankan Task Berulang**

GUI memang telah dikembangkan selama bertahun-tahun. Namun, sistem operasi mungkin tidak menyediakan semua menu dan tombol yang diperlukan untuk menjalankan task demi alasan keamanan. Akibatnya, sulit jika Anda harus menjalankan tugas yang berulang. Nah, CLI bisa mengatasinya.



Misalnya, untuk mengelola ratusan file dalam sebuah folder, Anda hanya perlu menjalankan satu perintah yang kemudian bisa diotomatiskan untuk melakukan task yang sama.

#### 4. Canggih

Sebagian besar OS modern tidak mengizinkan user ‘mengutak-atik’ proses inti sistem. Windows memiliki sistem keamanan, dan MacOS memiliki SIP (System Integrity Protection) yang membatasi Anda melakukan task yang dilindungi sistem. Dengan CLI, Anda bisa mengelola sistem sepenuhnya tanpa dibatasi larangan tersebut.

### Contoh Sintaksis di Command Prompt Windows

Berikut contoh penggunaan beberapa sintaksis di Command Prompt:

- **Mengubah direktori**  
Untuk cara masuk ke folder di CMD, gunakan **CD [path]**. Beri spasi di depan path yang dituju. Contoh:  
CD C:\Program Files
- **Mengganti nama file**  
Untuk mengganti nama file dalam folder tertentu, gunakan **REN [drive:][path] [nama sekarang] [target]**. Dengan menyebutkan nama folder, file yang diubah namanya akan disimpan di folder tersebut. Contoh:  
REN d:untitled.txt untitled1.txt
- **Menghapus file**  
Untuk menghapus file melalui command prompt, gunakan **DEL [nama file]**. Jika ingin menambahkan opsi seperti **force deletion** (hapus paksa), letakkan perintah di depan nama file. Contoh:  
DEL /F untitled.txt
- **Mengganti nama Volume Disk**  
Untuk mengganti nama volume disk tertentu, gunakan **LABEL [drive:][nama volume baru]**. Anda bisa menggunakan hingga 32 karakter pada volume NTFS dan 11 karakter pada volume FAT. Contoh:  
D:\> LABEL d:MyData

### Bash

Bash atau **Bourne Again Shell** adalah tipe shell yang digunakan di **MacOS** serta berbagai distribusi **Linux**, dan dikembangkan oleh **Free Software Foundation**. Tool ini juga bisa diinstall di Windows 10. Bash merupakan salah satu tipe shell yang bisa digunakan oleh user Linux, selain **Tchs shell**, **Ksh shell**, dan **Zsh shell**.

Di sebagian besar distribusi Linux, Bash ada dalam menu **Utilities**. Di desktop **Gnome**, namanya adalah **Terminal**, sedangkan untuk **KDE**, namanya adalah **Konsole**. Sementara itu, di MacOS, program ini disebut **Terminal.app**. Untuk menjalankannya, buka **Application -> Utilities -> Terminal**. Atau, cukup ketikkan **terminal** di pencarian Spotlight. Setelah terminal terbuka, Anda bisa langsung mengetikkan perintah. Perintah biasanya terdiri dari **perintah itu sendiri**, **argumen**, dan **opsi**. Perintah memuat instruksi yang akan dijalankan, argumen memberitahukan tempat perintah harus dijalankan, dan opsi meminta modifikasi hasil perintah.

Untuk menggunakan shell, Anda harus tahu sintaksisnya lebih dulu. Cara ini juga dikenal sebagai *shell scripting*, yaitu penggunaan script di CLI untuk menjalankan task tertentu. Meskipun ada banyak perintah yang bisa digunakan dengan CLI, semua perintah tersebut umumnya terbagi ke dalam **dua kategori**:

- Perintah untuk **menangani atau mengelola proses**
- Perintah untuk **menangani atau mengelola file**

Selalu pastikan bahwa Anda tidak salah mengetik perintah. Perhatikan setiap karakter yang digunakan, termasuk spasi. Pastikan juga Anda menggunakan huruf besar/kecil yang benar.

Jika karena alasan tertentu Anda ingin menghentikan proses yang sedang berjalan di Command Prompt atau Bash, tekan tombol **Control+C**.

### C. Perintah CMD

#### Perintah CMD untuk Network Configuration

Perintah CMD	Fungsi
Ping	Mengecek koneksi jaringan
Tracert	Trace route ke sebuah remote host
Pathping	Memberikan info hilangnya latensi dan paket untuk setiap node di jalur jaringan
ipconfig/all	Menampilkan konfigurasi koneksi
ipconfig/displaydns	Menampilkan konten DNS cache
ipconfig/flushdns	Menghapus DNS konten cache
ipconfig/release	Pelepasan semua konfigurasi
ipconfig/renew	Memperbarui semua koneksi
ipconfig/registerdns	Refresh DHCP & re-register DNS
ipconfig/showclassid	Menampilkan DHCP Class ID
ipconfig/setclassid	Memodifikasi DHCP Class ID
getmac	Menampilkan alamat MAC dari adaptor jaringan pengguna
nslookup	Mengecek IP address di Name Server
netstat	Menampilkan koneksi TCP/IP yang aktif
netstat -ano	Mengetahui apakah komputer kamu digunakan oleh orang lain atau mencari apa yang kamu lakukan
net view	Menampilkan nama perangkat yang terhubung ke WiFi kamu dengan nama PC mereka masing-masing
arp -a	Info perangkat yang terhubung ke WiFi dengan alamat IP, MAC, dan juga tipe dinamis atau statisnya
netsh wlan show profiles	Menampilkan seluruh jaringan WiFi yang pernah terhubung ke perangkat
netsh wlan show profile name="nama WiFi" key=clear	Menampilkan detail informasi jaringan WiFi termasuk password-nya

#### Perintah CMD untuk File&Folder Configuration

Perintah CMD	Fungsi
assoc	Menampilkan dan mengubah asosiasi nama ekstensi file
Attrib	Menampilkan, mengatur, atau menghapus atribut <i>read-only</i> , <i>archive</i> , <i>system</i> , dan <i>hidden</i> yang terpasang pada file atau folder
Cd	Menampilkan nama folder (direktori) atau mengubah lokasi/posisi direktori
Chdir	Memiliki fungsi sama seperti perintah cd
chkdsk	Memeriksa dan menampilkan laporan status disk berdasarkan file system
chkntfs	Memeriksa sistem file NTFS
Copy	Menyalin file dari suatu lokasi (direktori) ke lokasi lain

Color	mengganti warna tulisan di Command Prompt
convert	Mengubah partisi berjenis FAT menjadi NTFS
Date	Melihat dan mengganti tanggal
defrag	Melakukan <i>defragmentation</i>
Del	Menghapus file dan memasukkan ke <i>recycle bin</i>
deltree	Menghapus file secara permanen (tidak masuk ke dalam <i>recycle bin</i> )
Dir	Melihat daftar file dan sub-direktori yang ada dalam sebuah direktori
diskpart	Mengelola hard drive di komputer atau laptop
Add	Membuat partisi baru
assign	Memberikan huruf pada partisi baru
delete	Menghapus sebuah partisi
detail	Melihat informasi tentang partisi yang dipilih
driverquery	Menampilkan daftar driver yang di-indtall di komputer atau laptop
Exit	Keluar dari Command Prompt atau menutup proses <i>batch script</i> yang sedang berjalan
Find	Mencari teks tertentu dalam sebuah file
logoff	Menghentikan sesi <i>user</i> tertentu dari komputer atau laptop
Move	Memindahkan satu file atau lebih ke direktori lain
msg	Mengirim pesan ke user lain dalam sebuah jaringan komputer lokal
print	Mencetak file teks dari Command Prompt
pause	Menghentikan file <i>batch</i> yang sedang berjalan
rename	Mengubah nama file dan direktori

Contoh penerapan: `C:\>shutdown`

### Perintah CMD Lainnya Sesuai Abjad

Perintah CMD	Fungsi
addusers	Menambah dan daftar pengguna dalam file CSV
at	Jalankan perintah pada waktu tertentu
admodcmd	Mengubah konten secara massal dalam direktori aktif
arp	Memetakan alamat IP ke alamat perangkat keras
associat	Salah satu langkah asosiasi file
assoc	Menampilkan dan mengubah asosiasi nama ekstensi file
attrib	Menampilkan, mengatur, atau menghapus atribut <i>read-only</i> , <i>archive</i> , <i>system</i> , dan <i>hidden</i> yang terpasang pada file atau folder
cd	Menampilkan nama folder (direktori) atau mengubah lokasi/posisi direktori
bcdboot	Membuat dan memperbaiki partisi sistem
bcdedit	mengelola data konfigurasi boot
bitsadmin	Mengelola Layanan Transfer Cerdas Latar Belakang
bootcfg	Mengedit konfigurasi boot di Windows
break	Kombinasi tombol CTRL+C tidak dapat digunakan untuk menghentikan proses MS-DOS
cacls	Mengubah perizinan file

csvde	Impor atau Ekspor data dari direktori aktif
cscmd	Konfigurasi file offline di komputer klien
cprofile	Membersihkan profil yang ditentukan dari ruang yang terbuang dan menonaktifkan asosiasi file pengguna tertentu
coreinfo	Tampilkan pemetaan antara prosesor logis dan fisik
copy	enyalin file dari suatu lokasi (direktori) ke lokasi lain
color	mengganti warna tulisan di Command Prompt
convert	Mengubah partisi berjenis FAT menjadi NTFS
compress	Kompres satu atau lebih file
compact	Kompres file dan folder pada partisi NTFS
comp	Bandingkan isi dari dua file atau dua set file
cmstp	Menginstal atau menghapus profil layanan koneksi manajer
cmdkey	Mengelola nama pengguna dan kata sandi yang disimpan
cmd	Memulai shell CMD baru
cls	Bersihkan layar CMD
clip	Salin hasil dari setiap perintah (stdin) ke clipboard Windows
cleanmgr	Menggunakan file temp yang bersih dan mendaur ulang sampah secara otomatis
cipher	Mengenkripsi / mendekripsi file dan folder
choice	Terima input pengguna (melalui keyboard) ke file batch
chkntfs	Memeriksa sistem file NTFS
chcp	Menampilkan jumlah halaman kode konsol yang aktif
date	Melihat dan mengganti tanggal
defrag	Melakukan <i>defragmentation</i>
del	Menghapus file dan memasukkan ke <i>recycle bin</i>
deltree	Menghapus file secara permanen (tidak masuk ke dalam <i>recycle bin</i> )
delprof	Menghapus profil pengguna
devcon	Akses utilitas pengelola perangkat baris perintah
dsmgmt	Kelola Layanan Direktori Ringan Active Directory
dssrm	Hapus objek dari direktori aktif
dsmove	Ubah nama atau pindahkan objek direktori aktif
dsmod	Memodifikasi objek dalam direktori aktif
dsquery	Temukan objek dalam direktori aktif
dsget	Lihat objek dalam direktori aktif
dsadd	Menambahkan objek ke direktori aktif
dsacls	Lihat dan edit entri kontrol akses untuk objek di direktori aktif
driverquery	Menampilkan daftar driver perangkat yang diinstal
doskey	Mengedit baris perintah, memanggil kembali perintah, dan membuat makro
diskuse	Lihat ruang yang digunakan dalam folder (s)
diskshadow	Akses Layanan Salinan Disk Shadow
diskpart	Buat perubahan pada partisi penyimpanan, baik internal maupun terhubung
diskcopy	Salin data satu floppy disk yang lain
diskcomp	Bandingkan isi dua floppy disk
diruse	Menampilkan penggunaan disk
dirquota	Kelola kuota Sumber Daya Pengelola File Server

dir	Menampilkan daftar file dan folder
erase	Menghapus satu atau lebih file
endlocal	Akhir localisation perubahan lingkungan dalam file batch
echo	Mengaktifkan atau menonaktifkan fitur perintah-echoing, menampilkan pesan di layar
exit	Keluar dari baris perintah (Keluar dari skrip batch saat ini)
extract	Uncompress satu atau lebih file kabinet Windows
expand	Uncompress satu atau lebih .CAB file
explorer	Buka Windows Explorer
eventtriggers	Tampilkan dan konfigurasi pemicu peristiwa pada mesin lokal dan jarak jauh
eventcreate	Tambahkan acara khusus ke log peristiwa Windows (Hak admin diperlukan)
eventquery	Tampilkan daftar acara dan propertinya dari log kejadian
ftype	Tampilkan / Ubah asosiasi jenis ekstensi file
fsutil	Utilitas sistem file untuk mengelola properti file dan drive
format	Memformat disk
for	Jalankan perintah dalam lingkaran untuk file untuk parameter yang ditentukan
finger	Tampilkan informasi tentang pengguna pada komputer jarak jauh yang ditentukan
find	Mencari string teks tertentu dalam file
ftp	Gunakan layanan FTP untuk mentransfer file dari satu PC ke PC lainnya
freedisk	Memeriksa ruang kosong pada disk
forfiles	PMengaktifkan / menonaktifkan folder sementara
findstr	Menemukan pola string dalam file
fc	Membandingkan dua file
graftabl	Aktifkan kemampuan untuk menampilkan karakter tambahan dalam mode grafis
gpresult	Tampilkan Pengaturan Kebijakan Grup dan Set Kebijakan Hasil untuk pengguna
getmac	Menampilkan alamat MAC dari adaptor jaringan pengguna
gpupdate	Perbarui direktori lokal dan aktif berdasarkan pengaturan kebijakan grup
goto	Mengarahkan program batch ke saluran yang diidentifikasi oleh label
hostname	Menampilkan nama host komputer
help	Tampilkan daftar perintah dan lihat informasi online untuk mereka
inuse	Ganti file yang saat ini digunakan oleh OS (perlu direstart)
ipseccmd	Mengkonfigurasi kebijakan Keamanan IP
irftp	Mengirim file melalui tautan inframerah (fungsi inframerah diperlukan)
if	Pemrosesan bersyarat dalam program batch
icacls	Mengubah izin file dan folder
ipxroute	Tampilkan dan ubah informasi tabel routing yang digunakan oleh protokol IP
ipconfig	Tampilkan dan ubah konfigurasi IP
ifmember	Menampilkan grup yang merupakan pengguna aktif
iexpress	Membuat arsip zip self-extracting
logoff	Menghentikan sesi <i>user</i> tertentu dari komputer atau laptop
lpq	Menampilkan status antrian pencetakan
label	Edit disk label
local	Menampilkan keanggotaan kelompok-kelompok lokal
logman	Mengelola kinerja log monitor

lpr	mengirim file ke komputer yang menjalankan layanan Line Printer Daemon
logtime	Menampilkan log tanggal dan waktu dalam file
mstsc	Membuat koneksi <i>remote desktop</i>
msinfo32	Menampilkan informasi sistem
msiexec	Menginstall, memodifikasi, mengkonfigurasi menggunakan Windows Installer
msg	Mengirim pesan ke user lain dalam sebuah jaringan komputer lokal
move	Memindahkan satu file atau lebih ke direktori lain
moveuser	Memindahkan akun user ke sebuah domain atau di antara mesin
mountvol	Membuat, mendaftarkan, atau menghapus volume mount point
more	Tampilkan satu layar <i>otuput</i> pada waktu yang sama
makecab	Membuat file .CAB
macfile	Kelola server file untuk Mackintosh
munge	Cari dan Ganti teks dalam file
net	Kelola sumber daya jaringan
netdom	Domain Manager
netsh wlan show profiles	Menampilkan seluruh jaringan WiFi yang pernah terhubung ke perangkat
netsh wlan show profile name="nama WiFi" key=clear	Menampilkan detail informasi jaringan WiFi termasuk password-nya
nbstat	Tampilkan statistik jaringan (NetBIOS over TCP / IP)
nslookup	Mengecek IP address di Name Server
netstat	Menampilkan koneksi TCP/IP yang aktif
now	Tampilan saat ini Tanggal dan Waktu
ntrights	Edit hak user account
path	Menampilkan atau menetapkan path pencarian untuk file executable
pathping	Memberikan info hilangnya latensi dan paket untuk setiap node di jalur jaringan
pause	Menghentikan file <i>batch</i> yang sedang berjalan
perms	Tampilkan izin untuk pengguna
perfmon	Kinerja monitor
powercfg	Mengkonfigurasi pengaturan daya
print	Mencetak file teks dari Command Prompt
pause	Menghentikan file <i>batch</i> yang sedang berjalan
prnmngr	Tambah, menghapus, daftar printer menetapkan printer default
prompt	Mengubah command prompt
psinfo	Daftar informasi tentang sistem
pskill	Proses mematikan berdasarkan nama atau ID proses
pslist	Daftar informasi rinci tentang proses-proses
pspasswd	Ubah sandi account
psservice	Melihat dan mengatur layanan
pushd	Simpan dan kemudian mengubah direktori sekarang
qgrep	Cari file untuk baris yang cocok dengan pola tertentu
qprocess	Menampilkan informasi tentang proses
reg	Read, set, export, hapus kunci dan nilai-nilai

recover	Perbaiki file yang rusak dari disk yang rusak
regedit	Impor atau ekspor pengaturan registry
regini	Ubah Registry Permissions
ren	Mengubah nama file atau file
replace	Ganti atau memperbarui satu file dengan yang lain
rd	Hapus folder
rmtshare	Share folder atau printer\
route	Manipulasi tabel routing jaringan
runas	Jalankan program di bawah account pengguna yang berbeda
sc	Control layanan
schtasks	Jadwal perintah untuk dijalankan pada waktu tertentu
sclist	Tampilkan Layanan NT
setlocal	Pengendalian environment visibilitas variabel
setx	Set variabel environment secara permanen
share	Daftar atau mengedit file share atau share print
shift	Shift posisi digantikan parameter dalam sebuah file batch
shutdown	Shutdown komputer
sleep	Membuat komputer pada mode sleep selama beberapa detik tertentu
systeminfo	Menampilkan detail informasi konfigurasi tentang perangkat komputer
tasklist	Daftar menjalankan aplikasi dan services
taskkill	Hapus proses yang berjalan dari memori
time	Menampilkan atau mengatur waktu sistem
timeout	Penundaan pemrosesan dari sebuah batch file
title	Mengatur judul window untuk sesi cmd.exe
tree	tampilan grafis struktur folder
type	Menampilkan isi dari file teks
	tracert Trace route ke sebuah remote host
usrstat	Daftar domain nama pengguna dan terakhir login
ver	Tampilkan nomor versi OS yang diinstal
vol	Tunjukkan label volume disk dan nomor seri
vssadmin	Menampilkan salinan cadangan bayangan, menginstal penulis dan penyedia salinan bayangan
verify	Verifikasi apakah file disimpan dengan benar pada disk
waitfor	Digunakan untuk menyinkronkan acara antar komputer berjaringan
wevtutil	Dapatkan informasi tentang log dan penerbit acara
where	Temukan dan tampilkan file dalam direktori saat ini
whoami	Tampilkan informasi tentang pengguna aktif
windiff	Bandingkan isi dari dua file atau set file
winrm	Manajemen Jarak Jauh Windows
wuaucvt	Windows Update Agent untuk mengunduh file pembaruan baru
xcalcs	Ubah ACL untuk file dan folder
xcopy	Salin file atau pohon direktori ke folder lain

Tidak ada perintah CMD.

**Z**

Tidak ada perintah CMD.

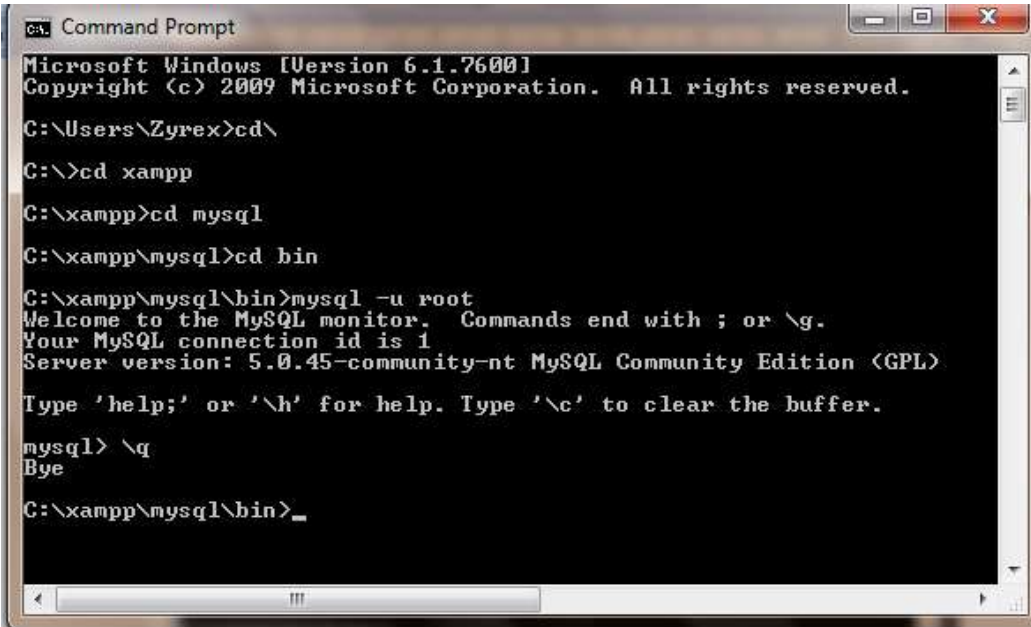
## 1. Format Perintah

Berikut adalah ketentuan-ketentuan memberi perintah pada MySQL:

- Setiap perintah harus diakhiri dengan tanda titik koma , kecuali untuk perintah tertentu, misal : quit
- Setiap perintah akan disimpan dalam buffer (memori sementara) untuk menyimpan histori perintah-perintah yang pernah diberikan.
- Perintah dapat berupa perintah SQL atau perintah khusus MySQL.
- Perintah-perintah dalam lingkungan MySQL tidak menerapkan aturan case sensitive, tetapi case insensitive yaitu perintah bisa dituliskan dalam huruf besar atau pun huruf kecil.
- Aturan case sensitive diterapkan pada penamaan objek-objek dalam database seperti nama database atau nama table, namun aturan ini hanya ada dalam lingkungan Unix dan Linux.

## 2. Start dan Stop MySQL

Berikut cara memulai MySQL dengan menggunakan bantuan XAMPP. Aktifkan XAMPP Control Panel Application, klik start apache dan mysql. Aktifkan command prompt, lalu ketik seperti gambar berikut:



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Zyrex>cd\

C:\>cd xampp

C:\xampp>cd mysql

C:\xampp\mysql>cd bin

C:\xampp\mysql\bin>mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> \q
Bye

C:\xampp\mysql\bin>
```

Sedangkan untuk stop atau keluar dari MySQL dapat menggunakan perintah : \q, exit dan quit.

## Instruksi Dalam SQL

Masuk ke database mysql di xampp

```
C:\Users\BANG MUSLIM>cd\

C:\>cd/xampp/mysql/bin

C:\xampp\mysql\bin>mysql -u root
```





## MATERI PERTEMUAN 15

### SQL ; Data Definition Language (DDL)



#### TAHUKAH KAMU...?

PHP memiliki beberapa fungsi untuk mencetak teks ke layar:

- fungsi `echo()`;
- fungsi `print()`;
- fungsi `printf()`.

Apa saja perbedaan fungsinya...?

DDL (*Data Definition Language*), DDL merupakan kelompok perintah yang berfungsi untuk mendefinisikan atribut-atribut basis data, tabel, atribut(kolom), batasan-batasan terhadap suatu atribut, serta hubungan antar tabel. Yang termasuk dalam kelompok DDL ini adalah CREATE, ALTER, dan DROP.

#### a. Membuat Database : **CREATE DATABASE** namadatabase;

Namadatabase tidak boleh mengandung spasi dan tidak boleh memiliki nama yang sama antar database. Berikut ini perintah untuk membuat database dengan nama rental : `CREATE DATABASE AHMADIMUSLIMPROJECT`; Syntax tambahan untuk menampilkan daftar nama database yang ada pada mysql menggunakan perintah : `SHOW DATABASES`;

```
CREATE DATABASE nama_database;
```

```
MariaDB [(none)]> create database ahmadimuslimproject2;  
Query OK, 1 row affected (0.01 sec)
```

```
MariaDB [(none)]>
```

Melihat database yang sudah dibuat

```
Show databases;
```

```
MariaDB [(none)]> show databases;
```

```
+-----+  
| Database |  
+-----+  
| ahmadimuslimproject2 |  
| bengkelkomputermuslim |  
| bengkelmuslimcmd |  
| gajiadttanzilcmd |  
| information_schema |  
| malasngoding |  
| muslim |  
| muslim_login |  
| mysql |  
| performance_schema |  
| phpmyadmin |  
| sbwidiastuti |  
| sippulsa |  
+-----+
```

```
13 rows in set (0.03 sec)
```

**b. Memilih Database : USE namadatabase;**

Sebelum membuat suatu tabel, terlebih dahulu harus memilih salah satu database sebagai database aktif yang akan digunakan untuk menyimpan tabel-tabel, Berikut ini perintah untuk menggunakan database dengan nama ahmadimuslimproject : USE ahmadimuslimproject;

Use nama\_database;

```
MariaDB [(none)]> use ahmadimuslimproject2;  
Database changed  
MariaDB [ahmadimuslimproject2]>
```

Menggunakan database

Use nama\_database;

**c. Syntax Menghapus Database : DROP DATABASE namadatabase;**

Database yang akan dihapus sesuai dengan namadatabase. Berikut ini perintah untuk menghapus database dengan nama rental : DROP DATABASE bengkelkomputermuslim;

```
MariaDB [bengkelmuslimcmd1]> drop database bengkelkomputermuslim;  
Query OK, 5 rows affected (0.10 sec)  
  
MariaDB [bengkelmuslimcmd1] > show databases;  
+-----+  
| Database |  
+-----+  
| ahmadimuslimproject2 |  
| aplikasisurat |  
| apoteksfl |  
| bengkelmuslimcmd |  
| db_absensi |  
| db_kas_masjid |  
| db_toko |  
| delia |  
| gajiadtanzilcmd |  
| information_schema |  
| inventory_barang |  
| jisung |  
| konveksi |  
| ksucatalog |  
| malasngoding |  
| miistore |  
| muslim |  
| muslim_login |  
| muslimproject2 |  
| mysql |  
| performance_schema |  
| perpustakaan |  
| phpmyadmin |  
| sbwidiastuti |  
| sippm |  
| sippulsa |  
| surat |  
| toko |  
+-----+  
28 rows in set (0.00 sec)
```

**d. Membuat Tabel : CREATE TABLE namatabel2 ( Field1 TipeData1,Field2 TipeData2);**

Nama tabel tidak boleh mengandung spasi (space). Field1 dan TipeData1 merupakan nama kolom pertama dan tipe data untuk kolom pertama. Jika ingin membuat tabel dengan kolom lebih dari satu, maka setelah pendefinisian tipe data sebelumnya diberikan tanda koma (,).

Membuat tabel sekaligus describe tabel tsb

```
Create table nama_tabel1 (  
Judul_field1 tippedata (panjang_data) not null auto_increment;  
Judul_field2 tippedata (panjang_data) not null;  
Judul_field3 tippedata (panjang_data) not null;  
Primary key (judul_field1),  
Index (judul_field2),  
constraint judul_field3 foreign key (judul_field3) references nama_tabel2  
(judul_field3)  
);
```

```

MariaDB [(none)]> use ahmadimuslimproject2;
Database changed
MariaDB [ahmadimuslimproject2]> create table pelanggan (
  -> id_pelanggan int not null,
  -> nama_pelanggan varchar (30) not null,
  -> desa_pelanggan varchar (50),
  -> kec_pelanggan varchar (30),
  -> hp_pelanggan varchar (14),
  -> primary key (id_pelanggan)
  -> );
Query OK, 0 rows affected (0.04 sec)

MariaDB [ahmadimuslimproject2]> desc pelanggan;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_pelanggan   | int(11)       | NO   | PRI | NULL    |       |
| nama_pelanggan | varchar(30)   | NO   |     | NULL    |       |
| desa_pelanggan | varchar(50)   | YES  |     | NULL    |       |
| kec_pelanggan  | varchar(30)   | YES  |     | NULL    |       |
| hp_pelanggan   | varchar(14)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.03 sec)

```

#### e. Menampilkan Tabel

Untuk menampilkan daftar nama tabel yang ada pada database yang sedangaktif/digunakan (dalam hal ini database rental) : SHOW TABLES;

Tampilkan tabel2 yang sudah berhasil dibuat:

Show tables;

```

MariaDB [ahmadimuslimproject2]> show tables;
+-----+
| Tables_in_ahmadimuslimproject2 |
+-----+
| layanan                         |
| pelanggan                       |
| pembayaran                     |
| petugas                        |
| transaksi                      |
+-----+
5 rows in set (0.00 sec)

```

#### f. Menampilkan Atribut/struktur Tabel : DESC namatabel;

Untuk menampilkan deskripsi tabel (dalam hal ini jenisfilm) syntaxnya adalah :DESC barang;

```

MariaDB [ahmadimuslimproject2]> desc pembayaran;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_bayar       | char(10)      | NO   | PRI | NULL    |       |
| id_transaksi   | char(10)      | YES  |     | NULL    |       |
| id_pelanggan   | char(10)      | YES  |     | NULL    |       |
| total_bayar    | int(11)       | NO   |     | NULL    |       |
| tanggal_bayar | date          | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)

```

#### g. Syntax Menghapus Tabel : DROP TABLE namatabel;

Tabel yang akan dihapus sesuai dengan namatabel, berikut ini perintah untukmenghapus tabel dengan nama jenisfilm : DROP TABLE BARANG;

```
MariaDB [ahmadimuslimproject21] > drop table kendaraan;
Query OK, 0 rows affected (0.02 sec)

MariaDB [ahmadimuslimproject21] > show tables;
+-----+
| Tables_in_ahmadimuslimproject2 |
+-----+
| layanan                          |
| pelanggan                       |
| pembayaran                     |
| petugas                        |
| transaksi                      |
+-----+
5 rows in set (0.00 sec)
```

#### h. Mendefinisikan Null/Not Null :

CREATE TABLE namatabel ( Field1 TipeData1 NOT NULL, Field2 TipeData2);

```
+-----+
| pembayaran : CREATE TABLE `pembayaran` (
| `id_bayar` int(11) NOT NULL,
| `id_transaksi` int(11) NOT NULL,
| `id_pelanggan` int(11) NOT NULL,
| `total_bayar` int(11) NOT NULL,
| `tanggal_bayar` date NOT NULL,
| PRIMARY KEY (`id_bayar`),
| KEY `id_transaksi` (`id_transaksi`),
| KEY `id_pelanggan` (`id_pelanggan`)
| > ENGINE=InnoDB DEFAULT CHARSET=latin1 ;
+-----+
```

#### i. Mendefinisikan Key Pada Tabel

Terdapat tiga cara untuk mendefinisikan primary key. Berikut ini adalah Syntax mendefinisikan primary key untuk Field1

CREATE TABLE namatabel(Field1 TipeData1 NOT NULL PRIMARY KEY, Field2 TipeData2);

Atau

```
MariaDB [bengkelmuslimcmd1] > create table layanan (
-> id_layanan int not null primary key,
-> nama_layanan varchar(50) not null,
-> harga_layanan int not null
-> );
Query OK, 0 rows affected (0.06 sec)
```

```
MariaDB [bengkelmuslimcmd1] > show tables;
```

```
+-----+
| Tables_in_bengkelmuslimcmd |
+-----+
| layanan                    |
| pelanggan                 |
| pembayaran                |
| petugas                   |
| transaksi                  |
+-----+
5 rows in set (0.00 sec)
```

```
MariaDB [bengkelmuslimcmd1] > desc layanan;
```

Field	Type	Null	Key	Default	Extra
id_layanan	int(11)	NO	PRI	NULL	
nama_layanan	varchar(50)	NO		NULL	
harga_layanan	int(11)	NO		NULL	

3 rows in set (0.02 sec)

CREATE TABLE namatabel ( Field1 TipeData1, Field2 TipeData2, PRIMARY KEY(Field1));

Atau

```

+
i layanan : CREATE TABLE `layanan` (
  `id_layanan` int(11) NOT NULL,
  `nama_layanan` varchar(50) NOT NULL,
  `harga_layanan` int(11) NOT NULL,
  PRIMARY KEY (`id_layanan`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ;

```

#### j. Menghapus Primary Key Pada Tabel

Cara 1 : Jika primary key dibuat dengan menggunakan alter table : ALTER TABLE namatabel DROP CONSTRAINT namaconstraint;

```

MariaDB [ahmadimuslimproject21]> alter table pembayaran drop foreign key
id_pelanggan;
Query OK, 0 rows affected (0.02 sec)
Records: 0 Duplicates: 0 Warnings: 0

```

```

MariaDB [ahmadimuslimproject21]> desc pembayaran;

```

Field	Type	Null	Key	Default	Extra
id_bayar	int(11)	NO	PRI	NULL	
id_transaksi	int(11)	NO	MUL	NULL	
id_pelanggan	int(11)	NO	MUL	NULL	
total_bayar	int(11)	NO		NULL	
tanggal_bayar	date	NO		NULL	

```

5 rows in set (0.02 sec)

```

Cara 2 : Jika primary key dibuat melalui create table : ALTER TABLE namatabel DROP PRIMARY KEY;

#### k. Menambah dan Menghapus Key Atribut di sebuah tabel (primary, foreign, index)

ALTER TABLE nama\_table1 DROP PRIMARY KEY;

```

MariaDB [bengkelmuslimcmd1]> alter table layanan drop primary key;
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0

```

```

MariaDB [bengkelmuslimcmd1]> desc layanan;

```

Field	Type	Null	Key	Default	Extra
no	int(11)	NO		NULL	
id_layanan	int(11)	NO		NULL	
nama_layanan	varchar(50)	NO		NULL	
harga_layanan	int(11)	NO		NULL	

```

4 rows in set (0.02 sec)

```

ALTER TABLE nama\_table1 ADD PRIMARY KEY(nama\_field1);

```

MariaDB [bengkelmuslimcmd1]> alter table layanan add primary key (id_layanan);
Query OK, 0 rows affected (0.07 sec)
Records: 0 Duplicates: 0 Warnings: 0

```

```

MariaDB [bengkelmuslimcmd1]> desc layanan;

```

Field	Type	Null	Key	Default	Extra
no	int(11)	NO		NULL	
id_layanan	int(11)	NO	PRI	NULL	
nama_layanan	varchar(50)	NO		NULL	
harga_layanan	int(11)	NO		NULL	

```

4 rows in set (0.02 sec)

```

Untuk meletakkan field diawal, tambahkan sintaks first :

ALTER TABLE PELANGAN ADD COLUMN KODE CHAR(5) FIRST;

```
MariaDB [bengkelmuslimcmd1]> desc layanan;
```

Field	Type	Null	Key	Default	Extra
no	int(11)	NO		NULL	
id_layanan	int(11)	NO	PRI	NULL	
nama_layanan	varchar(50)	NO		NULL	
harga_layanan	int(11)	NO		NULL	

4 rows in set (0.02 sec)

Untuk menyisipkan field setelah field tertentu, tambahkan sintaks after :

ALTER TABLE PELANGAN ADD COLUMN PHONE CHAR(5) AFTER ALAMAT;

ALTER TABLE nama\_tabel1 DROP nama\_field2;

```
MariaDB [ahmadimuslimproject21]> alter table kendaraan drop bahan_dasar;
```

Query OK, 0 rows affected (0.06 sec)  
Records: 0 Duplicates: 0 Warnings: 0

```
MariaDB [ahmadimuslimproject21]> desc kendaraan;
```

Field	Type	Null	Key	Default	Extra
id_kendaraan	int(11)	NO	PRI	NULL	
id_suplier	int(11)	YES	MUL	NULL	
nama_kendaraan	int(11)	NO		NULL	
kapasitas_cc	int(11)	NO	MUL	NULL	
tahun_produksi	int(11)	NO		NULL	
tanggal_beli	date	NO		NULL	
harga_jual	int(11)	NO		NULL	

7 rows in set (0.02 sec)

ALTER TABLE nama\_tabel DROP PRIMARY KEY;

```
MariaDB [ahmadimuslimproject21]> alter table kendaraan drop primary key;
```

Query OK, 0 rows affected (0.08 sec)  
Records: 0 Duplicates: 0 Warnings: 0

```
MariaDB [ahmadimuslimproject21]> desc kendaraan;
```

Field	Type	Null	Key	Default	Extra
id_kendaraan	int(11)	NO		NULL	
id_suplier	int(11)	YES	MUL	NULL	
nama_kendaraan	int(11)	NO		NULL	
kapasitas_cc	int(11)	NO	MUL	NULL	
tahun_produksi	int(11)	NO		NULL	
tanggal_beli	date	NO		NULL	
harga_jual	int(11)	NO		NULL	

7 rows in set (0.02 sec)

ALTER TABLE nama\_table1 ADD INDEX (nama\_field1);

```
MariaDB [ahmadimuslimproject21] alter table kendaraan add index (id_kendaraan);
Query OK, 0 rows affected (0.02 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [ahmadimuslimproject21] desc kendaraan;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_kendaraan | int(11)   | NO   | MUL | NULL    |       |
| id_suplier   | int(11)   | YES  | MUL | NULL    |       |
| nama_kendaraan | int(11)   | NO   |     | NULL    |       |
| kapasitas_cc | int(11)   | NO   | MUL | NULL    |       |
| tahun_produksi | int(11)   | NO   |     | NULL    |       |
| tanggal_beli | date      | NO   |     | NULL    |       |
| harga_jual   | int(11)   | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.02 sec)
```

ALTER TABLE nama\_table1 ADD FOREIGN KEY (nama\_field1) REFERENCES nama\_table2 (nama\_field1);

```
MariaDB [ahmadimuslimproject21] alter table kendaraan add foreign key (id_suplier) references suplier (id_suplier);
Query OK, 0 rows affected (0.13 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [ahmadimuslimproject21] desc kendaraan;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_kendaraan | int(11)   | NO   | PRI | NULL    |       |
| id_suplier   | int(11)   | YES  | MUL | NULL    |       |
| nama_kendaraan | int(11)   | NO   |     | NULL    |       |
| kapasitas_cc | int(11)   | NO   |     | NULL    |       |
| tahun_produksi | int(11)   | NO   |     | NULL    |       |
| tanggal_beli | int(11)   | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.02 sec)
```

SHOW CREATE TABLE: Perintah MySQL untuk Menampilkan Perintah Query Membuat struktur Table.

SHOW CREATE TABLE nama\_table1;

```
MariaDB [ahmadimuslimproject21] show create table kendaraan;
+-----+-----+
|      |      |
+-----+-----+
|      |      |
+-----+-----+
| kendaraan | CREATE TABLE `kendaraan` (
  `id_kendaraan` int(11) NOT NULL,
  `id_suplier` int(11) DEFAULT NULL,
  `nama_kendaraan` int(11) NOT NULL,
  `kapasitas_cc` int(11) NOT NULL,
  `tahun_produksi` int(11) NOT NULL,
  `tanggal_beli` date NOT NULL,
  `harga_jual` int(11) NOT NULL,
  KEY `kapasitas_cc` (`kapasitas_cc`),
  KEY `id_suplier` (`id_suplier`),
  KEY `id_kendaraan` (`id_kendaraan`),
  CONSTRAINT `kendaraan_ibfk_1` FOREIGN KEY (`id_suplier`) REFERENCES `suplier` (`id_suplier`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
+-----+-----+
```

ALTER TABLE nama\_table1 DROP foreign key nama\_table1\_ibfk\_3

```
MariaDB [ahmadimuslimproject21]> alter table kendaraan drop foreign key
kendaraan_ibfk_1;
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [ahmadimuslimproject21]> desc kendaraan;
+-----+-----+-----+-----+-----+-----+
| Field          | Type      | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_kendaraan   | int(11)   | NO   | MUL  | NULL    |       |
| id_suplier     | int(11)   | YES  | MUL  | NULL    |       |
| nama_kendaraan | int(11)   | NO   |      | NULL    |       |
| kapasitas_cc   | int(11)   | NO   | MUL  | NULL    |       |
| tahun_produksi | int(11)   | NO   |      | NULL    |       |
| tanggal_beli  | date      | NO   |      | NULL    |       |
| harga_jual     | int(11)   | NO   |      | NULL    |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.03 sec)
```

**l. Menambah Kolom Baru Pada Tabel : ALTER TABLE namatabel ADD fieldbaru tipe;**

Bila ingin merubah urutan field pada tabel di Mysql gunakan perintah berikut :

” ALTER TABLE nama\_table MODIFY COLUMN nama\_field tipe\_field AFTER nama\_field ”

Contoh kasus :

Misal di tabel mahasiswa ada field-field berikut :

1. Angkatan INT(4)
2. Nama VARCHAR (100)
3. Nim INT(10)
4. Alamat VARCHAR(255)

Nah, field NIM ingin anda tempatkan di posisi paling atas, caranya :

ALTER TABLE mahasiswa MODIFY COLUMN Nim INT(10) AFTER Angkatan

**m. Menambah Kolom Baru Pada Tabel : ALTER TABLE namatabel ADD fieldbaru tipe;**

Namatabel adalah nama tabel yang akan ditambah fieldnya. Fieldbaru adalah nama kolom yang akan ditambahkan, tipe adalah tipe data dari kolom yang akan ditambahkan.

Berikut ini contoh perintah untuk menambah kolom keterangan dengan tipe data varchar(25):

ALTER TABLE nama\_table ADD nama\_field1 tipe\_data (panjang\_data) after nama\_field2;

```
MariaDB [ahmadimuslimproject21]> alter table kendaraan add id_suplier
int after id_kendaraan;
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [ahmadimuslimproject21]> desc kendaraan;
+-----+-----+-----+-----+-----+-----+
| Field          | Type      | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_kendaraan   | int(11)   | NO   | PRI  | NULL    |       |
| id_suplier     | int(11)   | YES  |      | NULL    |       |
| nama_kendaraan | int(11)   | NO   |      | NULL    |       |
| kapasitas_cc   | int(11)   | NO   |      | NULL    |       |
| tahun_produksi | int(11)   | NO   |      | NULL    |       |
| tanggal_beli  | int(11)   | NO   |      | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.02 sec)

MariaDB [ahmadimuslimproject21]>
```



ALTER TABLE nama\_table1 ADD nama\_field1 tipe\_data (panjang\_data) NOT NULL;

```
MariaDB [ahmadimuslimproject21]> alter table kendaraan add harga int not null;
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [ahmadimuslimproject21]> desc kendaraan;
```

Field	Type	Null	Key	Default	Extra
id_kendaraan	int(11)	NO	PRI	NULL	
id_suplier	int(11)	YES	MUL	NULL	
nama_kendaraan	int(11)	NO		NULL	
kapasitas_cc	int(11)	NO	MUL	NULL	
tahun_produksi	int(11)	NO		NULL	
tanggal_beli	int(11)	NO		NULL	
harga	int(11)	NO		NULL	

```
7 rows in set (0.02 sec)
```

#### n. Mengubah Tipe Data atau Lebar Kolom Pada Tabel : ALTER TABLE NAMATABEL MODIFY COLUMN FIELD TIPE

Namatabel adalah nama tabel yang akan diubah tipe data atau lebar kolomnya. Field adalah kolom yang akan diubah tipe data atau lebarnya. Tipe adalah tipe data baru atau tipe data lama dengan lebar kolom yang berbeda. Berikut ini contoh perintah untuk mengubah tipe data untuk kolom keterangan dengan char(20) :

ALTER TABLE pelanggan MODIFY tgllahir tipe\_data (panjang\_data) NOT NULL;

```
MariaDB [ahmadimuslimproject21]> alter table kendaraan modify tanggal_beli date not null;
Query OK, 0 rows affected (0.09 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [ahmadimuslimproject21]> desc kendaraan;
```

Field	Type	Null	Key	Default	Extra
id_kendaraan	int(11)	NO	PRI	NULL	
id_suplier	int(11)	YES	MUL	NULL	
nama_kendaraan	int(11)	NO		NULL	
kapasitas_cc	int(11)	NO	MUL	NULL	
tahun_produksi	int(11)	NO		NULL	
tanggal_beli	date	NO		NULL	
harga	int(11)	NO		NULL	

```
7 rows in set (0.02 sec)

MariaDB [ahmadimuslimproject21]>
```

ALTER TABLE namatabel CHANGE nama\_lama nama\_baru tipe\_data (panjang\_data) after nama\_field1;

```
MariaDB [ahmadimuslimproject21]> alter table kendaraan change harga harga_jual int not null;
Query OK, 0 rows affected (0.04 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [ahmadimuslimproject21]> desc kendaraan;
```

Field	Type	Null	Key	Default	Extra
id_kendaraan	int(11)	NO	PRI	NULL	
id_suplier	int(11)	YES	MUL	NULL	
nama_kendaraan	int(11)	NO		NULL	
kapasitas_cc	int(11)	NO	MUL	NULL	
tahun_produksi	int(11)	NO		NULL	
tanggal_beli	date	NO		NULL	
harga_jual	int(11)	NO		NULL	

```
7 rows in set (0.03 sec)
```

ALTER TABLE nama\_tabel1 add nama\_field2 tipe\_data (panjang\_data) after nama\_field1;

```
MariaDB [ahmadimuslimproject21] alter table kendaraan add bahan_dasar v
archar (20) after kapasitas_cc;
Query OK, 0 rows affected (0.07 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [ahmadimuslimproject21] desc kendaraan;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_kendaraan | int(11)   | NO   | PRI | NULL    |       |
| id_suplier   | int(11)   | YES  | MUL | NULL    |       |
| nama_kendaraan | int(11)   | NO   |     | NULL    |       |
| kapasitas_cc | int(11)   | NO   | MUL | NULL    |       |
| bahan_dasar  | varchar(20) | YES  |     | NULL    |       |
| tahun_produksi | int(11)   | NO   |     | NULL    |       |
| tanggal_beli | date      | NO   |     | NULL    |       |
| harga_jual   | int(11)   | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.02 sec)
```

#### o. Mengubah Nama Kolom :

**ALTER TABLE namatabel CHANGE COLUMN namalamakolom namabarukolom tipedatarbaru;**

Namatabel adalah nama tabel yang akan diubah nama kolomnya, namalamakolom adalah kolom yang akan diganti namanya, namabarukolom adalah nama baru kolom, tipedatanya adalah tipe data dari kolom tersebut. Berikut ini contoh perintah untuk mengubah nama kolom keterangan menjadi ket :Merubah nama field harga-jual menjadi harga

```
MariaDB [ahmadimuslimproject21] alter table kendaraan change harga_jual
harga int;
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [ahmadimuslimproject21] desc kendaraan;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_kendaraan | int(11)   | NO   | MUL | NULL    |       |
| id_suplier   | int(11)   | YES  | MUL | NULL    |       |
| nama_kendaraan | int(11)   | NO   |     | NULL    |       |
| kapasitas_cc | int(11)   | NO   | MUL | NULL    |       |
| tahun_produksi | int(11)   | NO   |     | NULL    |       |
| tanggal_beli | date      | NO   |     | NULL    |       |
| harga       | int(11)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.03 sec)
```

#### p. Menambah dan Menghapus Kolom Pada Tabel :

ALTER TABLE namatabel DROP COLUMN namakolom;

ALTER TABLE nama\_tabel1 add nama\_field2 tipe\_data (panjang\_data) after nama\_field1;

```
MariaDB [ahmadimuslimproject21] alter table kendaraan add bahan_dasar v
archar (20) after kapasitas_cc;
Query OK, 0 rows affected (0.07 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [ahmadimuslimproject21] desc kendaraan;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_kendaraan | int(11)   | NO   | PRI | NULL    |       |
| id_suplier   | int(11)   | YES  | MUL | NULL    |       |
| nama_kendaraan | int(11)   | NO   |     | NULL    |       |
| kapasitas_cc | int(11)   | NO   | MUL | NULL    |       |
| bahan_dasar  | varchar(20) | YES  |     | NULL    |       |
| tahun_produksi | int(11)   | NO   |     | NULL    |       |
| tanggal_beli | date      | NO   |     | NULL    |       |
| harga_jual   | int(11)   | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.02 sec)
```

ALTER TABLE nama\_tabel1 DROP nama\_field2;

MariaDB [ahmadimuslimproject21] > alter table kendaraan drop bahan\_dasar;

Query OK, 0 rows affected (0.06 sec)

Records: 0 Duplicates: 0 Warnings: 0

MariaDB [ahmadimuslimproject21] > desc kendaraan;

Field	Type	Null	Key	Default	Extra
id_kendaraan	int(11)	NO	PRI	NULL	
id_suplier	int(11)	YES	MUL	NULL	
nama_kendaraan	int(11)	NO		NULL	
kapasitas_cc	int(11)	NO	MUL	NULL	
tahun_produksi	int(11)	NO		NULL	
tanggal_beli	date	NO		NULL	
harga_jual	int(11)	NO		NULL	

7 rows in set (0.02 sec)



## MATERI PERTEMUAN 15

### Data Control Language (DCL) dan Transaction Control Language (TCL)



#### TAHUKAH KAMU...?

- Cara membuat variabel dan menyimpan nilai di sana
- Cara mengambil nilai dari variabel
- Cara Mengenal tipe data apa saja yang dapat disimpan di dalam variabel
- Cara Konversi tipe data
- Cara Menghapus variabel dari memori

DCL adalah sub bahasa SQL yang berfungsi untuk melakukan pengontrolan data dan serverdatabasenya, seperti manipulasi user dan hak akses (priviledges). Yang termasuk perintah dalam DCL ada dua, yaitu GRANT dan REVOKE.

#### a. Grant

Perintah ini digunakan untuk memberikan hak akses oleh admin ke salah satu user atau pengguna. Hak akses tersebut bisa berupa hak membuat (CREATE), mengambil data (SELECT), menghapus data (DELETE), mengubah data (UPDATE), dan hak khusus lainnya yang berhubungan dengan sistem database.

Caranya buatlah user terlebih dahulu dengan perintah sebagai berikut:

```
CREATE USER "admin_sekolah"@"localhost" IDENTIFIED BY "admin";
```

Atau

```
CREATE USER "opt_sekolah"@"localhost" IDENTIFIED BY "opt_sekolah";
```

Kemudian berikan hak akses terhadap user yang telah dibuat dengan perintah sebagai berikut:

```
GRANT ALL PRIVILEGES ON sekolah.* TO "admin_sekolah"@"localhost";
```

*Catatan:* Hak akses ALL diberikan untuk user dapat menggunakan seluruh perintah SQL yakni CREATE, DROP, INSERT, UPDATE, DELETE, SELECT, ALTER

Untuk merubah password dapat dilakukan dengan perintah SQL seperti contoh sebagai berikut:

```
UPDATE mysql.user SET Password = PASSWORD('password_baru') WHERE User = 'root';
```

Sedangkan perintah SQL yang digunakan untuk menghapus user, dapat digunakan perintah sebagai berikut:

```
DROP user 'nama_user';
```

Perintah GRANT memungkinkan pemberian hak akses kepada pengguna

Tidak harus setiap pengguna database dapat mengakses seluruh data di database. Ada pengguna yang hanya dapat melakukan operasi di satu table saja. Bisa juga pengguna hanya dapat melakukan operasi SELECT saja tanpa bisa melakukan manipulasi data

Maka itulah diperlukan manajemen hak akses dengan GRANT

Sebelum kita menggunakan perintah GRANT terlebih dulu kita akan buat user di database dengan perintah

```
CREATE USER 'nama_user'@'localhost' IDENTIFIED BY 'password';
```

Note : untuk membuat user baru anda harus masuk sebagai root. Pahami cara masuk MySQL/MariaDB melalui terminal / command prompt menggunakan user root di Tutorial MySQL untuk pemula

Misal kita akan bikin user ahmadimuslim dengan password 12345

Login ke MySQL dengan username ahmadimuslim

```
$ ./mysql -u ahmadimuslim -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 209
Server version: 10.1.38-MariaDB Source distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]>
```

Saat akan melihat list database, user ahmadimuslim tidak dapat melihat keseluruhan data (berbeda jika melihat database dengan akses root)

Saat akan membuat database akan ditolak juga

Hal ini dikarenakan user ahmadimuslim tidak punya privilege terhadap database.

Mari kita beri akses dengan perintah GRANT melalui akses root dengan perintah

Tanda “ALL PRIVILEGES” dan asterik (\*) artinya user ahmadimuslim diberi akses untuk melakukan semua operasi seperti menambah, mengubah atau menghapus data di semua table / database

Coba masuk kembali dengan user ahmadimuslim dan buatlah database / table baru maka tidak akan bermasalah lagi

Beberapa opsi perintah GRANT antara lain

TIPE IZIN	KETERANGAN
ALL PRIVILEGES	Memberikan akses full
CREATE	Memberikan akses membuat table / database
DROP	Memberikan akses menghapus table / database
SELECT	Memberikan akses menambah record di table
INSERT	Memberikan akses merubah record di table
UPDATE	Memberikan akses menghapus record di table
DELETE	Memberikan akses menggunakan perintah SELECT

Cara menggunakannya adalah

```
GRANT tipe_izin ON nama_database.nama_table TO 'nama_user'@'localhost';
```

Contoh GRANT SELECT

Masuk ke user root dan berikan hak akses SELECT saja untuk user ahmadimuslim

Masuk ke user ahmadimuslim dan coba lakukan operasi SELECT dan INSERT

Operasi SELECT dibolehkan sedangkan operasi INSERT tidak bisa dilakukan

Contoh GRANT SELECT, INSERT, UPDATE, DELETE

Jika misalkan user hanya dibolehkan untuk melakukan perintah DML dan DQL tanpa akses merubah struktur table / database maka

#### b. Revoke

Perintah ini digunakan untuk mencabut hak akses yang telah diberikan kepada user. Dalam ini merupakan kebalikan dari perintah GRANT.

Contoh:

```
REVOKE SELECT ON universitas.* FROM 'admin_data'@'localhost'; REVOKE ALL ON dbsekolah.* TO 'user1'@'localhost';
```

Perintah REVOKE digunakan untuk mencabut kembali hak akses yang diberikan melalui perintah GRANT

Cara menggunakannya

```
REVOKE tipe_izin ON nama_database.nama_table FROM 'username'@'localhost';
```

Jika kita ingin menghapus akses INSERT di user ahmadimuslim untuk semua database dan table melalui root

Masuk menggunakan user ahmadimuslim dan coba masukkan record baru

Untuk mencabut seluruh hak akses user dapat menggunakan REVOKE ALL

```
REVOKE ALL ON nama_database.nama_table FROM 'username'@'localhost';
```

### 1. Transaction Control Language (TCL)

Transaction Control Language (TCL) adalah perintah SQL yang berhubungan dengan transaksi di database  
Perintah TCL antara lain

COMMIT -> Menyimpan transaksi secara permanen

ROLLBACK -> Mengembalikan database ke bentuk awal / COMMIT terakhir

## #1 Perintah COMMIT

Perintah COMMIT digunakan untuk menyimpan transaksi secara permanen di database

Saat melakukan perintah DML seperti INSERT, UPDATE, DELETE transaksi sebenarnya belum dilakukan secara permanen. Artinya operasi tersebut masih bisa di rollback / di batalkan

Jika ingin menyimpan transaksi sehingga tidak dapat di rollback kita gunakan perintah COMMIT

Kapan perintah COMMIT dibutuhkan?

Dalam suatu rangkaian operasi data, jika ada 1 atau lebih operasi yang mengalami kegagalan maka kita akan mengembalikan seperti ke bentuk semula. Jika tidak ada kesalahan maka seluruh rangkaian pernyataan akan di – COMMIT untuk menyimpan transaksi secara permanen

Gunakan Database universitas dan Table mahasiswa yang telah kita buat di materi DDL

INSERT INTO mahasiswa

VALUES

(21400200,"faqih","bandung"),

(21400201,"ina","jakarta"),

(21400202,"anto","semarang"),

(21400203,"dani","padang");

Selanjutnya kita cek data yang telah diinputkan

```
> SELECT * FROM mahasiswa;
```

```
+-----+-----+-----+
| nim | nama | alamat |
+-----+-----+-----+
| 21400200 | faqih | bandung |
| 21400201 | ina | jakarta |
| 21400202 | anto | semarang |
| 21400203 | dani | padang |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

Untuk memulai menggunakan COMMIT harus dimulai dengan

```
START TRANSACTION;
```

```
> START TRANSACTION;
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
> INSERT INTO mahasiswa VALUES (21400210,"Jaka","Kalimantan");
```

```
Query OK, 1 row affected (0.00 sec)
```

```
> COMMIT;
Query OK, 0 rows affected (0.07 sec)
```

```
> SELECT * FROM mahasiswa;
+-----+-----+-----+
| nim  | nama  | alamat |
+-----+-----+-----+
| 21400200 | faqih | bandung  |
| 21400201 | ina   | jakarta  |
| 21400202 | anto  | semarang |
| 21400203 | dani  | padang   |
| 21400210 | Jaka  | Kalimantan |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

## #2 Perintah ROLLBACK

Perintah ROLLBACK digunakan untuk mengembalikan database ke bentuk awal / COMMIT terakhir

Perintah COMMIT dan ROLLBACK saling berkaitan

Kapan perintah ROLLBACK dibutuhkan?

Dalam suatu rangkaian operasi data, jika ada 1 atau lebih operasi yang mengalami kegagalan maka kita akan mengembalikan seperti ke bentuk semula menggunakan perintah ROLLBACK

Untuk menggunakan COMMIT / ROLLBACK harus dimulai dengan

```
START TRANSACTION;
> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
```

```
> SELECT * FROM mahasiswa;
+-----+-----+-----+
| nim  | nama  | alamat |
+-----+-----+-----+
| 21400200 | faqih | bandung  |
| 21400201 | ina   | jakarta  |
| 21400202 | anto  | semarang |
| 21400203 | dani  | padang   |
| 21400210 | Jaka  | Kalimantan |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
> INSERT INTO mahasiswa VALUES (21400211,"Fitri","Surabaya");
Query OK, 1 row affected (0.00 sec)
```



```
> SELECT * FROM mahasiswa;
```

```
+-----+-----+-----+
| nim | nama | alamat |
+-----+-----+-----+
| 21400200 | faqih | bandung |
| 21400201 | ina | jakarta |
| 21400202 | anto | semarang |
| 21400203 | dani | padang |
| 21400210 | Jaka | Kalimantan |
| 21400211 | Fitri | Surabaya |
+-----+-----+-----+
```

6 rows in set (0.00 sec)

```
> ROLLBACK;
```

Query OK, 0 rows affected (0.11 sec)

```
> SELECT * FROM mahasiswa;
```

```
+-----+-----+-----+
| nim | nama | alamat |
+-----+-----+-----+
| 21400200 | faqih | bandung |
| 21400201 | ina | jakarta |
| 21400202 | anto | semarang |
| 21400203 | dani | padang |
| 21400210 | Jaka | Kalimantan |
+-----+-----+-----+
```

5 rows in set (0.00 sec)

Saat kita gunakan perintah ROLLBACK akan kembali ke database awal



## MATERI PERTEMUAN 16-17

### SQL ; Data Manipulation Language (DML)



#### TAHUKAH KAMU...?

Ada 6 Jenis operator dalam pemrograman PHP yang harus kita ketahui:

- Operator Aritmatika;
- Operator Penugasan atau *Assignment*;
- Operator *Increment & Decrement*;
- Operator Relasi atau pembandingan;
- Operator Logika;
- Operator Bitwise;
- Operator Ternary.

DML (Data Manipulation Language) DML adalah kelompok perintah yang berfungsi untuk memanipulasi data dalam basis data, misalnya untuk pengambilan, penyisipan, pengubahan dan penghapusan data. Perintah yang termasuk dalam kategori DML adalah : INSERT, DELETE, UPDATE dan SELECT.

#### a. I N S E R T

Perintah INSERT digunakan untuk menambahkan baris pada suatu tabel. Terdapat dua cara untuk menambah baris, yaitu:

Lihat dulu struktur fisikal tabel yang akan diisi/insert datanya

```
MariaDB [ahmadimuslimproject21]> desc layanan;
```

Field	Type	Null	Key	Default	Extra
id_layanan	int(11)	NO	PRI	NULL	
nama_layanan	varchar(50)	NO		NULL	
harga_layanan	int(11)	NO		NULL	

3 rows in set (0.02 sec)

INSERT INTO nama\_table SET

nama\_kolom1="Varibel1",

nama\_kolom2="Varibel2",

nama\_kolom3="Varibel3";

Atau

INSERT INTO nama\_tabel VALUES (data1, data2, dst...);

```
MariaDB [ahmadimuslimproject21] > insert into layanan values ('layanan01', 'ganti hardisk', '100000');
Query OK, 1 row affected, 1 warning (0.00 sec)

MariaDB [ahmadimuslimproject21] > select *from layanan;
+-----+-----+-----+
| id_layanan | nama_layanan | harga_layanan |
+-----+-----+-----+
| 0 | ganti hardisk | 100000 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Field utama (primary key) dari setiap tabel harus di rubah tipe datanya menjadi char dengan panjang data 10

```
MariaDB [ahmadimuslimproject21] > alter table layanan modify id_layanan char (10);
Query OK, 1 row affected (0.10 sec)
Records: 1 Duplicates: 0 Warnings: 0

MariaDB [ahmadimuslimproject21] > desc layanan;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_layanan | char(10) | NO | PRI | NULL | |
| nama_layanan | varchar(50) | NO | | NULL | |
| harga_layanan | int(11) | NO | | NULL | |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.02 sec)

MariaDB [ahmadimuslimproject21] >
```

Lalu rubah data integer (number) yang sudah terlanjur dimasukkan tadi menjadi tipe karakter (data string) dengan sintaks:

Update namatabel1 set namafield1='variabelbaru' where namafieldutama='keyatributvariabel';

```
MariaDB [ahmadimuslimproject21] > update layanan set id_layanan='1y001' where id_layanan='0';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

MariaDB [ahmadimuslimproject21] > select *from layanan;
+-----+-----+-----+
| id_layanan | nama_layanan | harga_layanan |
+-----+-----+-----+
| 1y001 | ganti hardisk | 100000 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Memasukkan data dalam jumlah banyak

Insert into namatabel1

Values

('variable1','variable2','variabel3','dst'),

('variable1','variable2','variabel3','dst'),

('variable1','variable2','variabel3','dst'),

(‘variable1’,‘variable2’,‘variabel3’,‘dst’);

```
MariaDB [lahmadimuslimproject21] insert into layanan
-> values
-> ('ly002','ganti ram','75000'),
-> ('ly003','ganti LCD','150000'),
-> ('ly004','ganti processor','100000');
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

Keterangan :

Jika data bertipe string, date atau time (contoh : action, horror, 2007-11-10) maka pemberian nilainya diapit dengan tanda petik tunggal ('horor') atau petik ganda ("horor"). Jika data bertipe numerik (2500, 400) maka pemberian nilainya tidak diapit tanda petik tunggal maupun ganda.

#### b. Delete

Perintah DELETE digunakan untuk menghapus satu baris, baris dengan kondisi tertentu atauseluruh baris. Syntax : DELETE FROM namatabel [WHERE kondisi];

Perintah dalam tanda [ ] bersifat opsional untuk menghapus suatu baris dengan suatukondisi tertentu. DELETE FROM nama\_tabel WHERE kolom=data;

```
MariaDB [lahmadimuslimproject21] delete from petugas where id_petugas='pt011';
Query OK, 1 row affected (0.00 sec)

MariaDB [lahmadimuslimproject21] select *from petugas;
```

id_petugas	nama_petugas	desa_petugas	kec_petugas	hp_petugas	jabatan
pt001	AHMADI MUSLIM	PAYA RAJA	banda mulia	82980342883	PIMPRO
pt002	Zainal Abidin	Tanah Terban	Karang Baru	82980342883	Bendaharawan
pt003	M. Yusuf	Medang Ara	Kuala Simpang	82980342883	Kabid Litbang
pt004	Muhammad	Paya Bedi	Karang Baru	82980342883	Kabit Humas
pt005	Suprayetno	Selamat	Kuala Simpang	82980342883	Kabid Sarana
pt006	M. Sabar	Pahlawan	Karang Baru	82980342883	Kabid Umum
pt007	Syawaluddin Lubis	Bundar	Kuala Simpang	82980342883	Aisten 1
pt008	Suzainal	Sungai Liput	Karang Baru	82980342883	Aisten 2
pt009	Sabaruddin	Landuh	Kuala Simpang	82980342883	Aisten 3
pt010	Hardianto	Pahlawan	Karang Baru	82980342883	Aisten 4
pt012	Kamal Ruzama	Sungai Liput	Kuala Simpang	82980342883	Aisten 6
pt013	Yoni Irawadi	Sungai Liput	Kuala Simpang	82980342883	Aisten 7

12 rows in set (0.00 sec)

DELETE FROM nama\_tabel;

Untuk menghapus seluruh data/record yang ada pada tabel:

#### c. Update

Perintah UPDATE digunakan untuk mengubah isi data pada satu atau beberapa kolom padasuatu table. Syntax :

UPDATE namatabel SET kolom1 = nilai1, kolom2 = nilai2 [WHERE kondisi];

UPDATE nama\_tabel SET kolom1=data1, kolom2=data2,... WHERE kolomprimary=dataprimery;

```
MariaDB [ahmadimuslimproject21]> select *from petugas;
```

id_petugas	nama_petugas	desa_petugas	kec_petugas	hp_petugas	jabatan
pt001	Nazaruddin	Paya Bedi	Kuala Simpang	82980342883	KTU
pt002	Zainal Abidin	Tanah Terban	Karang Baru	82980342883	Bendaharawan
pt003	M. Yusuf	Medang Ara	Kuala Simpang	82980342883	Kabid Lithang
pt004	Muhammad	Paya Bedi	Karang Baru	82980342883	Kabit Humas
pt005	Suprayetno	Selamat	Kuala Simpang	82980342883	Kabid Sarana
pt006	M. Sabar	Pahlawan	Karang Baru	82980342883	Kabid Umum
pt007	Syawaluddin Lubis	Bundar	Kuala Simpang	82980342883	Aisten 1
pt008	Suzainal	Sungai Liput	Karang Baru	82980342883	Aisten 2
pt009	Sabaruddin	Landuh	Kuala Simpang	82980342883	Aisten 3
pt010	Hardianto	Pahlawan	Karang Baru	82980342883	Aisten 4
pt011	H. Ajad Sabiluddin	Bundar	Kuala Simpang	82980342883	Aisten 5
pt012	Kamal Ruzama	Sungai Liput	Kuala Simpang	82980342883	Aisten 6
pt013	Yoni Irawadi	Sungai Liput	Kuala Simpang	82980342883	Aisten 7

```
13 rows in set (0.00 sec)
```

```
MariaDB [ahmadimuslimproject21]> update petugas set nama_petugas='ABANG MUSLIM', desa_petugas='PAYA RAJA', kec_petugas='banda mulia', jabatan='PIMPRO' where id_petugas='pt011';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

MariaDB [ahmadimuslimproject21]> update petugas set nama_petugas='AHMADI MUSLIM', desa_petugas='PAYA RAJA', kec_petugas='banda mulia', jabatan='PIMPRO' where id_petugas='pt001';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

MariaDB [ahmadimuslimproject21]> select *from petugas;
```

id_petugas	nama_petugas	desa_petugas	kec_petugas	hp_petugas	jabatan
pt001	AHMADI MUSLIM	PAYA RAJA	banda mulia	82980342883	PIMPRO
pt002	Zainal Abidin	Tanah Terban	Karang Baru	82980342883	Bendaharawan
pt003	M. Yusuf	Medang Ara	Kuala Simpang	82980342883	Kabid Lithang
pt004	Muhammad	Paya Bedi	Karang Baru	82980342883	Kabit Humas
pt005	Suprayetno	Selamat	Kuala Simpang	82980342883	Kabid Sarana
pt006	M. Sabar	Pahlawan	Karang Baru	82980342883	Kabid Umum
pt007	Syawaluddin Lubis	Bundar	Kuala Simpang	82980342883	Aisten 1
pt008	Suzainal	Sungai Liput	Karang Baru	82980342883	Aisten 2
pt009	Sabaruddin	Landuh	Kuala Simpang	82980342883	Aisten 3
pt010	Hardianto	Pahlawan	Karang Baru	82980342883	Aisten 4
pt011	ABANG MUSLIM	PAYA RAJA	banda mulia	082391827471	PIMPRO
pt012	Kamal Ruzama	Sungai Liput	Kuala Simpang	82980342883	Aisten 6
pt013	Yoni Irawadi	Sungai Liput	Kuala Simpang	82980342883	Aisten 7

```
13 rows in set (0.00 sec)
```

Perintah dalam tanda [ ] bersifat opsional untuk mengubah suatu baris dengan suatu kondisi tertentu.

#### d. Select

Perintah SELECT digunakan untuk menampilkan isi dari suatu tabel yang dapat dihubungkan dengan tabel yang lainnya.

- 1) Menampilkan data untuk semua kolom menggunakan asterisk (\*) :SELECT \*  
FROM namatabel;

Melihat isi data / record yang telah dimasukkan

Select \*from namatabel1;

```
MariaDB [ahmadimuslimproject21]> select *from layanan;
```

id_layanan	nama_layanan	harga_layanan
ly001	ganti hardisk	100000
ly002	ganti ram	75000
ly003	ganti LCD	150000
ly004	ganti processor	100000

```
4 rows in set (0.00 sec)
```

2) Menampilkan data untuk kolom tertentu :

SELECT kolom1,kolom2,kolom-n FROM namatabel;

SELECT nama\_kolom1, nama\_kolom2 FROM nama\_tabel;

```
MariaDB [ahmadimuslimproject21]> select id_pelanggan, total_bayar from pembayaran;
```

id_pelanggan	total_bayar
pl0001	410000
pl0002	230000
pl0003	620000
pl0004	500000

4 rows in set (0.00 sec)

```
MariaDB [ahmadimuslimproject21]> select id_pelanggan, nama_pelanggan from pelanggan;
```

id_pelanggan	nama_pelanggan
pl0001	Aprilia Lestari
pl0002	Arief Rizqi Faddilah
pl0003	Aulia Rahman
pl0004	Benni Ismail
pl0005	Deby Fahriza. D
pl0006	Desi Rahmadani
pl0007	Ditya Hermawan
pl0008	Dwi Armaya
pl0009	Firnanda Effendie Putra
pl0010	Fitria Mira Andela
pl0011	Fitria Yusli
pl0012	Goval Rahmanda
pl0013	Juwan Syahputra. S
pl0014	Lismayni
pl0015	M. Rais Syahizinda
pl0016	Maisyarah
pl0017	Marchellia Qientan Sayuti
pl0018	Mona Justisia
pl0019	Muhammad Alfa Rizzi
pl0020	Prilinurhaliza
pl0021	Putri Wahyuni
pl0022	Rapikah Hasanah

22 rows in set (0.00 sec)

```
MariaDB [ahmadimuslimproject21]> select nama_petugas, desa_petugas from petugas;
```

nama_petugas	desa_petugas
Nazaruddin	Paya Bedi
Zainal Abidin	Tanah Terban
M. Yusuf	Medang Ara
Muhammad	Paya Bedi
Suprayetno	Selamat
M. Sabar	Pahlawan
Syawaluddin Lubis	Bundar
Suzainal	Sungai Liput
Sabaruddin	Landuh
Hardianto	Pahlawan
H. Ajad Sabiluddin	Bundar
Kamal Ruzama	Sungai Liput
Yoni Irawadi	Sungai Liput

13 rows in set (0.00 sec)

- 3) Menampilkan data dengan kondisi data tertentu dengan klausa WHERE:  
SELECT \* FROM namatabel WHERE kondisi;

```
MariaDB [ahmadimuslimproject21] > select harga_layanan as harga_produk from layanan where harga_layanan <='200000';
```

harga_produk
50000
80000
50000
50000
35000
30000
25000
40000
25000
200000
80000
25000
25000
100000
100000
100000
25000
25000
25000
25000
25000

21 rows in set (0.00 sec)

Beberapa operator perbandingan yang dapat digunakan pada klausa WHERE adalah "="(sama dengan), > (lebih dari), < (kurang dari), <> (tidak sama dengan), >= (lebih dari atau sama dengan), <= (kurang dari atau sama dengan). Adapun operator lain, yaitu: AND, OR, NOT, BETWEEN-AND, IN dan LIKE.

- 4) Memberikan nama lain pada kolom :

SELECT namakolomlama AS namakolombaruFROM namatabel;

Berikut ini perintah untuk memberikan nama lain pada kolom jenis menjadi jenis\_film pada tabel jenisfilm:

```
MariaDB [ahmadimuslimproject21] > select harga_layanan as harga_produk from layanan;
```

harga_produk
50000
80000
50000
50000
35000
30000
25000
40000
25000
200000
80000
25000
25000
100000
100000
100000
25000
25000
25000
25000
25000

21 rows in set (0.00 sec)

SELECT JENIS AS TYPE FROM JENISFILM;

- 5) Menggunakan alias untuk nama tabel:

SELECT namalias .jenis, namalias .hargaFROM namatabel namalias;

Berikut ini perintah untuk memberikan alias pada tabel jenisfilm :

SELECT J.JENIS, J.HARGA FROM JENISFILM J;

6) Menampilkan data lebih dari dua tabel:

```
SELECT * FROM namatabel1, namatabel2,namatabel-n;
```

7) Nested Queries / Subquery (IN, NOT IN, EXISTS, NOT EXISTS)

Subquery berarti query di dalam query. Dengan menggunakan subquery, hasil dari query akan menjadi bagian dari query di atasnya. Subquery terletak di dalam klausa WHERE atau HAVING. Pada klausa WHERE, subquery digunakan untuk memilih baris-baris tertentu yang kemudian digunakan oleh query. Sedangkan pada klausa

HAVING, subquery digunakan untuk memilih kelompok baris yang kemudian digunakan oleh query.

Contoh 1: perintah untuk menampilkan data pada tabel jenisfilm yang mana data pada kolom jenis-nya tercantum pada tabel film menggunakan IN :

```
SELECT * FROM JENISFILM WHERE JENIS IN (SELECT JENIS FROM FILM);
```

```
MariaDB [ahmadinuslinproject21] > select *from transaksi where exists (select *from layanan where harga_layanan <100000);
```

id_transaksi	id_petugas	id_layanan	jumlah	sub_total
tr00001	pt002	ly01	1	50000
tr00001	pt002	ly02	1	80000
tr00001	pt002	ly03	1	50000
tr00001	pt002	ly04	2	100000
tr00001	pt002	ly05	2	70000
tr00001	pt002	ly06	2	60000
tr00002	pt003	ly04	1	50000
tr00002	pt003	ly05	2	70000
tr00002	pt003	ly06	2	60000
tr00002	pt003	ly07	2	50000
tr00003	pt001	ly02	4	320000
tr00003	pt001	ly03	6	300000
tr00004	pt001	ly04	10	500000

13 rows in set (0.00 sec)

```
MariaDB [ahmadinuslinproject21] > select *from transaksi;
```

id_transaksi	id_petugas	id_layanan	jumlah	sub_total
tr00001	pt002	ly01	1	50000
tr00001	pt002	ly02	1	80000
tr00001	pt002	ly03	1	50000
tr00001	pt002	ly04	2	100000
tr00001	pt002	ly05	2	70000
tr00001	pt002	ly06	2	60000
tr00002	pt003	ly04	1	50000
tr00002	pt003	ly05	2	70000
tr00002	pt003	ly06	2	60000
tr00002	pt003	ly07	2	50000
tr00003	pt001	ly02	4	320000
tr00003	pt001	ly03	6	300000
tr00004	pt001	ly04	10	500000

13 rows in set (0.00 sec)

atau menggunakan EXISTS

```
SELECT * FROM JENISFILM WHERE EXISTS (SELECT * FROM FILM WHERE HARGA > 2000);
```

Pada contoh di atas:

SELECT JENIS FROM FILM disebut subquery, sedangkan :

SELECT \* FROM JENISFILM berkedudukan sebagai query. Perhatikan, terdapat data jenis dan harga pada tabel jenisfilm yang tidak ditampilkan. Hal ini disebabkan data pada kolom jenis tidak terdapat pada kolom jenis di tabel film.

Contoh 2: perintah untuk menampilkan data pada tabel jenisfilm yang mana data pada kolom jenis-nya tidak tercantum pada tabel film menggunakan NOT IN:

```
SELECT * FROM JENISFILM WHERE JENIS NOT IN (SELECT JENIS FROM FILM);
```



```
MariaDB [ahmadimuslimproject21] > select * from transaksi where id_layanan not in (select nama_layanan from layanan);
```

id_transaksi	id_petugas	id_layanan	jumlah	sub_total
tr00001	pt002	ly01	1	50000
tr00001	pt002	ly02	1	80000
tr00001	pt002	ly03	1	50000
tr00001	pt002	ly04	2	100000
tr00001	pt002	ly05	2	70000
tr00001	pt002	ly06	2	60000
tr00002	pt003	ly04	1	50000
tr00002	pt003	ly05	2	70000
tr00002	pt003	ly06	2	60000
tr00002	pt003	ly07	2	50000
tr00003	pt001	ly02	4	320000
tr00003	pt001	ly03	6	300000
tr00004	pt001	ly04	10	500000

```
13 rows in set (0.00 sec)
```

atau menggunakan NOT EXISTS

```
SELECT * FROM JENISFILM WHERE NOT EXISTS (SELECT * FROM FILM WHERE HARGA > 2000);
```

#### 8) Operator comparison ANY dan ALL

Operator ANY digunakan berkaitan dengan subquery. Operator ini menghasilkan TRUE(benar) jika paling tidak salah satu perbandingan dengan hasil subquery menghasilkan

nilai TRUE. Ilustrasinya jika:

```
MariaDB [ahmadimuslimproject21] > select * from layanan where harga_layanan > any(select harga_layanan from layanan);
```

id_layanan	nama_layanan	harga_layanan
ly01	Instal Ulang (Paket Standar)	50000
ly02	Instal Ulang (Paket Komplit)	80000
ly03	Hang / Freeze	50000
ly04	Layar Biru / Bluescreen	50000
ly05	Lambat / Lemot	35000
ly06	Sering Mati Tiba Tiba	30000
ly08	Blank / Tidak Ada Tampilan	40000
ly10	Mati Total	200000
ly11	Kadang Nyala Kadang Tidak	80000
ly14	Tampilan Laptop Bergaris Horizontal/Vertical	100000
ly15	Gagal Instal Ulang	100000
ly16	Tidak Keluar Suara	100000

```
12 rows in set (0.00 sec)
```

Gaji > ANY (S)

Jika subquery S menghasilkan G1, G2, ..., Gn, maka kondisi di atas identik dengan: (gaji > G1) OR (gaji > G2) OR ... OR (gaji > Gn)

Contoh: perintah untuk menampilkan semua data jenisfilm yang harganya bukannya yang terkecil:

```
SELECT * FROM JENISFILM WHERE HARGA > ANY (SELECT HARGA FROM JENISFILM);
```

Operator ALL digunakan untuk melakukan perbandingan dengan subquery. Kondisi dengan ALL menghasilkan nilai TRUE (benar) jika subquery tidak menghasilkan apapun atau jika perbandingan menghasilkan TRUE untuk setiap nilai query terhadap hasil subquery.

Contoh : perintah untuk menampilkan data jenisfilm yang harganya paling tinggi: SELECT \* FROM JENISFILM WHERE HARGA >= ALL (SELECT HARGA FROM JENISFILM);

#### 9) Sintak ORDER BY

Klausula ORDER BY digunakan untuk mengurutkan data berdasarkan kolom tertentu sesuai dengan tipe data yang dimiliki. Contoh : perintah untuk mengurutkan data film berdasarkan kolom judul:

```
SELECT * FROM FILM ORDER BY JUDUL;
```

atau tambahkan ASC untuk pengurutan secara ascending (menaik) :SELECT \*  
FROM FILM ORDER BY JUDUL ASC;

atau tambahkan DESC untuk pengurutan secara descending (menurun):SELECT \*  
FROM FILM ORDER BY JUDUL DESC;

SELECT \* FROM nama\_tabel order by kolom\_dipilih ASC;

```
MariaDB [ahmadimuslimproject21]> select *from layanan order by harga_layanan asc;
```

id_layanan	nama_layanan	harga_layanan
ly21	Hardware Tidak Terdeteksi	25000
ly13	Laptop/Pc Kepanasan	25000
ly17	Tampilan Layar Besar Besar / Pecah Pecah	25000
ly09	Ada Bunyi Tiiit	25000
ly07	Restart Tiba Tiba	25000
ly18	Tidak Bisa Masuk Windows / Loading Terus	25000
ly19	Kena Virus (Pembersihan Virus)	25000
ly20	Tidak Bisa Koneksi Internet	25000
ly12	Ada Bunyi Berderik	25000
ly06	Sering Mati Tiba Tiba	30000
ly05	Lambat / Lemot	35000
ly08	Blank / Tidak Ada Tampilan	40000
ly01	Instal Ulang (Paket Standar)	50000
ly04	Layar Biru / Bluescreen	50000
ly03	Hang / Freeze	50000
ly02	Instal Ulang (Paket Komplit)	80000
ly11	Kadang Nyala Kadang Tidak	80000
ly14	Tampilan Laptop Bergaris Horisontal/Vertical	100000
ly15	Gagal Instal Ulang	100000
ly16	Tidak Keluar Suara	100000
ly10	Mati Total	200000

21 rows in set (0.00 sec)

SELECT \* FROM nama\_tabel order by kolom\_dipilih DESC;

```
MariaDB [ahmadimuslimproject21]> select *from layanan order by harga_layanan desc;
```

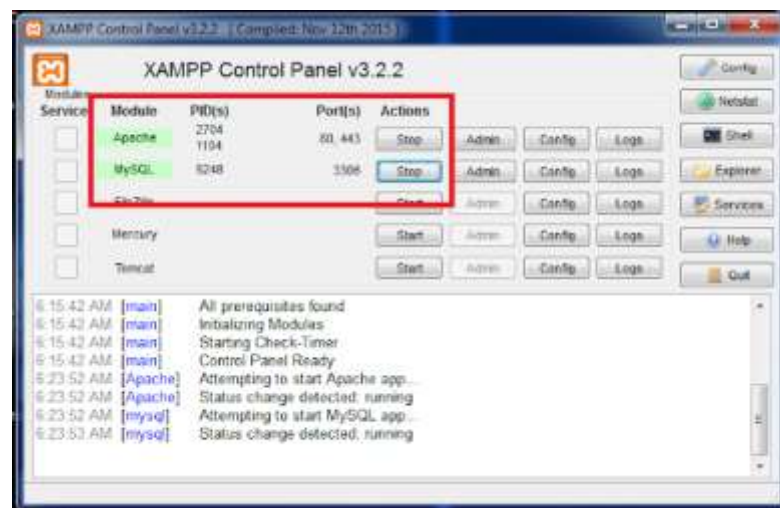
id_layanan	nama_layanan	harga_layanan
ly10	Mati Total	200000
ly16	Tidak Keluar Suara	100000
ly15	Gagal Instal Ulang	100000
ly14	Tampilan Laptop Bergaris Horisontal/Vertical	100000
ly11	Kadang Nyala Kadang Tidak	80000
ly02	Instal Ulang (Paket Komplit)	80000
ly01	Instal Ulang (Paket Standar)	50000
ly04	Layar Biru / Bluescreen	50000
ly03	Hang / Freeze	50000
ly08	Blank / Tidak Ada Tampilan	40000
ly05	Lambat / Lemot	35000
ly06	Sering Mati Tiba Tiba	30000
ly20	Tidak Bisa Koneksi Internet	25000
ly19	Kena Virus (Pembersihan Virus)	25000
ly18	Tidak Bisa Masuk Windows / Loading Terus	25000
ly17	Tampilan Layar Besar Besar / Pecah Pecah	25000
ly09	Ada Bunyi Tiiit	25000
ly07	Restart Tiba Tiba	25000
ly13	Laptop/Pc Kepanasan	25000
ly12	Ada Bunyi Berderik	25000
ly21	Hardware Tidak Terdeteksi	25000

21 rows in set (0.00 sec)

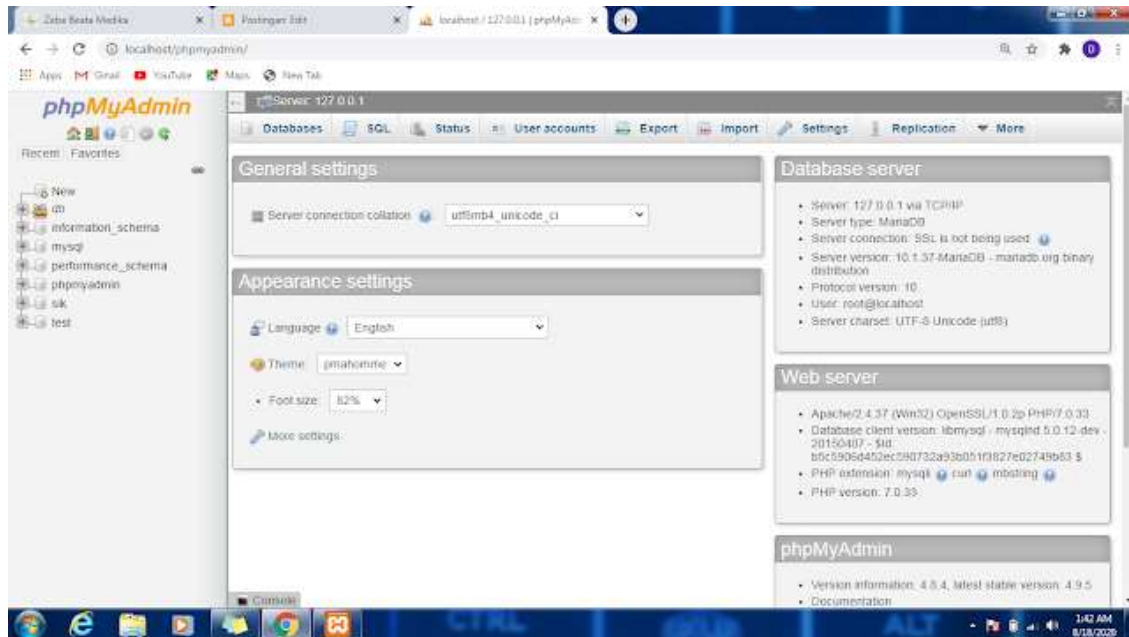
#### A. Insert Data Melalui CSV Ms Excel

Salah satu hal yang tak kalah penting dalam basis data adalah memasukkan data/record yang merupakan row atau baris pada entitas atau tabel. Sebenarnya kita bisa langsung input data secara langsung di mysql nya, namun bagaimana jika data yang akan diinput berjumlah ribuan? tentu saja tidak mungkin untuk harus input satu persatu. Untuk itu dalam kasus ini kita bisa langsung import datanya kedalam database mysql langsung dari data didalam file dengan extensi (.csv). Langsung saja kita ikuti langkah-langkah berikut ini untuk membuat import data excel ke mysql:

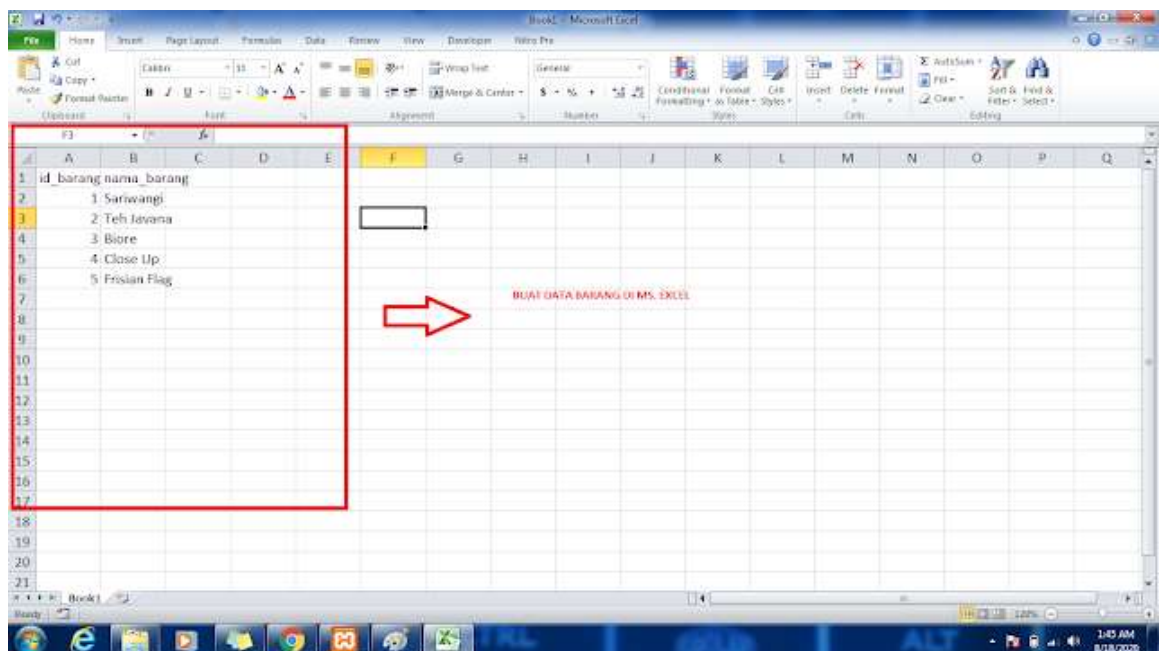
1. Silahkan buka terlebih dahulu xampp sebagai server local lalu menjalankan apache dan mysql dengan cara klik tombol start, seperti gambar dibawah ini:



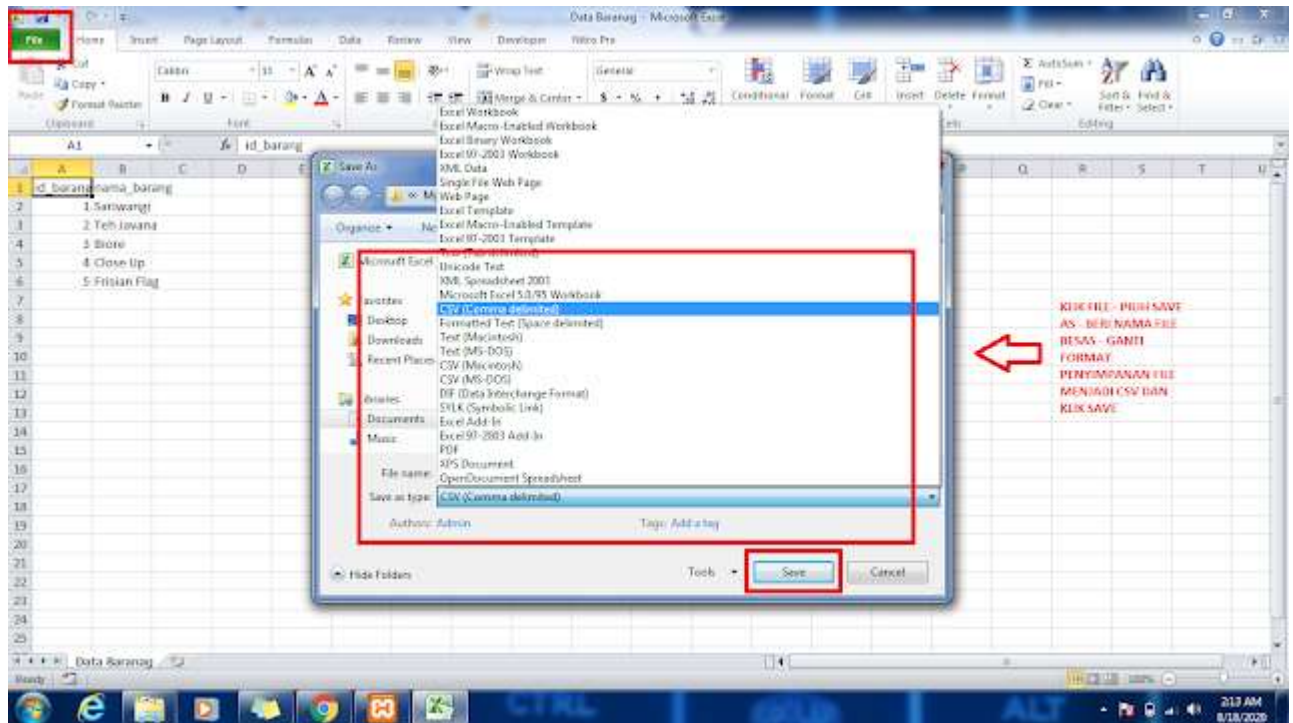
2. Dan tahapan selanjutnya buka phpmyadminnya dengan cara seperti gambar dibawah ini, dimana disana kita bisa membuat database baru dan tabel baru.



3. Tahapan selanjutnya buat satu buah file lembar kerja baru pada microsoft excel, seperti gambar dibawah ini:

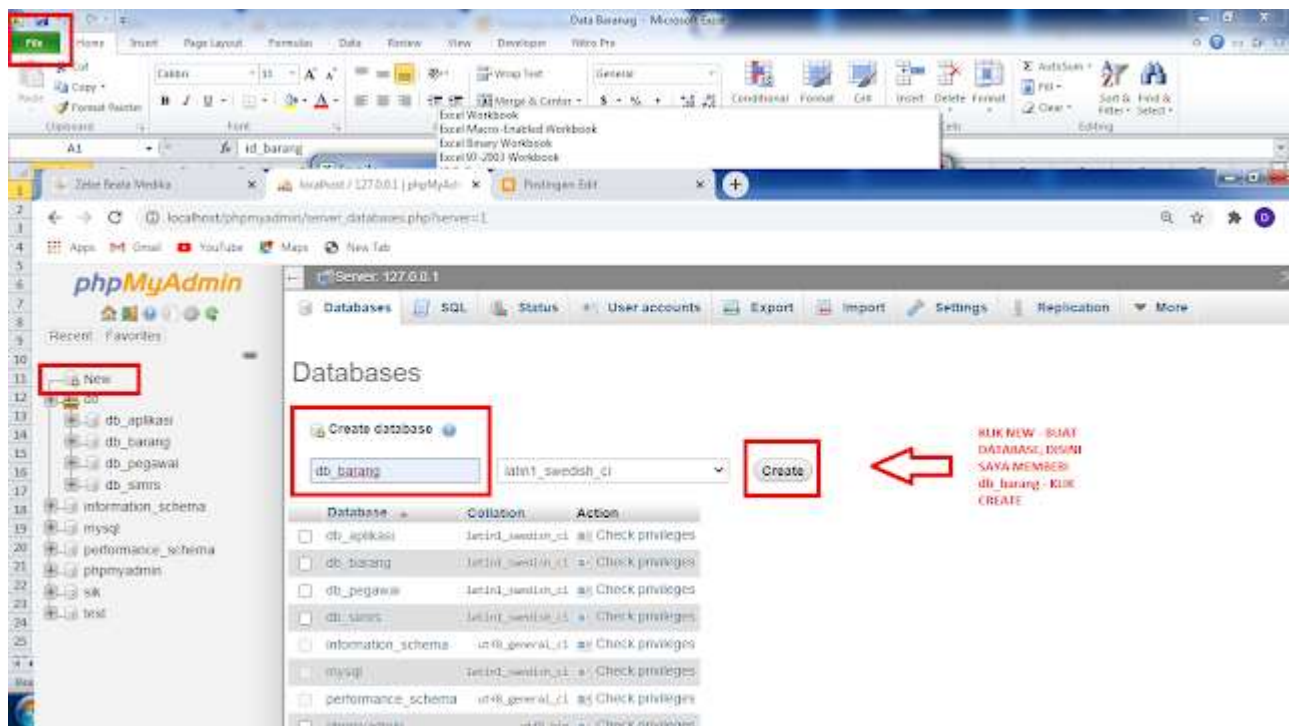


4. Jika tahapan diatas sudah selesai maka selanjutnya langsung saja menyimpannya dalam format csv, caranya seperti gambar dibawah ini:

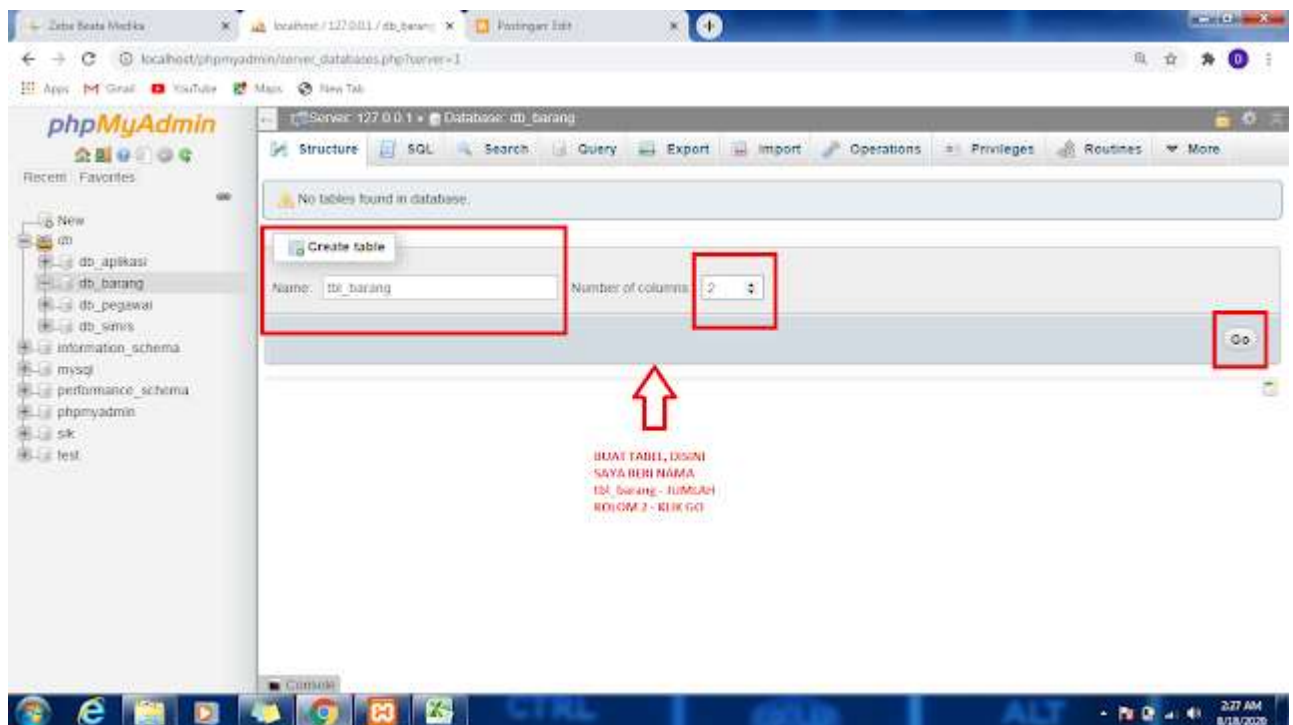


5. Selanjutnya kembali lagi pada phpmyadmin untuk membuat database baru dengan nama db\_barang, seperti gambar dibawah ini ya sobatku.

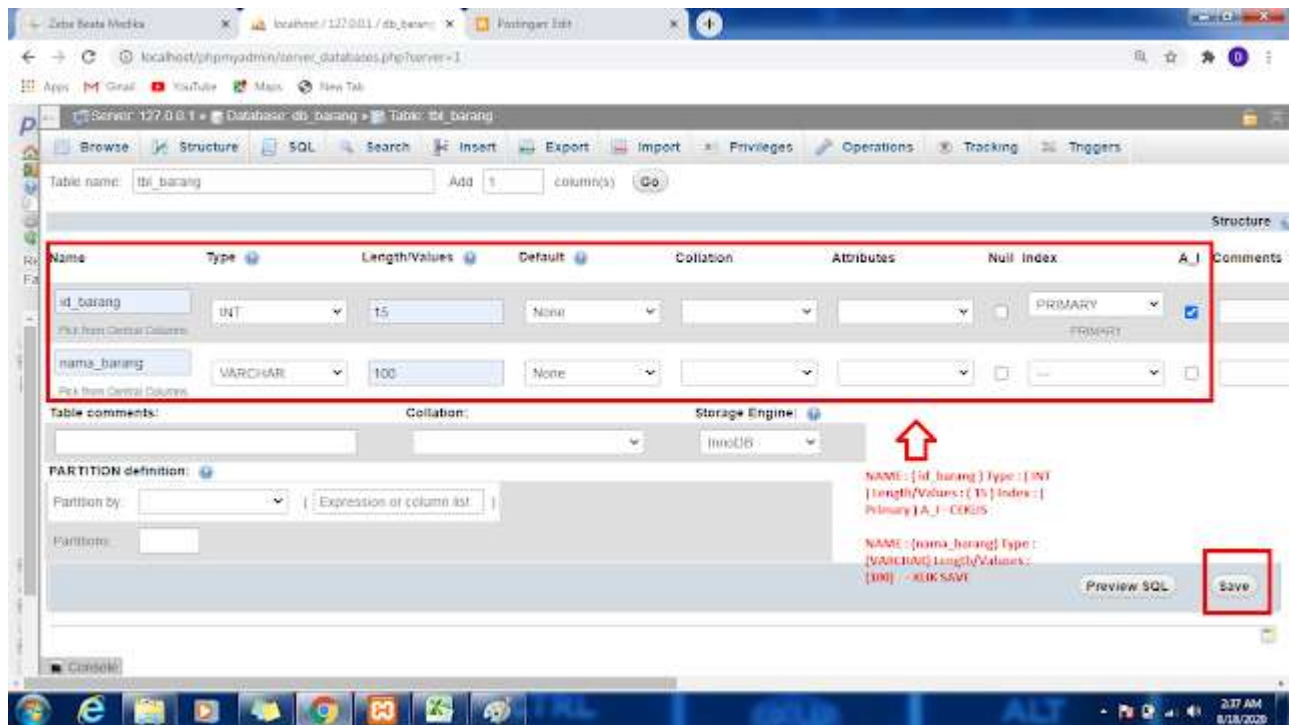




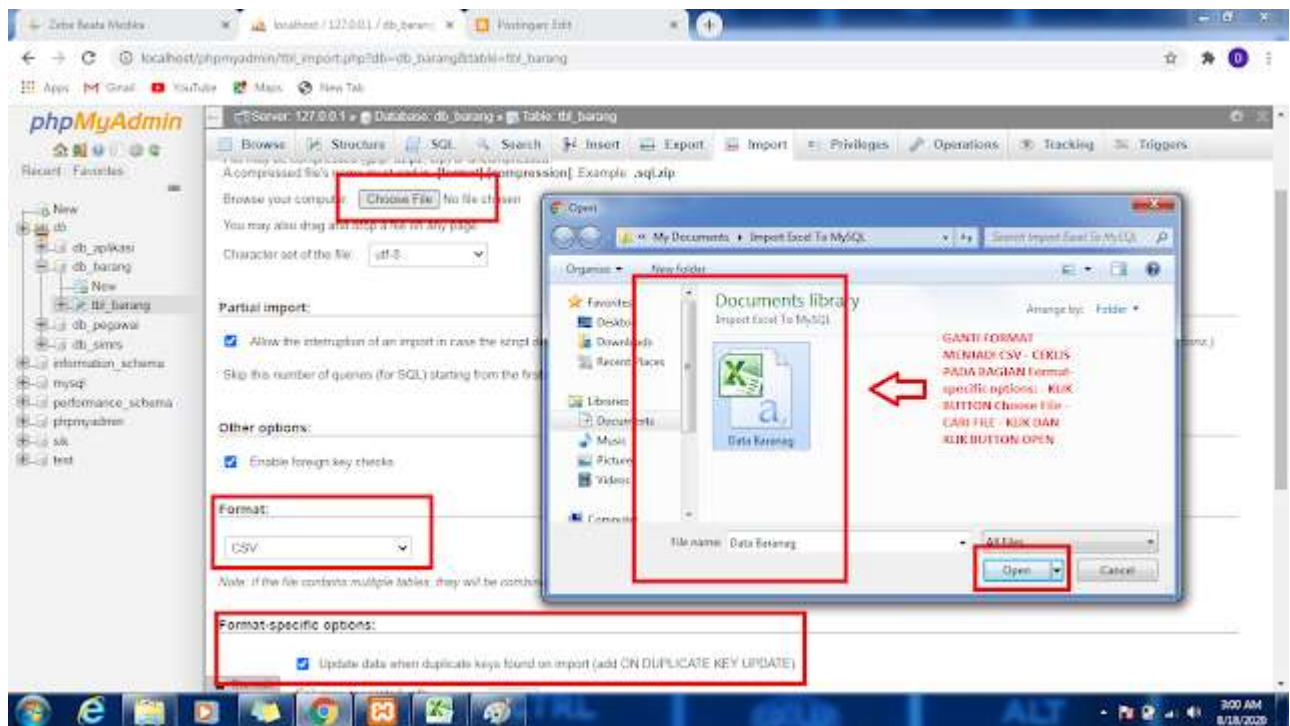
6. Selanjutnya jika tahapan diatas sudah selesai, maka buat tabel baru dengan nama tbl\_barang, caranya seperti gambar dibawah ini:



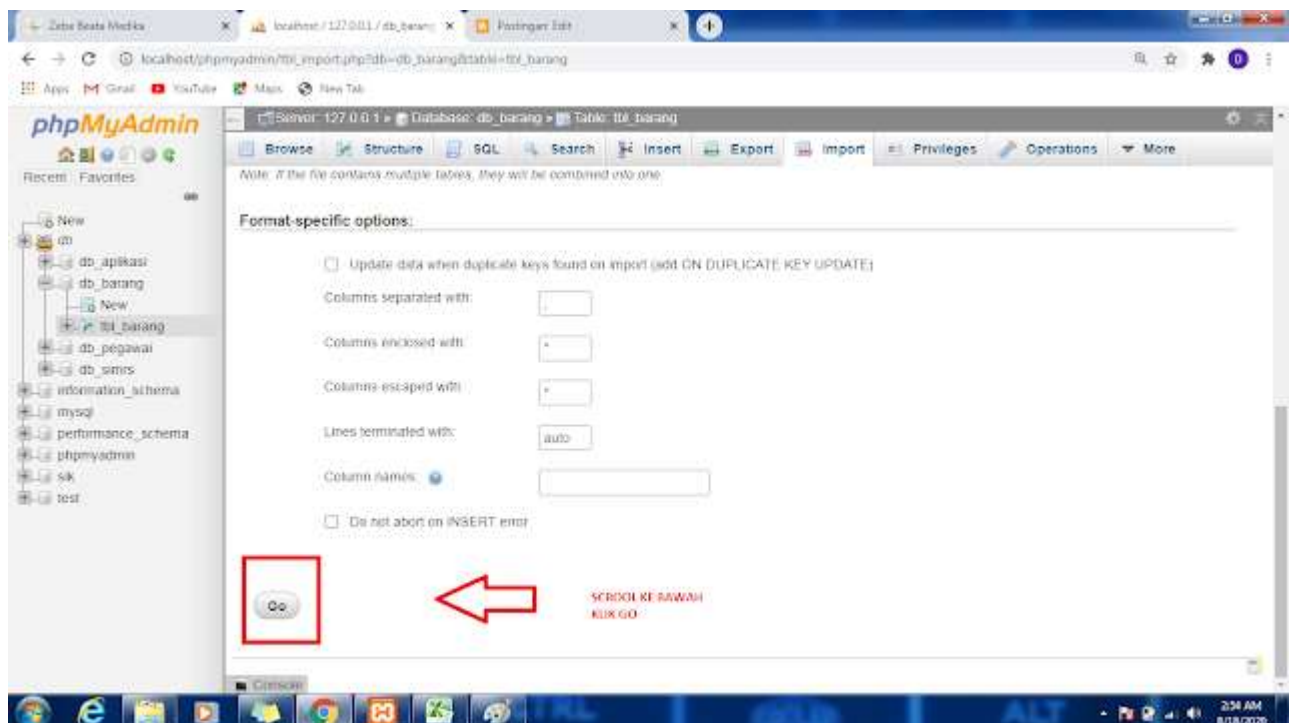
7. Dan selanjutnya kalian buat beberapa komponen yang ada dalam tabel yang buat, seperti gambar dibawah ini.



8. Untuk tahapan selanjutnya import data microsoft excel dengan format csv yang sudah buat, caranya dengan klik pada tbl\_barang lalu pilih import dan ganti formatnya menjadi csv lalu ceklis pada bagian format specific options dan klik tombol choose file lalu cari file excel atau csv yang sudah buat dan menyimpan selanjutnya pilih filenya dan klik open, seperti gambar dibawah ini:

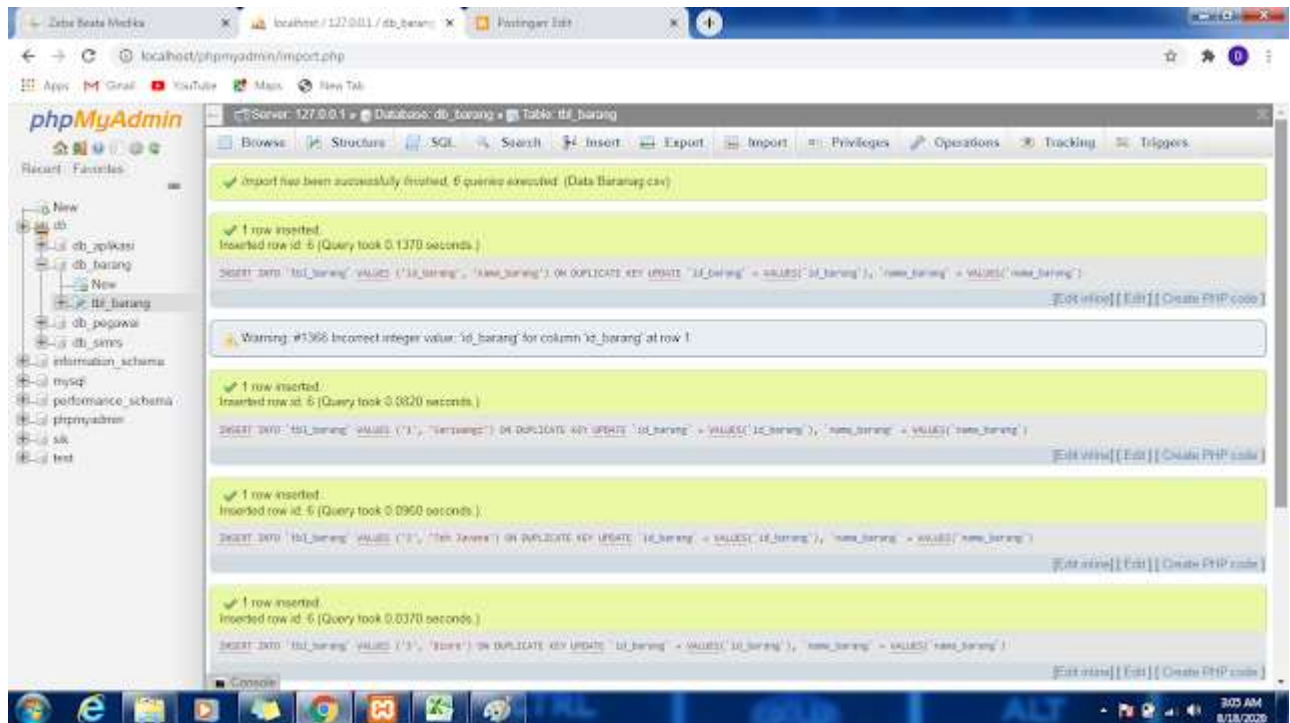


9. Dan tahapan selanjutnya setelah mengganti format file excel mencari format csv dan memasukkannya dengan choose file untuk memilih filenya selanjutnya scroll sampai ke bawah cari button Go dan klik, caranya seperti gambar dibawah ini:





10. Selanjutnya jika proses import data selesai maka akan muncul seperti gambar dibawah ini, bertanda berhasil import datanya.



11. Selanjutnya silahkan mengecek apakah data yang di import dari file excel sudah masuk atau belum, caranya seperti gambar dibawah ini ya sobat ora iso ngoding.



## B. Penggunaan View di MySQL

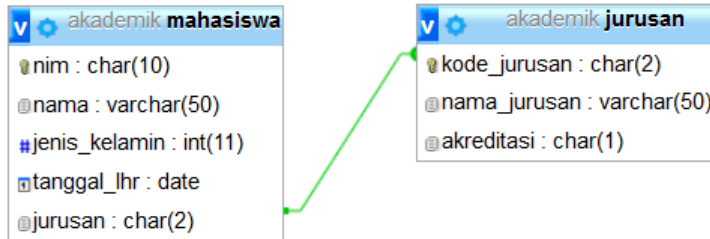
Format dasar membuat view di mysql adalah sebagai berikut:

```
CREATE VIEW nama_view AS Select_statement
```

Jika kita ingin menampilkan view. Perintahnya sama saat kita menampilkan data pada tabel yaitu menggunakan perintah SELECT:

```
SELECT * FROM nama_view
```

Sekarang, mari coba kita langsung masuk ke contoh nya untuk membuat view di mysql , disini saya mempunyai dua buah tabel, yaitu tabel mahasiswa dan jurusan.



Kedua tabel tersebut saling berelasi dengan mengacu pada field kode\_jurusan.

Contoh #1

```
CREATE VIEW mhs AS select nim,nama,jurusan FROM mahasiswa;
```

Perintah diatas saya membuat view dengan nama **mhs** yang mengambil data dari tabel **mahasiswa** hanya untuk kolom nim,nama dan jurusan. Untuk menampilkan data pada view **mhs** kita gunakan perintah select seperti pada tabel lainnya.

```
Command Prompt - mysql -u root
MariaDB [akademik]> select * from mhs;
+----+-----+-----+
| nim      | nama          | jurusan |
+----+-----+-----+
| 135310150 | Rian Hidayat  | TK      |
| 135310156 | Safitri Ayu   | TK      |
| 135410156 | Ahmad Riko    | TI      |
| 135410189 | Wawan Setiawan | TI      |
| 135510190 | Marshel Saraun | MI      |
| 135510191 | Nacha Saraun  | MI      |
| 135510920 | Dani Hermawan | SI      |
| 135518322 | Nita Daniyatun | TI      |
| 135518900 | Mili Wilian   | SI      |
| 135558944 | Bayu Mandalika | SI      |
| 135610157 | Dahlan Iskan  | SI      |
| 136349343 | Kory Ubi      | MI      |
| 147343998 | Chika Lestari | MI      |
| 158549583 | Candra Sidauruk | TK      |
| 158984545 | Juan Burnama  | SI      |
+----+-----+-----+
15 rows in set (0.00 sec)

MariaDB [akademik]>
```

Contoh #2

```
CREATE VIEW mhs2 AS
select nim,nama,nama_jurusan,akreditasi FROM mahasiswa
inner join jurusan on jurusan.kode_jurusan=mahasiswa.jurusan
```

```
WHERE jurusan.kode_jurusan='TK';
```

Pada contoh kedua ini kita membuat view dengan nama mhs2 yang mengambil data dari tabel mahasiswa dan jurusan.

Untuk data yang diambil dari tabel mahasiswa adalah data dari field nim dan nama. Sementara dari tabel jurusan diambil dari field nama\_jurusan dan akreditasi. Pada view disini kita juga menggunakan pengkondisian dengan menggunakan klausa WHERE, dimana data yang diambil hanya mahasiswa yang mempunyai jurusan **TK**.

```
Command Prompt - mysql -u root

MariaDB [akademik]> select * from mhs2;
+-----+-----+-----+-----+
| nim   | nama      | nama_jurusan | akreditasi |
+-----+-----+-----+-----+
| 135310150 | Rian Hidayat | Teknik Komputer | A         |
| 135310156 | Safitri Ayu  | Teknik Komputer | A         |
| 158549583 | Candra Sidauruk | Teknik Komputer | A         |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

MariaDB [akademik]>
```

Bagaimana Jika data pada tabel berubah ?

View di Mysql bersifat dinamis apabila data pada tabel yang menjadi acuan berubah maka otomatis di view juga akan ikut berubah.

### Menghapus View

Untuk menghapus View di mysql. Kita gunakan perintah **DROP**. Contoh disini saya menghapus view **mhs**.

```
DROP VIEW mhs;
```

```
Command Prompt - mysql -u root

MariaDB [akademik]> DROP VIEW mhs;
Query OK, 0 rows affected (0.00 sec)

MariaDB [akademik]>
```

Contoh diatas merupakan contoh view sederhana yang saya buat di MySQL. Jika kalian ingin belajar lagi mengenai cara membuat view di mysql, bisa kalian pelajari di beberapa situs pemrograman lainnya. Intinya ketika mengembangkan sebuah aplikasi yang skalanya sudah besar maka penggunaan view ini akan sangat bermanfaat dan memberikan pengaruh besar untuk performa sistem kita.

## C. Cara Backup Database MySQL Di Windows Dengan Command Prompt (CMD)

Untuk backup Database MySQL di windows bisa dilakukan dengan command prompt

yaitu :

```
cd c:\xampp\mysql\bin
```

Setelah itu ketik perintah backup mysqlnya

```
mysqldump -u root -p nama_database > c:\backup_nama_database.sql
```

jika minta password masukan passwordnya. klik enter. proses backup selesai, file backup ada di **C**

Atau

```
shell>mysqldump -u [user] -p [password] [database] -r "/path/namafile.sql"
```

atau

```
shell>mysqldump -u [user] -p [password] [database] > path/namafile.sql
```

untuk lebih jelasnya , misal kita punya xampp dan kita taruh di D maka:

dengan asumsi : user = root, password kosong, dan nama database = absensi

sebelumnya kita buat dulu satu folder '**backup**' di D:

```
D:\xampp\mysql\bin>mysqldump -u root -p absensi > D:\backup\absensi.sql
```

atau

```
D:\xampp\mysql\bin>mysqldump -u root -p absensi -r "D:\backup\absensi.sql"
```

maka database absensi akan tersimpan di folder D:\backup dengan nama absensi.sql

bila mau membackup hanya struktur database nya saja :

```
D:\xampp\mysql\bin>mysqldump -u root -p --no-data absensi > D:\backup\absensi.sql
```

atau

```
D:\xampp\mysql\bin>mysqldump -u root -p --no-data absensi -r "D:\backup\absensi.sql"
```

Bila mau membackup hanya datanya saja tanpa struktur :

```
D:\xampp\mysql\bin >mysqldump -u root -p --no-create-info absensi > D:\backup\absensi.sql
```

atau

```
D:\xampp\mysql\bin >mysqldump -u root -p --no-create-info absensi -r "D:\backup\absensi.sql"
```

untuk merestore database :

sebelumnya buat dulu database dengan nama misal absensi\_new , kemudian ketikkan di bawah ini :

```
D:\xampp\mysql\bin >mysql -u root -p absensi_new < D:\backup\absensi.sql
```

jangan lupa ya untuk membackup pakai tanda > sedangkan untuk merestore pakai tanda <

#### D. Cara Backup dan Restore MySQL Database Melalui PhpMyAdmin

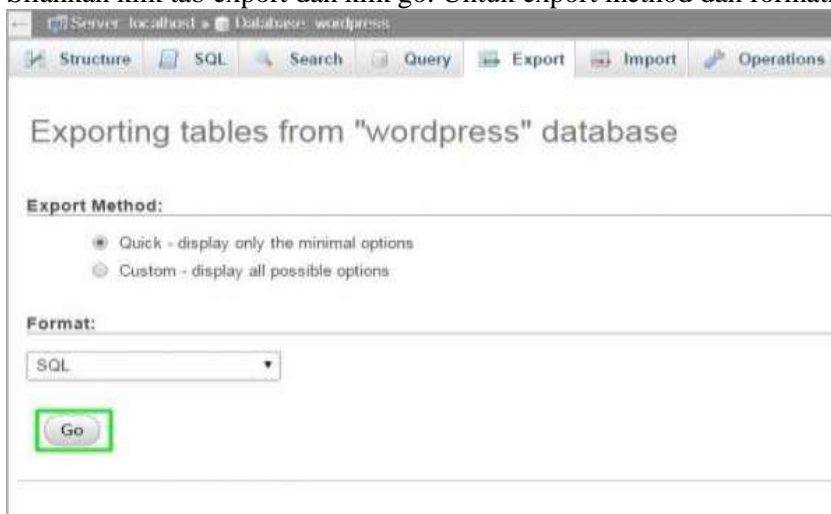
Cara melakukan backup dan restore melalui phpmyadmin terbilang sangat mudah. Anda hanya perlu memastikan bahwa anda sudah install phpmyadmin di server anda terlebih dahulu. Anda bisa membuka halaman phpmyadmin di IP/phpmyadmin atau domain/phpmyadmin, terkecuali anda sudah merubah URL phpmyadmin default ke URL lain.

##### **1. Cara Backup MySQL Database dengan PhpMyAdmin**

Login terlebih dahulu di halaman phpmyadmin anda. Klik database yang ingin anda backup. Karena data wordpress saya tersimpan di database wordpress, maka saya klik database wordpress.



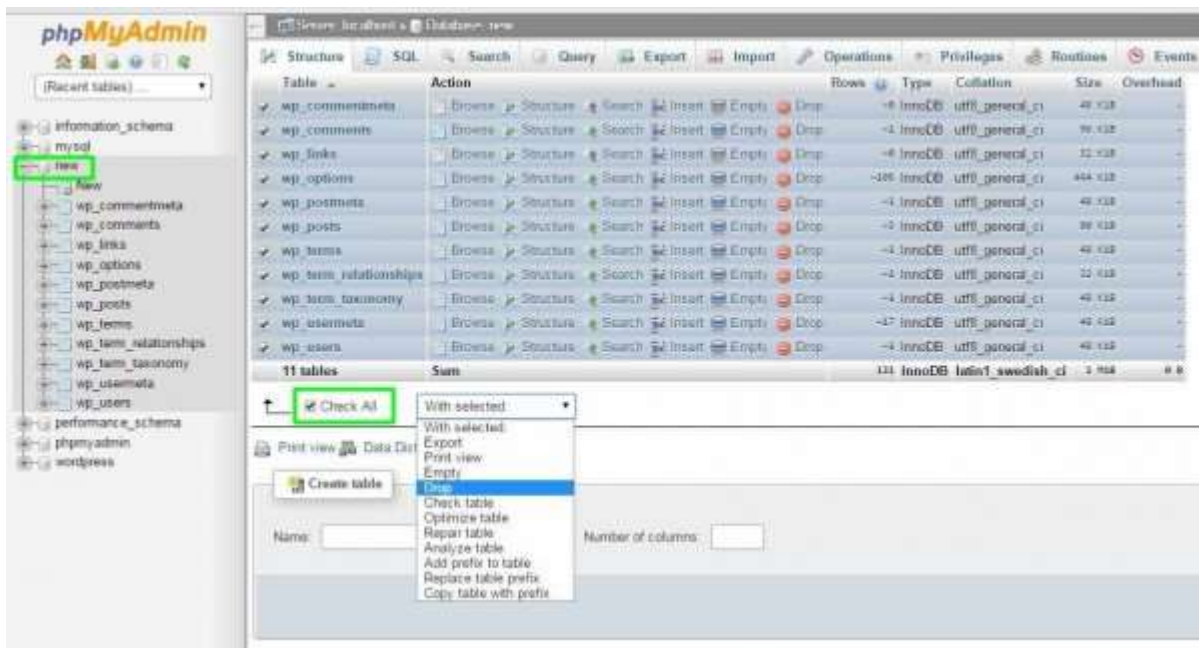
Silahkan klik tab export dan klik go. Untuk export method dan formatnya biarkan saja default.



Selanjutnya file sql database tersebut akan terdownload secara otomatis. Biarkan hingga prose download selesai dan database anda sudah berhasil terbackup di komputer lokal anda.

## 2. Cara Restore MySQL Database dengan PhpMyAdmin

Buat terlebih dahulu database baru. Anda dapat membuatnya dengan klik Server: localhost kemudian pilih Database dan create database. Klik database baru yang akan anda gunakan untuk restore file dari database lama. Jika anda sudah install wordpress di database tersebut biasanya ada tabel bawaan wordpress. Silahkan hapus terlebih dahulu semua tabel tersebut dengan cara Scroll ke bagian bawah dari tabel di database tersebut dan klik check all pada check box yang ada disebelah kiri. Selanjutnya di kotak with selected, silahkan klik drop.



Jika ada pertanyaan Do you really want to execute the following query? Klik saja yes.

Langkah berikutnya klik tab import dan silahkan choose file sql yang tadi anda download ke komputer anda.

Terakhir silahkan klik go untuk memproses upload file sql. Tunggu hingga proses upload selesai dan proses restore selesai dilakukan. Jika proses upload telah selesai, silahkan anda cek hasilnya di tab structure.

### 3. Backup dan Restore MySQL Database Melalui Terminal Console

Anda bisa menggunakan SSH Client seperti putty bagi yang ingin meremote server dari komputer windows atau aplikasi juicessh dan sejenisnya untuk android. Jika anda menggunakan linux, anda hanya tinggal buka terminal console saja. Cara backup dan restore mySQL melalui terminal console ini saya berikan point pentingnya saja, selanjutnya dapat anda kreasikan sesuai dengan kebutuhan dan situasi anda.

#### Cara Backup MySQL Database Melalui Terminal Console

Ada banyak cara yang dapat dilakukan untuk membackup dan restore database mysql melalui terminal console, salah satunya adalah melalui mysqldump. Untuk dapat melakukan backup mysql database dengan mysqldump, silahkan ikuti langkah-langkah berikut ini:

Login ke server anda seperti biasa.

Masukan perintah berikut :

```
mysqldump -u username -p database_yang_akan_dibackup > nama_backup.sql
```

Ganti username dengan user mysql anda, database\_yang\_akan\_dibackup dengan nama database yang akan anda backup dan nama\_backup.sql dengan nama backup yang anda inginkan. Jangan lupa .sql harus disertakan diakhir nama file backup tersebut.

Anda akan diminta untuk memasukkan password user mysql anda, silahkan anda masukkan saja. Tunggu sampai database anda sudah berhasil dibackup. Hasil backup akan tersimpan di direktori user (root).

```
login as: root
root@ [REDACTED]:~# 's password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 2.6.32-042stab092.3 i686)

 * Documentation:  https://help.ubuntu.com/
Last login: Tue Feb 24 02:09:38 2015 from 49.213.[REDACTED]
root@dedeerik:~# mysqldump -u root -p new > backup_new.sql
Enter password:
root@dedeerik:~#
```

### **Cara Restore MySQL Database Melalui Terminal Console**

Sebelum restore, anda harus membuat database baru terlebih dahulu dengan cara :

```
mysql -u username -p
```

```
CREATE DATABASE nama_database;
```

```
exit
```

Sekarang anda dapat melakukan redirect dump file tersebut ke dalam database yang baru saja anda buat melalui command :

```
mysql -u username -p nama_database < nama_backup.sql
```





## MATERI PERTEMUAN 20-21

### Akses Multitabel dengan Joint



#### TAHUKAH KAMU...?

Pada PHP ada 4 jenis perulangan yang bisa kita gunakan:

- Perulangan For
- Perulangan While
- Perulangan Do/While
- Perulangan Foreach

Apa yang akan kamu lakukan bila disuruh membuat daftar judul artikel dengan PHP?  
Apakah akan mencetaknya satu per satu dengan perintah **echo** seperti ini:

### Fungsi Agregasi

Fungsi agregat (aggregate) adalah fungsi yang menerima koleksi nilai dan mengembalikan nilai tunggal sebagai hasilnya. Stadar ISO mendefinisikan lima jenis fungsi agregasi.

a. C O U N T,

Perintah yang digunakan untuk menghitung jumlah baris suatu kolom pada tabel.

Contoh : Perintah untuk menghitung jumlah baris kolom jenis pada tabel jenisfilm:

```
SELECT COUNT(namafield) AS nama_alias FROM nama_tabel;
```

b . SUM

Perintah yang digunakan untuk menghitung jumlah nilai suatu kolom pada tabel.

Contoh : perintah untuk menghitung jumlah nilai kolom harga pada tabel jenisfilm :

```
SELECT SUM(namafield) AS nama_alias FROM nama_tabel;
```

c . AVG

Perintah yang digunakan untuk menghitung rata- rata dari nilai suatu kolom pada tabel.

Contoh : perintah untuk menghitung rata-rata dari kolom harga pada tabel jenisfilm:

```
SELECT AVG(namafield) AS nama_alias FROM nama_tabel;
```

d . MIN

Perintah yang digunakan untuk menampilkan nilai terkecil dari suatu kolom pada tabel.

Contoh : perintah untuk menampilkan nilai terkecil dari kolom harga pada tabel jenisfilm:

```
SELECT MIN(namafield) AS nama_alias FROM nama_tabel;
```

e . MAX

Perintah yang digunakan untuk menampilkan nilai terbesar dari suatu kolom pada table.

Contoh : perintah untuk menampilkan nilai terbesar dari kolom harga pada table jenisfilm :

```
SELECT MAX(namafield) AS nama_alias FROM nama_tabel;
```

### Contoh penerapan

Dalam latihan ini digunakan tabel dengan nama matakuliah, tetapi sebelumnya buatlah database bernama kampus, yang didalamnya ada tabel dengan nama matakuliah. Dengan struktur sebagai berikut:  
Kemudian masukkan data sebagai berikut:

Field	Type	Null	Key	Default	Extra
kode_mk	varchar(10)	NO	PRI	NULL	
nama_mk	varchar(20)	NO		NULL	
sks	int(2)	NO		NULL	
semester	int(2)	NO		NULL	

kode_mk	nama_mk	sks	semester
PTI123	Grafika Multimedia	3	5
PTI333	Basis Data Terdistri	3	5
PTI447	Praktikum Basis Data	1	3
PTI777	Sistem Informasi	2	3
TIK123	Jaringan Komputer	2	5
TIK333	Sistem Operasi	3	5
TIK342	Praktikum Basis Data	1	3

### Mengeliminasi Duplikasi Data

Untuk mengeliminasi data yang sama pada hasil query tambahkan perintah DISTINCT. Contoh:

```
SELECT DISTINCT nama_mk  
FROM matakuliah  
ORDER BY nama_mk
```

Hasilnya jika dieksekusi adalah

nama_mk
Basis Data Terdistri
Grafika Multimedia
Jaringan Komputer
Praktikum Basis Data
Sistem Informasi
Sistem Operasi

### Mendapatkan Banyak Data

Untuk mendapatkan jumlah data penggunaan perintah COUNT Contoh:

```
SELECT COUNT(*) AS jumlah  
FROM matakuliah
```

Hasilnya jika dieksekusi adalah

jumlah
7

### Mendapatkan jumlah Data

Mendapatkan jumlah data digunakan perintah SUM, berbeda dengan perintah COUNT, perintah SUM digunakan untuk menjumlah total data, contoh:

```
SELECT SUM(sks) AS total
FROM matakuliah
```

Hasilnya jika dieksekusi adalah

total
15

### Mendapatkan Nilai Rata-Rata

Untuk mendapatkan nilai rata-rata dari suatu nilai data dipergunakan fungsi agregasi berupa AVG. Contoh:

```
SELECT AVG(sks) AS rata_rata
FROM matakuliah;
```

Hasil yang diperoleh jika dieksekusi adalah sebagai berikut

rata_rata
2.1429

### Mendapatkan Nilai Minimum

Untuk mendapatkan nilai minimum dari beberapa data yang ada dalam tabel dipergunakan perintah MIN. Contoh:

```
SELECT MIN(sks) AS min
FROM matakuliah;
```

Hasil yang diperoleh jika dieksekusi adalah sebagai berikut

min
1

### Mendapatkan Nilai Maximum

Untuk mendapatkan nilai maximum dari beberapa data yang ada dalam tabel dipergunakan perintah MAX. Contoh:

```
MariaDB [kampus]> select max(sks) AS max from matakuliah;
```

max
3

### Pengelompokan Data

Pernyataan SQL untuk mengelompokkan semester berdasarkan jumlah kemunculannya. Contoh:

```
MariaDB [kampus]> SELECT semester, COUNT(semester) AS jumlah
-> FROM matakuliah
-> GROUP BY semester;
```

semester	jumlah
3	3
5	4

### Menyaring Pengelompokan Data

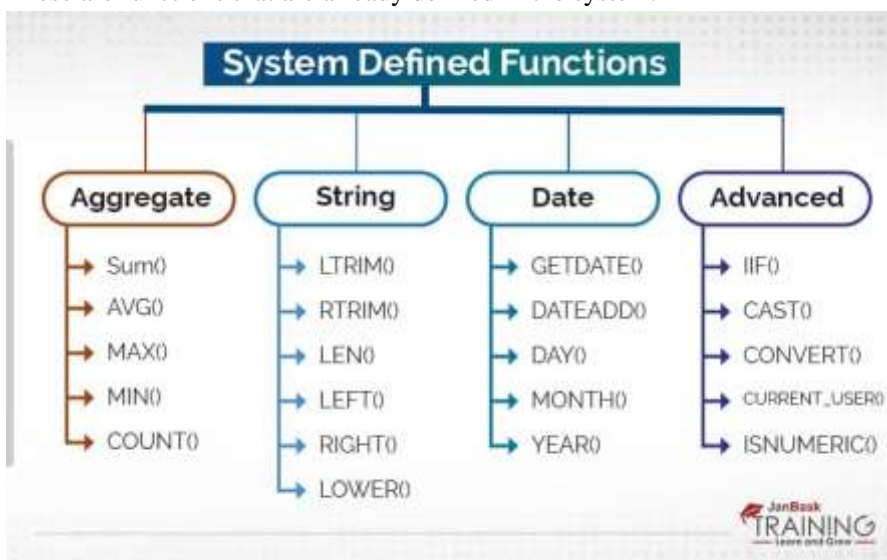
Untuk mengelompokkan data berdasarkan kriteria tertentu dipergunakan juga clausa WHERE Contoh:

```
MariaDB [kampus]> SELECT semester, COUNT(semester) AS jumlah
-> FROM matakuliah
-> WHERE semester > 3
-> GROUP BY semester;
```

semester	jumlah
5	4

## ADVANCE

These are functions that are already defined in the system.



## Different types of System Defined Functions

Aggregate Functions	String Functions	Date Functions	Advanced Functions
These are system defined functions which deal with numbers	These are system defined functions which deals with strings	These are system defined functions which deals with date.	These are system defined functions which performs certain complex task like logical conditions etc
Sum(), AVG(), MAX(), MIN(), COUNT() etc. are some example	LTRIM(), RTRIM(), LEN(), LEFT(), RIGHT(), LOWER() etc. are some of the example	GETDATE(), DATEADD(), DAY(), MONTH(), YEAR() etc. are some of the example	IIF(), CAST(), CONVERT(), CURRENT_USER(), ISNUMERIC() etc. are some examples.

### Aggregate Functions example

We will be using SalesOrderDetail of Adventureworks database for the below examples.

#### Sum()

```
Select sum (OrderQty) [Total Quantity] from [Sales].[SalesOrderDetail]
```

This sums up the OrderQty column value of SalesOrderDetail table.

The output is

#### Avg()

```
select avg(OrderQty) [Total Quantity] from [Sales].[SalesOrderDetail]
```

This gives the average of OrderQty column of SalesOrderDetail table.

#### Max()

```
select max(OrderQty) [Total Quantity] from [Sales].[SalesOrderDetail]
```

This gives out the maximum value of the OrderQty column of [SalesOrderDetail] table.

#### Min()

```
select min(OrderQty) [Total Quantity] from [Sales].[SalesOrderDetail]
```

This gives out the minimum value of the OrderQty column of [SalesOrderDetail] table.

#### Count()

```
Select count (*) from [Person]. [Person]
```

Returns total number of records in the Person table

### String Function Example

#### LTRIM()

Removes space from left side of the string

```
select ltrim(' tes')
```

#### RTRIM()

Removes space from left side of the string

```
select Rtrim('tes ')
```

#### Len()

Gives the length of the string.

```
select Len('Test')
```

#### LEFT()

```
select Left('Sanchayan',3)
```

This SQL Statement extracts three characters from the left side of the string.

#### RIGHT()

```
select Right('Sanchayan',3)
```

The above SQL Statement extracts three characters from the right side of the string

## Examples of Date function

### GETDATE()

Gives out the current date

```
select GETDATE()
```

### DATEADD()

```
SELECT DATEADD (month, 1, '20060830');
```

Add a month with the date value 20060830

### DAY()

```
select day('12/18/2019')
```

The SQL Statement gives the current day value of the date passed as parameter.

### MONTH()

```
select MONTH('12/18/2019')
```

The SQL Statement gives the current month value of the date passed as parameter.

### YEAR()

```
select YEAR('12/18/2019')
```

The SQL Statement gives the current year value of the date passed as parameter.

## Advance Function example

### IIF()

Can be used for if else condition in a single select statement.

The following query gives out MALE if gender is male and FEMALE otherwise.

```
select JobTitle,iif(Gender='M','MALE','FEMALE') [GENDER] from [HumanResources].[Employee]
```

### CAST()

```
SELECT CAST(25.65 AS int)
```

This converts the value 25.65 into integer.

The output is

### CONVERT()

Converts a string into a different data type here integer.

```
SELECT CONVERT(int, 25.65);
```

The output is as below

### CURRENT\_USER

This advance function gives out the current user of the system

```
select CURRENT_USER
```

### ISNUMERIC()

This function checks whether the parameter passed in it is numeric or not.

```
select ISNUMERIC(5)
```

This gives out 1 if true and 0 if false.

## User defined Functions

User-defined functions are functions that are developed by the user.

## Different types of User-Defined Functions

<u>Scalar Function</u>	<u>Table Valued Functions</u>
------------------------	-------------------------------

User-defined function that returns a single value	User-defined functions that returns more than one value
---	---

## Scalar Functions Example

The following code is a simple function which accepts two integer values and returns the sum of the two integers.

```
CREATE FUNCTION Func_Add_
(
    @val1 int,
    @val2 int
)
RETURNS int
BEGIN
    Return @val1 + @val2;
END
```

To execute the function we need to run the following command.

```
select [dbo].[Func_Add_] (2,3)
```

The output is

## Different types of table valued function

Inline table-valued function	Multi statement table-valued function
Returns a table object as output	Returns a table variable as output
Includes only one select statement	Include multiple statement
The processing time is faster	The processing time is slower

## Inline table valued function example

We will be using the SalesOrderHeader table of Adventureworks database. This particular function accepts a date as parameter and gives out the details of all the sales order details on that particular date.

```
CREATE FUNCTION func_inlinetablelevel (
    @order_date date
)
RETURNS TABLE
AS
RETURN
select * from [Sales].[SalesOrderHeader] where
OrderDate = @order_date;
To run the function we need to use the following statement
select * from func_inlinetablelevel('2011-05-31')
```

The output is

## Multi statement table valued function

This particular query accepts a sales order id and returns the total quantity sold against the order.

```
create FUNCTION tablemultivaluedfunctioneg (@Parameters int)
RETURNS @FunctionResultTableVariable TABLE (N int)
AS
BEGIN
    INSERT INTO @FunctionResultTableVariable
    SELECT OrderQty from [Sales].[SalesOrderDetail]
    where SalesOrderID=@Parameters ;
```

```
RETURN;  
END  
GO
```

WE can run the function using the following statement

```
select * from tablemultivaluedfunctioneg(43659)
```

The output is

## Different between Functions and Procedures

Stored Procedure	Functions
Compiled only once and executed again and again	Compiled every time before execution
It is optional to return a value	Function always returns a value
Cannot be called from a function	Can be called from a stored procedure
Cannot call procedure within a procedure	Can call a function from within a function

## Calling a function within a function

A function can be called from within a function. In this paragraph we will learn how to call a function from within a function.

Here in this example we will call the get date function from within the year function.

Let us see how it works.

Here goes the query statement

```
select year(getdate())
```

Here is the output

### Few more complicated function example

The first one is to create a Fibonacci number series using functions. We will pass the number of rows as parameter.

Fibonacci number series as we all know looks like this

```
0  
1  
1  
2  
3  
5  
8  
13
```

Now let us see how the code looks like

```
CREATE FUNCTION fn_Fibonacci(@max int)  
RETURNS @numbers TABLE(number int)  
AS  
BEGIN  
    Declare @n1 int = 0,@n2 int =1,@i int=0,@temp int  
    Insert Into @numbers Values(@n1),(@n2)  
    WHILE (@i<=@max-2)  
    BEGIN  
        Insert Into @numbers Values(@n2+@n1)  
        set @temp = @n2
```



```

Set @n2 = @n2 + @n1
Set @n1 = @temp
Set @i += 1
END
RETURN
END

```

To execute the function we need to write

```
select * from [dbo].[fn_Fibonacci] (15)
```

The output looks like below

## A. Evaluasi

### TES

1. Dapatkan kode\_mk, nama\_mk, semester, urutkan berdasar semester dan kode matakuliah.

Hasil:

kode_mk	nama_mk	sks	js
PTI447	Praktikum Basis Data	1	3
PTI777	Sistem Informasi	2	3
TIK342	Praktikum Basis Data	1	3
PTI123	Grafika Multimedia	3	5
PTI333	Basis Data Terdistribusi	3	5
TIK123	Jaringan Komputer	2	5
TIK333	Sistem Operasi	3	5

2. Dapatkan kode\_mk, nama\_mk, sks, dan js dari matakuliah. Urutkan berdasar nama matakuliah.

Petunjuk:

Misalkan js sama dengan 2 kali sks

Hasil:

kode_mk	nama_mk	sks	js
PTI333	Basis Data Terdistribusi	3	6
PTI123	Grafika Multimedia	3	6
TIK123	Jaringan Komputer	2	4
PTI447	Praktikum Basis Data	1	2
TIK342	Praktikum Basis Data	1	2
PTI777	Sistem Informasi	2	4
TIK333	Sistem Operasi	3	6

3. Dapatkan jumlah total sks dari tiap-tiap semester.

Hasil:

semester	total_sks
3	4
5	11

4. Tuliskan pernyataan SQL untuk mengelompokkan sks berdasarkan jumlah kemunculannya, di mana jumlah kemunculan sksnya lebih dari 2.

Hasil:

sks	jumlah
3	3

## B. Kunci Jawaban

### Rubrik Penilaian Tugas

No So al	Jawab an	Skor
1	<pre>MariaDB [kampus]&gt; SELECT kode_mk, nama_mk, semester -&gt; FROM matakuliah -&gt; ORDER BY semester AND kode_mk;</pre>	25
2	<pre>MariaDB [kampus]&gt; SELECT kode_mk, nama_mk, sks, 2*sks AS js -&gt; FROM matakuliah -&gt; ORDER BY nama_mk;</pre>	25
3	<pre>MariaDB [kampus]&gt; SELECT semester, SUM(sks) as total_sks -&gt; FROM matakuliah -&gt; GROUP BY semester;</pre>	25
4	<pre>MariaDB [kampus]&gt; SELECT sks, COUNT(sks) as jumlah -&gt; FROM matakuliah -&gt; WHERE sks&gt;2;</pre>	25
Jumlah Skor		100



## MATERI PERTEMUAN 20-21

### Akses Multitabel dengan Joint



#### TAHUKAH KAMU...?

Pada PHP ada 4 jenis perulangan yang bisa kita gunakan:

- Perulangan For
- Perulangan While
- Perulangan Do/While
- Perulangan Foreach

Apa yang akan kamu lakukan bila disuruh membuat daftar judul artikel dengan PHP?  
Apakah akan mencetaknya satu per satu dengan perintah `echo` seperti ini:

Join adalah cara untuk menghubungkan data yang diambil dari tabel-tabel melalui sebuah kolom yang menghubungkan mereka. Misal, pembaca mungkin ingin menghubungkan tabel alamat dengan tabel nomor telepon berdasarkan nama seseorang (contoh: "Berikan saya alamat dan nomor telepon seseorang yang bernama John Smith.")

Join merupakan sebuah konsep di dalam pengolahan data pada database. Konsep ini menggabungkan dua buah tabel atau lebih sehingga menghasilkan sebuah tabel baru yang bersifat temporary atau sementara. Melalui tabel baru ini akan dapat diperoleh hubungan tiap data pada tabel-tabel yang digabungkan. Tabel baru ini disebut sebagai *joined table*.

Pada MySQL dikenal beberapa macam join yang dapat dikelompokkan menjadi dua, yaitu inner join dan outer join. Inner join dapat dibagi kembali menjadi natural join dan cross join. Sedangkan outer join dapat dibedakan menjadi left outer join, right outer join, dan full outer join.

#### 1. Inner Join dan Natural Join

Inner join dan natural join merupakan join yang digunakan untuk menemukan persimpangan atau perpotongan antara dua buah tabel yang di-join-kan. Join ini akan mengembalikan atau menampilkan data-data yang saling berpasangan di antara kedua buah tabel. *Syntax* untuk *inner join* dan *natural join* adalah sebagai berikut:

##### **Cara #1. Inner Join dengan WHERE.**

Penggabungan dengan klausa WHERE memiliki bentuk umum sebagai berikut:

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

##### **Cara #2. Inner Join dengan klausa INNER JOIN.**

Berikut ini bentuk umumnya:

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

#### 2. Cross Join

*Cross join* merupakan *inner join* dengan seluruh kondisi *join* (tidak hanya data yang berpasangan) dianggap bernilai *true*. Sesuai dengan namanya, *join* ini akan mengembalikan semua

kemungkinanpasangan atau persilangan data pada tabel yang satu dengan data pada tabel yang lainnya. *Syntax* untuk *cross join* adalah sebagai berikut:

```
SELECT nama_kolom  
FROM tabel_1 CROSS JOIN tabel_2;
```

### 3. Left Outer Join dan Right Outer Join

*Left outer join* merupakan *join* yang akan mengembalikan seluruh data pada tabel sebelah kiri (*left table*) yang memiliki pasangan pada tabel sebelah kanan (*right table*) ditambah data-data pada *left table* yang tidak memiliki pasangan pada tabel sebelah kanan. Sedangkan *right outer*

*join* sebaliknya. Untuk data-data yang tidak memiliki pasangan, pada *joined table* yang dihasilkan data-data tersebut akan dipasangkan dengan data *null*. *Syntax* untuk *left outer join* dan *right outer join* adalah sebagai berikut:

```
SELECT nama_kolom  
FROM tabel_kiri LEFT OUTER JOIN tabel_kanan  
ON tabel_kiri.nama_kolom = tabel_kanan.nama_kolom;
```

```
SELECT nama_kolom  
FROM tabel_kiri RIGHT OUTER JOIN tabel_kanan  
ON tabel_kiri.nama_kolom = tabel_kanan.nama_kolom;
```

### 4. Full Outer Join

*Full outer join* merupakan kombinasi dari hasil *left outer join* dengan *right outer join*. MySQL tidak mengenal klausa *full outer join*. Oleh karena itu, untuk mendapatkan *joined table* hasil dari *full outer join* digunakan klausa atau operator UNION untuk menggabungkan hasil *query* yang menggunakan *left outer join* dengan hasil *query* yang menggunakan *right outer join*. *Syntax* untuk *full outer join* adalah sebagai berikut:

```
SELECT nama_kolom  
FROM tabel_kiri LEFT OUTER JOIN tabel_kanan  
ON tabel_kiri.nama_kolom = tabel_kanan.nama_kolom  
UNION  
SELECT nama_kolom
```

**FROM** *tabel\_kiri* **RIGHT OUTER JOIN** *tabel\_kanan*

**ON** *tabel\_kiri.nama\_kolom* = *tabel\_kanan.nama\_kolom*;

### c. Penggunaan Operator IS NULL

Seperti yang telah dijelaskan pada bagian *outer join*, data yang diperoleh dari suatu *query* dapat mengandung nilai *null*. Untuk mem-*filter query* agar hanya menampilkan data yang mengandung nilai *null* saja dapat menggunakan operator **IS NULL** pada klausa **WHERE**.

### d. Penggunaan Operator DISTINCT

Pada saat melakukan pengambilan data di dalam *database* terkadang dijumpai suatu kondisi dimana data yang sama diperoleh atau muncul lebih dari satu baris. Untuk menghilangkan kemunculan data yang berulang-ulang ini dapat menggunakan operator **DISTINCT** pada klausa **SELECT**. Dengan menggunakan operator ini, data yang sama hanya akan muncul satu kali atau hanya pada satu baris.

### e. Penggunaan Operator AS

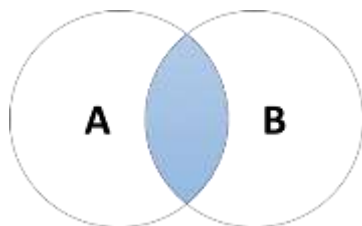
Operator **AS** berfungsi untuk mengubah nama kolom atau *field* pada suatu hasil *query*. Perubahan nama kolom ini hanya berlaku pada hasil *query* tersebut, tidak mengubah nama kolom asalnya.

Ada berapa jenis join?

Berikut ini adalah empat tipe join di SQL (bersama tiga variannya). Sebagai pelengkap dari penjelasan yang ada, kami telah menyediakan contoh kode SQL.

Inner Join

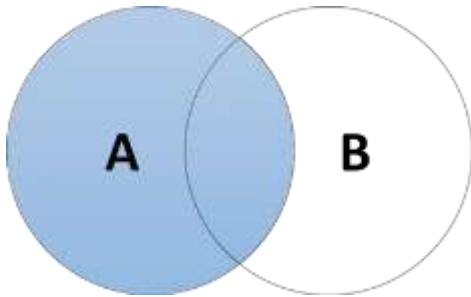
Inner join mungkin tipe join yang paling banyak dipakai. Inner join mengembalikan baris-baris dari dua tabel atau lebih yang memenuhi syarat.



```
SELECT columns
FROM TableA
INNER JOIN TableB
ON A.columnName = B.columnName;
```

Left [Outer] Join

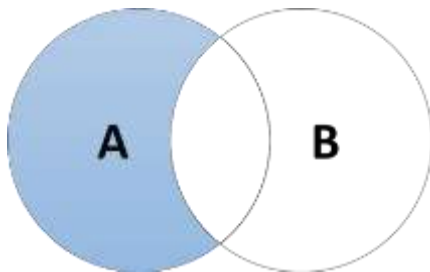
Left outer join (sering disingkat left join) akan mengembalikan seluruh baris dari tabel disebelah kiri yang dikenai kondisi **ON** dan hanya baris dari tabel disebelah kanan yang memenuhi kondisi join.



```
SELECT columns
FROM TableA
LEFT OUTER JOIN TableB
ON A.columnName = B.columnName
```

Left [Outer] Join without Intersection

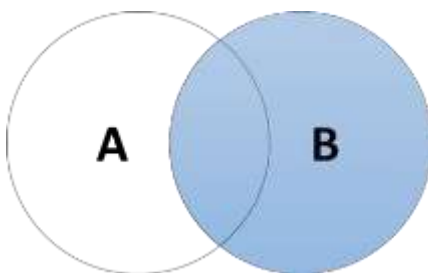
Join ini merupakan variasi dari left outer join. Pada join ini kita hanya akan mengambil data dari tabel sebelah kiri yang dikenai kondisi ON yang juga memenuhi kondisi join tanpa data dari tabel sebelah kanan yang memenuhi kondisi join.



```
SELECT columns
FROM TableA
LEFT OUTER JOIN TableB
ON A.columnName = B.columnName
WHERE B.columnName IS NULL
```

Right [Outer] Join

Right outer join (sering disingkat right join) akan mengembalikan semua baris dari tabel sebelah kanan yang dikenai kondisi ON dengan data dari tabel sebelah kiri yang memenuhi kondisi join. Teknik ini merupakan kebalikan dari left outer join.

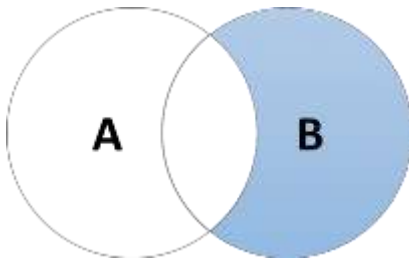


```
SELECT columns
FROM TableA
RIGHT OUTER JOIN TableB
ON A.columnName = B.columnName
```

Right [Outer] Join without Intersection

Teknik ini merupakan variasi dari right outer join. Pada join ini kita hanya akan

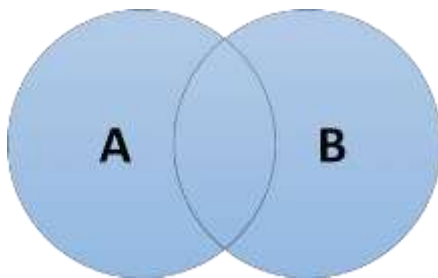
mengambil data dari tabel sebelah kanan yang dikenai kondisi ON yang juga memenuhi kondisi join tanpa data dari tabel sebelah kanan yang memenuhi kondisi join.



```
SELECT columns
FROM TableA
RIGHT OUTER JOIN TableB
ON A.columnName = B.columnName
WHERE A.columnName IS NULL
```

Full [Outer] Join

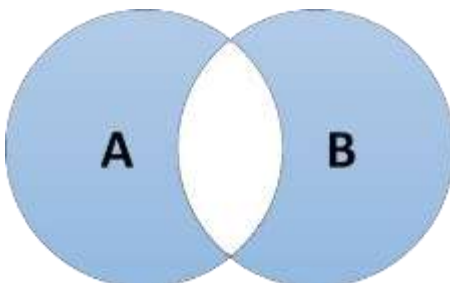
Full outer join (sering disingkat full join) akan mengembalikan seluruh baris dari kedua tabel yang dikenai ON termasuk data-data yang bernilai NULL.



```
SELECT columns
FROM TableA
FULL JOIN TableB
ON A.columnName = B.columnName
```

Full [Outer] Join without Intersection

Variasi lain dari full outer join yang akan mengembalikan seluruh data dari kedua tabel yang dikenai ON tanpa data yang memiliki nilai NULL.



```
SELECT columns
FROM TableA
FULL JOIN TableB
ON A.columnName = B.columnName
WHERE A.columnName IS NULL
OR B.columnName IS NULL
```



## STUDI KASUS I

JOIN merupakan perintah di MySQL untuk menggabungkan 2 table atau lebih berdasarkan kolom yang sama

JOIN di MySQL dibagi menjadi 3 cara

1. INNER JOIN
2. LEFT JOIN
3. RIGHT JOIN

Sebelum kita melanjutkan pembahasan tentang join kita akan membuat 2 table yaitu table **mahasiswa** dan **transaksi**

Table mahasiswa -> Data mahasiswa

Table transaksi -> Data transaksi peminjaman buku perpustakaan

```
CREATE TABLE mahasiswa
```

```
(  
  nim INT(10),  
  nama VARCHAR(100),  
  alamat VARCHAR(100),  
  PRIMARY KEY(nim)  
);
```

```
CREATE TABLE transaksi
```

```
(  
  id_transaksi INT(10),  
  nim INT(10),  
  buku VARCHAR(100),  
  PRIMARY KEY(id_transaksi)  
);
```

Masukkan data untuk table **mahasiswa** dan **transaksi**

```
INSERT INTO mahasiswa
```

```
VALUES  
(21400200,"faqih","bandung"),  
(21400201,"ina","jakarta"),  
(21400202,"anto","semarang"),  
(21400203,"dani","padang"),  
(21400204,"jaka","bandung"),  
(21400205,"nara","bandung"),  
(21400206,"senta","semarang");
```

```
INSERT INTO transaksi
```

```
VALUES  
(1,21400200,"Buku Informatika"),  
(2,21400202,"Buku Teknik Elektro"),  
(3,21400203,"Buku Matematika"),  
(4,21400206,"Buku Fisika"),  
(5,21400207,"Buku Bahasa Indonesia"),  
(6,21400210,"Buku Bahasa Daerah"),  
(7,21400211,"Buku Kimia");
```

```

1 > SELECT * FROM mahasiswa;
2 +-----+-----+-----+
3 | nim      | nama  | alamat |
4 +-----+-----+-----+
5 | 21400200 | faqih | bandung |
6 | 21400201 | ina   | jakarta |
7 | 21400202 | anto  | semarang |
8 | 21400203 | dani  | padang  |
9 | 21400204 | jaka  | bandung |
10 | 21400205 | nara  | bandung |
11 | 21400206 | senta | semarang |
12 +-----+-----+-----+
13 7 rows in set (0.00 sec)

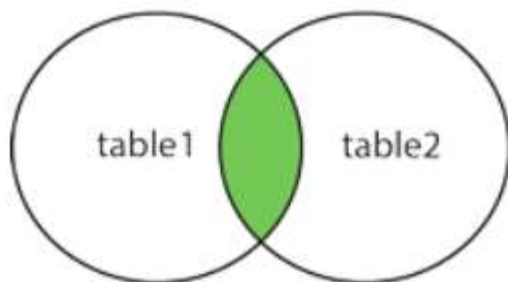
14
15 > SELECT * FROM transaksi;
16 +-----+-----+-----+
17 | id_transaksi | nim      | buku |
18 +-----+-----+-----+
19 | 1            | 21400200 | Buku Informatika |
20 | 2            | 21400202 | Buku Teknik Elektro |
21 | 3            | 21400203 | Buku Matematika |
22 | 4            | 21400206 | Buku Fisika |
23 | 5            | 21400207 | Buku Bahasa Indonesia |
24 | 6            | 21400210 | Buku Bahasa Daerah |
25 | 7            | 21400211 | Buku Kimia |
26 +-----+-----+-----+
27 7 rows in set (0.00 sec)

```

**Note :** Pelajari materi DDL dan DML untuk membuat struktur table dan nilainya

### INNER JOIN

INNER JOIN membandingkan record di setiap table untuk dicek apakah nilai sama atau tidak.



Jika nilai kedua table sama maka akan terbentuk table baru yang hanya menampilkan record yang sama dari kedua table

Cara penulisannya

SELECT \*

FROM table1

INNER JOIN table2

ON table1.field = table2.field;

Contoh, Mencari data dari table mahasiswa dan transaksi berdasarkan kolom NIM

SELECT \*

FROM mahasiswa

INNER JOIN transaksi

ON mahasiswa.nim = transaksi.nim;

```

1 SELECT *
2 FROM mahasiswa
3 INNER JOIN transaksi
4 ON mahasiswa.nim = transaksi.nim;

```

	nim	nama	alamat	id_transaksi	nim	buku
8	21400200	faqih	bandung	1	21400200	Buku Informatika
9	21400202	anto	semarang	2	21400202	Buku Teknik Elektro
10	21400203	dani	padang	3	21400203	Buku Matematika
11	21400206	senta	semarang	4	21400206	Buku Fisika

```

12
13 4 rows in set (0.00 sec)

```

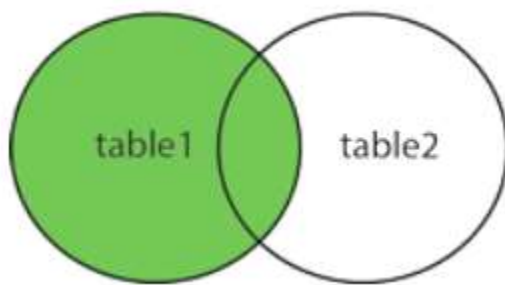
4 rows in set (0.00 sec)

Table mahasiswa mempunyai 7 record dan table transaksi mempunyai 7 record

Jika menggabungkan kedua data menggunakan INNER JOIN berdasarkan kolom NIM maka hanya tampil 4 data mahasiswa yang meminjam buku di perpustakaan

### LEFT JOIN

LEFT JOIN menghasilkan nilai berdasarkan table kiri (table1) dan nilai yang sama di table kanan (table2).



Jika table kanan tidak nilainya ada maka akan diisi nilai NULL

```

SELECT *
FROM table1
LEFT JOIN table2
ON table1.field = table2.field;

```

Contoh, Mencari data dari table mahasiswa dan transaksi berdasarkan kolom NIM

```

SELECT *
FROM mahasiswa
LEFT JOIN transaksi
ON mahasiswa.nim = transaksi.nim;

```

```
SELECT *
FROM mahasiswa
LEFT JOIN transaksi
ON mahasiswa.nim = transaksi.nim;
```

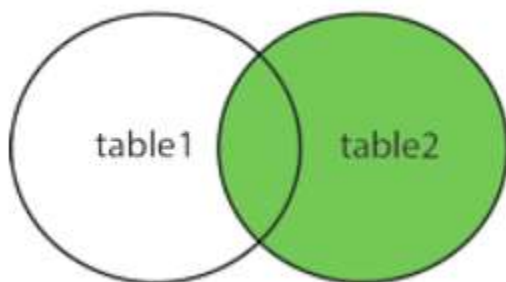
nim	nama	alamat	id_transaksi	nim	buku
21400200	faqih	bandung	1	21400200	Buku Informatika
21400202	anto	semarang	2	21400202	Buku Teknik Elektro
21400203	dani	padang	3	21400203	Buku Matematika
21400206	senta	semarang	4	21400206	Buku Fisika
21400201	ina	jakarta	NULL	NULL	NULL
21400204	jaka	bandung	NULL	NULL	NULL
21400205	nara	bandung	NULL	NULL	NULL

7 rows in set (0.00 sec)

Table kiri (mahasiswa) akan menjadi table master dan mencari nilai yang sama di table transaksi. Apabila ada mahasiswa yang tidak meminjam buku maka diberi nilai NULL

### RIGHT JOIN

Konsep RIGHT JOIN hampir sama seperti LEFT JOIN hanya yang menjadi master adalah table kanan (table 2)



Jika table kiri tidak nilainya ada maka akan diisi nilai NULL

```
SELECT *
FROM table1
RIGHT JOIN table2
ON table1.field = table2.field;
```

Contoh, Mencari data dari table mahasiswa dan transaksi berdasarkan kolom NIM

```
SELECT *
FROM mahasiswa
RIGHT JOIN transaksi
ON mahasiswa.nim = transaksi.nim;
```

```

SELECT *
FROM mahasiswa
RIGHT JOIN transaksi
ON mahasiswa.nim = transaksi.nim;

```

nim	nama	alamat	id_transaksi	nim	buku
21400200	faqih	bandung	1	21400200	Buku Informatika
21400202	anto	semarang	2	21400202	Buku Teknik Elektro
21400203	dani	padang	3	21400203	Buku Matematika
21400206	senta	semarang	4	21400206	Buku Fisika
NULL	NULL	NULL	5	21400207	Buku Bahasa Indonesia
NULL	NULL	NULL	6	21400210	Buku Bahasa Daerah
NULL	NULL	NULL	7	21400211	Buku Kimia

7 rows in set (0.00 sec)

Terdapat 7 transaksi peminjaman buku di perpustakaan. Bagi transaksi yang NIM mahasiswa tidak ada di table mahasiswa akan diberi nilai NULL

## LATIHAN

Dalam latihan ini digunakan dua buah tabel bernama Karyawan dan Departemen dengan relationship *bekerja pada*. Struktur tabelnya diperlihatkan sebagai berikut:

```

CREATE TABLE karyawan (
  nama varchar(30) NOT NULL,
  id_dep int(5) NOT NULL
) ENGINE=MyISAM;

CREATE TABLE departemen (
  id_dep int(5) NOT NULL,
  nama_dep varchar(30) NOT NULL,
  PRIMARY KEY (id_dep)
) ENGINE=MyISAM;

```

Data yang digunakan adalah sebagai berikut:

Tabel Karyawan

nama	id_dep
Agus	10
Budi	16
Citra	12
Dani	17

Tabel Departemen

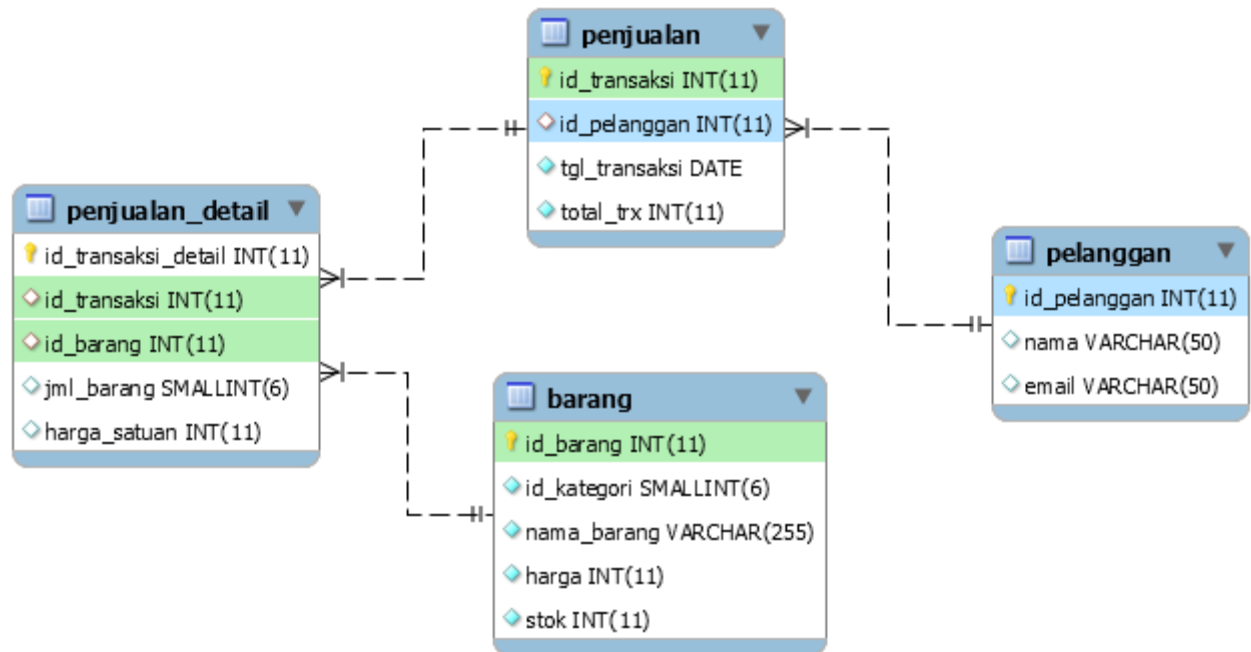
id_dep	nama_dep
10	Penelitian
11	Pemasaran
12	SDM
13	Keuangan

### Inner Join

Sebagaimana dijelaskan, *inner join* akan mengembalikan data di tabel A dan B yang sesuai. Sebagai contoh, kita bisa mendapatkan data karyawan yang memiliki departemen.

## SUDI KASUS II

Sebelum melangkah lebih jauh, kita perlu menyiapkan tabel yang akan kita gunakan dalam tutorial ini. Pada contoh kali ini, saya menggunakan empat tabel, yaitu: tabel **pelanggan**, **produk**, **transaksi**, dan **transaksi\_detail**. struktur dan hubungan keempat tabel tersebut tampak seperti pada gambar berikut:



Dari keempat tabel tersebut, kita hanya menggunakan beberapa diantaranya. Adapun contoh datanya adalah sebagai berikut:

### Tabel **barang**:

id_barang	id_kategori	nama_barang	harga	stok
1	1	RAM	230000	4
2	1	Mainboard	1250000	7
3	1	Mouse	80000	6
4	3	Mousepad	35000	3
5	3	Keyboard	80000	5

### Tabel **pelanggan**:

id_pelanggan	nama	email
1	Alfa	alfa@yahoo.com
2	Beta	beta@yahoo.com
3	Charlie	charlie@gmail.com
4	Delta	delta@gmail.com

### Tabel **penjualan**:

id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	1	2017-02-22	230000
2	3	2017-02-22	195000

3	2	2017-01-01	1710000
4	1	2017-02-04	310000
5	NULL	2017-02-10	80000

## I. JOIN Pada MySQL

Untuk menggabungkan tabel pada MySQL, kita gunakan klausa JOIN. Pada MySQL terdapat dua macam bentuk join, yaitu INNER JOIN, LEFT OUTER JOIN, dan RIGHT OUTER JOIN. Format penulisannya adalah sebagai berikut:

```
SELECT nama_kolom
FROM tabel
INNER JOIN | LEFT OUTER JOIN | RIGHT OUTER JOIN tabel ON kondisi
```

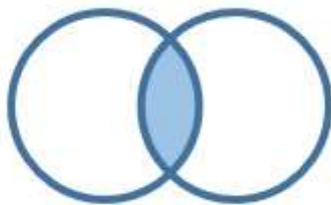
Selain menggunakan klausa ON untuk mendefinisikan kondisi, kita dapat menggunakan klausa USING, format penulisannya adalah:

```
SELECT nama_kolom
FROM tabel
INNER JOIN | LEFT OUTER JOIN | RIGHT OUTER JOIN tabel USING(nama_kolom)
```

Klausa USING ini akan menggunakan nama kolom yang ada di dalam tanda kurung untuk menghubungkan kedua tabel, kolom ini harus ada pada tabel yang ingin dihubungkan dan harus memiliki nama yang sama. Biasanya kolom yang berhubungan didefinisikan sebagai Primary Key dan Foreign Key, namun tidak masalah jika keduanya bukan primary key maupun foreign key.

## II. INNER atau CROSS JOIN

Cara pertama untuk menggabungkan tabel adalah menggunakan inner join. Dengan inner join, tabel akan digabungkan berdasarkan data yang sama, yang ada pada kedua tabel, jika di gambarkan dalam bentuk diagram venn, bentuk inner join seperti tampak pada gambar berikut:



Pada MySQL, penulisan INNER JOIN dapat dilakukan dengan dua cara yaitu (1) menggunakan klausa INNER JOIN (2) menggunakan klausa CROSS JOIN (3) cukup menggunakan klausa JOIN saja. Sobat bebas menggunakan keduanya asal konsisten, saya pribadi lebih prefer menggunakan JOIN saja, karena lebih simpel. Sebagai contoh kita akan menampilkan data pelanggan yang melakukan pesanan, query yang kita jalankan:

```
SELECT pl.id_pelanggan, nama, tgl_transaksi, total_transaksi
FROM pelanggan pl
JOIN penjualan pn ON pl.id_pelanggan = pn.id_pelanggan
```

Jika menggunakan klausa USING, maka query akan berbentuk seperti berikut:

```
SELECT pl.id_pelanggan, nama, tgl_transaksi, total_transaksi
FROM pelanggan pl
JOIN penjualan pn USING(id_pelanggan)
```

Hasil:

id_pelanggan	nama	tgl_transaksi	total_transaksi
1	Alfa	2017-02-22	230000
3	Charlie	2017-02-22	195000
2	Beta	2017-01-01	1710000
1	Alfa	2017-02-04	310000

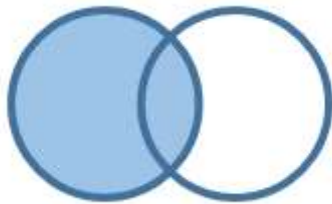
Pada contoh diatas, terlihat bahwa Pelanggan dengan nama Delta tidak muncul, hal ini disebabkan pelanggan tersebut tidak pernah melakukan transaksi. Transaksi dengan id\_trx 5 tidak muncul, karena transaksi tersebut memiliki nilai id\_transaksi NULL, sehingga tidak terhubung ke pelanggan manapun.

## II. OUTER JOIN

Cara kedua untuk menggabungkan tabel pada MySQL adalah menggunakan outer join. Pada outer join, data pada salah satu tabel akan ditampilkan semua, sedangkan data pada tabel yang lain hanya akan ditampilkan jika data tersebut ada pada tabel pertama. Pada MySQL, OUTER JOIN dibagi menjadi dua, yaitu LEFT OUTER JOIN dan RIGHT OUTER JOIN.

### 1 Left Outer Join.

Pada LEFT OUTER JOIN, semua data pada tabel sebelah kiri akan ditampilkan, sedangkan data pada tabel disebelah kanan hanya akan ditampilkan jika data terkait pada tabel tersebut muncul di tabel sebelah kiri. Jika di gambarkan dalam bentuk diagram venn, bentuk LEFT OUTER JOIN tampak pada gambar berikut:



LEFT OUTER JOIN dapat ditulis menggunakan dua cara, yaitu (1) dengan klausa LEFT OUTER JOIN, (2) cukup dengan klausa LEFT JOIN saja, sobat bebas memilih salah satu yang penting konsisten, saya sendiri prefer menggunakan bentuk kedua karena lebih simpel.

Contoh kita tampilkan semua data pelanggan beserta data transaksinya, jalankan query berikut:

```
SELECT pl.id_pelanggan, nama, tgl_transaksi, total_transaksi
FROM pelanggan pl
LEFT JOIN penjualan USING(id_pelanggan)
```

Hasil:

id_pelanggan	nama	tgl_transaksi	total_transaksi
1	Alfa	2017-02-22	230000
3	Charlie	2017-02-22	195000
2	Beta	2017-01-01	1710000
1	Alfa	2017-02-04	310000
4	Delta	NULL	NULL

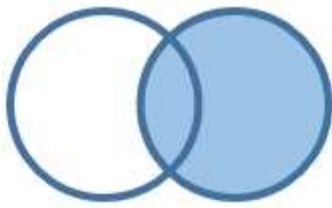
Pada contoh diatas, terlihat bahwa dengan LEFT JOIN, data pada tabel sebelah kiri, yaitu tabel pelanggan akan ditampilkan semua, sedangkan data pada tabel sebelah kanan hanya akan ditampilkan jika nilai kolom id\_pelanggan nya muncul pada tabel pertama, yaitu id\_pelanggan 1, 2, dan 3

### 2 Right Outer Join

Kebalikan dari LEFT OUTER JOIN, pada RIGHT OUTER JOIN, data pada tabel sebelah kanan akan ditampilkan semua, sedangkan data pada sebelah kiri hanya ditampilkan jika data terkait pada tabel tersebut muncul pada tabel sebelah kanan.

Jika digambarkan dalam bentuk diagram venn, maka, bentuk right outer join akan tampak seperti gambar berikut:





Sama seperti LEFT OUTER JOIN, RIGHT OUTER JOIN juga dapat ditulis menggunakan dua cara, yaitu ditulis secara utuh atau cukup RIGHT JOIN saja. Saya sendiri lebih prefer menggunakan RIGHT JOIN saja.

Contoh kita tampilkan semua data transaksi beserta data pelanggannya, jalankan query berikut:

```
SELECT pl.id_pelanggan, nama, tgl_transaksi, total_transaksi
FROM pelanggan pl
RIGHT JOIN penjualan USING(id_pelanggan)
```

Hasil:

id_pelanggan	nama	id_transaksi	tgl_transaksi	total_transaksi
1	Alfa	1	2017-02-22	230000
1	Alfa	4	2017-02-04	310000
2	Beta	3	2017-01-01	1710000
3	Charlie	2	2017-02-22	195000
NULL	NULL	5	2017-02-10	80000

Pada contoh diatas, terlihat bahwa semua data pada tabel disebelah kanan, yaitu tabel penjualan akan ditampilkan semua, sedangkan pada tabel sebelah kiri hanya ditampilkan yang data id\_pelanggan nya muncul pada tabel penjualan.

Perlu kah RIGHT JOIN?

Jika kita teliti lebih lanjut, sebenarnya right join hanya memindah posisi tabel, dari kiri ke kanan, contoh query pada right join dapat kita ubah dengan menjadi LEFT JOIN dengan mengubah posisi tabel, perhatikan contoh berikut:

```
SELECT pl.id_pelanggan, nama, id_transaksi, tgl_transaksi,
total_transaksi
FROM penjualan pl
LEFT JOIN pelanggan USING(id_pelanggan)
```

Perhatikan pada contoh diatas, kita balik posisi tabel penjualan dan tabel pelanggan. Jika query tersebut dijalankan, hasil yang kita peroleh adalah:

id_pelanggan	nama	id_transaksi	tgl_transaksi	total_transaksi
1	Alfa	1	2017-02-22	230000
1	Alfa	4	2017-02-04	310000
2	Beta	3	2017-01-01	1710000
3	Charlie	2	2017-02-22	195000
NULL	NULL	5	2017-02-10	80000

Perhatikan bahwa hasil diatas sama persis dengan hasil pada contoh right join, jadi kesimpulannya, agar memudahkan, cukup gunakan salah satu bentuk outer join saja, LEFT JOIN atau RIGHT JOIN, saya sendiri prefer menggunakan LEFT JOIN.

### III. IMPLISIT JOIN

Sejauh ini, kita menampilkan data dari beberapa tabel MySQL dengan menggunakan klausa JOIN.

Selain menggunakan klausa JOIN, terdapat satu cara lagi untuk menggabungkan tabel MySQL, yaitu menggunakan implisit join, disebut implisit join karena kita tidak menggunakan klausa JOIN,

pada implisit join, kriteria hubungan antar tabel di definisikan pada klausa **WHERE**. Sebagai contoh, mari kita gabungkan tabel pelanggan dan penjualan, jalankan query berikut:

```
SELECT pl.id_pelanggan, nama, id_transaksi, tgl_transaksi,
       total_transaksi
FROM pelanggan pl, penjualan pn
WHERE pl.id_pelanggan = pn.id_pelanggan
```

Hasil yang kita peroleh:

id_pelanggan	nama	id_transaksi	tgl_transaksi	total_transaksi
1	Alfa	1	2017-02-22	230000
3	Charlie	2	2017-02-22	195000
2	Beta	3	2017-01-01	1710000
1	Alfa	4	2017-02-04	310000

Perhatikan bahwa hasil tersebut sama persis dengan hasil pada contoh **INNER JOIN**, sehingga dapat disimpulkan bahwa implisit join = inner join.

Implisit join mensyaratkan kedua tabel memiliki data yang sama ( **WHERE pl.id\_pelanggan = pn.id\_pelanggan** ), sehingga implisit join ini hanya berlaku pada **INNER JOIN**, dan tidak bisa digunakan untuk **OUTER JOIN**.

Implisit **JOIN** ini merupakan cara lama ketika pertama kali standar SQL dibuat, setelah muncul standar yang lebih baru (SQL2) maka mulai digunakanlah klausa **JOIN**. Saya sendiri prefer menggunakan klausa **JOIN** karena lebih mudah dibaca dan dipahami, terutama hubungan antara tabel yang digabungkan.

Pada bentuk klausa **JOIN**, hubungan antar tabel dinyatakan pada klausa **ON** atau **USING**, sedangkan filter datanya dilakukan pada klausa **WHERE**, misal:

```
SELECT pl.id_pelanggan, nama, id_transaksi, tgl_transaksi,
       total_transaksi
FROM pelanggan pl
LEFT JOIN penjualan pn USING (id_pelanggan)
WHERE pl.id_pelanggan = 2 OR pl.id_pelanggan = 1
```

sedangkan pada implisit **JOIN**, hubungan antar tabel dan filter datanya, semua didefinisikan pada klausa **WHERE**, misal:

```
SELECT pl.id_pelanggan, nama, id_transaksi, tgl_transaksi,
       total_transaksi
FROM pelanggan pl, penjualan pn
WHERE pl.id_pelanggan = pn.id_pelanggan
      AND (pl.id_pelanggan = 2 OR pl.id_pelanggan = 1)
```



## MATERI PERTEMUAN 22-23

### Trigger dan Store Prosedure



#### TAHUKAH KAMU...?

Pada kesempatan ini, kita akan membahas:

- Fungsi dengan Parameter
- Parameter dengan Nilai Default
- Fungsi yang Mengembalikan Nilai
- Memanggil Fungsi di dalam Fungsi

Banyak fungsi *build-in* dari php yang sering kita gunakan, seperti **print()**, **print\_r()**, **unset()**, dll. Selain fungsi-fungsi tersebut, kita juga dapat membuat fungsi sendiri sesuai kebutuhan

TRIGGER adalah kumpulan kode SQL yang berjalan secara otomatis untuk mengeksekusi perintah INSERT, update, delete. biasanya trigger akan dijalankan sebelum atau sesudah proses insert, update, delete (Perintah DML)

#### Cara penulisan TRIGGER

```
delimiter $$
create trigger nama_trigger
{before | after} {insert | update | delete }
  on nama_table
  for each row
begin
  kode sql
end$$
delimiter ;
```

Untuk memulai menggunakan TRIGGER kita gunakan CREATE TRIGGER dilanjutkan nama TRIGGER yang ingin dibuat

{BEFORE | AFTER} adalah waktu TRIGGER akan dijalankan, apakah sebelum atau sesudah database dimodifikasi oleh perintah DML

{INSERT | UPDATE | DELETE} adalah perintah DML yang mengaktifkan TRIGGER

Lebih detail waktu TRIGGER akan dijelaskan di tabel berikut

Waktu TRIGGER Keterangan TRIGGER

BEFORE INSERT TRIGGER dijalankan sebelum record dimasukkan ke database

AFTER INSERT TRIGGER dijalankan sesudah record dimasukkan ke database

BEFORE UPDATE TRIGGER dijalankan sebelum record dirubah di database

AFTER UPDATE TRIGGER dijalankan sesudah record dirubah database

BEFORE DELETE TRIGGER dijalankan sebelum record dihapus di database

AFTER DELETE TRIGGER dijalankan sesudah record dihapus di database  
ON mendefinisikan table yang mengaktifkan TRIGGER

BEGIN END adalah pernyataan yang membungkus kode TRIGGER

Pastikan diawal gunakan DELIMITER \$\$ dan diakhir dikembalikan ke DELIMITER ; seperti dalam membuat Stored Procedure

Hands On

Pada Hands-On TRIGGER akan dibuat 2 table yaitu table mahasiswa dan table log\_mahasiswa

Table mahasiswa -> menyimpan data mahasiswa

Table log\_mahasiswa -> menyimpan perubahan data mahasiswa

Jadi setiap ada perubahan data (UPDATE) alamat mahasiswa pada table mahasiswa maka akan disimpan di table log\_mahasiswa tentang histori perubahan data alamat tersebut.

Dengan adanya log perubahan data mahasiswa maka akan memudahkan dalam melihat histori data mahasiswa yang pernah berubah dalam sistem.

Table mahasiswa

```
CREATE TABLE mahasiswa
(
    nim INT(10),
    nama VARCHAR(100),
    alamat VARCHAR(100),
    PRIMARY KEY(nim)
);

INSERT INTO mahasiswa
VALUES
(21400200, "faqih", "bandung"),
(21400201, "ina", "jakarta"),
(21400202, "anto", "semarang"),
(21400203, "dani", "padang"),
(21400204, "jaka", "bandung"),
(21400205, "nara", "bandung"),
(21400206, "senta", "semarang");
```

P.S. Referensi contoh membuat table dan record dapat dibaca di materi DDL dan DML

Table log\_mahasiswa

```
CREATE TABLE log_mahasiswa
(
    id_log INT(10) AUTO_INCREMENT,
    nim INT(10),
    alamat_lama VARCHAR(100),
    alamat_baru VARCHAR(100),
    waktu DATE,
    PRIMARY KEY(id_log)
);
```

Membuat TRIGGER

Kita akan menyimpan data perubahan alamat sebelum perintah UPDATE dijalankan

```
DELIMITER $$
CREATE TRIGGER update_alamat_mahasiswa
  BEFORE UPDATE
  ON mahasiswa
  FOR EACH ROW
BEGIN
  INSERT INTO log_mahasiswa
  set nim = OLD.nim,
  alamat_lama=old.alamat,
  alamat_baru=new.alamat,
  waktu = NOW();
END$$
DELIMITER ;
```

Keyword OLD digunakan untuk mengambil data kolom di table yang lama sedangkan keyword NEW digunakan untuk mengambil data kolom di table yang baru

Sekarang kita akan coba update alamat mahasiswa dengan NIM 21400200. Sebelum diupdate alamat mahasiswa dengan NIM 21400200 adalah "bandung"

Kita ganti alamat "bandung" menjadi "surabaya"

```
UPDATE mahasiswa
SET alamat = 'surabaya'
WHERE nim = 21400200;
```

Sekarang coba lakukan perintah SELECT untuk melihat table log\_mahasiswa

```
> SELECT * FROM log_mahasiswa;
+-----+-----+-----+-----+-----+
| id_log | nim  | alamat_lama | alamat_baru | waktu  |
+-----+-----+-----+-----+-----+
| 1      | 21400200 | bandung    | surabaya    | 2019-11-02 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Oke, record baru secara otomatis telah ditambahkan ke table log\_mahasiswa untuk mahasiswa dengan NIM 21400200 yang telah diubah alamat awal "bandung" menjadi "surabaya"

Sedangkan pada table mahasiswa alamat yang tercantum adalah alamat yang baru

```
> SELECT * FROM mahasiswa WHERE nim=21400200;
+-----+-----+-----+
| nim  | nama  | alamat  |
+-----+-----+-----+
| 21400200 | faqih | surabaya |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Stored Procedure adalah sebuah fungsi berisi kode SQL yang dapat digunakan kembali. Dalam Stored Procedure juga dapat dimasukkan parameter sehingga fungsi dapat digunakan lebih dinamis berdasarkan parameter tersebut

### Cara penulisan Stored Procedure

```
DELIMITER $$
```

```
CREATE PROCEDURE nama_procedure()  
BEGIN  
    kode sql  
END$$
```

```
DELIMITER ;
```

Sedangkan untuk menjalankan Stored procedure adalah

```
CALL nama_procedure();  
Hands ON
```

Sebelum mencoba menggunakan Stored Procedure kita harus mempunyai table yang siap digunakan. Referensi contoh membuat table dan record dapat dibaca di materi DDL dan DML. Buat table mahasiswa dan isi nilainya

```
CREATE TABLE mahasiswa  
(  
    nim INT(10),  
    nama VARCHAR(100),  
    alamat VARCHAR(100),  
    PRIMARY KEY(nim)  
);  
  
INSERT INTO mahasiswa  
VALUES  
(21400200,"faqih","bandung"),  
(21400201,"ina","jakarta"),  
(21400202,"anto","semarang"),  
(21400203,"dani","padang"),  
(21400204,"jaka","bandung"),  
(21400205,"nara","bandung"),  
(21400206,"senta","semarang");
```

```
> SELECT * FROM mahasiswa;  
+-----+-----+-----+  
| nim  | nama  | alamat  |  
+-----+-----+-----+  
| 21400200 | faqih | bandung |  
| 21400201 | ina   | jakarta |  
| 21400202 | anto  | semarang |  
| 21400203 | dani  | padang  |  
| 21400204 | jaka  | bandung |  
| 21400205 | nara  | bandung |  
| 21400206 | senta | semarang |  
+-----+-----+-----+  
7 rows in set (0.00 sec)
```

Membuat Stored Procedure selectMahasiswa() untuk mendapatkan seluruh NIM dan NAM mahasiswa

```
DELIMITER $$

CREATE PROCEDURE selectMahasiswa()
BEGIN
    SELECT nim, nama FROM mahasiswa;
END$$

DELIMITER ;
```

Untuk memanggil Stored Procedure selectMahasiswa()

CALL selectMahasiswa()  
Hasilnya adalah

```
> CALL selectMahasiswa();
+-----+-----+
| nim   | nama   |
+-----+-----+
| 21400200 | faqih |
| 21400201 | ina   |
| 21400202 | anto   |
| 21400203 | dani   |
| 21400204 | jaka   |
| 21400205 | nara   |
| 21400206 | senta   |
+-----+-----+
7 rows in set (0.00 sec)
```

Jadi, setiap ingin menampilkan NIM dan NAMA mahasiswa kita tidak perlu membuat kode SQL seperti biasanya. Cukup simpan kode SQL di Stored Procedure dan bisa kita panggil dan gunakan berulang – ulang

Stored Procedure dengan Parameter

Kita juga memasukkan parameter di Stored Procedure agar menjadi lebih dinamis

Contoh, kita ingin membuat kode SQL untuk mencari data mahasiswa berdasarkan alamat.

```
DELIMITER $$

CREATE PROCEDURE alamatMahasiswa
(
    alamatMhs VARCHAR(100)
)
BEGIN
    SELECT *
    FROM mahasiswa
    WHERE alamat = alamatMhs;
END$$

DELIMITER ;
```

Di dalam nama Stored Procedure terdapat parameter alamatMhs dengan tipe data varchar. Saat masuk ke kode SQL yaitu mencari mahasiswa dengan alamat yang sama dengan parameter yang diinputkan. Cara memanggilnya adalah

CALL alamatMahasiswa("bandung")  
Hasilnya

```
> CALL alamatMahasiswa("bandung");
+-----+-----+-----+
| nim   | nama  | alamat |
+-----+-----+-----+
| 21400200 | faqih | bandung |
| 21400204 | jaka  | bandung |
| 21400205 | nara  | bandung |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

Dengan Stored Procedure alamatMahasiswa() kita mencari data mahasiswa yang berasal dari "bandung"

DML dengan Stored Procedure

Stored Procedure tidak hanya bisa diterapkan di Data Query Language (DQL) tetapi juga Data Manipulation Language (DML)

Misal kita ingin memasukkan data mahasiswa dengan Stored Procedure

```
DELIMITER $$
CREATE PROCEDURE insertMahasiswa
(
    nimMhs INT(10),
    namaMhs VARCHAR(100),
    alamatMhs VARCHAR(100)
)
BEGIN
    INSERT INTO mahasiswa
    VALUES (nimMhs, namaMhs, alamatMhs);
END$$
DELIMITER ;
```

Jika kita ingin memasukkan record baru di table mahasiswa maka akan ada 3 parameter saat memanggil Stored Procedure insertMahasiswa()

```
> CALL insertMahasiswa(21400207,"joni","jakarta");
Query OK, 1 row affected (0.05 sec)
```

```
> SELECT * FROM mahasiswa;
+-----+-----+-----+
| nim   | nama  | alamat |
+-----+-----+-----+
| 21400200 | faqih | bandung |
| 21400201 | ina   | jakarta |
| 21400202 | anto  | semarang |
| 21400203 | dani  | padang  |
| 21400204 | jaka  | bandung |
| 21400205 | nara  | bandung |
| 21400206 | senta | semarang |
| 21400207 | joni  | jakarta |
+-----+-----+-----+
8 rows in set (0.00 sec)
```

Selanjutnya jika ingin memasukkan data mahasiswa ke table mahasiswa tidak perlu membuat kode INSERT berkali-kali. Kita bisa gunakan Stored Procedure insertMahasiswa()



untuk menggantikan proses INSERT yang biasanya kita gunakan

Jadi Stored Procedure sangat penting dan akan memudahkan dalam menggunakan kode yang ingin dieksekusi secara berulang – ulang.

VIEW adalah perintah untuk membuat table virtual yang menyimpan kode SQL. Dengan view kita bisa membuat kode SQL yang kompleks dikemas menjadi satu table sederhana

View akan menyimpan kode SQL yang kompleks tadi menjadi single table virtual yang lebih mudah untuk digunakan

Membuat view

Materi MySQL dibagi menjadi beberapa tutorial antara lain: Cara penulisan view

CREATE VIEW <nama view>

AS

Kode SQL

Saat kita mengeksekusi CREATE VIEW maka akan terbentuk table virtual yang menyimpan kode SQL

Contoh, Kita akan membuat kode SQL yang menghubungkan table mahasiswa dan table transaksi secara INNER JOIN dan menyimpannya ke view

Note : Baca dan pahami cara JOIN di MySQL

Kode untuk INSERT table mahasiswa dan transaksi

```
INSERT INTO mahasiswa
VALUES
(21400200,"faqih","bandung"),
(21400201,"ina","jakarta"),
(21400202,"anto","semarang"),
(21400203,"dani","padang"),
(21400204,"jaka","bandung"),
(21400205,"nara","bandung"),
(21400206,"senta","semarang");

INSERT INTO transaksi
VALUES
(1,21400200,"Buku Informatika"),
(2,21400202,"Buku Teknik Elektro"),
(3,21400203,"Buku Matematika"),
(4,21400206,"Buku Fisika"),
(5,21400207,"Buku Bahasa Indonesia"),
(6,21400210,"Buku Bahasa Daerah"),
(7,21400211,"Buku Kimia");
```

Kemudian kita join table mahasiswa dan table transaksi berdasarkan field NIM secara INNER JOIN

Hasilnya adalah

```
+-----+-----+-----+-----+
| nim | nama | alamat | buku |
+-----+-----+-----+-----+
| 21400200 | faqih | bandung | Buku Informatika |
| 21400202 | anto | semarang | Buku Teknik Elektro |
| 21400203 | dani | padang | Buku Matematika |
| 21400206 | senta | semarang | Buku Fisika |
```

```
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Dengan view kita bisa membuat table virtual yang menyimpan query join di atas

```
CREATE VIEW transaksiMhs AS
SELECT mahasiswa.nim, nama, alamat, buku
FROM mahasiswa
INNER JOIN transaksi
ON mahasiswa.nim = transaksi.nim
```

Jadi kita telah membuat table virtual dengan view dengan nama transaksiMhs.

Cara menggunakannya adalah seperti melakukan query table biasa

```
SELECT *
FROM transaksiMhs
```

Contoh lain, misal ingin query nama mahasiswa yang meminjam buku matematika

```
> SELECT nama, buku FROM transaksiMhs WHERE buku="Buku Matematika";
```

```
+-----+-----+
| nama | buku  |
+-----+-----+
| dani | Buku Matematika |
+-----+-----+
1 row in set (0.00 sec)
```

Menghapus view

Cara penulisan untuk menghapus view yang sudah tidak digunakan

```
DROP VIEW <nama view>;
Contoh menghapus view transaksiMhs
DROP VIEW transaksiMhs
```