ISBN 978-623-6081-18-1 (PDF)

Basis Data

Ika Ratna Indra Astutik, S.Kom, MT.
Mohammad Alfan Rosid, M.Kom.

Universitas Muhammadiyah Sidoarjo

Basis Data

Ika Ratna Indra Astutik, S.Kom, MT.
Mohammad Alfan Rosid, M.Kom.

Universitas Muhammadiyah Sidoarjo

BUKU AJAR

Universitas Muhammadiyah Sidoarjo

BUKU AJAR
BASIS DATA UNTUK INFORMATIKA

Oleh:
Ika Ratna Indra Astutik, S.Kom, M.T.
Mochamad Alfan Rosid, S.Kom, M.Kom.

Diterbitkan oleh **UMSIDA PRESS**

UNIVERSITAS MUHAMMADIYAH SIDOARJO

2020

BUKU AJAR
BASIS DATA UNTUK INFORMATIKA

Penulis:
Ika Ratna Indra Astutik, S.Kom, M.T.
Mochamad Alfan Rosid, S.Kom, M.Kom.

ISBN :
978-623-6081-18-1

Editor:
Muhammad Suryawinata, S.Pd, M.Kom

Design Sampul dan Tata Letak:
Mochammad Nashrullah, S.Pd.
Amy Yoga Prajati, S.Kom

Penerbit:
UMSIDA Press
Anggota IKAPI No. 218/Anggota Luar Biasa/JTI/2019
Anggota APPTI No.002 018 1 09 2017

Redaksi
Universitas Muhammadiyah Sidoarjo
Jl. Mojopahit No 666B
Sidoarjo, Jawa Timur

Cetakan Pertama, September 2020

KATA PENGANTAR

Assalamualaikum wa Rahmatullah wa Barakatuh

Puji syukur kami panjatkan ke hadirat Allah SWT karena atas rahmat dan karunia-Nya kami bisa menyelesaikan pembuatan buku ajar "Basis Data Untuk Informatika" bagi mahasiswa Informatika Universitas Muhammadiyah Sidoarjo.

Tim penulis mengucapkan terimakasih kepada:

1. Dr. Hindarto, M.T., Dekan Fakultas Sains dan Teknologi yang memberikan arahan dan motivasi kepada penulis dalam menyelesaikan buku ajar ini.
2. Ir. Sumarno., Kaprodi Informatika yang telah memberikan dukungan untuk menyusun buku ajar ini.
3. Rekan-rekan dosen pengampu Mata Kuliah Basis Data di prodi Informatika yang telah berbagi pengalaman dalam mengampu mata kuliah tersebut.

Penulis mengharapkan adanya saran dan kritik agar buku ajar Basis Data untuk Informatika bisa lebih baik lagi dan tentunya sesuai dengan peraturan yang berlaku. Terima kasih

Wassalamualaikum wa Rahmatullah wa Barakatuh

Tim Penulis

DAFTAR ISI

BATANG TUBUH DAN
SUB-CAPAIAN PEMBELAJARAN MATA KULIAH

BAB	Sub-Capaian Pembelajaran Mata Kuliah
BAB I PENGENALAN BASIS DATA	1. Mahasiswa mampu memahami definisi Basis data. 2. Mahasiswa mampu memahami keunggulan penerapan Basis data. 3. Mahasiswa mampu memahami konsep dasar Basis data.
BAB II LINGKUNGAN BASIS DATA	1. Mahasiswa mampu memahami Arsitektur Basis data. 2. Mahasiswa mampu memahami model data di Basis data.
BAB III DATABASE MANAGEMENT SYSTEM (DBMS)	1. Mahasiswa mampu memahahami sejarah DBMS. 2. Mahasiswa mampu memahami entitas, atribut dan key di basis data. 3. Mahasiswa mampu mengaplikasikan konsep basis data ke DBMS.
BAB IV ER MODEL DAN RDBM	1. Mahasiswa mampu memahami dan menganalisa hubungan antar tabel di Basis data. 2. Mahasiswa mampu mengaplikasikan model entity relationship di DBMS. 3. Mahasiswa mampu mengaplikasikan model RDBM.
BAB V NORMALISASI	1. Mahasiswa mampu memahami konsep normalisasi data.

	2. Mahasiswa mampu mengaplikasikan konsep normalisasi di DBMS.
BAB VI BAHASA BASIS DATA	1. Mahasiswa mampu memahami DDL, DML dan DCL di Basis data. 2. Mahasiswa mampu mengaplikasikan konsep DDL, DML dan DCL di DBMS.
BAB VII MYSQL	1. Mahasiswa mampu memahami sejarah dan perkembangan MySQL. 2. Mahasiswa mampu mengoperasikan perintah-perintah yang ada di MySQL. 3. Mahasiswa mampu mengaplikasikan konsep konsep Basis data ke MySQL.
BAB VIII DATA DEFINITION LANGUAGE (DDL)	1. Mahasiswa mampu memahami perintah-perintah DDL di Basis data. 2. Mahasiswa mampu mengaplikasikan konsep DDL di DBMS.
BAB IX DATA MANIPULATION LANGUAGE (DML)	1. Mahasiswa mampu memahami perintah-perintah DML di MySQL. 2. Mahasiswa mampu mengaplikasikan konsep DML di MySQL.
BAB X FUNGSI DI MYSQL	1. Mahasiswa mampu memahami fungsi di MySQL. 2. Mahasiswa mampu mengaplikasikan konsep Fungsi di MySQL.
BAB XI QUERY DAN VIEW	1. Mahasiswa mampu memahami query dan view di MySQL. 2. Mahasiswa mampu mengaplikasikan konsep Query dan view di MySQL melalui studi kasus.

BAB XII HAK AKSES USER	1. Mahasiswa mampu memahami hak akses user di MySQL. 2. Mahasiswa mampu mengaplikasikan konsep hak akses user di MySQL melalui studi kasus.

BAB 1
PENGENALAN BASIS DATA

Bab ini menjelaskan tentang pengetahuan dasar basis data khususnya data dan informasi.

1.1. Data dan Informasi

Untuk memahami basis data maka terlebih dahulu harus mengenal tentang data dan informasi. Karena ada sebuah pernyataan yang menyebutkan bahwa data yang dikumpulkan merupakan hal yang sangat penting bagi kehidupan manusia sehari-hari khususnya pada saat era milenia sekarang ini. Pernyataan tersebut memang terbukti karena sehari-hari manusia memerlukan dan menggunakan data dalam segala hal termasuk melakukan perancangan serta pengambilan keputusan.

Contoh yang sangat sederhana yaitu mahasiswa sangat memerlukan data yaitu nilai tugas, nilai ujian UTS atau UAS, nilai indeks prestasi semester, nilai indeks prestasi komulatif, biaya kuliah dan lain sebagainya. Begitu juga dengan instansi, perusahaan atau organisasi baik yang berskala kecil maupun besar sangat membutuhkan data untuk pengambilan keputusan. Saat ini, semua perusahaan sudah menggunakan sistem informasi untuk mengolah datanya, dimana sistem informasi tersebut sangat memerlukan data-data untuk bisa menghasilkan informasi yang diperlukan perusahaan. Sehingga antara sistem informasi dan data-data memiliki hubungan yang sangat erat dan tidak mungkin bisa di pisahkan satu dengan yang lainnya (Indrajani, 2018).

Sebelum membahas lebih jauh tentang basis data maka harus diketahui pengertian tentang data dan informasi, yaitu sebagai berikut :

1. Data

   Menurut Edhy Sutanta (2011), Data merupakan sesuatu yang menyangkut barang, kejadian, aktivitas, atau transaksi yang telah tercatat, dikelompokkan dan disimpan tapi belum memiliki makna atau arti bagi penggunanya. Data sering sekali disebut sebagai sebuah bahan mentah informasi. Data bisa berbentuk Teks, Citra, Audio dan Video atau Data Terformat(jam, tanggal, nilai mata uang) (Sutanta, 2011).

Gambar 1.1. Macam-macam data

Data di bedakan menjadi beberapa macam yaitu antara lain :

a. Menurut sifatnya
   - ❖ Data Kualitatif

     data kualitatif adalah data yang ditampilakn tidak berbentuk angka.

     Contoh :

     Kuesioner Pertanyaan tentang suasana kerja, kualitas pelayanan sebuah perguruan tinggi, kualitas produk perusahaan, dll.
   - ❖ Data Kuantitatif

     Data kuantitatif adalah data yang ditampilkan dalam bentuk angka.

     Contoh :

     besarnya penbisaan perusahaan, harga saham, dll.

b. Menurut sumbernya :

- ❖ Data Internal

  data yang diperoleh dari dalam suatu perusahaan atau organisasi yang menggambarkan keadaan perusahaan atau organisasi tersebut.

  Contoh :

  Data profil perusahaan, jumlah pegawai, jumlah modalnya, atau jumlah produksinya, dll.

- ❖ Data Eksternal

  Data eksternal adalah data yang diperoleh dari luar suatu perusahaan organisasi yang bisa menggambarkan faktor-faktor yang mungkin mempengaruhi hasil kerja suatu perusahaan atau organisasi.

  Contoh :

  daya beli masyarakat mempengaruhi hasil penjualan suatu perusahaan

c. Menurut cara memperolehnya :

- ❖ Data Primer (primary data)

  Data primer adalah data yang dikumpulkan sendiri oleh perorangan atau suatu organisasi secara langsung dari obyek yang diteliti untuk kepentingan studi yang bersangkutan.

  Contoh :

  Data interview, observasi

- ❖ Data Sekunder (*secondary data*)

  Data sekunder adalah data yang diperoleh atau dikumpulkan dan disatukan oleh studi-studi sebelumnya atau yang diterbitkan oleh berbagai instansi lain.

  Contoh :

  Data dokumentasi dan arsip-arsip resmi

d. Menurut waktu pengumpulannya :

- ❖ Data cross section

  Data yang dikumpulkan pada suatu waktu tertentu (*at a point of time*) untuk menggambarkan keadaan dan kegiatan pada waktu tersebut.

  Contoh :

  Data penelitian yang menggunakan kuesioner.

- ❖ Data berkala (*time series data*)

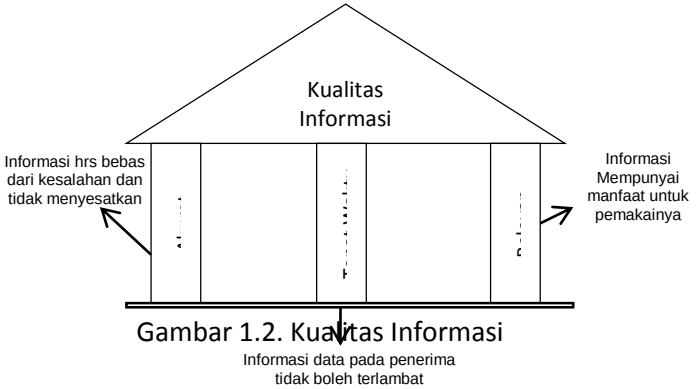
  Data yang dikumpulkan dari waktu ke waktu untuk melihat perkembangan suatu kejadian/kegiatan selama periode tersebut

  Contoh :

  Perkembangan masyarakat yang terkena serangan jantung mulai tahun 2014-2016

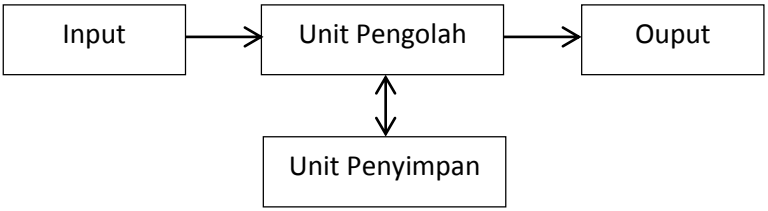
2. Informasi

Informasi, adalah data yang telah dikelola dalam bentuk tertentu untuk memberikan makna atau arti bagi penerimanya. Data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau mendatang. Kualitas informasi tergantung pada :



Gambar 1.2. Kualitas Informasi

Perubahan data menjadi sebuah informasi data di lihat pada gambar 1.3. Dalam gambar tersebut Input merupakan data yang akan diolah atau di proses oleh unit pengolah dalam perusahaan, sedangkan output merupakan informasi yang dihasilkan dari pengelolaan data yang di inputkan untuk memberikan informasi yang dibutuhkan oleh perusahaan. Suatu unit penyimpanan atau memori sekunder juga sangat di butuhkan sebagai alat penyimpan data dalam bentuk basis data.



Gambar 1.3. Perubahan data menjadi Informasi (Sutanta, 2003)

1.2. Definisi Basis Data (*Database*)

Konsep basis data telah muncul dan mulai berkembang seiring dengan adanya kebutuhan pengolahan data-data untuk memenuhi kebutuhan akan informasi. Dengan adanya perkembangan komputer pada tahun 1945, telah memunculkan pandangan dan pengetahuan baru tentang konsep penyimpanan data-data di dalam basis data. Pada tahap ini data-data diolah berdasarkan prinsip *file processing*. Generasi DBMS pertama didesain oleh Charles Bachman di perusahaan General Electric pada awal tahun 1960 yang disebut sebagai Penyimpanan Data Terintegrasi (*Integrated Data Store*).

Kemudian dilanjutkan penemuan tentang model data relational pada tahun 1970 oleh Edgar Codd di Laboratorium Penelitian di San Jose. Di tahun 1980, model relasional menjadi DBMS yang paling banyak dipakai pengguna. Pada tahun 1990 an Bahasa *query* SQL dikembangkan untuk basis data relasional oleh IBM.

Basis data atau biasa disebut Database dan Sistem Basis data menjadi hal yang utama dalam kehidupan masyarakat modern saat ini khususnya dalam pengelolahan data menjadi sebuah informasi. Penggunaan basis data di Perguruan Tinggi, sekolah, bank, perusahaan, perpustakaan, supermarket, dll.

Basis data terdiri dari dua kata yaitu "basis" dan "data", dimana basis mempunyai arti Markas, Tempat Berkumpul, Tempat Bersarang, Asas, Dasar, Gudang. Sedangkan data mempunyai arti Fakta tentang obyek yang diteliti atau dikumpulkan dalam dunia nyata. Jadi basis data bisa di artikan :

a. Himpunan kelompok data atau arsip yang saling berhubungan atau berelasi yang di tata sedemikian rupa agar nanti bisa dimanfaatkan kembali dengan cepat dan mudah.
b. kumpulan data-data yang saling berhubungan atau berelasi yang disimpan secara bersama sedemikian rupa dan tanpa ada pengulangan (redundansi) yang tidak perlu dengan tujuan untuk memenuhi berbagai kebutuhan pengguna.
c. kumpulan file atau tabel atau arsip yang saling berhubungan atau berelasi yang disimpan dalam suatu media penyimpanan elektronik.

Menurut Edhy Sutanta (2018) basis data merupakan suatu kumpulan data yang saling terhubung yang disimpan secara bersamaan pada suatu media penyimpanan dan tidak diperlukan suatu kerangkapan data (walaupun ada maka kerangkapan data-

data tersebut harus seminimal mungkin dan terkontrol, data-data tersimpan dengan cara-cara tertentu sehingga mudah untuk dipakai, data-data tersebut bisa digunakan oleh lebih dari satu program-program aplikasi secara optimal. Data-data di simpan tanpa mengalami ketergantungan dengan program yang akan digunakan (Sutanta, 2018).

Basis data memiliki beberapa kriteria-kriteria penting yang harus dipenuhi, yaitu :

1. Basis data berorientasi pada data tidak berorientasi pada program yang akan menggunakannya.
2. Data-data dalam basis data bisa digunakan oleh pemakai yang berbeda-beda atau beberapa program aplikasi tanpa harus mengubah basis data.
3. Data-data di dalam basis data bisa berkembang dengan mudah dan cepat baik valume maupun strukturnya.
4. Data-data yang ada di dalam basis data bisa memenuhi kebutuhan sistem-sistem baru secara mudah.
5. Data-data bisa digunakan dengan cara yang berbeda-beda tergantug tujuannya.
6. Kerangkapan data sangat kecil terjadi.

Menurut Fatta (2007) basis data adalah Kumpulan dari tabel-tabel yang saling berhubungan yang disimpan dalam media penyimpanan tertentu. Atau bisa juga di artikan kumpulan data-data yang saling berhubungan yang disimpan secara bersama tanpa adanya pengulangan data. Basis data merupakan sekumpulan data-data yang saling berhubungan atau saling ber-relasi. Dimana relasi tersebut ditunjukan dengan sebuah kunci key dari tiap-tiap file yang ada (Fatta, 2007).

1.3. Kenapa Diperlukan Basis Data

Dalam pengembangan suatu sistem informasi, basis data mempunyai peran yang sangat penting dan diperlukan karena :

a. Basis data merupakan salah satu komponen penting suatu sistem informasi dikarenakan basis data berfungsi untuk menyediakan informasi yang diperlukan pengguna.
b. Basis data bisa menentukan kualitas informasi yang dihasilkan oleh sebuah sistem informasi.
c. Data-data dalam basis data akan bisa saling berhubungan atau ber-relasi dengan data yang lainnya.
d. Basis data mengurangi adanya duplikasi data (*data redundancy*).
e. Basis data bisa mengurangi terjadinya pemborosan tempat penyimpanan luar (eksternal).

Orang-orang yang mempunyai peran petting dalam perancangan basis data antara lain :

a. Pemakai akhir dan vendor *Database Management System* (DBMS).
b. Programmer aplikasi basis data (database).
c. Administrator Basis data (*database Administrator*).

Tujuan perancangan basis data adalah sebagai berikut :

a. Kecepatan dan Kemudahan (*Speed*)
   Pengguna basis data bisa menyimpan data, melakukan perubahan atau manipulasi terhadap data, menampilkan kembali data dengan lebih mudah dan cepat dibandingkan dengan cara manual ataupun elektronis.
b. Efisiensi Ruang Penyimpanan (*Space*)

Dengan basis data kita mampu melakukan penekanan jumlah redundansi (pengulangan) data, baik dengan menerapkan sejumlah pengkodean atau dengan membuat relasi-relasi antara kelompok data yang saling berhubungan.

c. Keakuratan (*Accuracy*)
   Agar data sesuai dengan aturan dan batasan tertentu dengan cara memanfaatkan pengkodean atau pembentukan relasi antar data bersama dengan penerapan aturan/batasan (constraint) tipe data, domain data, keunikan data dsb.
d. Ketersediaan (*Availability*)
   Agar data bisa diakses oleh tiap-tiap pengguna yang membutuhkan, dengan penerapan teknologi jaringan serta melakukan pemindahan/penghapusan data yang sudah tidak digunakan / kadaluwarsa untuk menghemat ruang penyimpanan.
e. Kelengkapan (*Completeness*)
   Agar data yang dikelola senantiasa lengkap baik relatif terhadap kebutuhan pemakai maupun terhadap waktu, dengan melakukan penambahan baris-baris data ataupun melakukan perubahan struktur pada basis data; yakni dengan menambahkan field pada tabel atau menambah tabel baru.
f. Keamanan (*Security*)
   Agar data yang bersifat rahasia atau proses yang vital tidak jatuh ke orang / pengguna yang tidak berhak, yakni dengan penggunaan account (username dan password) serta menerapkan pembedaan hak akses tiap-tiap pengguna terhadap data yang bisa dibaca atau proses yang bisa dilakukan.
g. Kebersamaan (*Sharebility*)
   Agar data yang dikelola oleh sistem mendukung lingkungan multiuser (banyak pemakai), dengan menjaga / menghindari

munculnya problem baru seperti inkonsistensi data (karena terjadi perubahan data yang dilakukan oleh beberapa user dalam waktu yang bersamaan) atau kondisi deadlock (karena ada banyak pemakai yang saling menunggu untuk menggunakan data).

1.4. Keuntungan Penerapan Basis Data

Basis data yang dikembangkan dengan benar dan sesuai dengan kriteria atau batasan pegelolahan data bisa memberikan keuntungan sebagai berikut :

1. Kerangkapan data bisa diminimalkan. Apabila file-file basis data dalaam program aplikasi di ciptakan oleh perancang yang berbeda pada waktu yang berselang cukup lama maka beberapa bagian data akan mengalami kerangkapan. Pengembangan basis data yang disesuaikan dengan definisi basisi data bisa terhindarkan dari terjadinya kerangkapan data.
2. Inkonsistensi data bisa di hindari. Basis data yang terbebas dari kerangkapan data akan terhindar dari munculnya data-data yang tidak konsisten.
3. Data dalam basis data bisa digunakan secara bersama (multiuser). Dalam rangka meningkat kinerja sistem dan untuk memperoleh respons waktu yang cepat, beberapa sistem mengizinkan banyak pemakai untuk bisa meng-update data secara simultan.
4. Standarisasi data bisa dilakukan.
5. Pembatasan untuk keamanan data bisa dilakukan. Data-data yang ada di dalam basis data bisa diatur sehingga hanya pemakai tertentu yang mempunyai wewenang saja yang dapt mengaksesnya.
6. Integritas data bisa terjaga dan terpelihara (*maintenance*).

7. Perbedaan kebutuhan data antar pemakai bisa di seimbangkan. Tiap-tiap pemakai dalam sistem memiliki kebutuhan yang berbeda-beda. Pengembangan basis data yang benar bisa mampu menyeimbangkan perbedaan-perbedaan kebutuhan tersebut.

1.5. Konsep Dasar Basis Data

Beberapa istilah yang digunakan dalam pengelolaan basis data adalah sebagai berikut :

a. Field / Atribut

   Field merupakan implementasi dari suatu atribut data. Field merupakan unit terkecil dari data yang berarti (meaningful data) yang di simpan dalam suatu file atau basis data.

b. Record / Baris / Tupel

   Record merupakan koleksi dari field-field yang disusun dalam format yang telah ditentukan. selama desain sistem, record akan di klasifikasikan sebagai fixed-length record atau variable-length record. dimana Fixed-length record adalah tiap record mempunyai field, jumlah field dan ukuran logis yang sama sedangkan variable-length record merupakan mengizinkan record-record yang berbeda dalam ile yang sama memiliki panjang yang berbeda Istilah lain adalah baris atau Tupel

c. File dan Tabel

   Record-record yang serupa di kelompokkan dalam grup-grup yang disebut file. jadi file merupakan kumpulan semua kejadian dari struktur record yang diberikan. File juga bisa di artikan Kumpulan record sejenis yang mempunyai panjang atribut sama namun berbeda isi datanya. Tabel merupakan ekuivalen basis data relasional dari sebuah file

1.6. Ringkasan

1. Basis data adalah kumpulan file atau tabel atau arsip yang saling berhubungan atau berelasi yang disimpan dalam suatu media penyimpanan elektronik.
2. Struktur logis basis data terdiri dari field, record, tabel dan basis data.
3. Tujuan perancangan basis data adalah Kecepatan dan Kemudahan, Efisiensi Ruang Penyimpanan, Keakuratan, Ketersediaan, Kelengkapan, Keamanan, Kebersamaan.

1.7. Evaluasi

1. Definisikan basis data sesuai dengan pemahaman Anda.
2. Apa saja keuanggulan dalam penerapan basis data ?
3. Jelaskan perbedaan antara field, record dan tabel ?
4. Dalam pengembangan sebuah sistem informasi, basis data mempunyai peran yang penting, jelaskan mengapa demikian?

BAB 2
LINGKUNGAN BASIS DATA

Bab ini menjelaskan arsitektur basis data, bahasa dalam basis data dan model data di basis data.

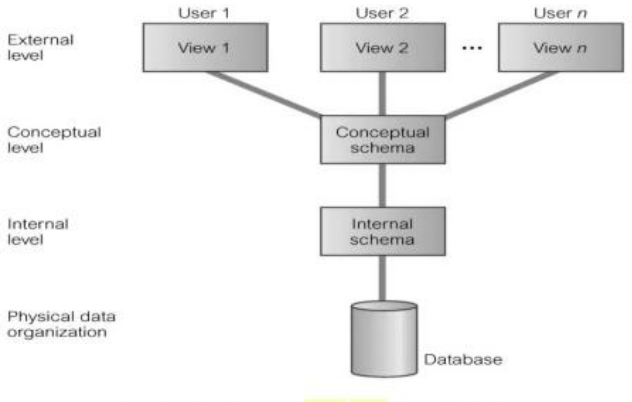
2.1. Arsitektur Basis Data

Arsitektur basis data menyediakan pengguna suatu pandangan abstrak mengenai data, dengan menyembunyikan detil bagaimana data di simpan dan di manipulasi sesuai dengan tujuan utama basis data seperti yang di tunjukkan gambar 2.1. Ada 3 level atau tingkat dalam arsitektur basis data di lihat dari sudut pandang pengguna (user) terhadap basis data yaitu sebagai berikut :

1. Tingkat Eksternal (*view Level*)
   Merupakan level tertinggi dari abstraksi data. pada level ini hanya menunjukkan sebagian saja dari basis data yang bisa dilihat dan di pakai, basis data yang hanya bagi pengguna tertentu saja.
2. Tingkat Logik (*Conceptual Level*)
   Pada Level ini data di gambarkan dengan apa (*what*) yang sebenarnya disimpan dalam basis data dan hubungannya dengan data yang lain yang digambarkan sebagai Semua entitas beserta atribut dan hubungannya, Batasan data, Informasi semantik tentang data, Keamanan dan integritas informsi.
3. Tingkat Fisik (*Internal Level*)
   Merupakan level terendah, yang menunjukkan bagaimana (*how*) data di simpan secara fisik di dalam media penyimpanan sehingga tingkat ini memperhatikan alokasi ruang penyimpanan data dan indeks, Deskripsi record untuk

penyimpanan, Penempatan record, penempatan data dan teknik encription.

Tujuan dari 3 level arsitektur basis data supaya seluruh pengguna bisa mengakses data yang sama dan tidak perlu mengetahui detail tampilan fisik dari basis data. Pengguna tidak akan terpengaruh Apabila ada perubahan dari tampilan basis data. Database Administrator bisa mengubah struktur basis data tanpa mempengaruhi sudut pandang pengguna.Database Administrator bisa mengubah struktur konsep basis data tanpa mempengaruhi semua pengguna database.



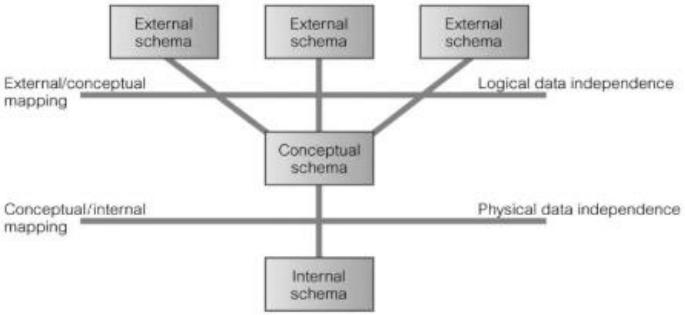
Gambar 2.1. 3 Level Basis Data (Jayanti, 2018)

2.2. Data Independence

Arsitektur basis data bertujuan untuk memudahkan dalam memelihara kemandirian data (*data independence*). Yaitu adanya perubahan dalam satu level tidak memengaruhi level yang lain. Terdapat dua jenis data independence yaitu :

a. Physical Data Independence dimana perubahan level internal tanpa menggunakan skema konseptual atau eksternal basis data.
b. Logical Data Independence dimana perubahan level konseptual tanpa mengganggu skema eksternal.

Prinsip *data independence* di terapkan pada pengelolaan basis data supaya Database Administrator bisa dengan mudah mengubah isi, lokasi, struktur basis data tanpa mengganggu program aplikasi yang sudah ada. software developer bisa memperkenalkan program-program baru tanpa mengganggu program yang sudah ada dan untuk memudahkan dalan perkembangan program aplikasi.



Gambar 2.2. Data Independence dan 3 level arsitektur basis data

2.3. Bahasa Dalam Basis Data

Dalam basis data terdapat tiga bentuk bahasa dalam basis data (*Database Language*) yaitu sebagai berikut :

1. *Data Definition Language* (DDL)
   memperbolehkan pengguna untuk mendeskripsikan struktur basis data, misalnya merinci tipe dan batasan data yang akan

disimpan dalam basis data serta menentukan kunci field dan relasinya.

2. *Data Manipulation Language* (DML)

   Bahasa yang memperbolehkan pengguna untuk mengakses atau memaniulasi data yang ada di basis data. DML mengacu pada kumpulan perintah-perintah yang bisa digunakan untuk melakukan manipulasi data seperti menyimpan data ke dalam tabel, mengubah data, menghapus data dan menampilkan data kembali.

   Terdapat 2 jenis Data Manipulation Language (DML), yaitu :

   a. *Procedural*, pengguna menentukan data apa yang diinginkan serta bagaimana menbisakannya.
   b. *Non Procedural*, membuat pengguna daat menentukan data apa saja yang diinginkan tanpa menyebutkan bagaimana cara menbisakannya.

   Manipulasi dalam pengelolaan basis data antara lain :

   a. Melakukan penyisipan atau penambahan data baru ke basis data.
   b. Penghapusan data dari basis data.
   c. Pengubahan data di basis data.
   d. Mengambil atau menampilkan informasi yang tersimpan di basis data.

3. *Data Control Language* (DCL)

   Menyediakan akses terkontrol ke basis data, misalnya *security system, concurrency control system* dan *recovery control system.* DCL memungkinkan membagi pengguna berdasarkan hak akses ke basis data atau mencabut hak akses pengguna tersebut.

2.4. Model Data Dalam Basis Data

Model data merupakan sekumpulan konsep untuk menerangkan tentang data, hubungan antara data satu dengan data yang lain serta batasan-batasa data. Untuk menggambarkan data pada tingkat eksternal dan konseptual. maka digunakan model data berbasis objek atau record

1. Model Entity Relationship

   Model ini didasarkan pada persepsi data terhadap dunia nyata yag terdiri dari sekumpulan objek yang biasa di sebut dengan entity dan hubungan antar objek yang di sebut relationship.

   Pemodelan data dengan model entity relationship diagram menggunakan diagram entity relationship atau biasa dikenal dengan *Entity Relationship Diagram* (ERD). ERD mempunyai notasi sebagai berikut :

   a. Kotak persegi panjang untuk menggambarkan himpunan enity.
   b. Ellips untuk menggambarkan atribut.
   c. Belah ketupat untuk menggambarkan hubungan antara himpunan entity.
   d. Garis untuk menghubungkan antar entitas dalam Entity Relationship Diagram.

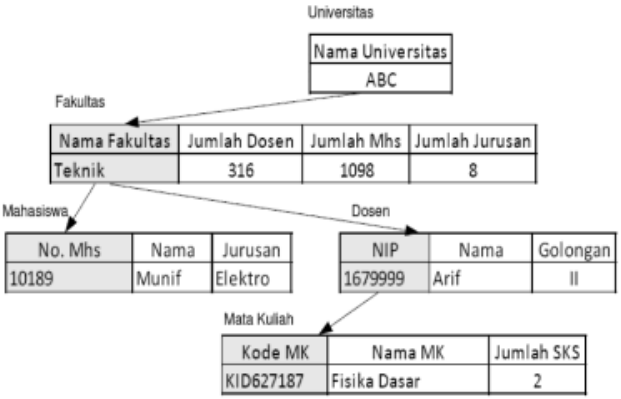
2. Model Relational

   Model ini menggambarkan data dalam bentuk tabel-tabel. Asosiasi antar tabel didefinisikan lewat kunci tamu atau disebut *Foreign Key* (FK). Dengan menggunakan model ini, pencarian field dari tabel-tabel yang ada di basis data bisa dilakukan dengan cepat dan mudah. Pencarian atribut yang berhubungan pada tabel yang berbeda bisa dilakukan dengan menghubungkan terlebih dahulu tabel-tabel tersebut dengan menggunakan atribut yang sama.

3. Model Hirarki

   Model ini menyerupai pohon yang dibalik. Model Hirarki menggunakan pola hubungan orang tua dan anak. Pada puncak hierarki disebut dengan akar (*root*). Tiap Entitas tingkat atas (*parent*) mempunyai satu atau lebih subentitas (*children*), sehingga tiap-tiap entitas hanya boleh mempunyai satu induk, tetapi bisa mempunyai banyak anak. Hubungan antar entitas pada model ini dinyatakan dalam satu - banyak (*one to many*) atau satu - satu (*one to one*).

   Misalkan pada perguruan tinggi mempunyai banyak fakultas, tiap-tiap fakultas mempunyai banyak dosen, mahasiswa, jurusan dan lain sebagainya. Tanda panah pada diagram menunjukkan derajat hubungan satu ke banyak yang di tunjukkan pada gambar 2.3.



Gambar 2.3. Model Hierarki

2.5. Ringkasan

1. Arsitektur basis data menyediakan pengguna suatu pandangan abstrak mengenai data, dengan menyembunyikan detil

bagaimana data di simpan dan di manipulasi sesuai dengan tujuan utama basis data.

2. Arsitektur basis data terdiri dari Tingkat Eksternal (view Level), Tingkat Logik (Conceptual Level) dan Tingkat Fisik (Internal Level).
3. Bahasa basis data terdiri dari Data Definition Language (DDL), Data Manipulation Language (DML) dan Data Control Language (DCL).
4. Dalam basis data terdapat beberapa model antara lain Model Entity Relationship, Model Relational dan Model Hirarki.

2.6. Evaluasi

1. Sebutkan dan jelaskan bahasa yang ada pada basis data.
2. Pada basis data terdapat beberapa model data diantaranya adalah Model Entity Relationship, Model Relational dan Model Hirarki pada basis data, Jelaskan perbedaan antara model data tersebut.

BAB 3
DATABASE MANAGEMENT SYSTEM (DBMS)

Pada bab ini akan di jelaskan tentang Sejarah DBMS, Definisi dan Fungsi DBMS, Komponen DBMS, Arsitektur DBMS Multi User pada basis data.

3.1. Sejarah DBMS

Database Management System (DBMS) pertama kali diciptakan oleh Charles Bachman dari perusahaan General Electric tahun 1960 yang disebut dengan penyimpanan Data Terintegrasi (integrated Data Store). Dibuat untuk model data jaringan yang kemudian di standarisasikan oleh Conference on Data System Language (CODASYL). Tahun 1960 akhir IBM megembangkan sistem manajemen informasi (Information Management System) DBMS. IMS di bentuk dari representasi data model data hierarki.

Pada tahun 1970, Edgar COdd yang bekerja di laboratorium penelitian di San Jose mengusulkan model data relational. Tahun 1980, model data relational menjadi paradigma DBMS yang paling doninan. Kemudian mulai dikembangkan bahasa Query Structure Query Language (SQL) sebagai bagian dari proye R dari IBM.Tahun 1980 SQL mulai di standarisasikan oleh American Nasional Standart Institute (ANSI) dan International Standarts Organizatoion (ISO) melahirkan SQL-92 yaitu sebuah program yang digunakan untuk mengeksekusi secara bersam-sama dalam basis data yang dikenal dengan transaksi.

Saat ini DBMS bisa digunakan untuk menyimpan data yang bisa diakses melali web browser. Query bisa di generrate memlalui formulir web dan format jawabannya menggunakan bahasa HTML untuk memudahkan tampilan data pada browser. Basis data

sekarang sduah bisa di akses melalui jaringan komputer dan internet.Contohnya basis data multimedia, video interaktif, perpustakaan digital dan lain-lain.

3.2. Definisi dan Fungsi DBMS

Sistem manajemen basis data merupakan sistem yang terkomputerisasi dengan tujuan utamanya yaitu memelihara informasi serta membuat informasi tersebut tersedia dan bisa di akses saat dibutuhkan. Sistem basis data merupakan sekumpulan basis data dalam suatu sistem yang mungkin tidak ada hubungan satu sama lain, tetapi secara keseluruhan mempunyai hubungan sebagai sebuah sistem dengan didukung oleh komponen lainnya.

Sistem basis data bisa juga di definisikan sebagai sekumpulan subsistem yang terdiri atas basis data dengan para pemakai yang menggunakan basis data secara bersama-sama, personal-personal yang merancang dan mengelola basis data, teknik-teknik untuk merancang dan mengelola basis data, serta sistem komputer mendukungnya.

Dari pengertian tersebut bisa di simpulkan bahwa sistem basis data mempunyai kriteria sebagai berikut :

1. Basis data sebagai inti dari sistem basis data.
2. Perangkat lunak (*Software*) untuk perancangan dan pengelolaan basis data.
3. Perangkat keras (*Hardware*) sebagai pendukung operasi pengelolaan data.
4. Manusia (*Brainware*) yang mempunyai peran penting dalam sistem tersebut, yaitu sebagai pemakai atau para spesialis informasi yang mempunyai fungsi sebagai perancang atau pengelola.

Fungsi Database Management System (DBMS) adalah sebagai berikut :

1. Penyimpanan, pengambilan dan perubahan data (*Data storage, retrieval, and update*)

   Sebuah DBMS menfasilitasi pengguna untuk menyimpan, memperoleh, dan mengubah data di dalam basis data. Hal tersebut merupakan fungsi dasar dari sebuah DBMS.

2. Katalog yang bisa diakses pemakai (*A user-accessible catalog*)

   Sebuah DBMS menyediakan katalog yang menyimpan item-item data yang disimpan dan bisa diakses oleh pengguna. Fitur utama dari arsitektur ANSI-SPARC adalah system catalog terintegrasi yang berisi data mengenai schema, pengguna (user), aplikasi, dimana katalog bisa diakses oleh pengguna sama seperti DBMS. System catalog merupakan salah satu komponen fundamental dari sistem. Didalam DBMS, system catalog menyimpan:

   a. Nama, tipe, dan ukuran dari item data;
   b. Nama relationship;
   c. Integrity constraint dari data;
   d. Nama pengguna yang memiliki otorisasi dalam mengakses data;
   e. Item data yang bisa diakses oleh tiap pengguna serta tipe akses yang diijinkan, misalnya akses untuk memasukkan, mengubah, menghapus, dan membaca data;
   f. Skema eksternal, konseptual, dan internal serta pemetaan antar skema;
   g. Statistik penggunaan, seperti frekuensi transaksi dan jumlah pengaksesan objek didalam basis data.

3. Mendukung Transaksi (*Transaction Support*)

Sebuah DBMS menyediakan mekanisme untuk memastikan apakah sebuah transaksi berhasil dijalankan secara utuh atau tidak dijalankan sama sekali, serta memastikan bahwa basis data selalu berada di consistent state. Sebuah transaksi bisa mengakses atau mengubah isi dari basis data.

4. Melayani kontrol concurrency (*Concurrency control services*)
   Sebuah DBMS harus menyediakan mekanisme untuk memastikan basis data dilakukan update secara benar saat beberapa pengguna mengubah basis data secara bersamaan karena DBMS memungkinkan beberapa pengguna untuk mengakses shared data secara bersamaan.
5. Melayani recovery (*Recovery services*)
   Sebuah DBMS harus menyediakan mekanisme recovery basis data Apabila terjadi kerusakan. Seperti yang sudah disebutkan pada poin keempat yakni Apabila sebuah transaksi gagal maka basis data harus dikembalikan ke consistent state. Kegagalan sebuah transaksi bisa berupa system crash, media failure, error yang terjadi pada perangkat keras atau perangkat lunak yang menyebabkan transaksi dibatalkan.
6. Melayani autorisasi (*Authorization services*)
   Sebuah DBMS menyediakan mekanisme agar hanya pengguna terotorisasi yang bisa mengakses basis data. Hal tersebut berkaitan dengan keamanan basis data dari pengguna yang tidak memiliki otorisasi mengakses basis data, baik yang disengaja maupun tidak.
7. Mendukung komunikasi data (*Support for data communication*)
   Sebuah DBMS memiliki kemampuan berintegrasi dengan perangkat lunak untuk komunikasi, karena pengguna bisa mengakses basis data secara langsung atau melalui jaringan.
8. Melayani integrity (*Integrity services*)

Sebuah DBMS memastikan baik data didalam basis data maupun pengubahan data selalu memenuhi aturan. Integritas basis data berkaitan dengan kebenaran dan konsistensi dari data yang disimpan, dimana berkaitan dengan constraint yang merupakan aturan didalam basis data yang tidak bisa dilanggar.

9. Melayani data independence (*Services to promote data independence*)

   Sebuah basis data memiliki fasilitas untuk mendukung independensi dari program terhadap struktur actual dari basis data.

10. Melayani utility (*Utility services*)

    Sebuah DBMS memiliki beberapa utility service untuk membantu administrator basis data dalam melakukan manajemen basis data secara efektif. Misalnya: fasilitas import data, monitoring data, analisis statistik (ubinus, 2018).

3.3. Komponen DBMS

Database Management System (DBMS) mempunyai komponen-komponen penyusun antara lain sebagai berikut :

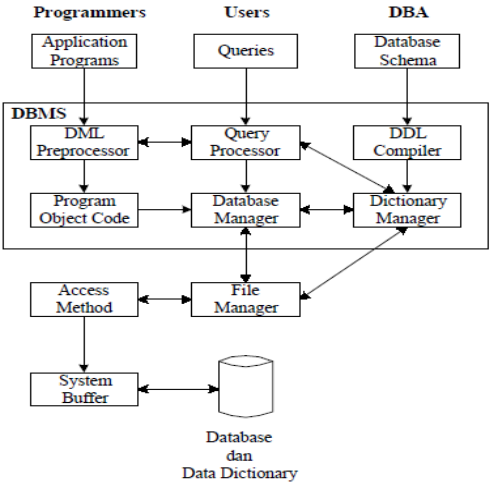
1. Query Processsor
2. Komponen yang merubah bentuk query ke dalam instruksi tingkat rendah ke database manager
3. Database Manager

   Database manager menerima query dan menguji skema eksternal dan konseptual untuk menentukan apakah record-record dibutuhkan untuk memenuhi permintaan. Kemudian DM memanggil file manager untuk menyelesaikan permintaan

4. File Manager

   Memanipulasi penyimpanan file dan mengatur alokasi ruang penyimpanan pada disk.

5. DML Preprocessor
   Modul yang merubah perintah DML embedded ke dalam program aplikasi dalam bentuk fungsi-fungsi yang memanggil dalam host language.
6. DDL Compiler
   Merubah perintah DDL menjadi kumpulan tabel yang berisi metadata.
7. Dictionary Manager
   Mengatur akses dan memelihara data dictionary. Data dictionary diakses oleh komponen DBMS yang lain.



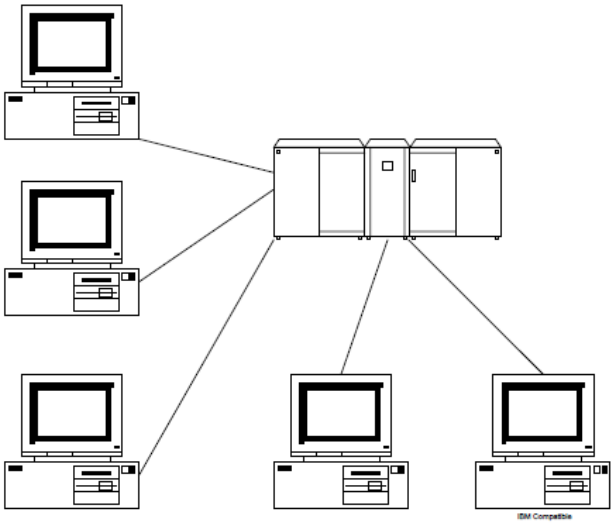
Gambar 3.1. Komponen DBMS

3.4. Arsitektur DBMS Multi User

1. Teleprocessing
   Arsitektur tradisional untuk sistem multi user adalah teleprocessing, dimana satu komputer dengan sebuah CPU dan

sejumlah terminal seperti pada gambar 3.2. Semua pemrosesan dikerjakan dalam batasan fisik komputer yang sama. Terminal untuk pemakai berjenis 'dumb', yang tidak bisa berfungsi sendiri dan masing-masing dihubungkan ke komputer pusat. Terminal-terminal tersebut mengirimkan pesan melalui subsistem pengontrol komunikasi pada sistem operasi ke program aplikasi, yang bergantian menggunakan layanan DBMS.

Dengan cara yang sama, pesan dikembalikan ke terminal pemakai. Arsitektur ini menempatkan beban yang besar pada komputer pusat yang tidak hanya menjalankan program aplikasi tetapi juga harus menyelesaikan sejumlah pekerjaan pada terminal seperti format data untuk tampilan di monitor.

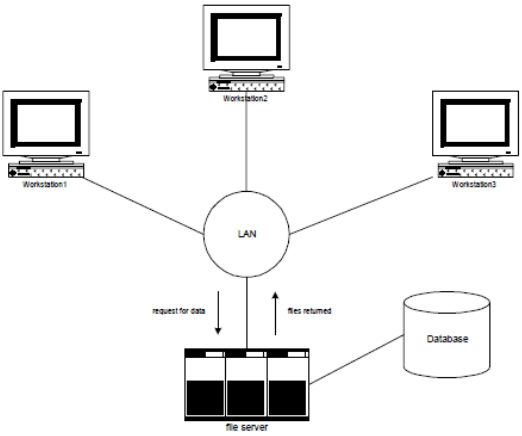


Gambar 3.3. Arsitektur Tellecprocessing

2. File-Server

Proses didistribusikan ke dalam jaringan sejenis LAN (*Local Area Network*). File server mengendalikan file yang diperlukan

oleh aplikasi dan DBMS. Meskipun aplikasi dan DBMS dijalankan pada masing-masing workstation tetapi tetap meminta file dari file server Apabila diperlukan seperti di tunjukkan gambar 3.4.



Gambar 3.4. Arsitektur File Server

Dengan cara ini, file server berfungsi sebagai sebuah hard disk yang digunakan secara bersamaan.

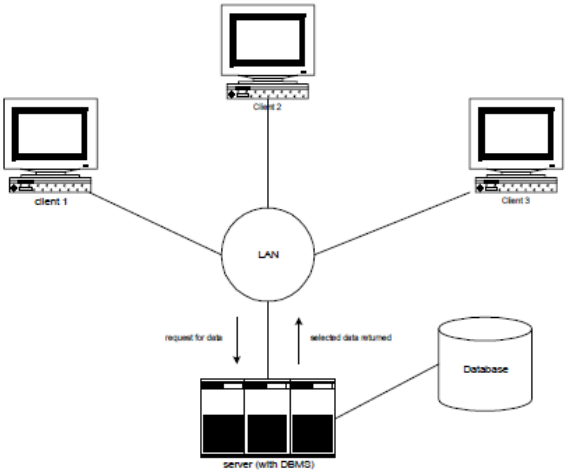
Kerugian arsitektur file-server adalah :

a. Terdapat lalulintas jaringan yang besar
b. Masing-masing workstation membutuhkan copy DBMS
c. Kontrol terhadap concurrency, recovery dan integrity menjadi lebih kompleks karena sejumlah DBMS mengakses file secara bersamaan

3. Client Server

Untuk mengatasi kelemahan arsitektur-arsitektur di atas maka dikembangkan arsitektur client-server. Client-server menunjukkan cara komponen software berinteraksi dalam

bentuk sistem. Sesuai dengan namanya, ada sebuah pemroses client yang membutuhkan sumber dan sebuah server yang menyediakan sumbernya. Tidak ada kebutuhan client dan server yang harus diletakkan pada mesin yang sama. Secara ringkas, umumnya server diletakkan pada satu sisi dalam LAN dan client pada sisi yang lain.



Gambar 3.5. Arsitektur Client Server

Dalam konteks basis data, client mengatur interface berfungsi sebagai workstation tempat menjalankan aplikasi basis data. Client menerima permintaan pemakai, memeriksa sintaks dan generate kebutuhan basis data dalam SQL atau bahasa yang lain. Kemudian meneruskan pesan ke server, menunggu response dan bentuk response untuk pemakai akhir. Server menerima dan memproses permintaan basis data kemudian mengembalikan hasil ke client.

4. Data Dictionary

Data dictionary adalah tempat penyimpanan informasi yang menggambarkan data dalam basis data. Data dictionary biasa disebut juga dengan metadata atau data mengenai data. Modul pengontrol otorisasi menggunakan data dictionary untuk memeriksa apakah seorang pemakai perlu mempunyai wewenang.

Untuk mengerjakan pemeriksaan tersebut data dictionary menyimpan :

a. nama-nama pemakai yang mempunyai wewenang untuk menggunakan DBMS
b. nama-nama data item yang ada dalam basis data
c. data item yang bisa diakses oleh pemakai dan jenis akses yang diijinkan, misalnya: insert, update, delete atau read

Sedangkan untuk memeriksa integritas data, data dictionary menyimpan :

a. nama-nama data item dalam basis data
b. jenis dan ukuran data item
c. batasan untuk masing-masing data item

Sistem data dictionary bisa dibedakan atas sistem aktif dan pasif. Sistem aktif selalu konsisten dengan struktur basis data karena secara otomatis dikerjakan oleh sistem. Sebaliknya, sistem pasif tidak konsisten terhadap perubahan basis data yang dilakukan oleh pemakai.

3.5. Ringkasan

1. Sistem manajemen basis data merupakan sistem yang terkomputerisasi dengan tujuan utamanya yaitu memelihara

informasi serta membuat informasi tersebut tersedia dan bisa di akses saat dibutuhkan.

2. Kriteria sistem basis data terdiri dari Basis data sebagai inti dari sistem basis data, Perangkat lunak (*Software*), Perangkat keras (*Hardware*), Manusia (*Brainware*).
3. Fungsi Database Management System (DBMS) adalah Penyimpanan, pengambilan dan perubahan data (*Data storage, retrieval, and update*), Katalog yang bisa diakses pemakai (*A user-accessible catalog*), Mendukung Transaksi (*Transaction Support*), Melayani kontrol concurrency (*Concurrency control services*), Melayani recovery (*Recovery services*), Melayani autorisasi (*Authorization services*), Mendukung komunikasi data (*Support for data communication*), Melayani integrity (*Integrity services*), Melayani data independence (*Services to promote data independence*), Melayani utility (*Utility services*).
4. Arsitektur DBMS Multi User terdiri atas Teleprocessing, File-Server, Client Server, Data Dictionary.

3.6. Evaluasi

1. Apa yang Ada ketahui tentang Database Management System?
2. Sebutkan serta jelaskan kriteria yang ada pada DBMS beserta contohnya.

BAB 4
ER MODEL DAN RDBM

Pada bab ini akan di jelaskan tentang konsep dasar entity relationship, Komponen-komponen Entity Relationship, Menggambar entity relationship, Kelebihan dan kekurangan entity relationship, konsep dasar relational database, karakteristik relasi, komponen relasi, kunci relasi.

4.1. Konsep Dasar ER-Model

ER Model atau biasa disebut *Entity Relationship Model* merupakan model untuk menjelaskan hubungan antara data satu dengan data yang lain di dalam basis data berdasarkan pandangan bahwa dunia nyata terdiri dari objek-objek dasar yang mempunyai hubungan (relasi) antara objek-objek tersebut. Model data entity relationship dikembangkan berdasarkan obyek-obyek. Entity Relationship model di gambarkan dalam bentuk diagram yang di sebut *diagram entity relationship* (ER_Diagram). Untuk menggambarkan sebuah ER_Diagram di gunakan simbol-simbol tertentu.

Penggunaan ER_Model relatif mudah dipahami, bahkan oleh pengguna yang belum mengenal konsep basis data. Bagi analis sistem, ER_Diagram berguna untuk memodelkan sistem-sistem yang basis datanya akan dikembangkan. Model ini juga membantu analis sistem untuk bisa merancang sistem saat melakukan analisis sistem dan perancangan basis data karena model ini bisa menunjukkan macam-macam data yang dibutuhkan oleh pengguna dan kerelasian antara data satu dengan data yang lain di dalam basis data. Bagi pengguna, model ini sangat membantu dalam memahami

sistem dan rancangan basis data yang akan dikembangkan oleh analis sistem (Sutanta, 2011).

4.2. Komponen-komponen Entity Relationship

Diagram Entity Relationship (ER_Diagram) terdiri atas tiga komponen penyusun antara lain entitas, atribut dan kerelasian antar entitas. Entitas merupakan objek dasar yang terlibat dalam suatu sistem. Atribut berperan sebagai penjelas entitas yang ada, sedangkan kerelasian menunjukkan hubungan yang terjadi di antara dua entitas atau lebih (Silberschatz, dkk, 2011).

4.2.1. Entitas (Entity) di Basis Data

Entitas merupakan obyek-obyek dasar yang saling terkait di dalam sistem, obyek dasar bisa berupa orang, benda, atau hal yang keterangannya perlu disimpan di dalam sebuah basis data. Atau bisa juga di artikan individu yang mewakili sesuatu yang nyata dan bisa dibedakan dari individu yang lain. Menurut Suntanta (2011) untuk menggambarkan entitas-enitas digunakan aturan sebagai berikut :

1. Entitas digambarkan dengan simbol persegi panjang.
2. Penamaan entitas dituliskan di dalam simbol persegi panjang.
3. Penamaan entitas berupa kata benda tunggal.
4. Penamaan entitas sebisa mungkin menggunakan nama yang mudah dimengerti dan bisa menyatakan maknanya dengan jelas.

Sering kali penamaan entitas bisa tersusun atas lebih dari satu kata. Untuk memenuhi aturan-aturan penggambaran tersebut maka sering digunakan tanda penghubung garis bawah atau under score (_), menyatakan bahwa kata-kata tersebut dianggap sebagai kata tunggal. Contohnya sebagai berikut :

1. Untuk menyatakan entitas program studi bisa menggunakan **Program_Studi** bukan **Program** atau **Studi** karena lebih mudah dipahaminya dan memungkinkan penggunaan Program atau Studi akan mempunyai arti berbeda dari sudut pandang pengguna lain.
2. Untuk menyatakan nama entitas mata kuliah bisa menggunakan **Mata_kuliah** bukan **Mata** atau **Kuliah** karena pengguna lebih mudah memahami serta penggunaan nama Mata atau Kuliah mempunyai arti berbeda dari sudut pandang pengguna lain.

Penamaan entitas-entitas yang ada di basis data haruslah singkat dan jelas karena apabila terlalu panjang maka akan menyulitkan terutama dalam penerapan ke program atau sistem dan pengguna lebih memahami nama yang lebih singkat, misalkan :

1. Apabila ingin menyatakan nama entitas orang tua atau wali mahasiswa lebih baik menggunakan nama Wali_mhs bukan OrangTua_Wali_mhs karena cukup jelas di pahami.
2. Dan apabila ingin menyatakan nama entitas provinsi atau Dati II asal mahasiswa lebih baik dinamakan Provinsi_Mahasiswa karena sudah jelas.

Penamaan sebuah entitas bisa disingkat akan tapi tidak dianjurkan dikarenakan akan mempersulit pengguna dalam memahami maksud dari nama entitas tersebut seperti :

1. Untuk menyatakan nama entitas program studi dipakai nama **prodi** saja, singkatan ini sering dipakai sehingga apabila dipakai pengguna maka masih bisa memahami maksudnya.

2. Untuk menyatakan nama entitas mahasiswa dipakai nama **Mhs**, singkatan ini sering digunakan sehingga apabila dipakai maka pengguna masih bisa memahami maksudnya.

Penentuan entitas-entitas di dalam suatu sistem perlu dilakukan dengan hati-hati dan cermat. Tidak semua orang, benda, atau suatu hal bisa disebut entitas. Hanya orang, benda dan hal yang berkaitan dengan sistem yang akan di bangun yang perlu di simpan dalam basis data yang bisa disebut entitas seperti contoh pada tabel 4.1, tabel 4.2, tabel 4.3 dan tabel 4.4.

Tabel 4.1. Contoh entitas berupa orang

Objek Dasar	Simbol Entitas
Mahasiswa	Mahasiswa
Dosen	Dosen
Wali Mahasiswa	Wali_Mahasiswa

Tabel 4.2. Contoh entitas berupa Benda

Objek Dasar	Simbol Entitas
Gedung	Gedung
Ruang	Ruang

Tabel 4.3. Contoh entitas berupa Hal

Objek Dasar	Simbol Entitas

Program Studi	Program_Studi
Mata Kuliah	Mata_Kuliah
Jabatan	Jabatan

Entitas-entitas yang ada di basis data dibagi menjadi beberapa istilah antara lain sebagai berikut :

1. Isian Entitas

   Isian entitas merupakan data-data yang dimasukkan (diinputkan) ke dalam entitas. Contoh isian entitas adalah :

   a. Mahasiswa dengan NIM 201080200001
   b. Mahasiswa dengan nama Bunga
   c. Program studi dengan nama Informatika
   d. Fakultas dengan nama Teknik

2. Entitas Reguler

   Entitas reguler merupakan entitas-entitas dominan karena entitas ini tidak bergantung pada entitas yang lain. Contoh entitas reguler sebagai berikut :

   a. Mahasiswa
   b. Program_Studi
   c. Mata_Kuliah
   d. Dosen

3. Entitas Dependen

   Entitas dependen juga bisa disebut entitas tidak bebas atau entitas lemah (*Weak Entity*) / entitas subordinat. Entitas ini bergantung pada entitas yang lain. Entitas ini bisa ada apabila

ada entitas lain sebagai acuannya dan biasanya yang jadi acuan adalah entitas reguler. Contoh entitas ini adalah :
Entitas Wali_Mahasiswa bergantung pada entitas Mahasiswa.
Untuk menggambarkan entitas dependen digunakan simbol persegi panjang dengan garis ganda seperti contoh entitas Wali_Mahasiswa dalam Tabel 4.1.

4.2.2. Atribut (Attribute) Pada Entitas

Tiap-tiap entitas yang ada di basis data mempunyai atribut. Atribut merupakan ciri-ciri yang membedakan antara entitas satu dengan entitas yang lainnya. Atribut juga sering disebut sebagai properti. Atribut merupakan keterangan-keterangan yang terkait dengan entitas yang perlu disimpan dalam basis data. Atribut berfungsi sebagai penjelas sebuah entitas. Untuk menggambarkan sebuah atribut digunakan aturan-aturan sebagai berikut :

1. Atribut di gambarkan dalam sebuah simbol elips.
2. Penamaan atribut di tulis di dalam simbol elips.
3. Penamaan atribut berupa kata benda tunggal.
4. Penamaan atribut sebisa mungkin memakai nama yang mudah dipahami dan bisa menyatakan maknanya dengan jelas.
5. Atribut-atribut dihubungkan dengan entitas yang bersesuaian dengan menggunakan sebuah garis di basis data.

Penamaan atribut-atribut haruslah jelas dan mudah dipahami oleh pengguna. Nama-nama yang dipakai juga harus menunjukkan makna yang di maksudkan. Apabila memerlukan tanda pemisah pakailah tanda penghubung garis bawah atau underscore (_). Penyingkatan nama atribut juga diperbolehkan asal mudah dipahami pengguna dan jelas artinya. Contoh atribut-atribut untuk

entitas dalam subsistem pengelolaan data-data akademik bisa dilihat pada tabel 4.4 berikut ini :

Tabel 4.4. Contoh Atribut pada Entitas

Simbol Entias	Atribut
Mahasiswa	Nim,nama_mahasiswa,kode_prodi,kelas,thn_masuk,kelamin,tanggal_lahir,tempat_lahir,alamat,status
Program_Studi	Kode_prodi,nama_prodi,jenjang,status,kode_fakultas
Fakultas	Kode_fakultas, nama_fakultas, status, nama_dekan
Mata_kuliah	Kode_mk, nama_mk, sks, semester, status, kode_prodi
Dosen	Nik, nama_dosen, kelamin, tanggal_lahir, tempat_lahir, alamat, status, no_telpon

Menurut Sutanta (2011) atribut pada sebuah entitas bisa di klasifikasikan dalam dua kelompok, yaitu :

1. Atribut Sederhana atau *Simple Attribute*, yaitu apabila atribut berisikan hanya satu komponen nilai. Contoh atribut sederhana dengan nilainya dalam entitas mahasiswa sebagai berikut :
   a. Nim : 201080200001
   b. Kode_prodi : 10802 (Informatika)
   c. Status : A (Aktif)
2. Atribut Komposit atau *Composite attribute*, yaitu apabila suatu atribut berisikan lebih dari satu komponen nilai. Contoh atribut komposit dengan nilainya sebagai berikut :
   a. Nama_mahasiswa : Bunga Sari Rejeki
      Terdiri atas beberapa komponen nilai, yaitu :
      Nama depan : Bunga

Nama tengah : Sari
Nama akhir : Rejeki

b. Tanggal_lahir : 01-01-2000
Terdiri atas beberapa komponen nilai, yaitu :
Tanggal : 01
Bulan : 01
Tahun : 2000

Atribut nama bisa dianggap atribut sederhana atau komposit tergantung pada nilai datanya. Apabila terdiri dari hanya satu kata maka termasuk atribut sederhana. Akan tetapi apabila terdiri lebih dari satu kata maka atribut tersebut adalah atribut komposit.

3. Atribut bernilai tunggal atau *Single-Valued Attribute*
adalah atribut yang memiliki paling banyak hanya satu nilai untuk tiap-tiap baris data. Contoh atribut bernilai tunggal adalah sebagai berikut :
   a. Nim : 201080200001
   b. Kelas : A1
4. Atribut bernilai banyak atau *Multi-Valued Attribute*
adalah atribut yang bisa berisi lebih dari satu nilai tetapi dengan jenis data yang sama. Contoh atribut bernilai banyak adalah sebagai berikut :
   a. Publikasi : Jurnal terakreditasi, proseding
5. Atribut turunan atau *Derived Attribute*
merupakan atribut yang nilai-nilainya diturunkan dari atribut-atribut yang lain. Contoh atribut turunan adalah sebagai berikut :
   a. Atribut Usia diturunkan dari atribut tanggal lahir
   b. Atribut lama kerja diturunkan dari atribut tanggal mulai kerja

4.2.3. Atribut-atribut Kunci

Pada tiap-tiap entitas selalu terdapat atribut kunci yang berupa satu atribut yang bisa mewakili sebuah record. Misalnya Nomor Induk Mahasiswa (NIM) merupakan atribut kunci dari entitas Mahasiswa. Apabila melakukan pencarian data-data mahasiswa cukup hanya dengan menyebutkan NIM tersebut, maka mahasiswa akan bisa diketahui nama, kelamin, program studi, alamat, tanggal lahir, tempat lahir dan lain-lain. Atribut-atribut kunci ada beberapa Jenis yaitu sebagai berikut :

1. *Candidate Key* atau Kunci Kandidat

   Kunci kandidat merupakan satu atribut yang menjeaskan secara unik satu kejadian dari suatu entitas. Atau atribut-atribut yang bisa dipilih menjadi sebuah *primary key*. Apabila satu kunci kandidat berisikan lebih dari satu atribut maka disebut sebagai composite key atau kunci gabungan.

   Contohnya :

   Entitas mahasiswa memiliki atribut nim, nama mahasiswa, kode prodi,tanggal lahir, tempat lahir, alamat,no ktp. Maka atribut yang bisa menjadi kunci kandidat adalah :

   a. Atribut Nim karena unik tidak memungkinkan ada data ganda.
   b. Atribut No KTP karena unik tidak memungkinkan ada data ganda.
   c. Atribut Nama mahasiswa sering digunaka sebagai kunci pencarian namun tidak bisa dikatakan kunci karena seseorang bisa mempunyai nama yang sama.
   d. Atribut Nama mahasiswa dan tanggal lahir, mungkin bisa digunakan sebagai kunci karena sangat kecil seseorang yang mempunyai nama dan tanggal lahir sama.

e. Atribut Nama mahasiswa + tanggal lahir + tempat lahir bisa digunakan sebagai kunci.
f. Atribut Alamat tidak bisa digunakan sebagai kunci.

2. *Primary Key* atau Kunci Utama

   Primary key merupakan satu atribut yang tidak hanya menjelaskan secara unik suatu kejadian tapi juga bisa mewakili tiap-tiap kejadian dari entitas. Tiap-tiap atribut yang menjadi kunci kandidat akan berpeluang menjadi primary key akan tetapi sebaiknya dipilih satu saja yang bisa mewakili secara menyeluruh pada entitas yang ada. Sebagai petunjuk, berikut ini adalah ciri-ciri dari atribut-atribut yang bisa dipertimbangkan sebagai identifier atau primary key yaitu :

   a. Atribut yang mempunyai nilai tidak berubah-ubah
   b. Tidak berisikan nilai kosong (*null*)
   c. Tidak berisikan data nama atau lokasi yang mungkin sering berubah-ubah.

   Contoh :

   Nim karena unik dan tidak memungkinkan ada data ganda serta mewakili dengan menyeluruh semua entitas mahasiswa, tiap-tiap mahasiswa pasti memiliki sebuah nim.

3. *Alternate Key* atau Kunci Alternatif

   Alternate key merupakan kunci kandidat yang tidak digunakan sebagai *primary key*. Sering kali kunci alternatif digunakan sebagai kunci pengurutan dalam suatu laporan.

4. *Foreign Key* atau Kunci Tamu

   Foreign key merupakan satu atribut yang melengkapi satu hubungan (*relationship*) yang menuju ke induknya. Kunci tamu ditempatkan pada entitas anak serta mempunyai kesamaan dengan kunci primary key direlasikan. Atribut-atribut yang bukan key, akan tetapi adalah key pada entitas yang lain. atribut

yang digunakan untuk merelasikan dari entitas primer (utama atau master) ke entitas yang lain (sekunder).

Contoh : Kode prodi merupakan foreign key pada entitas mahasiswa tapi pada entitas program studi, kode prodi merupakan primary key.

4.2.4. Kerelasian Antara Entitas di Basis Data (*Relationship*)

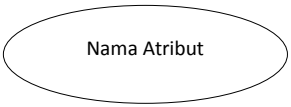
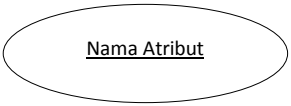
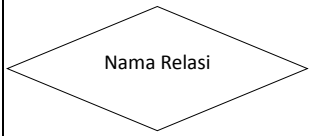
Kerelasian antara entitas satu dengan entitas yang lain menjelaskan hubungan antara dua entitas. Kerelasian merupakan suatu kejadian yang terjadi di antara dua buah entitas yang nilainya perlu disimpan dalam sebuah basis data. Kejadian yang tidak memerlukan penyimpanan dalam basis data (meskipun betul-betul ada) bukan termasuk kerelasian.

Aturan-aturan penggambaran kerelasian antara entitas di basis data adalah sebagai berikut :

1. Suatu Kerelasian digambarkan dengan simbol belah ketupat.
2. Nama kerelasian ditulis di dalam simbol belah ketupat.
3. Kerelasian berfungsi untuk menghubungkan dua entitas.
4. Nama kerelasian menggunakan kata kerja aktif tunggal yang diawali dengan awalan me.
5. Penamaan kerelasian sebisa mungkin menggunakan nama yang mudah dimengerti dan bisa menyatakan maknanya dengan sangat jelas.

Tabel 4.5. Simbol-simbol ER Diagram

Simbol	Deskripsi
Entitas/Entity Nama Entitas	Entitas merupakan data inti yang akan disimpan; bakal tabel pada basis data

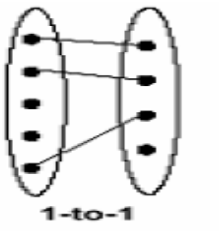
Atribut 	Field atau kolom data yang butuh disimpan dalam suatu Entitas
Atribut Primary Key 	Field atau kolom data yang butuh disimpan dalam suatu Entitas dan digunakan sebagai kunci akses record yang diinginkan; biasanya berupa id
Relasi 	Relasi yang menghubungkan antar entitas. Nama kerelasian berupa kata kerja aktif (diawali dengan awalan me-), tunggal.
Asosiasi 1 n	Penghubung antara relasi dan entitas dimana ke dua ujungnya memiliki multiplicity kemungkinan jumlah pemakaian

A. Jenis Kerelatian Antara Entitas (*Relationship*) di Basis Data

Kerelasian ada beberapa jenis yang dikelompokkan sebagai berikut yaitu :

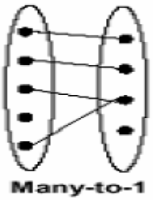
1. Kerelasian jenis 1-ke-1 atau satu ke satu atau *one to one*

   Kerelasian ini terjadi apabila kejadian pada dua entitas yang berhubungan hanya dimungkinkan terjadi satu kali kejadian pada kedua entitas yang terlibat. Apabila nilai yang dipakai sebagai penghubung pada entitas kesatu hanya dimungkinkan muncul satu kali pada entitas kedua yang saling berhubungan. Misalnya **tiap-tiap** mahasiswa hanya mempunyai **seorang** wali mahasiswa.



Gambar 4.1 Relasi satu ke satu

2. Kerelasian jenis n-ke-1 atau banyak ke satu atau *many to one*
   Kerelasian ini terjadi apabila kejadian di antara dua entitas yang berhubungan hanya dimungkinkan terjadi satu kali dalam entitas kesatu dan bisa terjadi lebih dari satu kali kejadian pada entitas yang kedua. Apabila nilai yang dipakai sebagai penghubung pada entitas kesatu dimungkinkan muncul lebih dari satu kali pada entitas kedua yang saling berhubungan.
   Misalkan **lebih dari satu** mahasiswa bisa memilih hanya **satu** program studi (n-ke-1) tetapi sebaliknya **satu** program studi bisa dipilih oleh **lebih dari satu** mahasiswa (1-ke-n).



Gambar 4.2 Relasi banyak ke satu

3. Kerelasian jenis n-ke-n atau banyak ke banyak atau *many to many*
   Kerelasian jenis ini terjadi ketika terdapat kejadian di antara dua entitas yang berhubungan dimungkinkan terjadi lebih dari satu kali dalam entitas kesatu dan entitas kedua.

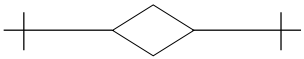
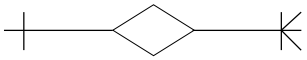
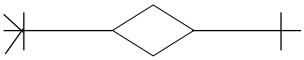
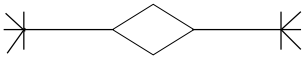
Misalkan **lebih dari satu** mahasiswa bisa memilih **lebih dari satu** mata kuliah (n-ke-n).

Gambar 4.3 Relasi banyak ke banyak

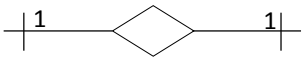
B. Simbol Kerelasian Antara Entitas di Basis Data

Seperti yang dijelaskan sebelumnya, ER_Model ditunjukkan dengan menggunakan sebuah diagram yang biasa disebut Diagram ER (ER_Diagram). Untuk menggambarkan kerelasian antara entitas satu dengan yang lain bisa memakai salah satu dari pilihan simbol berikut :

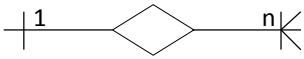
1. Pilihan pertama

Jenis Kerelasian	Simbol yang digunakan
1-ke-1	
1-ke-n	
n-ke-1	
n-ke-n	

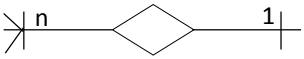
2. Pilihan kedua

Jenis Kerelasian	Simbol yang digunakan
1-ke-1	

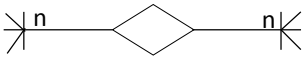
1-ke-n



n-ke-1



n-ke-n



Saat menggambar sebuah ER_Diagram bisa digunakan salah satu dari beberapa pilihan diatas sesuai dengan keinginan perancang, yang penting pemakaian simbol di lakukan dengan konsisten (tetap tidak berubah-ubah). Misalkan untuk menunjukkan hubungan antara entitas mahasiswa dan mata_kuliah dengan nama kerelasian mengikuti memiliki hubungan banyak ke banyak (n-ke-n) yang di gambarkan sebagai berikut :

Gambar 4.4 Relasi antara entitas mahasiswa dan mata_kuliah

Gambar 4.5 Relasi antara entitas mahasiswa dan mata_kuliah

4.3. Menggambar ER_Diagram

Untuk menggambarkan sebuah ER_Diagram yang lengkap bisa dilakukan dengan mengikuti langkah-langkah berikut ini :

a. Mengidentifikasikan tiap-tiap entitas yang terlibat.
b. Mengidentifikasikan tiap-tiap atribut pada tiap-tiap entitas.
c. Mengidentifikasikan tiap-tiap kerelasian yang ada berikut jenisnya yang terjadi di antara entitas.
d. Menggambarkan simbol entitas, atribut, dan kerelasian antara entitas-entitas yang ada di basis data sedemikian rupa sehingga simbol-simbol kerelasian bisa digambarkan dengan sangat jelas dan tidak saling bertabrakan.

4.4. Kelebihan dan Kekurangan ER_Diagram

Apabila diterapkan dengan benar maka pemakaian ER_Diagram dalam pemodelan data pada basis data memberikan keuntungan bagi perancangan maupun pengguna antara lain :

1. Mempermudah pengguna dalam menganalisis sistem-sistem yang akan dikembangkan.
2. Mempermudah pengguna saat merancang basis data.
3. Rancangan basis data yang dikembangkan didasarkan pada ER_Diagram umumnya telah dirancang secara optimal.
4. Didalam beberapa kesempatan, penggunaan simbol grafis (termasuk ER_Diagram) lebih mudah dipahami oleh para pengguna dibandingkan bentuk naratif.
5. Perancangan dengan menggunakan ER_Diagram pengguna umumnya mudah memahami sistem-sistem yang ada serta basis data yang dirancang oleh pengguna.

Kekurangan ER_Diagram di antaranya adalah :

1. Adanya kebutuhan media yang sangat luas.
2. Sering kali ER_Diagram tampil sangat rumit dan sulit dipahami oleh pengguna.

4.5. Contoh Penerapan ER_Diagram

Contoh penerapan ER_Diagram berikut ini merupakan bagian yang sangat kecil dari sebuah ER_Diagram yang lengkap karena adanya keterbatasan media. Contoh yang diambil adalah pengolahan data-data akademik yang sudah pernah di conthkan pada pembahasan-pembahasan di bab sebelumnya. Berikut ini langkah-langkah menggambar ER_Diagram :

1. Mengidentifikasikan tiap-tiap entitas yang terlibat
   Entitas yang terkait dengan pengolahan data-data akademik antara lain :

   Tabel 4.6 Daftar Entitas Basis data

No	Nama Entitas
1	Mahasiswa
2	Program Studi
3	Mata Kuliah
4	Dosen

2. Mengidentifikasi tiap-tiap atribut pada tiap-tiap entitas.
   Atribut-atribut yang dibutuhkan untuk masing-masing entitas dalam pengolahan data-data akademik antara lain sebagai berikut :

   Tabel 4.7 Daftar atribut pada tiap-tiap entitas

Entias	**Atribut**
Mahasiswa	Nim,nama_mhs,kode_prodi,kelas,thn_masuk,kelamin,tanggal_lahir,tempat_lahir,alamat, status
Program_studi	Kode_prodi,nama_prodi,jenjang,status
Mata_kuliah	Kode_mk, nama_mk, sks, semester, status, kode_prodi

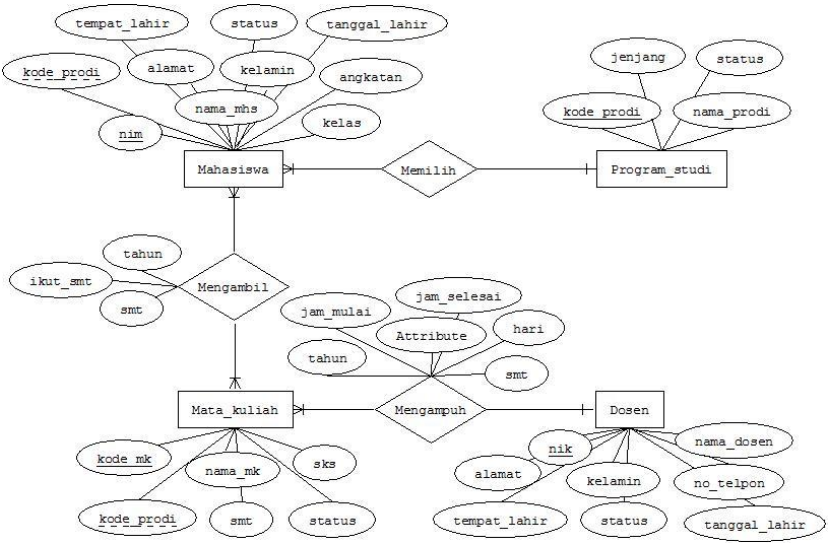
Dosen	Nik, nama_dosen, kelamin, tanggal_lahir, tempat_lahir, alamat, status, no_telpon

3. Mengidentifikasi tiap-tiap kerelasian yang mungkin terbentuk antara entitas-entitas yang ada.
   Kerelasian antara entitas-entitas yang mungkin terbentuk dalam pengolahan data-data akademik adalah sebaga berikut :

Tabel 4.8 Daftar kerelasian antara entitas

Entitas yang berhubungan		Nama Kerelasian	Jenis kerelasian	Representasi
Entitas I	Entitas II			
Mahasiswa	Program_studi	Memilih	n-ke-1	Penghubung : Kode_prodi dalam entitas mahasiswa
Mahasiswa	Mata_kuliah	Mengambil	n-ke-n	File KRS : Nim, smt, tahun, kode_mk, sks, ikut_smt
Dosen	Mata_kuliah	Mengampuh	1-ke-n	Jadwal_mengajar : Id_jadwal, Nik, kode_mk, sks, tahun, smt, hari, jam_mulai, jam_selesai, ruang

4. Menggambarkan ER_Diagramnya.
   Penggambaran ER_Diagram ada 2 cara yaitu menggambar dengan menyertakan atribut ditiap-tiap entitas dan relasi atau hanya menggambarkan entitas dan relasinya antara entitas saja dengan atribut yang berbeda. tapi lebih disarankan menggambar entitas dengan atributnya karena lebih memperjelas maksud gambar ER_Diagram. Berikut ini gambar ER_Diagram untuk pengolahan data akademik :



Gambar 4.6 ER_Diagram Akademik

4.6. Definisi RDBM

RDBM merupakan singkatan dari *Relasional Database Model* yang biasa disebut basis data relasional dikenalkan pertama kali oleh EF. Codd pada tahun 1970. Model basis data ini menunjukkan suatu cara yang dipakai untuk mengelola data-data secara fisik dalam memori sekunder yang akan berpengaruh pula terhadap cara-cara membentuk dan mengelompokkan keseluruhan data yang berkaitan dengan sistem yang sedang dirancang (Sutanta, 2004).

Model basis data RDBM sampai saat ini menjadi salah satu model yang paling banyak digunakan pengguna. Karena konsep dan terminologi yang dipakai dalam model RDBM hampir sama dengan kondisi sesungguhnya yang dihadapi oleh para pengguna sehingga sangat memudahkan pengguna dalam memahaminya.

Dalam RDBM dijelaskan hubungan logik (*logic*) antara data dalam basis data dengan menggambarkan dalam bentuk relasi-relasi yang berupa tabel-tabel mendatar yang terdiri atas sejumlah baris yang menunjukkan suatu record dan kolom yang menunjukkan suatu atribut tertentu. Relasi dibuat dengan sangat baik sehingga bisa menghilangkan kerangkapan data (*redundancy data*) yang tidak berguna. Dalam basis data kerelasian antara tabel satu dengan tabel yang lain ditunjukkan dengan suatu key yaitu primary ke dan foreign key.

4.7. Terminologi RDBM

Model relational basis data mempunyai terminologi sendiri yang berbeda dengan model-model lainnya. Terminologi berkaitan dengan penggunaan istilah-istilah basis data yang bersifat khusus yang harus dimegerti oleh pengguna karena dibutuhkan untuk menghindari terjadinya kerancauan saat menggunakan Database Management System (DBMS). Beberapa istilah yang ada pada RDBM antara lain :

Tabel 4.9 Istilah pada terminologi di RDBM

Istilah formal	**Istilah non formal**	**Keterangan**
Elemen data (data element), rinci data (data item), entri (entry)		Sebuah nilai data aktual
Atribut (attribute)	Kolom, field	Sekelompok rinci data yang mempunyai arti. Atribut memiliki tipe, ukuran, dan domain yang sama

Record/Tuple	Baris/Rekaman	Sekumpulan atribut yang mempunyai hubungan terhadap obyek tertentu.
Relasi (relation)	Tabel	Sekumpulan record yang sejenis secara relasi
Derajat	Aritas	Jumlah atribut dalam sebuah relasi
Kardinalitas		Jumlah record dalam sebuah relasi
Kerelasian (relasionship)		Hubungan antar relasi
Unary relation		Relasi yang tersusun oleh satu atribut
Binary relation		Relasi yang tersusun oleh dua atribut
Ternary relation		Relasi yang tersusun oleh tiga atribut
n-ary relation		Relasi yang tersusun oleh n atribut
Key		Satu atau gabungan atribut yang bersifat unik yang digunakan untuk mengidentifikasi setipa record dalam relasi
Primary key		Key yang dipilih sebagai kunci utama dalam relasi
Foreing key	Kunci tamu/kunci asing	Satu atau gabungan sembarang atribut yang menjadi PK dalam relasi lain yang mempunyai hubungan secara logik

Domain	Himpunan nilai yang memenuhi syarat
Schema	Deskripsi hubungan logik secara global termasuk didalamnya nama, deskripsi tipe , ukuran atribut dan relasi antar tabel dalam sistem basis data
Subschema	Deskripsi hubungan logik secara terpisah, termasuk didalamnya nama, deskripsi tipe dan ukuran atribut serta antar relasi dalam subsistem basis data

4.8. Karakteristik dan Komponen Relasi

Sebuah relasi dalam model data RDBM mempunyai beberapa karakteristik yang wajib dipenuhi karena karateristik dalam suatu relasi sangatlah penting karena akan menjadi dasar bagi perancangan struktur-struktur relasi yang digunakan sebelum penyimpanan data yang di lakukan dalam basis data. Berikut ini karakteristik dari RDBM :

1. Semua data yang dimasukkan pada suatu atribut dan record tertentu harus memiliki nilai tunggal (*single value*) dan harus berupa nilai yang tidak bisa dibagi-bagi lagi (*atomic value*).
2. Semua data yang dimasukkan pada atribut tertentu pada relasi-relasi harus memiliki tipe dan ukuran yang sama.

3. Masing-masing atribut pada sebuah relasi memiliki nama yang unik.
4. Pada sebuah relasi tidak boleh ada dua atau lebih record data yang identik.

Tiap-tiap relasi yang terjadi di RDBM selalu terbentuk atas du komponen yaitu sebagai berikut :

1. Intensi (intention) merupakan definisi penamaan suatu relasi dan batasan integritas yang terdiri atas kesatuan pada kunci primer (*primery key*) serta batasan integritas referensial yang ada pada kunci penghubung (*foreign key*). Intensi biasanya mempunyai sifat tetap dimana penamaan sebuah relasi sangat kecil mengalami perubahan-perubahan.
2. Ekstensi (*extention*) adalah nilai-nilai nyata sebuah data yang tersimpan dalam tabel pada saat tertentu. Ekstensi kadang berubah karena nilai-nilai data akan selalu mengalami perubahan-perubahan seperti penambahan, pengubahan atau penghapusan.

4.9. Kunci dan Aturan kunci Relasi

Dalam basis data kunci relasi sangat dibutuhkan untuk mengakses data dari relasi atau untuk merancang kerelasian antara entitas/relasi. Kunci relasi adalah satu atau bisa juga gabungan atribut yang mempunyai sifat unik yang bisa digunakan untuk membedakan tiap-tiap record dalam suatu relasi. Sehingga nilai dalam atribut yang dipakai sebagai kunci relasi tidak diperbolehkan ada yang sama untuk keseluruhan record dalam relasi.

Berdasarkan jumlah atribut pembentuknya, kunci relasi bisa dikelompokkan dalam dua jenis, antara lain :

1. *Simple key* (Kunci sederhana) merupakan kunci relasi yang terdiri atas sebuah atribut. Kunci ini terbentuk apabila sifat unik dari atribut telah dipenuhi dengan menggunakan hanya satu atribut saja. Contoh dalam tabel mahasiswa kunci sederhana adalah nim karena atribut ini bersifat unik sehingga bisa digunakan sebagai kunci relasi.
2. Composite Key (Kunci komposit) adalah kunci yang terbentuk atas gabungan dari beberapa atribut. Hal tersebut terjadi dikarenakan agar bisa mencapai sifat unik tidak bisa dipenuhi oleh satu atribut saja tetapi harus menggabungkan lebih dari satu atribut.

Berdasarkan macam-macamnya kunci relasi terdari dari :

1. Kunci Kandidat (*Candidate key*) atau CK.
2. Kunci Primer (*Primary key*) atau PK.
3. Kunci Alternatif (*Alternate key*) atau AK.
4. Kunci penghubung atau biasa disebut dengan kunci tamu (*Foreign key*) atau FK.

Sedangkan aturan-aturan pada kunci relasi utama adalah berkaitan dengan beberap batasan integritas antara lain integritas entitas serta integritas referensial yang dijelaskan sebagai berikut :

1. *Entity Integrity* (Integritas entitas)
   Dalam aturan integritas entitas mempunyai batasan secara kesatuan terhadap nilai data pada atribut-atribut bahwa nilai yang digunakan sebagai *Primary key* (PK) tidak boleh kosong (null). Untuk data yag bertipe karakter (character), pada tiap-tiap record yang ada dalam entitas atau relasi tidak boleh ada data yang panjang karakternya nol atau kosong (spaci). Sedangkan pada data yang bertipe numerik di tiap-tiap record

yang ada dalam entitas atau relasi juga tidak boleh mempunyai data dengan nilai nol (0). Peraturan ini memberikan kepastian bahwa tiap-tiap record dalam sebuah entitas atau relasi akan bisa diakses berdasarkan nilai Primary Keynya.

2. *Referencial integrity* (Integritas referensial)
   Aturan-aturan pada integritas referesial mempunyai batasan yaitu di dalam sebuah relasi antara dua atau lebih tabel di basis data yang dihubungkan dengan suatu kunci penghubung (*Foreign key*) atau FK, maka hubungan antara relasi tersebut harus memastikan bahwa tiap-tiap nilai data pada Foreign Key dalam relasi anak harus mempunyai record dengan nilai data yang sama dengan relasi yang dihubungkan (*primary key*).

4.10. Kerelasian Antara Entitas

Kerelasian menggambarkan hubungan antara entitas dalam basis data. Kerelasian antara entitas bisa ditunjukkan oleh Foreign Key dan juga bisa ditunjukkan menggunakan diagram yang biasa disebut diagram kerelasian antara entitas. Jenis kerelasian antara entitas di RDBM sama dengan jenis kerelasian di ER_Model yang bertujuan untuk menunjukkan hubungan antar data di dalam basis data. Kerelasian di RDBM antara lain :

1. Kerelasian 1-ke-1 (*one to one*) antara entitas.
2. Kerelasian 1-ke-n (*one to many*) antara entitas.
3. Kerelasian n-ke-1 (*many to one*) antara entitas.
4. Kerelasian n-ke-n (*many to many*) antara entitas.

Untuk menggambarkan diagram hubungan kerelasian antara relasi bisa dilakukan dengan langkah-langkah sebagai berikut :

1. Langkah pertama menuliskan tiap-tiap atribut dan relasi yang ada dalam bentuk tabel-tabel satu kolom dimana header tabel

memuat nama entitas dan isi dari tabel memuat nama-nama atribut-atributnya.

2. Langkah kedua menentukan kunci primer (PK) dan kunci penghubung (FK) apabila di tabel tersebut terdapat kunci dalam tiap relasi. Setelah itu memberikan tanda bintang satu (*) untuk atribut-atribut yang berfungsi sebagai kunci primer dan tanda bintang dua (**) pada atribut-atribut yang berfungsi sebagai kunci penghubung (FK).
3. Langkah ketiga menggambarkan hubungan (kerelasian) antar entitas dengan menghubungkan tiap-tiap FK dengan atribut-atribut yang disesuaikan dengan relasi atribut induknya menggunakan tanda garis.
4. Langkah terakhir menggambarkan jenis hubungan (kerelasian) antara entitas-entitas dengan menambahkan tanda panah berganda untuk jenis relasi banyak (*many*) dan panah tunggal untuk jenis relasi satu (*one*).

4.11. Contoh Penerapan RDBM

Untuk memperjelas tentang RDBM berikut ini contoh penerapannya bagaimana langkah-langkah menggambarkan diagram kerelasian antar relasi-relasi dalam tabel di basis data akademik. Berikut langkah-langkah RDBM :

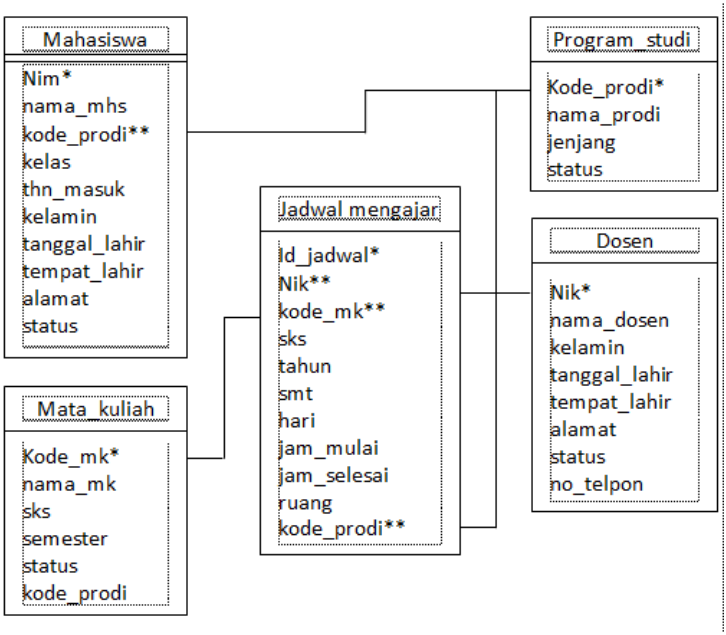
1. Langkah pertama adalah menjabarkan atribut dan relasi ke dalam tabel di basis data akademik

Gambar 4.7 Atribut dan relasi ke dalam tabel di basis data

2. Langkah selanjutnya menentukan atribut-atribut yang menjadi Primary key dan foreign key (kunci penghubung) di tiap-tiap tabel dalam basis data akademik. Dimana bintang satu (*) merupakan primary key dan pemberian bintang dua (**) adalah foreign key.

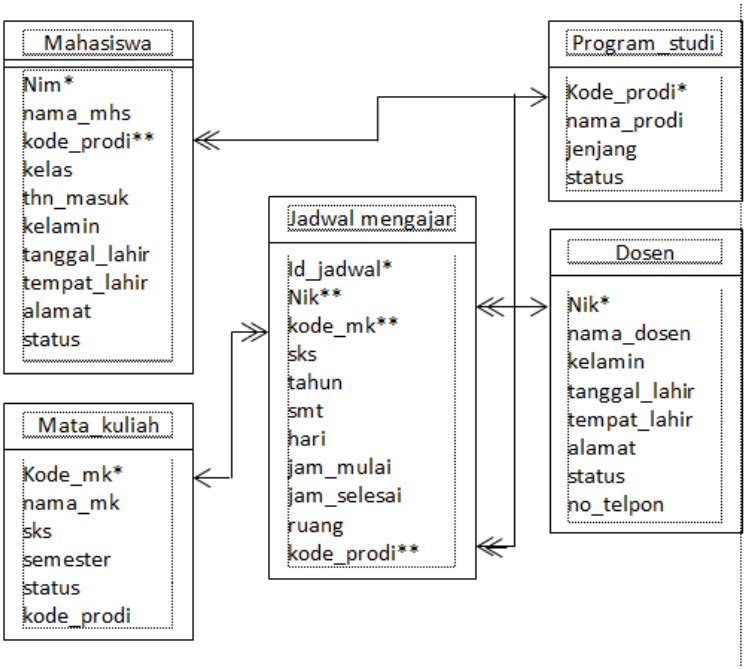
Gambar 4.8 penentuan atribut yang menjadi PK dan atribut FK

3. Langkah ketiga yaitu menggambarkan hubungan antara kunci primer (PK) dengan kunci penghubung (FK). Dimana PK digambarkan dengan panah satu dan FK digambarkan dengan panah dua.



Gambar 4.9 Menghubungkan antara tabel di basis data

4. Langkah terakhir yaitu memberikan jenis kerelasian yang terjadi antara tabel satu dengan tabel yang lain. Basis data merupakan tabel yang saling berelasi sehingga tiap-tiap tabel yang ada di basis data harus saling menyambung tidak ada tabel yang tidak berelasi.



Gambar 4.10 RDBM dengan jenis kerelasian antar tabel pada basis data

4.12. Ringkasan

1. ER Model atau biasa disebut *Entity Relationship Model* merupakan model untuk menjelaskan hubungan antara data satu dengan data yang lain di dalam basis data berdasarkan pandangan bahwa dunia nyata terdiri dari objek-objek dasar yang mempunyai hubungan (relasi) antara objek-objek tersebut.

2. Komponen Entity Relatinship antara lain Entitas, Atribut, Atribut kunci, Kerelasian antar entitas.
3. Atribut kunci dalam basis data terdiri atas Candidate Key atau Kunci Kandidat, Primary Key atau Kunci Utama, Alternate Key atau Kunci Alternatif, Foreign Key atau Kunci Tamu.
4. Kerelasian merupakan suatu kejadian yang terjadi di antara dua buah entitas yang nilainya perlu disimpan dalam sebuah basis data.
5. Model basis data RDBMS menunjukkan suatu cara yang dipakai untuk mengelola data-data secara fisik dalam memori sekunder yang akan berpengaruh pula terhadap cara-cara membentuk dan mengelompokkan keseluruhan data yang berkaitan dengan sistem yang sedang dirancang.
6. Kerelasian antara entitas meliputi Kerelasian 1-ke-1 (*one to one*) antara entitas. Kerelasian 1-ke-n (*one to many*) antara entitas. Kerelasian n-ke-1 (*many to one*) antara entitas. Kerelasian n-ke-n (*many to many*) antara entitas.

4.13. Evaluasi

1. Carilah sebuah contoh sistem informasi yang paling Anda pahami. Berdasarkan contoh tersebut
   a. Identifikasi semua entitas yang terlibat
   b. Identifikasi semua atribut pada masing-masing entitas
   c. Identifikasi kerelasian antar entitas yang terjadi
   d. Gambarkan ER Diagramnya

BAB 5
NORMALISASI

Bab ini membahas tentang normalisasi data serta tahapan-tahapannya antara lain Bentuk tidak normal, Bentuk normal kesatu, Bentuk normal kedua, Bentuk normal ketiga, Boyce-Codd normal form, Penerapan bentuk normalisasi.

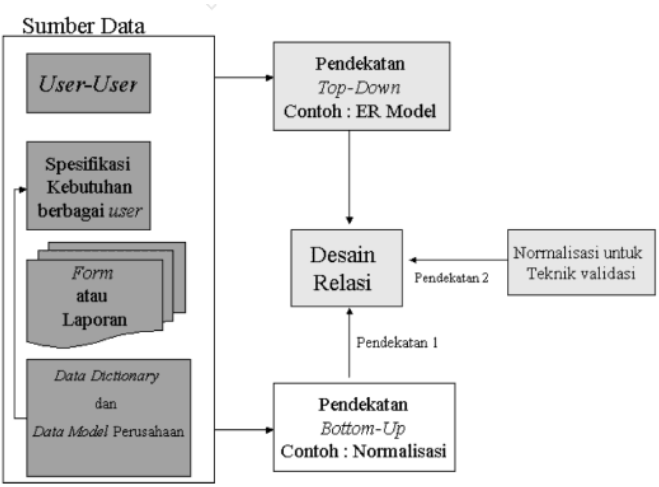
5.1. Definis Normalisasi

Normalisasi adalah teknik menggunakan pendekatan bottom-up yang digunakan untuk membantu mengidentifikasikan hubungan. dimulai dari menguji hubungan yaitu functional dependecies antara atribut (indrajani, 2015). Pengertian lainnya normalisasi merupakan proses pengelompokkan data elemen menjadi tabel-tabel yang menunjukkan entity dan relasinya (Kristanto, 1994). Normalisasi juga bisa diartikan suatu teknik untuk mengorganisasikan data kedalam tabel-tabel untuk memenuhi kebutuhan pemakai didalam suatu organisasi (Qalbi, 2014).

Tujuan dari Normalisasi Data adalah :

1. Untuk menghilangkan kerangkapan data atau redundacy data pada atribut.
2. Untuk mengurangi kompleksitas.
3. Untuk mempermudah pemodifikasian data.

Peranan normalisasi dalam perancangan basis data dlam penggunaan pendekatan bottom-up dan teknik validasi. Teknik validasi digunakan untuk memeriksa struktur relasi yang dihasilkan oleh ER Modeling sudah baik atau tidak seperti yang di tunjukkan gambar 5.1.



Gambar 5.1. Peranan Normalisasi dalam perancangan basis data

(Indrajani, 2015)

Dari gambar tersebut terlihat sumber data terdiri dari pengguna (user), spesifikasi kebutuhan berbagai user, berbagai form atau laporan, data dictionary dan data model organisasi atau perusahaan. Kemudian terdapat pendekatan top-down dn bottom-up dimana pendekatan tersebut akan menghasilkan desain relasi antar entitas (tabel) di basis data juga ada peranan normalisasi pada bottom-up dan teknik validasi.

Dalam normalisasi terdapat istilah kebergantungan fungsi (function dependecy) yaitu antara satu atribut mempunyai kebergantungan fungsi dengan atribut lain di tabel tersebut melalui suatu relasi. Misalkan ada relasi R yang mempunyai atribut Y dan atribut X, dimana atribut Y bergantung fungsi pada atribut X hanya Apabila tiap-tiap nilai X punya hubungan dengan tepat satu nilai Y

dalam R (dalam tiap-tiap satu waktu). Contoh terdapat entitas atau tabel masiswa dengan atribut sebagai berikut nim, nama, kelamin, tanggal_lahir, tempat lahir dan lain sebagainya. Isi dari atribut nama bergantung pada nomor induk mahsiswa (nim), sehingga bisa dikatakan bahwa atribut nama berganting secara fungsi pada nim dan nim menunjukkan fungsi nama. Apabila megetahui nim mahasiswa maka bisa juga diketahui nama mahasiswa tersebut yang bisa dinotasikan sebagai berikut :

NIM → Nama atau Nama = f (nim)

5.2. Bentuk-bentuk Normalisasi

Pada proses normalisasi perlu dikenal terlebih dahulu definisi dari bentuk-bentuk normalisasi antara lain :

5.2.1. Bentuk tidak normal (unformalized form)

Bentuk ini merupakan kumpulan data yang akan direkam di basis data, tidak ada keharusan untuk mengikuti format-format tertentu, bisa saja data tidak lengkap atau terduplikasi. Data yang dikumpulkan apa adanya sesuai dengan kedatangan.

5.2.2. Bentuk Normal Kesatu (1NF atau First Normal Form)

Bentuk normal kesatu mempunyai ciri-ciri antara lain tiap-tiap data dibentuk dalam file dasar (flat file) dan data dibentuk dalam satu record demi satu record. Tidak ada set atribut yang berulang-ulang atau atribut yang bernilai ganda.

Proses UNF ke 1NF adalah sebagai berikut :

1. Tentukan satu atau kumpulan atribut sebagai kunci untuk tabel unormlized.
2. Identifikasikan grup yang berulang dalam tabel unormalized yang berulang untuk kunci atribut.

3. Hapus grup yang berulang dengan cara :
    a. masukkan data yang semestinya ke dalam kolom yang kosong pada baris yang berisikan data yang berulang (*flattening the table*).
    b. Menggantikan data yang ada degan menulis ulang dari kunci atribut yang sesungguhnyake dalam relasi terpisah.

Contoh Penerapan sebagai berikut :

a. Entitas kelas terdiri dari kode_kelas, nama_kelas dan instruktur merupakan bentuk 1NF karena tidak ada yang berganda dan tiap atribut memiliki satu penertian yang tunggal. Contoh kelas dengan datanya :

Kode_kelas	Nama_kelas	Dosen
1234	Basis Data	Suseno
5678	Matematika Diskrit	Rejeki
1775	Pemrograman	Indira

b. Entitas mahasiswa terdiri dari nim, nama_mahasiswa, dosen_wali, kelas1, kelas2, kelas3. Mahasiswa yang mempunyai nim, nama dan dosen wali mengikuti 3 mata kuliah atau kelas. Dalam data ini ada perulangan kelas 3 kali maka ini bukan bentuk 1NF. Contoh data sebagai berikut :

Nim	Nama	Dosen_wali	Kelas 1	Kelas 2	Kelas 3
20200001	Siti	Suseno	1234	5678	
20200002	Eka	Rejeki	1234	1775	5678

Maka bentuk normal kesatu (1NF) dari data di atas adalah sebagai berikut :

nim	Nama	Dosen_wali	Kode_kelas
20200001	Siti	Suseno	1234
20200001	Siti	Suseno	5678
20200002	Eka	Rejeki	1234
20200002	Eka	Rejeki	1775
20200002	Eka	Rejeki	5678

5.2.3. Bentuk normal kedua (2NF atau second normal form)

Bentuk normal kedua mempunyai syarat yaitu bentuk data telah memenuhi kriteria bentuk normal kesatu, atribut bukan kunci haruslah bergantung secara fungsi pada kunci utama, atau primery key, sehingga untuk bentuk normal kedua haruslah sudah ditentukan kunci-kunci atribut. Kunci atribut harus unik dan bisa mewakili atribut lain yang menjadi anggotanya.

Proses 1FN ke 2NF adalah sebagai berikut :

1. Idetifikasikan primary key untuk relasi 1FN
2. Identifikasikan functional depedencies dalam relasi.
3. Apabila terdapat partial dependencies terhadap primary key, maka hapus dengan menempatkan dalam relasi yang baru bersama dengan salinnan determinannya.

Dari contoh relasi mahasiswa pada bentuk normal kesatu, terlihat bahwa kunci utama (primary key) adalah nomor induk mahaiswa (nim). Nama mahasiswa dan dosen wali bergantung pada fungsi nim, tetapi kode kelas bukanlah fungsi dari mahasiswa maka file atau tabel mahasiswa dipecah menjadi 2 relasi yaitu sebagai berikut :

- Relasi Mahasiswa

nim	Nama	Dosen_wali
20200001	Siti	Suseno

20200002	Eka	Rejeki

❖ Relasi ambil kelas

nim	Kode_kelas
20200001	1234
20200001	5678
20200002	1234
20200002	1775
20200002	5678

5.2.4. Bentuk normal ketiga (3NF atau three normal form)

Untuk menjadi bentuk normal ketiga maka relasi haruslah dalam bentuk normal kedua dan semua atribut bukan primer tidak punya hubungan yang transitif. Dengan kata lain, tiap-tiap atribut bukan kunci haruslah bergantun pada primery key secara menyeluruh. Contoh bentuk ketiga(3NF) sama dengan contoh pada bentuk yang kedua (2NF) karena seluruh atribut yang berada di contoh kedua bergantung penuh pada kunci utamanya.

5.2.5. Boyce-Codd Normal Form (BCNF)

Boyce-Codd Normal Form mempunyai paksaan yang lebih kuat dari normal bentuk ketiga. Untuk menjadi BCNF relasi antar tabel harus dalam bentuk normal kesatu dan tiap-tiap atribut harus bergantung fungsi dengan atribut superkey.

Penerapan BCNF bisa dilihat pada contoh berikut ini, terdapat relasi seminar, kunci utama (primary key) adalah nim+seminar. Mahasiswa bisa mengambil satu atau dua seminar. Tiap-tiap seminar membutuhkan 2 instruktur dan tiap-tiap mahasiswa dibimbing oleh salah satu diantara 2 instruktur seminar tersebut. Tiap-tiap instruktur boleh hanya mengambil satu seminar

saja. Pada contoh ini nim dan seminar menunjukkan seorang instruktur.

Relasi Seminar

nim	Seminar	Instruktur
20200001	201	Alan
20200002	201	Ikina
20200003	202	Treno
20200004	202	Amalia
20200005	202	Amalia

Bentuk relasi seminar adalah bentuk normal ketiga, tetapi tidak BCNF karena kode seminar masih bergantung fungsi pada instruktur, apabila tiap-tiap instruktur bisa mengajar hany pada satu seminar. Seminar bergantung secara fungsi pada satu atribut bukan superkey seperti yang disyaratkan oleh BCNF, maka relasi seminar dipecah menjadi 2 yaitu :

- Relasi Mengajar

Instruktur	Seminar
Alan	201
Ikina	201
Treno	202
Amalia	202

- Relasi Seminar-Instruktur

nim	Instruktur
20200001	Alan
20200002	Ikina
20200003	Treno
20200004	Amalia

20200005	Amalia

(Kristanti, 1994)

5.2.6. Penerapan Bentuk Normalisasi dalam Sebuah Kasus

Proses Perancangan basis data menggunakan teknik normalisasi bisa dimulai dari dokumen dasar yang dipakai dalam sistem. Seperti Faktur pembelian barang, berdasarkan faktur tersebut bisa dilakukan analisa data apa saja yang akan diperlukan dalam pembentukan basis data sesuai dengan kebutuhan organisasi atau perusahaan. Contoh penerapan Normalisasi sebagai berikut :

PT. Jasa Raya Makmur
Jl. Mojopahit 56 Sidoarjo

Faktur Pembelian Barang

No. Faktur : 123
Kode Supplier : 224
Tanggal : 15 / 04/ 2020
Nama Supplier : PT. Nakara

Kode Barang	Nama Barang	Jumlah
A01	Kursi	2
A02	Meja	3
A03	Lemari	2
A04	Rak Piring	2
A05	Rak Televisi	4

Maka langkah-langkah dalam normalisasi adalah sebagai berikut :

1. Langkah I Bentuk Unnormalized

Data-data pada faktur diatas dibentuk menjadi tabel unnormalized, dengan mencantumkan semua atribut data yang

ada. Dari sebanyak 3 Faktur pembelian tersebut diperoleh data pembelian sebagai berikut:

No_faktur	tanggal	Kode_ supplier	Nama_supplier	Kode_ barang	Nama_barang	jumlah
123	15012020	224	PT Nakara	A01	Kursi	2
				A02	Meja	3
				A03	Lemari	2
				A04	Rak Piring	2
				A05	Ral Televisi	4
124	17022020	245	PT Mutiara	A01	Kursi	4
				A02	Meja	2
125	29032020	335	PT Elang Jaya	A05	Rak Televisi	3

Menuliskan semua data yang akan disimpan bagian yang sama (double) tidak usah dituliskan sehingga terlihat record yang tidak lengkap, sulit untuk membayangkan bagaimana bentuk record yang harus dibentuk untuk menyimpan data tersebut.

2. Langkah II Bentuk Normal kesatu (1NF)

Bentuk data yang ada di tabel menjadi normal kesatu (1NF) dengan memisahkan data pada atribut yang tepat dan bernilai atomi (tidak bisa di pecah) juga seluruh record harus lengkap. Perhatikan tabel pembelian di atas, pada tabel tersebut terdapat kolom yang memiliki multivalue (yaitu; kode_barang, nama_barang, jumlah), maka kolom tersebut harus dipisah dengan membuat tabel lain dan menyertakan kolom primary key dari tabel asal sebagai foreign key pada tabel baru tersebut.

Tabel Pembelian

No_faktur	Tanggal	Kode_supplier	Nama_supplier
123	15012020	224	PT Nakara
124	17022020	245	PT Mutiara
125	29032020	335	PT Elang Jaya

Tabel Daftar_barang_pembelian

Kode_ barang	Nama_barang	jumlah	No_faktur
A01	Kursi	2	123
A02	Meja	3	123
A03	Lemari	2	123
A04	Rak Piring	2	123
A05	Ral Televisi	4	123
A01	Kursi	4	124
A02	Meja	2	124
A05	Rak Televisi	3	125

3. Langkah III Bentuk Normal kedua (2NF)

Pada bentuk normal kedua ini normalisasi dilakukan dengan mencari kunci-kunci atribut yang bisa digunakan sebagai patokan dalam pencarian data dan bersifat unik. Sehigga data-data yang ada tergantung secara fungsional pada kunci utama (*primary key*). Ini berati bahwa tiap-tiap kolom yang bukan kunci harus tergantung secara fungsional pada kolom kunci utama (*primary key*). Apabila ada kolom yang tidak tergantung secara fungsional terhadap kunci utama, maka kolom harus dipisah dengan membuat tabel lain atau tabel baru dan pada tabel baru tersebut juga ditambahkan kolom untuk primary key dari tabel asal sebagai relasi antar tabel.

Berdasarkan pada tabel-tabel pada proses normalisasi bentuk kesatu maka bisa dicari atribut unik yang diambil sebagai kunci kandidat. Pada tabel pembelian jelas terlihat bahwa no_faktur adalah unik sehingga bentuk Functional Dependenciesnya adalah :
No_faktur → tanggal, kode_supplier, Nama_supplier

Sedangkan pada tabel daftar_barang_pembelian bentuk Functional Dependeciesnya adalah :

- **Kode_barang → nama_barang**
- **Kode_barang –/–> jumlah**, karena untuk tiap-tiap set baris dimungkinkan untuk suatu nilai yang sama pada kolom kode_barang, terdapat nilai yang berbeda pada kolom jumlah.
- **Kode_barang –/–> no_faktur**, karena untuk tiap-tiap set baris dimungkinkan untuk suatu nilai yang sama pada kolom kode_barang, terdapat nilai yang berbeda pada kolom no_faktur.

kolom jumlah dan no_faktur tidak tergantung secara fungsional terhadap primary key, sehingga :

- Jumlah dan no_faktur harus dipindahkan ke tabel baru serta ditambahkan kode_barang sebagai primary key pada tabel asal ke tabel baru sehingga terbentuk relasi dengan tabel asal.
- Pada tabel baru tersebut tentukan primary key nya, sehingga atribut non primary key akan tergantung secara fungsional pada primary key tersebut. Dalam hal ini primary key-nya adalah kombinasi dari kolom kode_barang dan no_faktur, sehingga Fungsional Dependesiny adalah :

  kode_barang, no_faktur → jumlah
- Berikan nama tabel baru dan tabel yang mengalami perubahan dengan nama yang sesuai dengan datanya.

Sehingga tabel-tabel sekarang akan menjadi seperti berikut:

- Tabel Pembelian

No_faktur	Tanggal	Kode_supplier	Nama_supplier
123	15012020	224	PT Nakara
124	17022020	245	PT Mutiara
125	29032020	335	PT Elang Jaya

❖ Tabel barang

Kode_ barang	Nama_barang
A01	Kursi
A02	Meja
A03	Lemari
A04	Rak Piring
A05	Ral Televisi
A01	Kursi
A02	Meja
A05	Rak Televisi

❖ Tabel Daftar_pembelian

Kode_ barang	jumlah	No_faktur
A01	2	123
A02	3	123
A03	2	123
A04	2	123
A05	4	123
A01	4	124
A02	2	124
A05	3	125

4. Langkah IV Bentuk Normal ketiga (3NF)

Sesuai aturan 3NF, tabel atau relasi dalam bentuk 2NF serta tiap-tiap kolom bukan kunci tidak memiliki hubungan yang transitif terhadap primary key. Ini berati bahwa tiap-tiap kolom yang bukan kunci harus tergantung pada kolom kunci utama secara menyeluruh.

Dimana :

a. Apabila primary key adalah berupa kombinasi kolom maka kolom yang bukan primary key tidak boleh tergantung hanya pada salah satu kolom primary key (no transitif dependency). Apabila ada kolom yang memiliki hubungan secara transitif terhadap primary key, maka kolom tersebut harus dipisah dengan membuat tabel lain (tabel baru) dan pada tabel baru tersebut juga tambahkan kolom sebagai tempat bagi kolom ketergantungannya dan menjadi primary key pada tabel baru, kolom tersebut akan menjadi relasi yang menghubungkan tabel asal dengan tabel baru.

b. Pada tabel pembelian, jelas terlihat bahwa kolom nama_suppler sebenarnya tergantung pada kolom kode_supplier, karena kolom kode_supplier tergantung pada nomor_faktur (primary_key) maka nama_supplier juga tergantung pada nomor_faktur tetapi ketergantungannya hanya transitif, melalui kolom kode_supplier.
   no_faktur →kode_supplier, nama_supplier
   kode_supplier →nama_supplier
   Karena nama_supplier tergantung pada kode_supplier, maka pada tabel baru harus disertakan kolom kode_supplier sebagai primary key pada tabel baru. Sedangkan kolom kode_supplier pada tabel pembelian tidak dihapus (tetap ada, sebagai relasi dengan tabel baru).

❖ Tabel Pembelian

No_faktur	Tanggal	Kode_supplier
123	15012020	224
124	17022020	245
125	29032020	335

❖ Tabel barang

Kode_ barang	**Nama_barang**
A01	Kursi
A02	Meja
A03	Lemari
A04	Rak Piring
A05	Ral Televisi
A01	Kursi
A02	Meja
A05	Rak Televisi

❖ Tabel Supplier

Kode_supplier	Nama_supplier
224	PT Nakara
245	PT Mutiara
335	PT Elang Jaya

❖ Tabel Daftar_pembelian

Kode_ barang	**jumlah**	**No_faktur**
A01	2	123
A02	3	123
A03	2	123
A04	2	123
A05	4	123
A01	4	124
A02	2	124
A05	3	125

5.3. Ringkasan

1. Normalisasi adalah teknik menggunakan pendekatan bottom-up yang digunakan untuk membantu mengidentifikasikan hubungan. dimulai dari menguji hubungan yaitu functional dependecies antara atribut
2. Tujuan dari Normalisasi Data adalah Untuk menghilangkan kerangkapan data atau redundacy data pada atribut, Untuk mengurangi kompleksitas, Untuk mempermudah pemodifikasian data.
3. Bentuk-bentuk Normalisasi antara lain Bentuk tidak normal (*unformalized form*), Bentuk Normal Kesatu (*1NF atau First Normal Form*), Bentuk normal kedua (*2NF atau second normal form*), Bentuk normal ketiga (*3NF atau three normal form*), Boyce-Codd Normal Form (BCNF).

5.4. Evaluasi

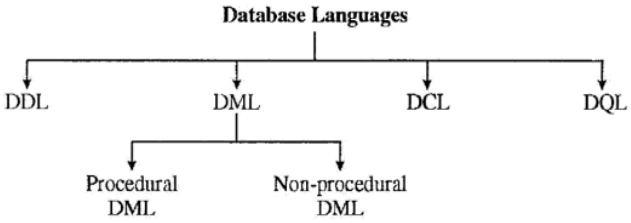
1. Berikan contoh penerapan normalisasi dalam suatu basis data mulai dari bentuk tidak normal sampai bentuk normal ketiga.

BAB 6
BAHASA BASIS DATA

Bab ini membahas tentang bahasa yang digunakan untuk membuat dan mengelola basis data yaitu antara lain Data Definition Language, Data Manipulation Language, Data Control Language dan Query.

6.1. Definisi Bahasa Basis Data

Bahasa yang digunakan untuk mendefinisikan, mengelolah dan memanipulasi basis data. Secara normal bahasa basis data adalah SQL (*Structure Query Langauge*). SQL kini telah menjadi bahasa standart dalam basis data, karena lebih mudah dan efisien untuk dipelajari daripada bahasa basis data yang lain. SQL berasal dari Sequel, kemudian di standarisasikan oleh ANSI (*American National Standard Institute*) sebagai Bahasa Query Database Relational. SQL di tujukan bagi pemakai yang ,ebih suka menyampaikan maksudnya pada penyususnan kode-kode bahasa pemrograman. sebuah sistem basis data menyediakan beberapa bahasa yaitu sebagai berikut :



Gambar 5.1. Hierarki Bahasa Basis Data
(Subandi, 2018)

6.2. Data Definition Language (DDL)

DDL adalah struktur basis data yang menggambarkan desain basis data secara keseluruhan. DDL digunakan untuk mendefinisikan struktur dan kerangka dari basis data yaitu antara lain :

1. Membentuk basis data, tabel, indeks.
2. Mengubah struktur tabel.
3. Menghapus basis data, tabel atau indeks.

Perintah DDL : create database, create table, create view, alter tabel, drop database, drop table, drop index, drop view.

6.3. Data Manipulation Language (DML)

DML merupakan bentuk bahasa basis data yang digunakan untuk melakukan manipulasi dan mengambil data pada basis data. Manipulasi data meliputi perintah :4. Qu

1. insert berfungsi untuk memasukkan atau menyisipkan penambahan data baru ke basis data.
2. Update berfungsi untuk mengubah data pada basis data.
3. Select berfungsi untuk melihat data atau memilih data pada basis data.
4. Delete berfungsi untuk menghapus data pada basis data.

sebagai contoh untuk menampilkan data pada tabel mahasiswa di basis data menggunakan perintah : **select * from mahasiswa**. DML merupakan bahasa yang bertujuan memudahkan pemakai untuk mengakses data sesuai dengan model data. Ada dua jenis Data Manipulation Language yaitu :

1. Prosedural yaitu mensyaratkan agar pemakai menentukan data apa yang diinginkan serta bagaimana cara menbisakannya. Contoh : dBASE III, FoxBASE.
2. Non Prosedural yaitu membuat pemakai bisa menentukan data apa yang diinginkan tanpa harus menyebutkan bagaimana cara menbisakannya. Contoh : SQL (Structural Query Language).

6.4. Data Control Language (DCL)

DCL merupakan bahasa yang digunakan untuk komponen dari pernyataan SQL untuk mengakses data dari basis data. DCL juga digunakan untuk pengaturan hak akses pengguna pada basis data yang meliputi :

1. Menugaskan hak akses terhadap basis data kepada pengguna atau grup pengguna, contoh : Grant
2. Membatalkan hak akses pengguna terhadap basis data, contoh Revoke.

Perintah-perintah DCL antara lain : Commit, RollBack,Grant, Revoke, dan lain sebagainya.

6.5. Data Query Language (DQL)

Data Query Language merupakan komponen dari SQL yang pernyataan-pernyataan yang di ajukan untuk mengizinkan dalam pegambilan dari basis data, biasanya di ambil dari beberapa tabel. DQL adalah pernyataan yang diajukan untuk mengambil informasi

yang merupakan bagian dari DML yang digunakan untuk pengambilan informasi. Beberapa perintah query antara lain :

1. Join Query untuk menggabungkan antara tabel satu dengan tabel yang lain.
2. Left Join Query untuk menggabungkan antara tabe satu dengan tabel yang lain tapi menitikberatkan pada data di tabel sebelah kiri.
3. Right Join Query untuk menggabungkan antara tabe satu dengan tabel yang lain tapi menitikberatkan pada data di tabel sebelah kanan.

Sebagai contoh untuk menampilkan data program studi yang di ambil oleh mahasiswa maka harus menggabungkan antara tabel mahasiswa dan tabel program studi karena yang disimoan di tabel mahasiswa hanya kode program studi saja jadi Apabila ingin mengetahui nama program studi harus mengambil dari tabel program studi dengan menggunakan perintah berikut :

```
select a.nim,a.nama_mahasiswa,b.nama from
mahasiswa a join program_studi b on
a.kode_prodi=b.kode_prodi.
```

6.6. Ringkasan

1. Bahasa yang digunakan untuk mendefinisikan, mengelolah dan memanipulasi basis data. Secara normal bahasa basis data adalah SQL (*Structure Query Langauge*).
2. Bahasa yang digunakan untuk membuat dan mengelola basis data yaitu antara lain Data Definition Language, Data Manipulation Language, Data Control Language dan Query.
3. *Data Query Language* merupakan komponen dari SQL yang pernyataan-pernyataan yang di ajukan untuk mengizinkan

dalam pegambilan dari basis data, biasanya di ambil dari beberapa tabel.

6.7. Evaluasi

1. Jelaskan perbedaan antara Data Manipulasi Language dan Data Query Language serta berikan contohnya.
2. Sebutkan dan jelaskan perintah-perintah yang terdapat pada DDL, DML dan DCL.

BAB 7
MYSQL

Pada bab ini akan mempelajari tentang sejarah MySQL, Kelebihan an kekurangan MySQL, Instalasi MySQL, Tipe data di MySQL.

7.1. Sejarah MySQL

MySQL merupakan salah satu jenis basis data server yang sangat terkenal, dikarenakan MySQL menggunakan SQL (*Structured Query Language*) sebagai bahasa dasar untuk mengakses data di basis data. MySQL termasuk RDBMS (*Relational Database Management System*) yang lebih populer pada pengguna pemrograman web, terutama di lingkungan Linux. Namun saat ini telah tersedia MySQL untuk platform sistem operasi Windows.

MySQL menbisa penghargaan sebagai basis data terbaik untuk server Linux versi Lnux Magazine tahun 2001 dan 2002 dan sebagai basis data terfavorit tahun 2000.

MySQL bersifat gratis (*free*) pengguna tidak perlu membayar untuk menggunakannya. Namun MySQL terdiri dari dua lisensi yaitu :

1. Lisensi gratis (free) (free software atau open source GNU General Public License). MySQL lisensi ini bebas digunakan, dimodifikasi source programnya dengan catatan harus di publikasikan ke pengguna (user).
2. Lisensi komersial (Non GPL) dimana pemakai harus membayar sejumlah biaya kapada MySQL sebagai pemegang hak cipta, sesuai dengan jenis layanan yang tersedia.

MySQL bisa digunakan pada berbagai platform sistem operasi khusus pada operasi windows. MySQL besifat shareware (pegguna akan dikenakan biaya setelah melakukan modifikasi dan digunakan untuk keperluan produksi). Perangkat lunak MySQL bisa di download melalui website MySQL yaitu http://www.mysql.com atau http://www.mysql.org (Saputro, 2005).

Pada tanggal 20 April 2009 MySQL di akuisisi oleh Oracle dan sejak saat itu berkembang isu Oracle bahwa yang memiliki produk basis data yang berkompetisi dengan MySQL maka Oracle akan mematikan MySQL. Namun sampai sejauh ini hal tersebut belum terbukti MySQL tetap bisa di pakai oleh semua pengguna.

Gambar 7.1. Logo MySQL

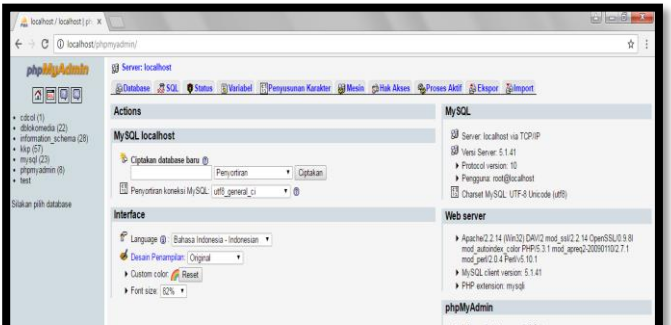
MySQL mempunyai 2 tampilan yang bisa diakses oleh user atau pemakai yaitu :

1. Berbasis GUI

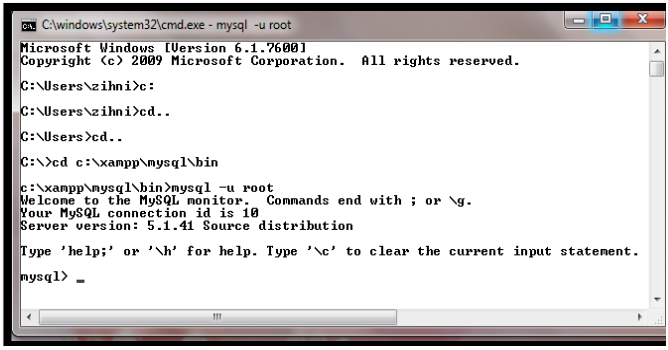
   Pada tampilan berbasis GUI, pengguna tidak mengetikkan perintah terlalu banyak hanya pada saat membutuhkan saja karena dalam tampilan berbasis GUI, pengguna tinggal mengklik apa yang diinginkan misalnya membuat basis data atau membuat tabel tanpa harus mengetikkan perintah CREATE DATABASE atau CREATE TABLE pengguna langsung memasukkan nama basis data yang akan dibuat pada form yang sudah disediakan oleh MySQL.

2. Berbasis Console

   Pada tampilan console pengguna harus mengetikkan perintah-perintah apabila ingin melakukan pengolahan data atau pembuatan basis data. Tampilan ini cocok untuk pegguna yang baru belajar tentang MySQL karena bisa memahami perintah-perintah MySQL dengan baik. MySQL yang cara mengaksesnya menggunakan command prompt (text).



Gambar 7.2. MySQL berbasis QUI



Gambar 7.3. MySQL berbasis Console

7.2. Kelebihan dan Kekurangan MySQL

MySQL mempunyai kelebihan bisa diakse oleh banyak bahasa pemrograman sebagai “frontend”. MySQL merupakan basis data server yang ideal untuk data segala ukuran dengan kemampuan mempunyai kecepatan yang sangat tinggi dalam melakukan proses data, multi-threaded, multi user dan query. Ukuran file yang dihasilkan MySQL lebih kecil dibanding file basis data yang lain.

Kelebihan MySQL dalam mengelolah data adalah :

Kecepatan, MySQL mempunyai kecepatan paling baik dibanding RDBMS lainnya.

1. **Mudah di gunakan**, perintah dalam MySQL dan aturan-aturannya relatif mudah dingat dan diimplementasikan karena MySQL menggunakan SQL sebagai bahasa standar database.
2. **Kecepatan,** berdasarkan hasil pengujian tabel 1 dan tabel 2, MySQL mempunyai paling baik dibandingkan RDBMS lainnya.
3. **Open source**, MySQL sudah menggunakan konsep open source, artinya siapapun bisa ikut dalam mengembangkan MySQL dan hasil pengembangannya di publikasikan kepada pemakai.
4. **Kapabilitas**, MySQL mampu memproses data yang tersimpan dalam database dengan jumlah 50 juta record, 60.000 tabel dan 5.000.000.000 juta baris.
5. **Biaya murah**, pemakai bisa menggunakan MySQL tanpa harus mengeluarkan biaya yang cukup mahal selama mengikuti konsep open source.
6. **Keamanan**, MySQL menerapkan sistem keamanan dan hak akses secara bertingkat, termasuk dukungan dengan keamanan data secara pengacakan lapisan data.
7. **Lintas platform**, MySQL bisa dijalankan pada beberapa sistem operasi di antaranya yaitu Linux, Windows, FreeBSD, Novel Netware, Sun Solaris, SCO Open Unix dan IBM's AIX.

Sedangkan kekurangan MySQL adalah sebagai berikut :

1. Banyak mengklaim kurang support terhadap pemrograman Visual atau Desktop, sehingga sedikit yang menggunakan untuk aplikasi visual.
2. Karena berlisensi GPL sehingga sulit menbisakan update untuk masalah yang mendesak, sehingga perusahaan skala menengah keatas lebih memilih RDBMS berlisensi dan disupport seperti Oracle dan MS SQL Server.

3. Sangat diragukan dalam menangani data yang berskala besar, karena ada beberapa opini yang pro dan kontra terhadap kemampuan MySQL terhadap pengolahan data yang besar.

7.3. SQL Standart Basis Data

Structure Query Language (SQL) bukan bahasa pemrograman atau perangkat lunak (*software*). Namun, SQL merupakan bahasa standar yang digunakan untuk mengolah basis data. Sedangkan perangkat lunak MySQL menggunakan SQL dalam mengolah basi data. SQL memiliki kemampuan melakukan query dan memanipulasi data. SQL dikatakan tidak termasuk sebagai bahasa pemrograman karena tidak memiliki kemampuan sebaga berikut :

1. Melakukan uji kondisi suatu pernyataan.
2. Melakukan uji pengulangan
3. Melakukan uji pencabangan.

Namun SQL mampu melakukan proses pembuatan basis data, tabel, melakukan perubahan struktur tabel, memanajemen hak akses basis data, menbisakan informasi, melakukan pemutakhiran data, menghapus tabel, menghapus atau mengubah data dan lain sebagainya.

Standar SQL didefiisikan oleh ISO (International Stadarts Organization) dan ANSI (the American Nasional Standars Institute) dengan sebutan SQL86. SQL merupakan subbahasa standar yang khusus digunakan untuk melakukan akses basis data relasional. Perintah SQL dikelompokkan sebagai berikut :

1. Data Definition laguange (DDL)

   DDL atau bahasa pendefinisian data merupakan perintah-perintah yang digunakan untuk membuat dan mendefinisikan basis data dan struktur tabelnya. Perintah-perintah DDL antara

lain : CREATE DATABASE, CREATE TABLE, CREATE VIEW, DROP VIEW, DROP DATABASE, DROP TABLE, ALTER TABLE.

2. Data Manipulation Language (DML)

   DML atau bahasa manipulasi data merupakan perintah-perintah yang digunakan untuk melakukan proses manipulasi atau pengelolaan data yang ada dalam basis data atau tabel. Seperti INSERT, UPDATE, DELETE dan SELECT

3. Data Control language (DCL)

   DCL atau bahasa pengendali data merupakan kelompok perintah-perintah SQL yang digunakan untuk melakukan otorisasi terhadap hak akses suatu data dan pengalokasian ruang. Seperti perintah GRANT, REVOKE, COMMIT DAN ROLLBACK.

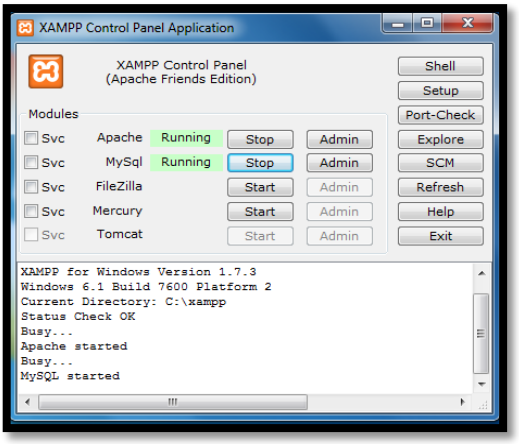
7.4. Aturan Perintah dalam MySQL

Dalam penggunaan RDBMS MySQL terdapat aturan-aturan yng harus dipatuhi agar penggunaan bahasa basis data tidak mengalami kesaahan saat digunakan. Aturan tersebut yaitu sebagai berikut :

1. Tiap-tiap perintah harus diakhiri dengan tanda titik koma (;) atau tanda \g.
2. Perintah bisa berupa perintah SQL atau perintah khusus untuk MySQL.
3. Perintah yang bukan MySQL bisa dipendekkan dengan menggunakan tanda \ dan huruf denpan perintah.
4. Perintah bisa diberikan dalam beberapa baris. Apabila tidak maka diakhiri dengan tanda titik koma (;) atau \g maka baris berikutnya masih dianggap satu kesatuan dengan perintah sebelumnya.
5. Perintah SQL dalam MySQL bisa dituliskan dalam huruf besar maupun kecil karena MySQL tidak sensitif case.

7.5. Instalasi MySQL

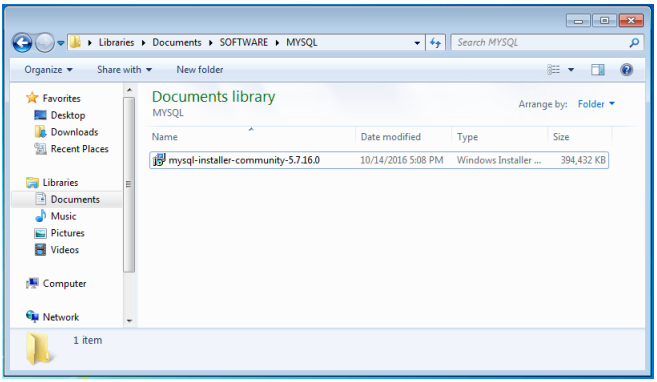
Cara instalasi MySQL bisa dilakukan dengan mudah karena MySQL sudah menyediakan setup untuk semua sistem operasi, pengguna tinggal mengunduh setup berdasarkan sistem operasi yang dipakainya. Pengguna bisa memilih menginstall MySQL saja atau memilih instalasi MySQL melalui Web server misalnya Xampp atau Appserv karena perangkat lunak xampp sudah satu paket dengan database mysql dan bahasa pemrograman PHP sehingga pengguna tidak perlu menginstall MySQL lagi.



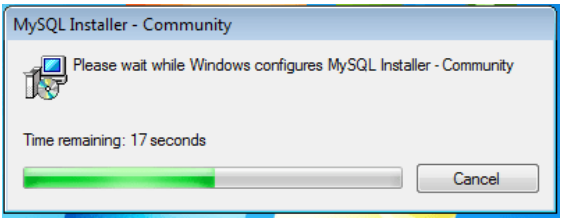
Gambar 7.4. Tampilan Xampp

Sebelum melakukan instalasi download terlebih dahulu software nya di link https://dev.mysql.com/downloads/mysql/, didalam buku ini menggunakan MySQL versi 5.7.16.0 community karena apabila menggunakan versi enterprise harus membayar yang biasanya digunakan untuk perusahaan besar. Berikut langkah-langkah instalasi MySQL :

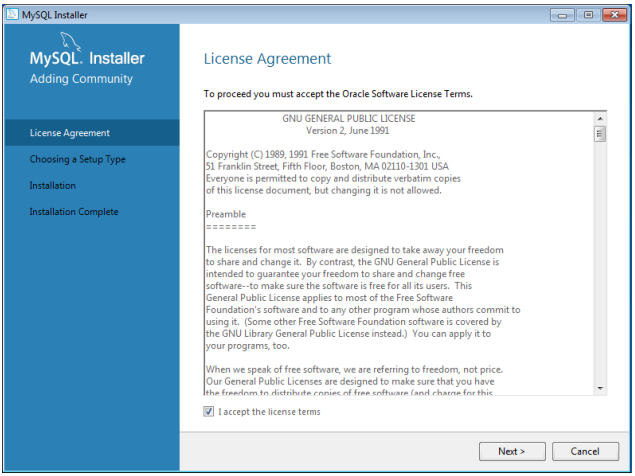
1. Buka folder tempat meyimpan file unduhan MySQL kemudian Jalankan file mysql-installer-community-5.7.16.0.msi yang sudah di unduh dengan meng-klik 2x file tersebut.



2. Setelah itu akan muncul dialog keterangan Windows Configures MySQL Installer, tunggu sampai proses loading ini selesai.

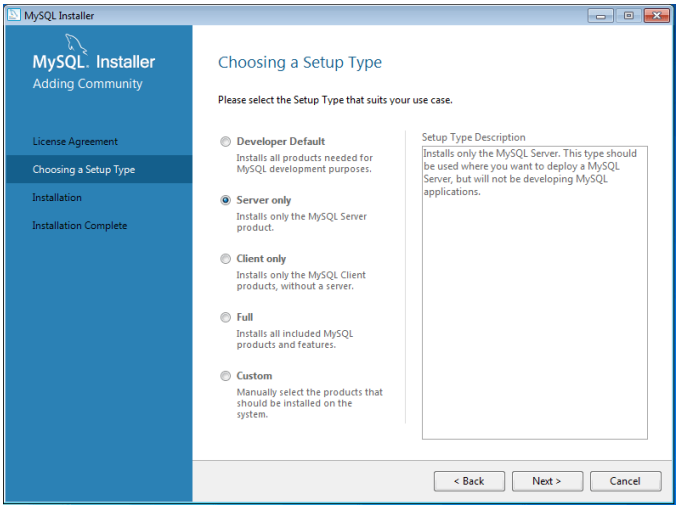


3. Seperti pada umumnya menginstall aplikasi, akan muncul tampilan License Agreement, silahkan centang **checkbox I Accept the license terms** lalu klik **Next**.

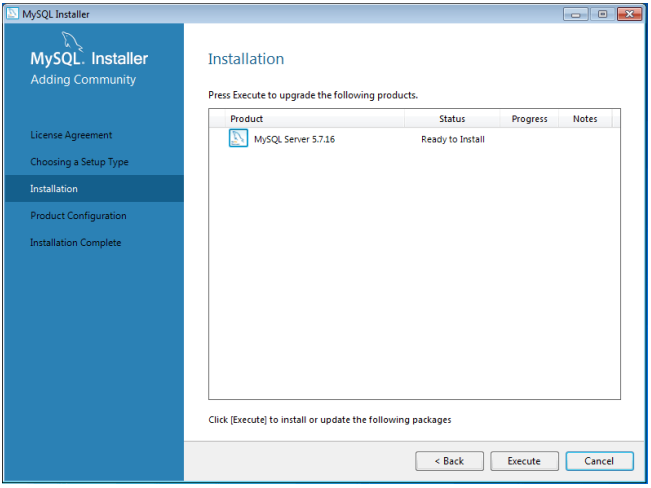


4. Lalu akan muncul tampilan Choosing a Setup Type, disini ada beberapa opsi pilihan proses installasi, yaitu:
   a. Developer Default
   b. Server Only
   c. Client Only
   d. Full
   e. Custom

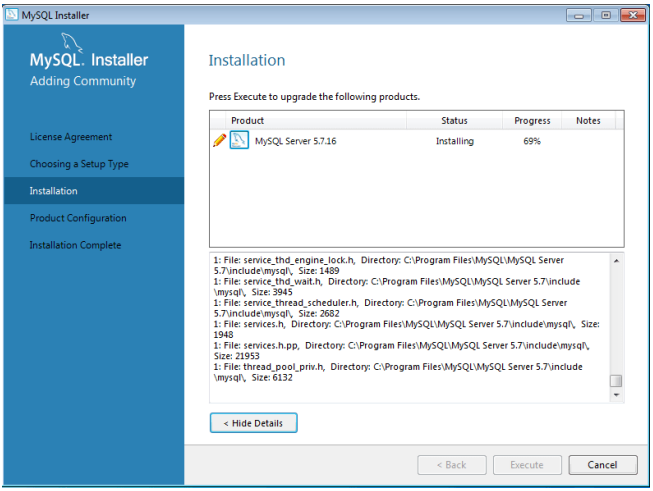
   Dimana perbedaan yang mendasar dari masing-masing opsi adalah program apa saja yang ingin di install didalamnya, kemudian pilih **Server Only** lalu **klik Next** maka akan muncul tampilan sebagai berikut :



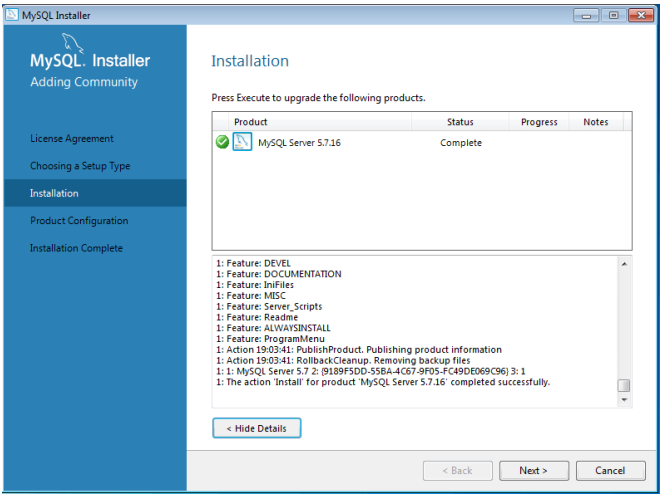
5. Selanjutkan akan muncul daftar program apa saja yang akan dipakai, karena hanya akan menginstall basis data mysql server saja maka yang muncul hanya satu saja, yaitu **MySQL Server**. Setelah itu klik **Execute**.



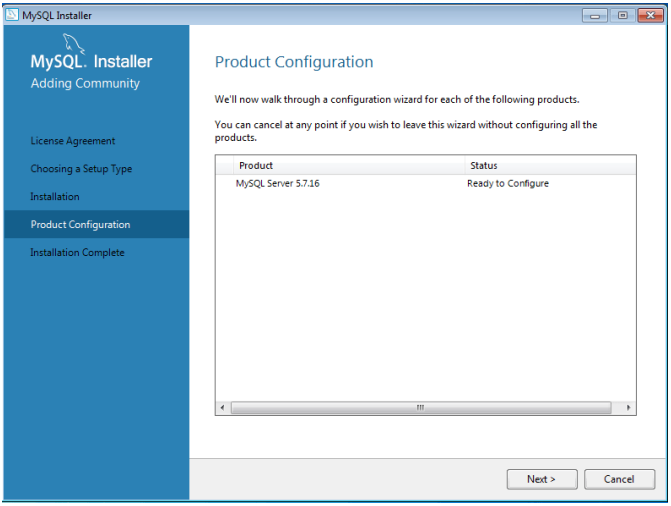
6. Maka Proses Install Database MySQL berjalan, Apabila ingin melihat detail proses install database mysql silahkan klik tombol Show details, seperti gambar dibawah ini.



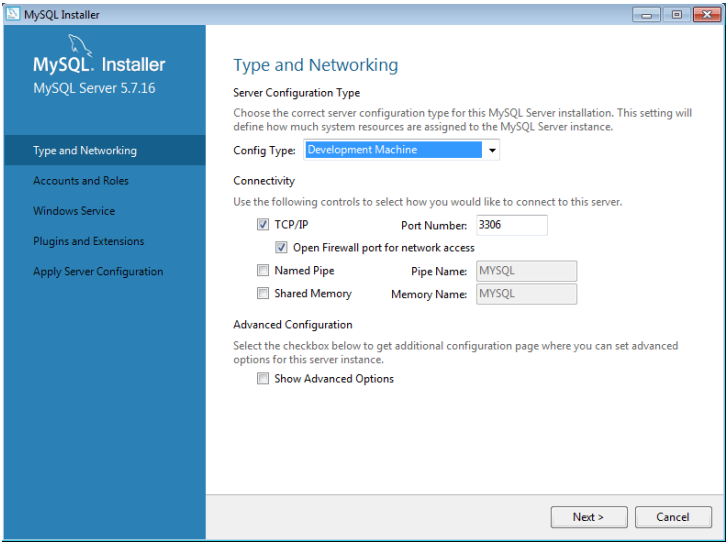
7. Setelah itu akan tampil keterangan bahwa proses intall database mysql telah complete pada bagian status dan pada bagian detail proses install database mysql akan muncul fitur apa saja didalamnya, lalu klik Next.



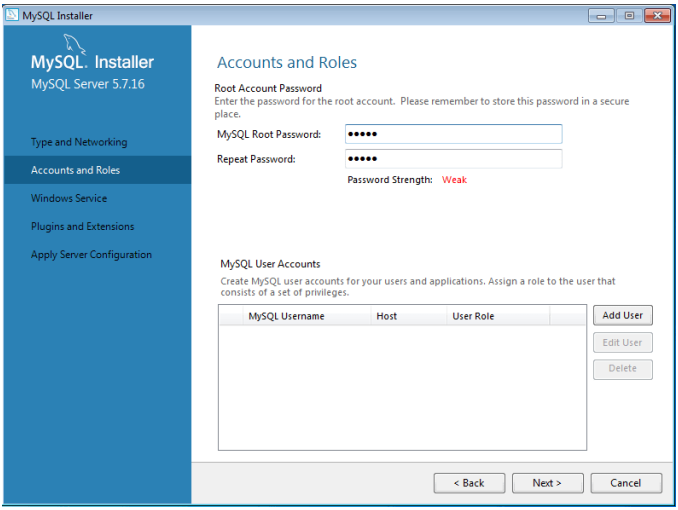
8. Sampai disini akan muncul pengaturan konfigurasi MySQL, Klik Next.



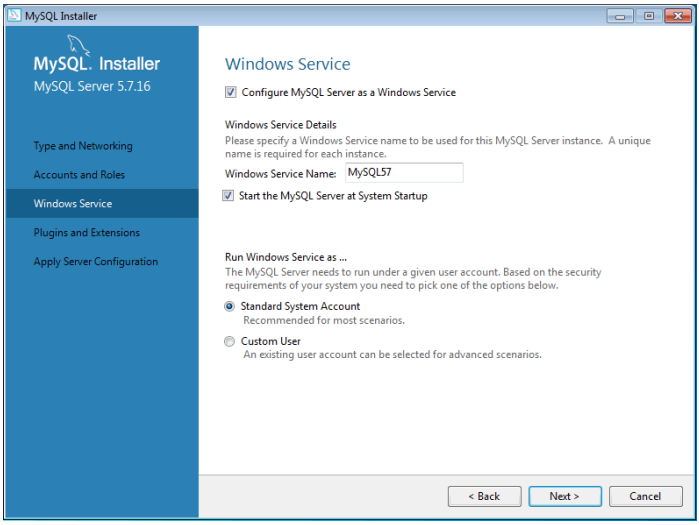
9. Akan muncul tampilan konfigurasi awal MySQL Type and networking, untuk detail konfigurasi silahkan disamakan dengan konfigurasi dibawah ini, Apabila telah sesuai klik **Next**.
   a. Config Type: Development Machine
   b. Centang TCP/IP
   c. Masukan Port Number: 3306
   d. Centang Open Firewall Port for network access
   e. Kosongkan bagian yang lain



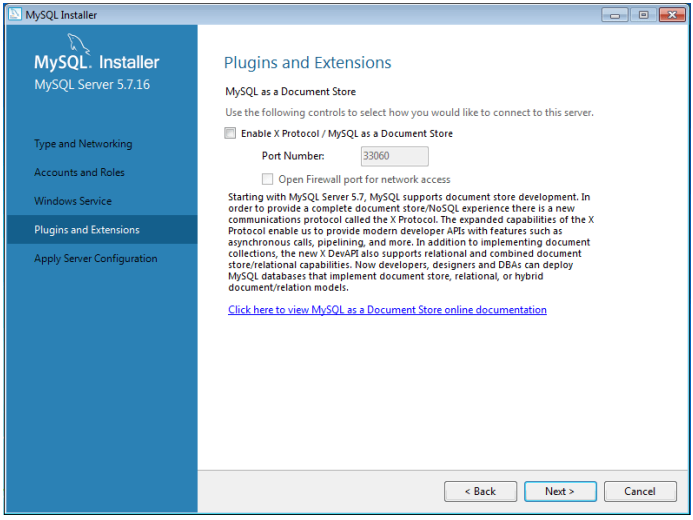
10. Setelah itu akan muncul bagian Konfigurasi Accounts and Roles, pada bagian silahkan isikan password sesuai dengan keinginan. Untuk dibagian bawah dilewatkan saja, kecuali Apabila ingin menambah pengguna di MySQL ini, Apabila telah selesai silahkan klik Next.



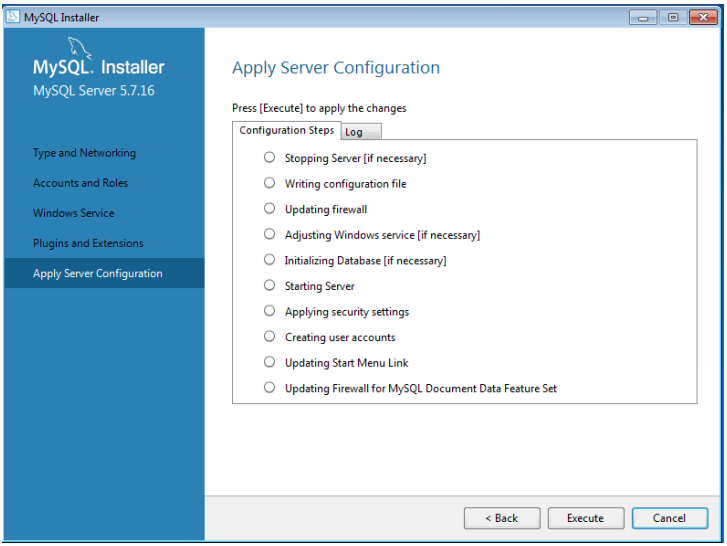
11. Setelah itu akan muncul bagian konfigurasi Windows Service, konfigurasi ini yang akan menangani proses stratup program MySQL, sehingga apabila komputer baru saja diaktifkan maka secara otomatis MySQL bisa langsung digunakan tanpa perlu di aktifkan terlebih dahulu, silahkan ikuti pengaturan dibawah ini, Apabila telah selesai klik Next.
    a. Centang Configure MySQL Server as a Windows Service
    b. Pada kolom Windows Service Name silahkan isi sesuai keinginan, namun penulis membiarkan secara default atau bawaan, ini hanya masalah penamaan service saja.
    c. Lalu centang pada Start the MySQL Server at System Startup.
    d. Lalu pilih Standard System Account.



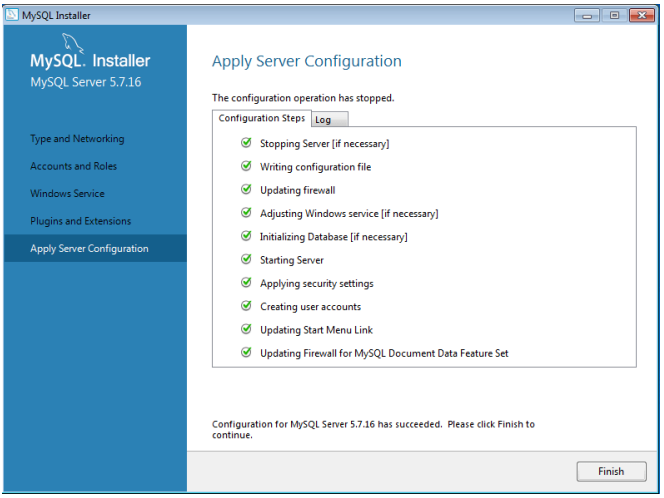
12. Setelah itu akan dibawa kebagian Plugins and Extension, pada konfigurasi ini biarkan semuanya kosong. Lalu klik Next.



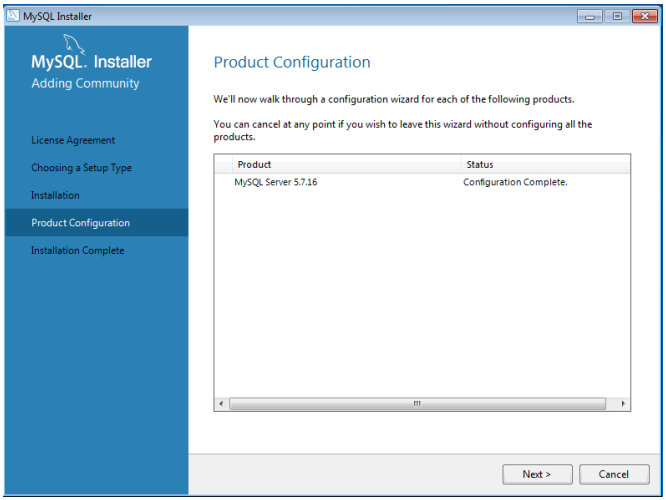
13. Sampai disini semua konfigurasi akan dijalankan secara otomatis dan akan terlihat apa saja yang akan dikonfigurasi oleh sistem, silahkan klik tombol Execute.



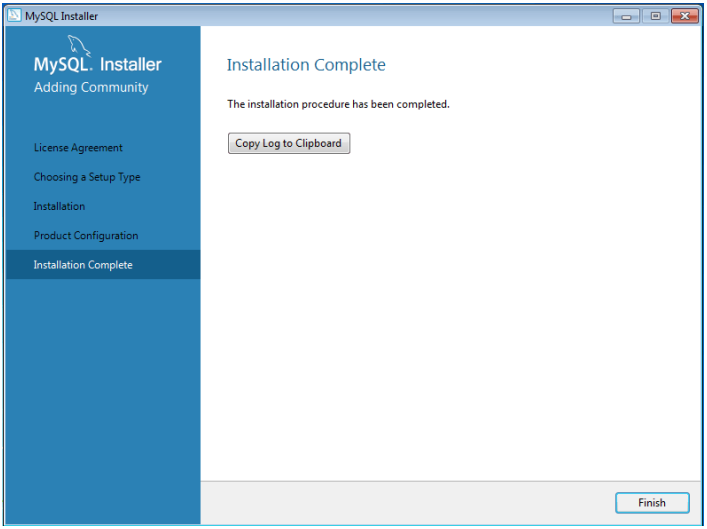
14. Apabila seluruh daftar konfigurasi telah selesai semua, maka daftar itu akan secara otomatis satu persatu muncul icon ceklis warna hijau, icon itu menunjukan bahwa masing-masing konfigurasi telah selesai. Lalu klik Finish.



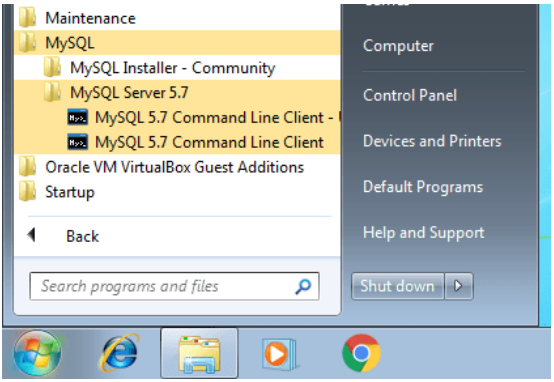
15. Setelah itu langkah instalasi akan dikembalikan lagi ke bagian Product Configuration, tetapi tampilan berbeda dengan sebelumnya karena akan menampilkan daftar keterangan modul apa saja yang telah selesai di konfigurasi. Lalu Klik Next.



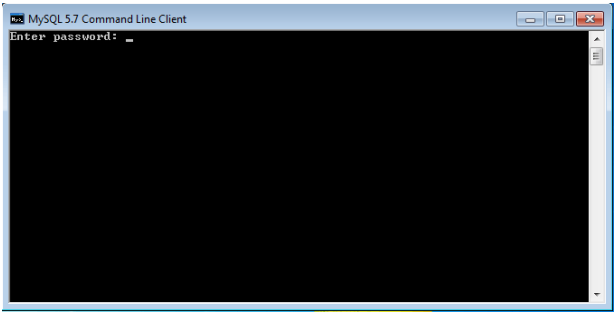
16. Terakhir akan muncul keterangan Installation Complete, dibagian ini ada tombol Copy Log to Clipboard, tombol ini berfungsi untuk meng-copy seluruh keterangan Log Proses Installasi, biarkan saja tombol tersebut. Silahkan Klik Finish.



17. Sampai disini proses install database MySQL telah selesai, namun untuk bisa memastikannya silahkan klik tombol start apabila menggunakan windows 7 atau windows 10, tapi apabila menggunakan windows 8, silahkan cari melalui kolom pencarian pada bagian kanan layar.



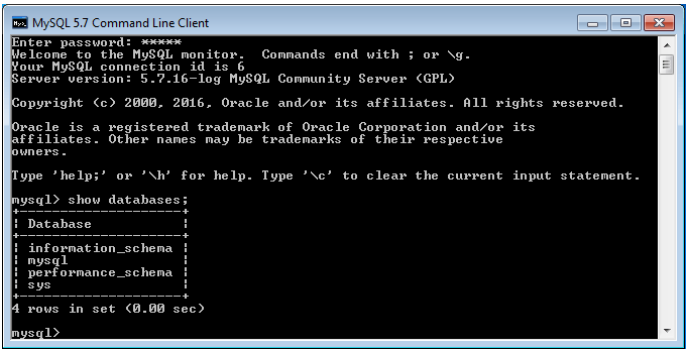
18. Apabila sudah klik start, klik All Programs -> MySQL -> MySQL 5.7 lalu pilih MySQL 5.7 Command Line Client, maka akan muncul Commpand Promp, dan apabila diharuskan memasukkan password, silahkan masukan Password yang sebelumnya telah ditentukan pada bagian konfigurasi MySQL.



19. Apabila sudah berhasil masuk ke MySQL, silahkan ketikan perintah berikut ini:

    SHOW DATABASES;

Lalu klik enter, maka akan muncul sejumlah database, seperti tampilan dibawah ini.



20. Dari tampilan di atas adalah daftar database bawaan pada MySQL, database tersebut adalah konfigurasi MySQL itu sendiri, untuk keluar silahkan ketikan perintah dibawah ini.
    QUIT;
    (Sumber : https://bahasaweb.com/install-database-mysql/)

Apabila menggunakan web server Xampp maka untuk masuk ke MySQL langkah-langkahya sebagai berikut :

1. Melalui DOS Prompt, masuk ke direktori utama MySQL dengan cara sebagai berikut (yang diketik hanya yang digaris bawah) :
   C:\> cd c:\xampp\mysql\bin
2. Setelah itu ketikkan perintah berikut (yang diketik hanya yang digaris bawah) :
   C:\xampp\mysql\bin> mysql –u root atau
   C:\xampp\mysql\bin> mysql –u root –p
3. Apabila diminta untuk memasukkan password, isikan password yang digunakan pada saat instalasi.

7.6. Tipe Data di MySQL

Dalam basis data terdapat tabel-tabel untuk menyimpan data. Tabel merupakan bentuk fisik data yang terdiri atas baris dan kolom. Tiap-tiap kolom selalu menyimpan tipe data sejenis dan tiap-tiap baris terdiri dari berbagai tipe data yang berbeda. Tipe data merupakan jenis nilai yang bisa ditampung pada suatu variable, bisa berupa angka (numerik), teks, ataupun berupa gambar. Tipe data bisa di bagi menjadi :

1. Tipe data Angka (Numerik)
   merupakan tipe data yang bisa kita gunakan pada suatu variabel konstanta yang bisa menyimpan nilai berupa angka.

No	Tipe Data	Fungsi	Jangkauan	Ukuran
1	Tinyint	Digunakan untuk menyimpan data bilangan bulat positif dan negatif.	-128 s/d 127	1 byte (8 bit)
2	Smallint	Digunakan untuk menyimpan data bilangan bulat positif.	-32.768 s/d 32.767	2 byte (16 bit)
3	Mediumint	Digunakan untuk menyimpan data bilangan bulat positif.	-8.388.608 s/d 8.388.607	3 byte (24 bit)
4	Int	Digunakan untuk menyimpan data bilangan bulat positif.	-2.147.483.648 s/d 2.147.483.647	4 byte (32 bit)

5	Bigint	Digunakan untuk menyimpan data bilangan bulat positif.	± 9,22 x 1018	8 byte (64 bit)
6	Float	Digunakan untuk menyimpan data bilangan pecahan positif dan negatif presisi tunggal.	-3.402823466E+38 s/d -1.175494351E-38, 0, dan 1.175494351E-38 s/d 3.402823466E+38	4 byte (32 bit)
7	Double	Digunakan untuk menyimpan data bilangan pecahan positif dan negatif presisi ganda.	-1.79…E+308 s/d -2.22…E-308, 0, dan 2.22…E-308 s/d 1.79…E+308	5 byte (64 bit)
8	Real	Merupakan sinonim dari Double.	-1.79…E+308 s/d -2.22…E-308, 0, dan 2.22…E-308 s/d 1.79…E+308	6 byte (64 bit)
9	Decimal	Digunakan untuk menyimpan data bilangan pecahan positif dan negatif.	-1.79…E+308 s/d -2.22…E-308, 0, dan 2.22…E-308 s/d 1.79…E+308	7 byte (64 bit)

10	Number	Merupakan sinonim dari Decimal.	-1.79…E+308 s/d -2.22…E-308, 0, dan 2.22…E-308 s/d 1.79…E+308	8 byte (64 bit)

2. Tipe data String (Teks)
   Merupakan tipe data yang digunakan untuk menyimpan data berupa teks.

No	Tipe Data	Fungsi	Jangkauan
1	Char	Digunakan untuk menyimpan data string ukuran tetap.	0 s/d 255 karakter
2	Varchar	Digunakan untuk menyimpan data string ukuran dinamis.	0 s/d 255 karakter (versi 4.1), 0 s/d 65.535 (versi 5.0.3)
3	Tinytext	Digunakan untuk menyimpan data text.	0 s/d 255 karakter (versi 4.1), 0 s/d 65.535 (versi 5.0.3)
4	Text	Digunakan untuk menyimpan data text.	0 s/d 65.535 (216 – 1) karakter
5	Mediuntext	Digunakan untuk menyimpan data text.	0 s/d 224 – 1 karakter
6	Longtext	Digunakan untuk menyimpan data text.	0 s/d 232 – 1 karakter

3. Tipe data Tanggal (*Date*) dan Waktu (*Time*)
   tipe data date dan time ini biasanya untuk menampung data tanggal dan waktu, ada beberapa tipe data date dan time, sebagai berikut ini:

No	Tipe Data	Fungsi	Jangkauan	Ukuran
1	Date	Digunakan untuk menyimpan data tanggal	1000-01-01 s/d 9999-12-31 (YYYY-MM-DD)	3 byte
2	Time	Digunakan untuk menyimpan data waktu.	-838:59:59 s/d +838:59:59 (HH:MM:SS)	3 byte
3	Datetime	Digunakan untuk menyimpan data tanggal dan waktu.	1000-01-01 00:00:00' s/d '9999-12-31 23:59:59	8 byte
4	Year	Digunakan untuk menyimpan data tahun dari tanggal.	1900 s/d 2155	1 byte

4. Tipe data Blob
   Merupakan tipe data yang digunaka untuk menyimpan data biner, biasanya digunakan untuk menampung kode-kode biner atau object dari suatu file misal foto.

No	Tipe Data	Fungsi	Jangkauan
1	Bit	Digunakan untuk menyimpan data biner	64 digit biner

2	Tinyblob	Digunakan untuk menyimpan data biner	255 byte
3	Blob	Digunakan untuk menyimpan data biner.	216 – 1 byte
4	Mediumblob	Digunakan untuk menyimpan data biner.	224 – 1 byte
5	longblob	Digunakan untuk menyimpan data biner.	232 – 1 byte

5. Tipe Data lainnya
   Selain tipe data diatas, MySQL memiliki tipe data yang lainnya yang mungkin tipe data ini akan terus bertambah seiring perkembangannya.

No	Tipe Data	Fungsi	Jangkauan
1	Enum	Enumerasi (kumpulan data).	1 s/d 65535 string.
2	Set	Combination (himpunan data).	1 s/d 255 string anggota.

Dalam menggunakan tipe data perlu memperhatikan penggunaannya disesuaikan dengan kebutuhan dan kesesuaian data yang dipakai dan perllu diingat juga bahwa tiap-tiap nilai ukuran yang diberikan pada field atau kolom di MySQL akan menambah beban memori penyimpanan, sehingga tipe dalam menetukan tipe dan ukuran harus benar-benar sesuai dengan kriteria data yang ingin dimasukan.

7.7. Ringkasan

1. MySQL merupakan salah satu jenis basis data server yang sangat terkenal, dikarenakan MySQL menggunakan SQL (Structured Query Language) sebagai bahasa dasar untuk mengakses data di basis data.
2. MySQL bersifat gratis (free) pengguna tidak perlu membayar untuk menggunakannya.
3. MySQL bisa digunakan pada berbagai platform sistem operasi khusus pada operasi windows.
4. Perangkat lunak MySQL bisa di download melalui website MySQL yaitu http://www.mysql.com atau http://www.mysql.org
5. Tipe data pada MySQL yaitu Tipe data Angka, Tipe data teks, Tipe data tanggal, Tipe data Blob

7.8. Evaluasi

1. Apa yang Anda ketahui tentang MySQL.
2. Sebutkan langkah-langkah instalasi MySQL serta install MySQL di Laptop Anda.

BAB 8
DATA DEFINITION LANGUAGE (DDL)

8.1. Perintah Dasar DDL

Data Definition Language (DDL) merupakan bahasa dasar dalam tahap awal penerapan basis data di MySQL yang digunakan untuk membuat dan mengahapus basis data, membuat, mengubah

dan menghapus tabel dan view dan lain sebagainya. Perintah-perintah dalam DDL adalah sebagai berikut :

Perintah	**Keterangan**
Create database	Membuat basis data
Drop database	Menghapus basis data
Create table	Membuat tabel
Alter table	Mengubah atau menyisipkan kolom ke dalam tabel
Drop table	Menghapus tabel dari basis data
Create index	Membuat index
Drop index	Menghapus index

8.2. Menciptakan Basis Data

Langkah pertama yang dilakukan untuk bisa menyimpan data di basis data adalah membuat basis data, mengaktifkan basis data atau menghapus basis data tersebut.

A. Membuat Basis Data

Sebelum membuat tabel harus membuat basis data terlebi dahulu, perintah untuk membuat basis data baru adalah sebagai berikut :

```
CREATE DATABASE <nama_database>;
```

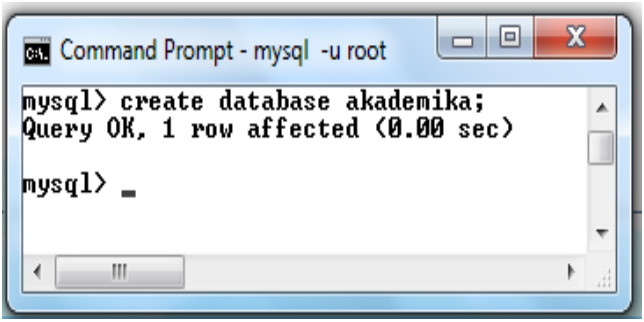
Dalam MySQL tiap-tiap nama basis data otomatis juga sebagai nama direktori, sehingga tiap-tiap basis data tidak akan saling terpengaruh. Yang perlu diingat penamaan basis data harus jelas dan mudah dipahami bisa terdiri dari satu kata yang menunjukkan

basis data apa yang akan di bangun misalnya pengolahan data dibidang akademik maka nama basis data adalah akademika. Penamaan basis data juga bisa menggunakan dua kata apabila menggunakan spasi bisa diganti dengan tanda garis bawah (_),Apabila muncul Query OK maka basis data berhasil dibuat.

.

```
Mysql>CREATE DATABASE akademika;
```

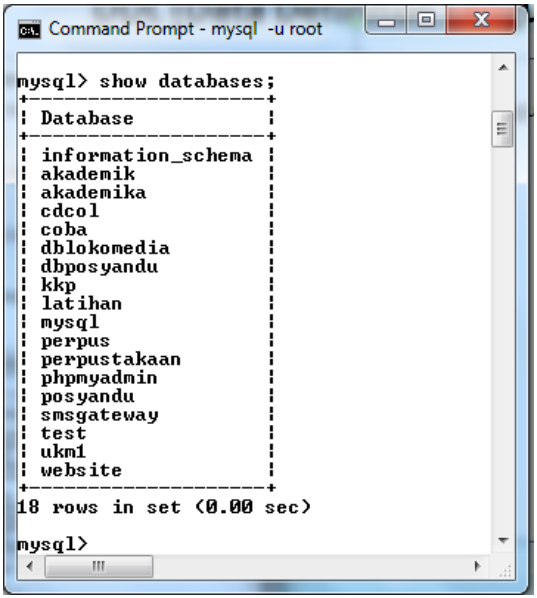
Tampilan hasil eksekusi script :



B. Melihat Daftar Basis Data

Untuk melihat atau menampilkan daftar basis data yang ada di Mysql bisa menggunakan perintah :

```
SHOW DATABASES;
```



C. Menghapus Basis Data

Untuk melakukan penghapusan terhadap basis data yang sudah dibuat maka menggunakan perintah :

```
DROP DATABASE <nama_database>;
```

Dimana basis data yang akan dihapus harus sesuai dengan nama basis data yag dibuat. Berikut ini contoh perintah untuk menghapus database dengan nama akademika :

```
Mysql> DROP DATABASE akademika;
```

Perlu diperhatikan Apabila akan menghapus basis data pertimbangkan terlebih dahulu apakah sudah tidak terpakai lagi atau sudah ada backupnya Apabila tidak maka apabila butuh data di basis data tersebut maka harus memulai lagi dari awal.

D. Mengaktifkan Basis Data

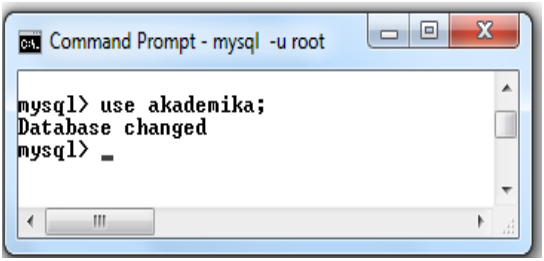
Sebelum membuat suatu tabel, terlebih dahulu harus mengaktifkan basis data yang akan digunakan untuk menyimpan tabel-tabel tersebut dengan perintah :

```
USE <nama_database>
```

Contoh :

```
Mysql>use akademika;
```

Maka akan tampil hasil sebagai berikut :



8.3. Menciptakan Tabel Dalam Basis Data

Dalam basis data tabel atau field berfungsi untuk menyimpan record atau data.

A. Membuat tabel

Sebelum melakukan pengolahan data maka harus membuat tabel perintahnya adalah sebagai berikut :

```
CREATE TABLE namatabel
(
```

```
Field1 TipeData1 ([lebar]),
Field2 TipeData2 ([lebar]),
...
Fieldn TipeDatan ([lebar])
);
```

Dalam pembuatan tabel perlu memperhatikan hal-hal berikut ini :

1. Nama tabel tidak boleh mengandung spasi (*space*) tetapi Apabila menginginkan ada spasi harus menggunakan tanda penghubung (nama_tabel).
2. Field1 merupakan atribut pertama dan TipeData1 merupakan tipe data untuk atribut pertama.
3. Apabila ingin membuat tabel dengan atribut lebih dari satu, maka setelah pendefinisian tipe data sebelumnya diberikan tanda koma (,).

Berikut ini contoh perintah untuk membuat tabel dengan nama prodi :

```
mysql> create table prodi (
        kode_prodi varchar(5),
        nama_prodi varchar(50),
        jenjang varchar(2));
```

B. Mendefinisikan null/not null

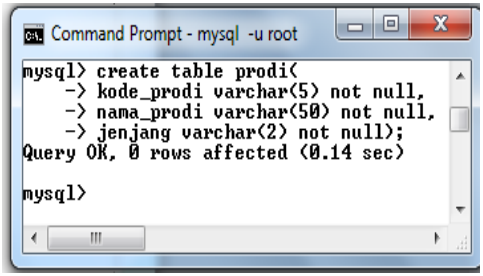
Ketika membuat tabel, beberapa atribut (field) harus diatur agar atribut (field) tertentu harus diisi oleh pengguna. Biasanya field ini adalah sebagai field utama atau kunci, juga sebagai identikasi yang bersifat unik sehingga field tersebut tidak boleh kosong. Perintahnya adalah sebagai berikut :

```
CREATE TABLE namatabel
(
Field1 TipeData1 ([lebar]) NOT NULL,
Field2 TipeData2 ([lebar]) NOT NULL,
...
Fieldn TipeDatan ([lebar])
);
```

Contoh:

```
mysql> create table prodi (
       kode_prodi varchar(5) not null,
       nama_prodi varchar(50) not null,
       jenjang varchar(2) not null);
```

Tampilan hasil :

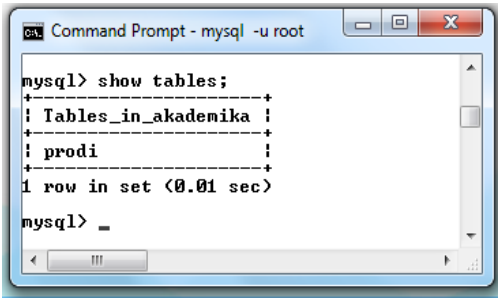


C. Melihat Daftar Tabel dalam Basis Data

untuk melihat tabel apa saja yag ada di dalam basis data aktif bisa digunakan perintah SHOW TABLES seperti contoh berikut ini :

```
Mysql> SHOW TABLES;
```

Maka akan tampil sebagai berikut :

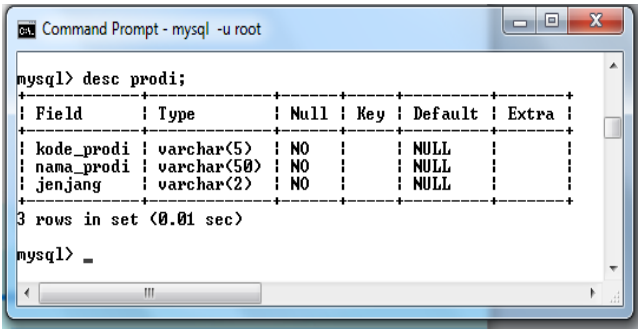


D. Melihat Struktur Tabel

Untuk melihat struktu tabel yang baru saja dibuat menggunakan perintah DESC. Misalkan ingin melihat struktur tabel prodi maka perintahnya adalah :

```
Mysql>DESC prodi;
```

Maka hasil tampilannya adalah :



E. Menentukan Kunci Utama (Primary Key) Pada Tabel

Key adalah satu gabungan dari beberapa atribut yang bisa membedakan data pada semua basis data (row) dalam tabel secara unik. Key di dalam basis data berfungsi sebagai suatu cara untuk

mengidentifikasi dan menghubungkan satu tabel dengan tabel yang lain. Penerapan kunci utama dalam tabel menggunakan perintah not null primary key. Ada beberapa cara untuk menerapkan primary key di field seperti berikut ini

1. Penerapan 1 : primary key diletakkan langsung di field yang akan menjadi kunci :

```
mysql> create table prodi (
       kode_prodi varchar(5) not null
       primary key,
       nama_prodi varchar(50) not null,
       jenjang varchar(2) not null);
```

2. Penerapan 2 : primary key ditempatkan diakhir :

```
mysql> create table prodi (
        kode_prodi varchar(5) not null,
        nama_prodi varchar(50) not null,
        jenjang varchar(2) not null,
        primary key(kode_prodi));
```

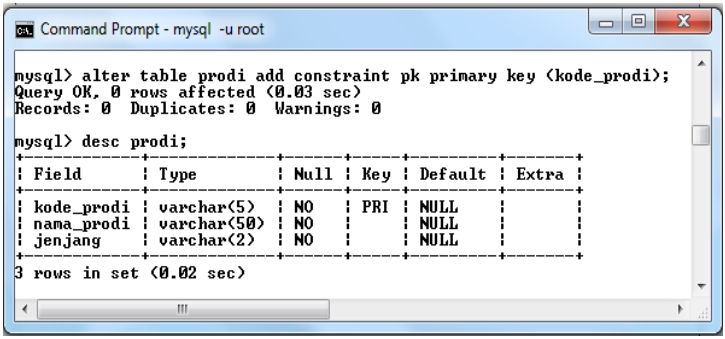
3. Penerapan 3 : dengan mengubah tabel yang sudah terbentuk :

```
mysql> create table prodi (
        kode_prodi varchar(5) not null,
        nama_prodi varchar(50) not null,
        jenjang varchar(2) not null);
```

Primary key ditambahkan setelah tabel selesai di buat menggunakan perintah ALTER TABLE :

```
Mysql> alter table prodi add constraint pk
primary key (kode_prodi);
```

Apabila diterapkan di MySQL maka hasilnya seperti dibawah ini :



F. Menghapus Primary key

Untuk menghapus primary key menggunakan perintah ALTER TABLE. Penghapusan primary key ada beberapa cara yang bisa digunakan antara lain :

1. Cara 1 : Apabila primary key dibuat dengan menggunakan alter table :

```
ALTER TABLE <nama_tabel> DROP CONSTRAINT
<nama_constraint>;
```

Cara 2 : Apabila primary key dibuat melalui create table :

```
ALTER TABLE <nama_tabel> DROP PRIMARY KEY;
```

Berikut ini perintah yang digunakan untuk menghapus primary key pada tabel prodi :

```
Mysql> alter table prodi drop primary key;
```

G. Mendefinisikan Nilai Bawaan (Default)

Nilai default adalah nilai yang otomatis diberikan oleh sistem untuk suatu atribut ketika ada penambahan baris baru, sementara nilai pada atribut tersebut tidak diisi oleh pengguna.

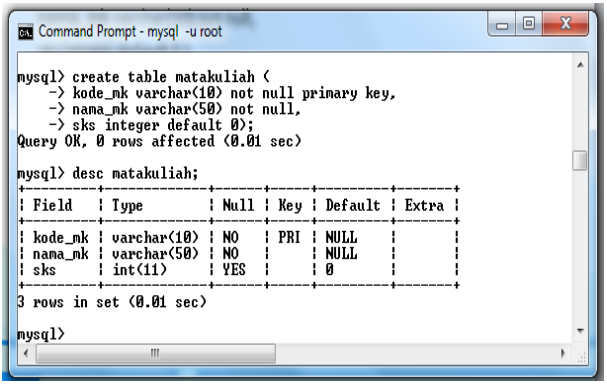
Perintah :

```
CREATE TABLE <nama_tabel>
(
Field1 TipeData1 ([lebar]),
Field2 TipeData2 DEFAULT nilai
);
```

Dimana nilai adalah nilai default dari atribut tersebut. Berikut contoh penerapan Default :

```
Mysql> create table matakuliah (
       Kode_mk varchar(10) not null primary
       key,
       nama_mk varchar(50) not null,
       sks integer default 0 );
```

Tampilan hasil :



H. Menentukan Foreign Key Pada Tabel

Foreign Key adalah satu atribut atau satu set atribut sebagai key penghubung antara dua tabel sebagai pelengkap satu relationship (hubungan). Foreign key di tabel satu akan terhubung ke primary key di tabel lain yang mempunyai tipe dan data yang sama (dari anak ke induknya). Perintah untuk menentukan foreign key pada tabel adalah sebagai berikut :

```
CREATE TABLE <nama_tabel>
(
Field1 TipeData1 ([lebar]),
Field2 TipeData2 ([lebar]),
FOREIGN KEY (Field2) REFERENCES
<nama_tabel_induk> (<nama_fiel_dinduk>)ON
UPDATE CASCADE
ON DELETE NO ACTION
)
```

atau

```
ALTER TABLE <nama_tabel> ADD CONSTRAINT
<nama_constraint> FOREIGN KEY (<nama_field>)
REFERENCES <nama_tabel_induk>
(<nama_field_induk>) ON UPDATE CASCADE ON
DELETE NO ACTION;
```

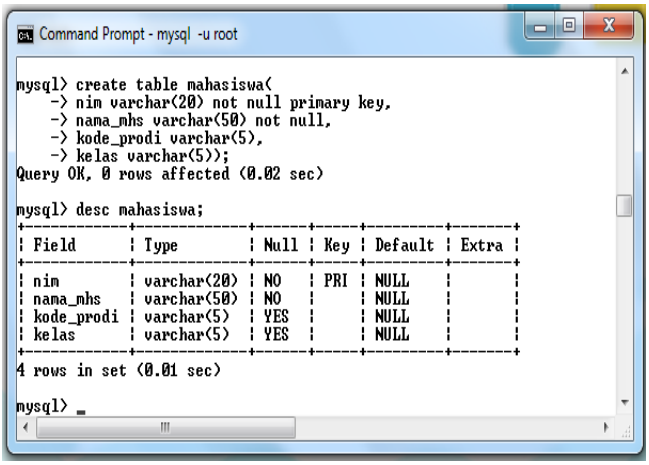
Apabila sebuah primary key terhubungan ke tabel atau entitas lain, maka keberadaan primary key pada entity tersebut di sebut sebagai foreign key. Untuk membuat foreign key, maka harus dipastikan bahwa tabel dan atribut yang dirujuk (tabel induk dari

foreign key) sudah didefinisikan terlebih dahulu. Berikut contoh penerapan foreign key :

Membuat tabel mahasiswa, dimana terdapat field sebagai primary key di tabel lain namun menjadi foreign key di tabel mahasiswa yaitu field kode_prodi :

```
Mysql> create table mahasiswa (
       nim varchar(20) not null primary key,
       nama_mhs varchar(50) not null,
       kode_prodi varchar(5),
       kelas varchar(5));
```

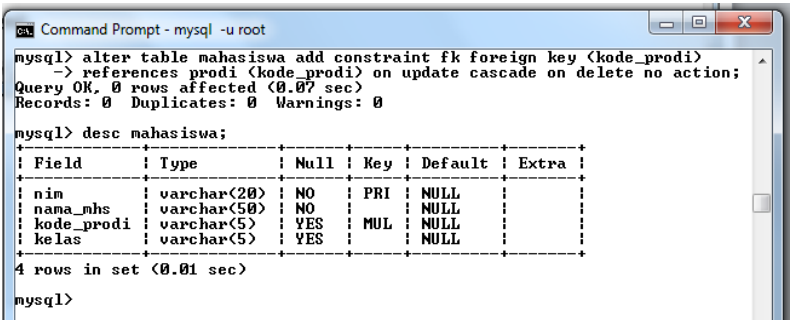
Hasil tampilan dari script diatas adalah sebagai berikut :



Penambahan Foreign key :

```
Mysql> alter table mahasiswa add constraint
fk foreign key (kode_prodi) references prodi
(kode_prodi) on update cascade on delete no
action;
```

Tampilan hasil :



I. Menghapus Foreign key

Foreign key juga bisa dihapus menggunakan perintah :

```
ALTER TABLE <nama_tabel> DROP FOREIGN KEY
<nama_constraint>;
```

Contoh :

```
Mysql> alter table mahasiswa drop foreign key
fk;
```

J. Mengubah Struktur Tabel

Tabel yang sudah dibuat bisa dilakukan perubahan strukturnya seperti penambahan atribut (field), penghapusan atribut (field) bahkan mengganti lebar field dari tabel tersebut. Perintah yang digunakan adalah ALTER TABLE.

1. Menambah Atribut Baru Pada Tabel menggunakan perintah sebagai berikut :

```
ALTER TABLE <nama_tabel> ADD <field_baru><tipe>;
```

Dimana :

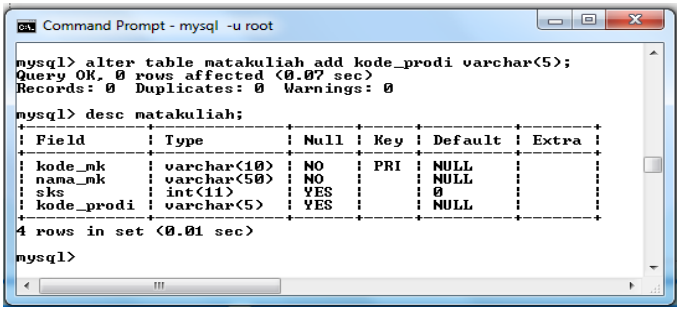
a. namatabel adalah nama tabel yang akan ditambah fieldnya.
b. Fieldbaru adalah nama atribut yang akan ditambahkan.
c. Tipe adalah tipe data dari atribut yang akan ditambahkan.

Contoh :

Berikut ini perintah untuk menambah atribut kode_prodi dengan tipe data varchar(5) ke dalam tabel matakuliah :

```
Mysql> alter table matakuliah add kode_prodi
       varchar(5);
```

Tampilan hasil script :



K. Mengubah Nama Atribut (Field) pada Tabel

Terkadang dalam perancangan tabel terdapat kesalahan pemberian nama atau ingin mengubah nama yang sudah ada maka menggunakan perintah CHANGE COLUMN seperti berikut ini :

ALTER TABLE <nama_tabel> CHANGE COLUMN <nama_lama_field> <nama_baru_field>< tipe_datanya><ukurannya>;

Dimana :

1. Namatabel adalah nama tabel yang akan diubah nama atributnya,
2. Namalamafield adalah atribut yang akan diganti namanya,
3. Namabarufield adalah nama baru atribut,
4. Tipedatanya adalah tipe data dari atribut tersebut.
5. Ukurannya adalah ukuran data dari atribut tersebut.

Berikut ini perintah untuk mengubah nama atribut kode_prodi menjadi kd_prodi :

```
mysql> alter table matakuliah change column
       kode_prodi kd_prodi char(5);
```

L. Menghapus Atribut (Field) Pada Tabel

Atribut atau field yang sudah dibuat bisa dihapus dengan menggunakan perintah DROP COLUMN akan tetapi untuk menghapus harus memperhitungkan apakah field sudah tidak penting lagi atau sudah tidak digunakan atau masih aktif digunakan karena Apabila sudah terhapus maka tidak bisa di kembalikan kecuali restore data dari backup data.

```
 ALTER TABLE <nama_tabel> DROP COLUMN
<nama_kolom>;
```

Berikut ini perintah untuk menghapus atribut kd_prodi pada tabel matakuliah :

```
Mysql> alter table matakuliah drop kd_prodi;
```

M. Menghapus Tabel

Tabel sudah di buat bisa di hapus dengan menggunakan perintah DROP TABLE. Untuk melakukan penghapusan tabel harus di pertimbangkan terlebih dahulu karena kadang data-data yang ada di tabel masih diperlukan lagi. Tidak apa-apa di hapus Apabila ada bakcup tapi Apabila tabel masih kosong boleh langsung dihapus. Perintah untuk menghapus tabel adalah sebagai berikut:

```
DROP TABLE <nama_tabel>;
```

Tabel yang akan dihapus sesuai dengan namatabel, berikut ini perintah untuk menghapus tabel matakuliah :

```
Mysql> drop table matakuliah;
```

8.4. Ringkasan

1. Data Definition Language (DDL) merupakan bahasa dasar dalam tahap awal penerapan basis data di MySQL yang digunakan untuk membuat dan mengahapus basis data, membuat, mengubah dan menghapus tabel dan view dan lain sebagainya.
2. Perintah-perintah DDL antara lain create database, drop database, create table, alter table, drop table,create index dan drop index.

8.5. Evaluasi

1. Berdasarkan hasil jawaban basis data pada bab 4 (ER Diagram) buatlah database dan tabel menggunakan perintah-perintah DDL di MySQL berbasis console.

BAB 9
DATA MANIPULATION LANGUAGE (DML)

Pada bab ini akan membahas tentang mengelola dan memanipulasi data di MySQL yaitu antara lain menambah data, melihat data, mengubah data dan menghapus data dengan menggunakan perintah *insert, select, update dan delete*.

9.1. Perintah Dasar DML

Data Manipulasi Language (DML) Data Manipulation Language (DDL) merupakan perintah-perintah yang berfungsi untuk melakukan manipulasi data-data yang ada didalam tabel. Bentuk manipulasi yang bisa dilakukan oleh DML diantaranya adalah :

1. Memasukkan data baru ke dalam tabel
2. Pengubahan data
3. Menampilkan data dengan kreiteria tertentu
4. Menampilkan data secara terurut.
5. Melakukan pencarian kembali data lama
6. Penghapusan data

Perintah-perintah tersebut adalah sebagai berikut :

Perintah	**Keterangan**

Insert	Menambahkan suatu data pada tabel
Update	Mengubah nilai pada sebuah data
Select	Melihat data yang ada di tabel
Delete	Menghapus data pada tabel

9.2. Menambah Data

Penambahan data pada tabel menggunakan perintah INSERT . Dimana aturan Penulisannya adalah sebagai berikut :

1. Apabila yang dimasukkan berupa angka maka tidak menggunakan tanda petik (‘).
2. Apabila yang dimasukkan berupa karakter dan tanggal menggunakan tanda petik (‘).

Terdapat empat cara untuk menambah baris, yaitu :

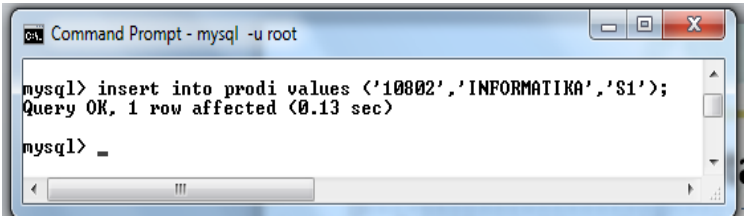
1. Menambah baris dengan mengisi data langsung pada tiap-tiap kolom tanpa menyertakan struktur tabel.

```
INSERT INTO <nama_tabel> VALUES (nilai1,
nilai2,nilai-n);
```

Contoh :

```
Mysql> insert into prodi values (’10802’,
       ’INFORMATIKA’,’S1’);
```

Tampilan Hasil :



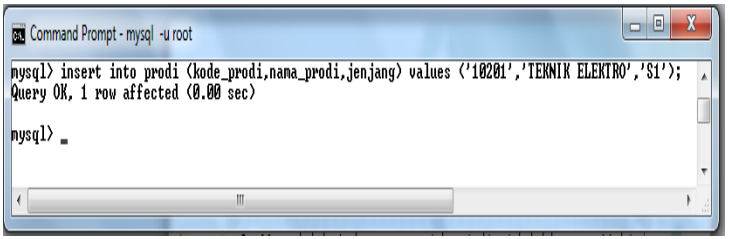
2. Menambah baris dengan menyertakan struktur tabel dalam mengisi data pada tiap-tiap kolom.

```
INSERT INTO <nama_tabel> (kolom1,kolom2,
kolom-n) VALUES (nilai1,nilai2,nilai-n);
```

Contoh penerapannya sebagai berikut :

```
Mysql> insert into prodi (kode_prodi,
      nama_prodi,jenjang) values ('10201',
      'TEKNIK ELEKTRO','S1');
```

Hasil tampilan :



3. Menambah data lebih dari satu

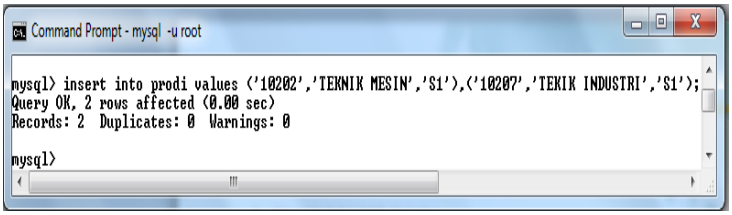
Apabila ingin langsung memasukkan 2 baris data atau lebih dalam satu perintah INSERT MySQL, tinggal menambahkan isi data untuk baris berikutnya dibelakang perintah dengan format penulisan sebagai berikut:

```
   INSERT INTO nama_tabel VALUES
(nilai_kolom1a
   , nilai_kolom2a,...), (nilai_kolom1b,
   nilai_kolom2b,...);
```

Contoh penerapannya sebagai berikut :

```
Mysql> INSERT INTO prodi values ('10202',
'TEKNIK MESIN','S1'),('10207','TEKNIK
INDUSTRI','S1');
```

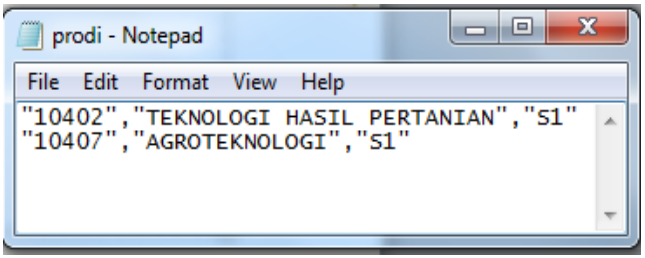
Tampilan Hasil script sebagai berikut :



4. Menambah data dari file .txt

   Menambah data ke tabel juga bisa dilakukan dengan menggunakan file yaitu berkestensi .txt bisa menggunakan notepda atau notepad++. Langkah-langkah penambahan melalui file sebagai berikut :

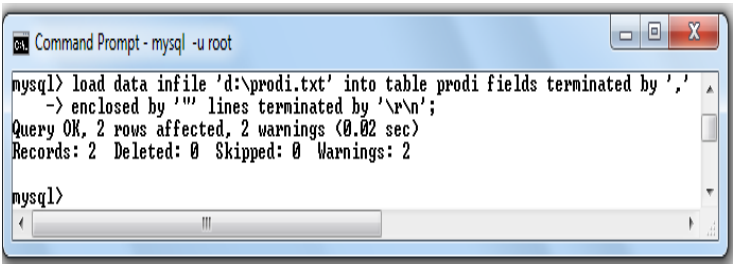
   a. Siapkan file .txt yang akan d tambahkan atau di masukkan ke tabel.



b. Simpan dengan nama sesuai dengan tabel yang akan di tambahkan datanya atau dengan nama yang lain.

c. Ketikkan perintah :

LOAD DATA INFILE 'nama_file.txt' INTO TABLE nama_tabel_yang_dituju FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINE TERMINATED BY '\r\n';



9.3. Menampilkan Data

Untuk menampilkan isi data atau melihat data dari suatu tabel terutama untuk memastikan apakah data yang di tambahkan sudah berhasil masuk ke tabel menggunakan perintah SELECT. Ada Beberapa cara untuk menampilan antara lain :

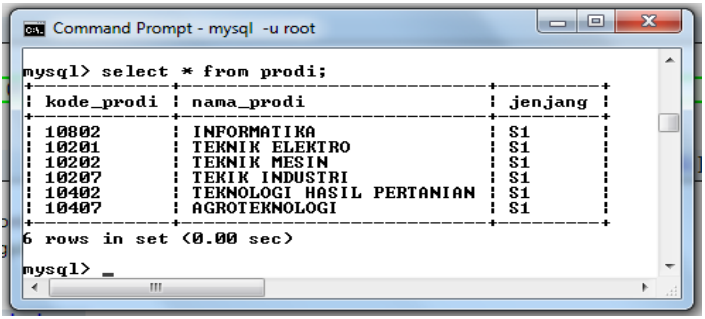
1. Menampilkan data untuk semua kolom menggunakan asterisk (*) perintahnya sebagai berikut :

```
SELECT * FROM <nama_tabel>;
```

Contoh :

```
Mysql> select * from prodi;
```

Tampilan Hasil script :



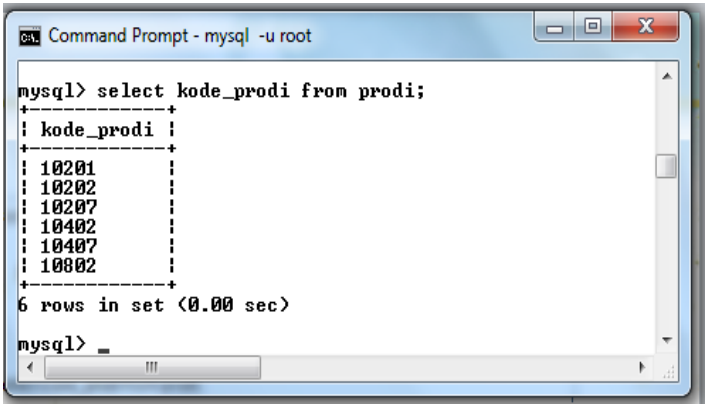
2. Menampilkan data untuk kolom tertentu dengan menyebutkan nama-nama kolom yang akan ditampilkan. Perintahnya sebagai berikut :

```
SELECT kolom1,kolom2,kolom-n FROM
<nama_tabel>;
```

Contoh :

```
Mysql> select kode_prodi from prodi;
```

Tampilan hasil :



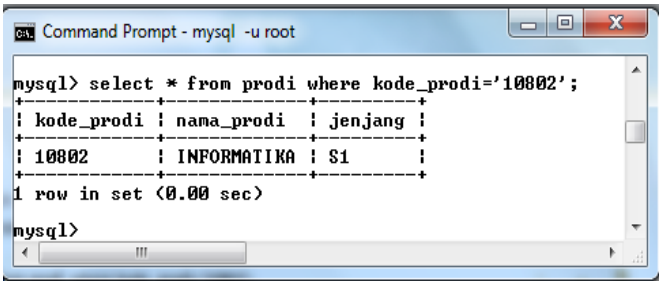
3. Menampilkan data dengan kondisi data tertentu menggunakan klausa WHERE. Perintanya adalah :

```
SELECT * FROM <nama_tabel> WHERE
<kondisi>;
```

Contoh :
perintah untuk menampilkan data pada tabel prodi dimana kode prodi adalah 10802

```
Select * from prodi where kode_prodi='10802';
```

Tampilan hasil script :



9.4. Mengubah Data

Untuk mengubah isi data pada satu atau beberapa kolom pada suatu tabel digunakan perintah UPDATE. Perintah update adalah sebagai berikut :

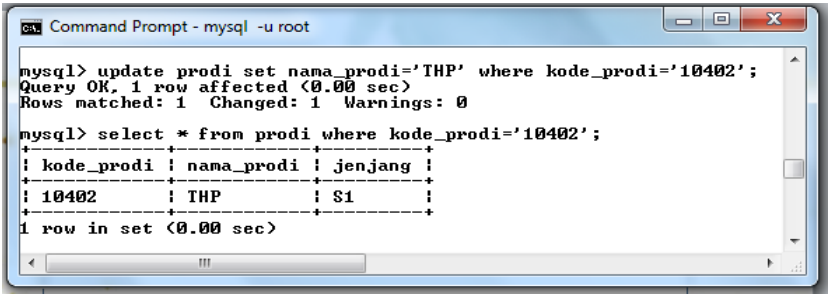
```
UPDATE namatabel SET kolom1 = nilai1, kolom2
= nilai2 [WHERE kondisi];
```

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

Contoh :

```
Update prodi set nama_prodi='THP' WHERE kode_prodi='10402';
```

Tampilan hasil :



9.5. Menghapus data

Data yag sudah tidak digunakan bisa dilakukan penghapusan untuk mengurangi beban penyimpanan basis data yaitu dengan menggunakan perintah DELETE. Perintah ini digunakan untuk menghapus satu baris, baris dengan kondisi tertentu atau seluruh baris. Ada dua cara penghapusan data yaitu

1. Menghapus keseluruhan data di tabel

   ```
   DELETE FROM <nama_tabel>;
   ```

   Dimana <nama_tabel> adalah nama tabel yang akan dihapus datanya. Contoh Apabila ingin menghapus seluruh baris pada tabel prodi :

   ```
   Mysql> delete from prodi;
   ```

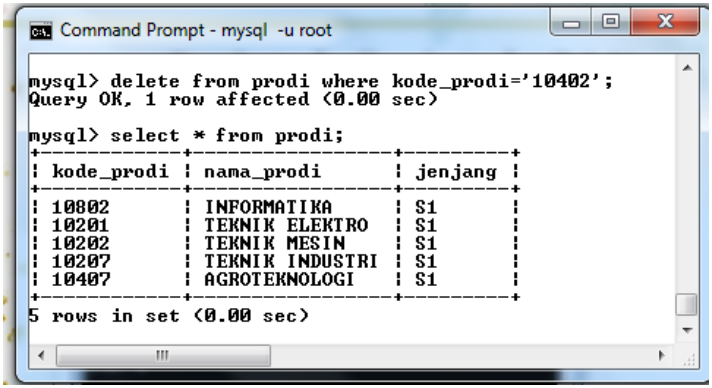
2. Menghapus data berdasarkan kondisi tertentu bisa menggunaka perintah :

   ```
   DELETE FROM <nama_tabel> WHERE <kondisi>;
   ```

Perintah menggunakan klausa WHERE bersifat opsional untuk menghapus suatu baris dengan suatu kondisi tertentu. Contoh :

```
Mysql> delete from prodi where kode_prodi='10402';
```

Tampilan Hasil :



9.6. Mendalami Perintah SELECT

Perintah SELECT tidak hanya disertai klausa FROM dan WHERE saja tetapi SELECT juga mempunyai banyak klausa yang memudahkan pegguna dalam memanipulasi dan mengolah data di basis data.

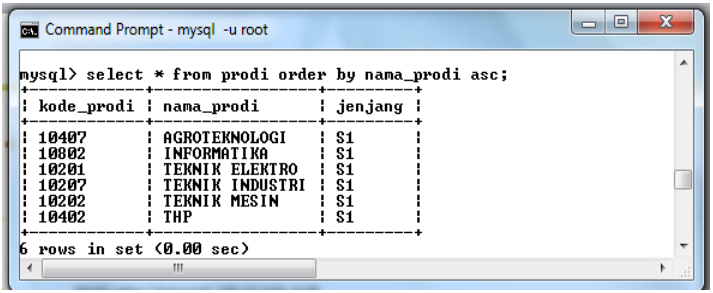
A. Mengurutkan Data

Untuk mengurutkan data berdasarkan kolom tertentu sesuai dengan tipe data yang dimiliki bisa menggunakan perintah SELECT disertai ORDER BY. Pengurutan bisa dilakukan secara naik (ascending) dari A – Z maupun turun (descending) dari Z- A. Pengurutan juga bisa dilakukan berdasarkan sebuah field atau beberapa field gabungan. Contoh perintah untuk mengurutkan data prodi berdasarkan kolom tertentu misal nama_prodi :

```
Mysql> select * from prodi order by nama_prodi;
```

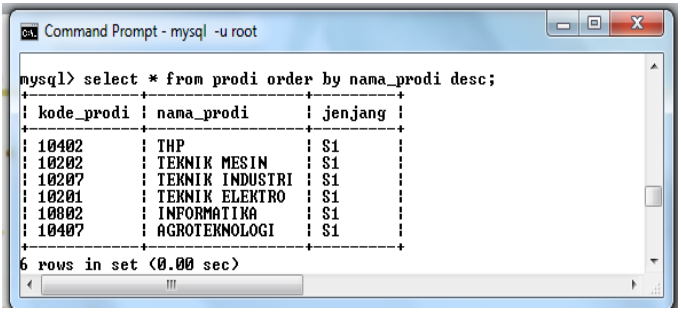
Untuk mengurutkan data secara ascending (menaik) bisa ditambahkan ASC sebagai berikut :

```
Mysql> select * from prodi order by nama_prodi asc;
```



atau tambahkan DESC untuk pengurutan secara descending (menurun) seperti contoh berikut ini :

```
Mysql> select * from prodi order by
nama_prodi
      desc;
```



B. Mengelompokkan Data

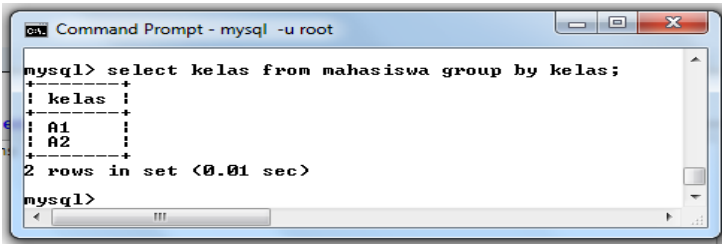
Proses pengelompokkan di sini adalah mengelompokkan beberapa nilai mejadi terwakili oleh satu nilai saja. Sehingga data yang mempunyai nilai agreagat (misalkan nama mahasiswa) tidak

bisa dijadikan acuan dalam klausa GROUP BY karena mempunyai nilai yng kurang spesifik. Nilai yang akan dijadikan acuan haruslah nilai yang bersifat spesifik dan tunggal. Perintahnya :

```
select <nama_field> from <nama_tabel> group by
<nama_field_kelompok>;
```

Nama_field_kelompok biasanya disesuaikan dengan nama_field Contoh Pengelompokan data berdasarkan kelas.

```
Mysql>select kelas from mahasiswa group by kelas;
```



C. Menggunakan Klausa Having

Klausa HAVING digunakan berpasangan dengan klausa GROUP BY. Kegunaannya adalah untuk menentukan kondisi dalam GROUP BY dan hanya kelompok yang memenuhi HAVING saja yang dihasilkan. Misalkan :

```
Myql > select nim,nama_mhs,kode_prodi,
kelas
       From Mahasiswa group by kode_prodi
        having kode_prodi='10201';
```

Tampilan Hasil :

D. Menggunakan Klausa Limit

Klausa ini digunakan untuk untuk menentukan jumlah data yang akan ditampilkan pada layar. Perinta klausa LIMIT adalah sebagai berikut :

```
SELECT <nama_kolom> FROM <nama_tabel> LIMIT <batasan>,<jumlah>;
```

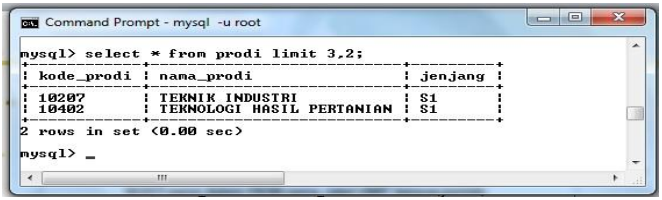
Dimana :

1. Batasan merupakan nomor recordset awal yang akan diseleksi
2. Jumlah merupakan banyak data yang akan ditampilkan mulai dari nomor batasan

Contoh penerapannya sebagai berikut, misalkan akan menampilkan data mulai dari nomer 3 ke atas sejumlah 2 data maka perintahnya adalah :

```
Mysql > select * from prodi limit 3,2;
```

Maka hasilnya sebagai berikut :



E. Menggunakan Klausa AS

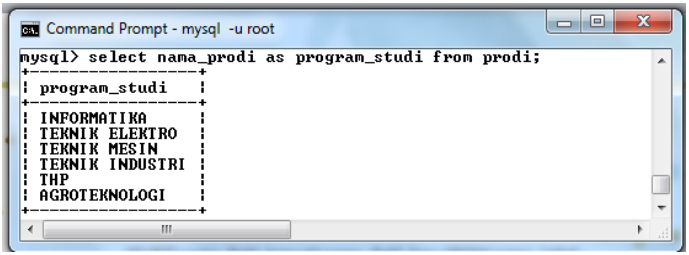
Perintah select untuk Memberikan nama lain pada kolom (field) di tabel. Biasanya digunakan untuk memperjelas nama tabel sehingga pengguna paham field tersebut menampilkan data apa. Perintah klausa ini sebagai berikut :

```
SELECT <nama_field_lama> AS <nama_field_baru>
FROM <nama_tabel>;
```

Contoh peerapan klausa AS :

```
Mysql> select nama_prodi as program_studi from
       prodi;
```

Maka hasilnya sebagai berikut :



F. Menggunakan Alias pada Nama Tabel

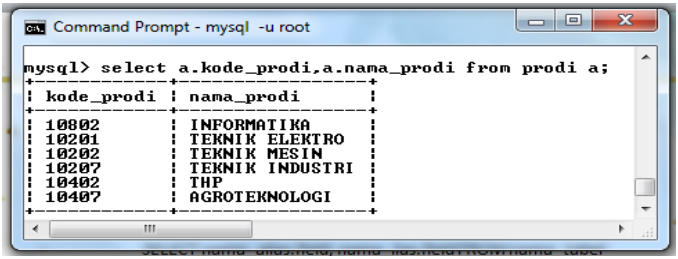
Penamaan tabel juga bisa diubah ke nama yang lain (perumpamaan) dikenal dengan alias tabel. Perumpaan nama tabel biasanya di terapkan saat menggunakan join query yang bertujuan mempersingkat dan mempermudah dalam membuat join query. Perintah alias sebagai berikut :

```
SELECT <nama_alias.field, nama_lias.field>
FROM <nama_tabel nama_alias>;
```

Contoh penerapannya :

```
Msql > select a.kode_prodi,a.nama_prodi from
       prodi a;
```

Maka tampilan hasilnya sebagai berikut :



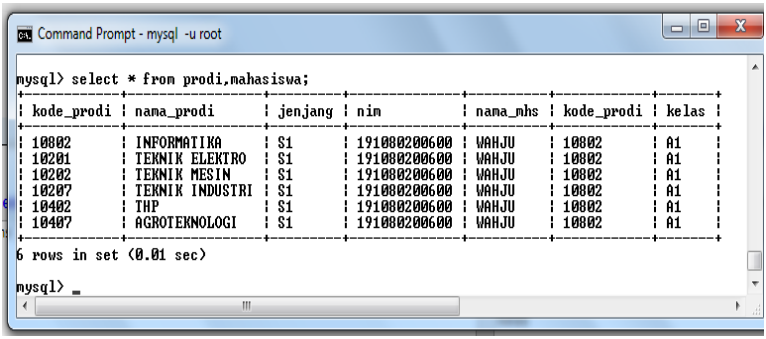
G. Menampilkan Data Lebih Dari satu tabel

Menampilkan data dengan menggunakan perintah SELECT tidak hanya untuk satu tabel saja tetapi bisa beberapa tabel tergantung kebutuhan pengguna. Tapi perlu diingat Apabila akan menampilkan semua data pada seua tabel maka tampilan hasil da layar monitor tidak akan rapi (membingungkan) sehingga lebih disarankan menampilan data sesuai dengan kebutuhan saja. Perintah untuk menampilkan data lebih dari satu tabel adalah sebagai berikut :

```
SELECT * from <nama_tabel1>,<nama_tabel2>,
<nama_tabel-n>;
```

Contoh Penerapan sebagai berikut :

```
Mysql> select * from prodi,mahasiswa;
```



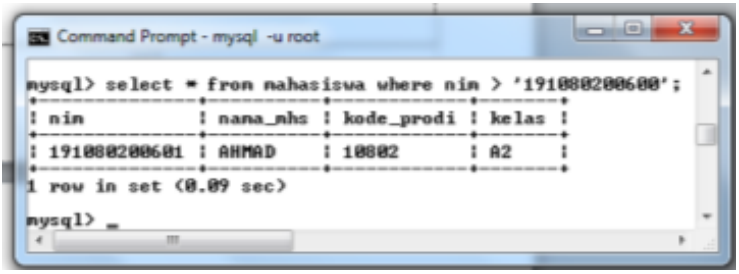
9.7. Operator Relasi Di MySQL

Operator Relasi merupakan operator yang digunakan untuk membandingkan suatu nilai dengan nilai yang lain. Biasanya operator ini digunakan bersamaan dengan oprator logika dalam membantu menampilkan informasi dengan kriteria tertentu.

Operator	Keterangan
=	Sama dengan
>	Lebih besar
<	Lebih kecil
>=	Lebih besar atau sama dengan
<=	Lebih kecil atau sama dengan
<>	Tidak sama dengan

Contoh penerapannya yaitu menampilkan data mahasiswa yang nim lebih besar '191080200600'. Maka hasil tampak sebagai berikut :

Mysql > select * from mahasiswa where nim > '191080200600';



9.8. Operator Logika Di MySQL

Dalam menampilkan data menggunakan SELECT bisa di gabung dengan operator logika dimana oprator ini terdiri atas tiga macam yaitu OR, AND dan NOT.

Operator	Fungsi	Keterangan
OR atau ||	Atau	Contoh : Hitam OR Putih Lulus || Gagal
AND atau &&	Dan	Contoh : Teori AND Praktik Teori && Praktik
NOT atau !	Sebagai negasi atau pembalik nilai	NOT kerja atau !kerja (artinya bekerja)

Masing-masing operator logika dijelaskan sebagai berikut dan diserta contoh dalam MySQL.

1. Operator OR
   Operator OR merupakan operator logika dimana ada 2 kondisi Apabila salah satu terpenuhi atau bernilai benar maka hasilnya adalah benar. Sintaks dari operator OR adalah :

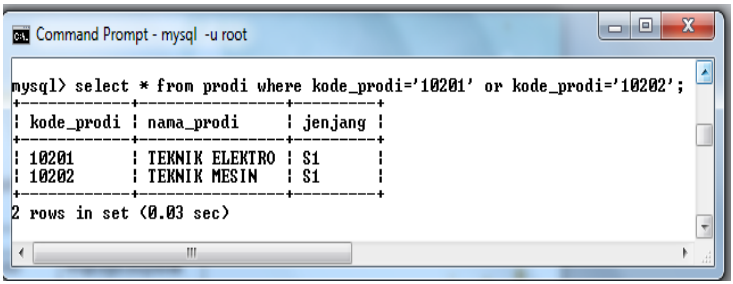
   Kondisi1 OR kondisi2

Kondisi1	Kondisi2	Hasil
False	False	False
False	True	True
True	False	True
True	True	True

   Berikut ini contoh penerapan OR :

```
   Mysql>select * from prodi where
kode_prodi='
         10201' Or kode_prodi='10202';
```

   Tampilan hasil :



2. Operator AND
   Operator AND mempunyai ciri-ciri sebagai berikut :

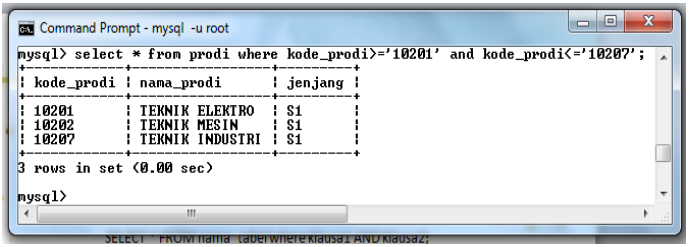
a. Apapun kondisinya bila di-AND-kan dengan nilai yang salah hasilnya akan tetap salah.
b. Hasil akan bernilai benar apabila kedua kondisi bernilai benar.

Kondisi1	Kondisi2	Hasil
False	False	False
False	True	False
True	False	False
True	True	True

Contoh :
perintah untuk menampilkan data pada prodi dimana kode prodi mulai ‘10201’ sampai ‘10207’ :

```
mysql>select * from prodi where
kode_prodi=’
      10201’ and kode_prodi=’10207’;
```

Tampilan Hasil :



3. Operator NOT
Operator ini berguna untuk melakukan pembalikan atau lawan dari nilai logika. Perintahnya adalah

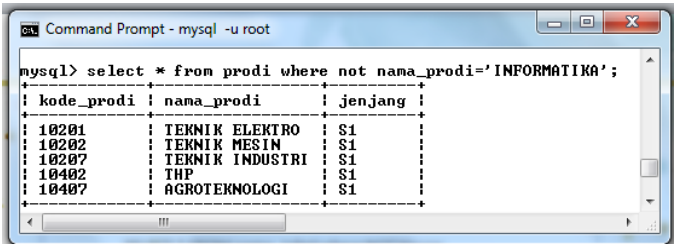
NOT Kondisi

Kondisi	Hasil
Not True	False
Not False	False

Contoh :
Perintah untuk menampilkan data pada tabel prodi dimana nilai pada kolom nama prodi tidak sama dengan INFORMATIKA.

```
Mysql>select * from prodi where not
      nama_prodi='INFORMATIKA';
```

Tampilan Hasil :



9.9. Operator Pembanding Di MySQL

Operator ini digunakan untuk menampilkan data dengan kondisi tertentu. Berikut ini beberapa klausa yang termasuk dalam operator pembanding.

Operator	Keterangan
IS NOT NULL	Menampilkan data tidak kosong
IS NULL	Menampilkan data koson

BETWEEN	Menampilkan data antara kondisi1 dan kondisi2
IN	Menampilkan data yang berada dalam pilihan yang ada
NOT IN	Menampilkan data yang tidak berada dalam pilihan yang ada
LIKE	Menampilkan data sesuai dengan kriteria
NOT LIKE	Menampilkan data yang tidak sesuai dengan kriteria

Operator pembanding biasanya disesuaikan dengan tampilan data yang diinginkan. Berikut ini contoh penerapan oparator pembanding :

1. Operator Null dan Is Not Null

   NULL adalah suatu tanda khusus bernilai NULL yang artinya belum berisi data. Null bukanlah 0 (nol), Null juga berbeda dari karakter kosong (‘ ’). Secara standart field ini bertipe numerik selalu bersifat NULL. Sebaliknya, bila dinyatakan dengan NOT NULL maka field tersebut harus mempunyai nilai. Contoh Penerapannya :

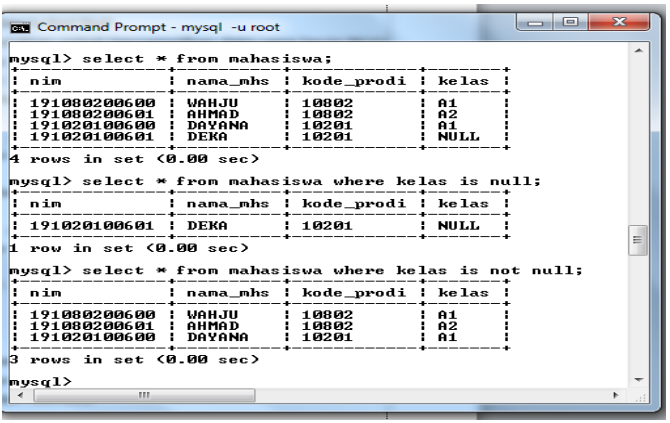
   a. Menampilkan data mahasiswa yang kelasnya NULL :

   ```
   Mysql> select * from mahasiswa where
   kelas is null;
   ```

   b. Menampilkan data mahasiswa yang kelasnya NOT NULL:

   ```
   Mysql> select * from mahasiswa where
   kelas is not null;
   ```

   Tampilan Hasil :

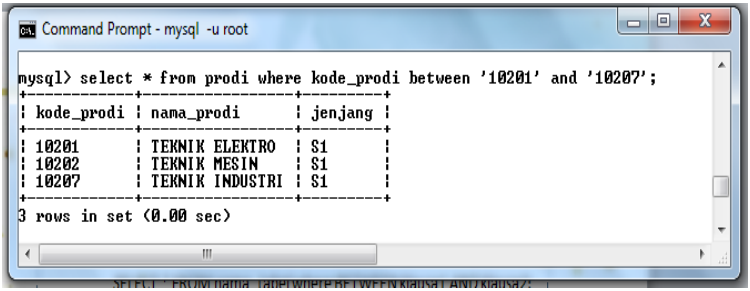


2. Operator Between

   Operator ini digunakan untuk Menampilkan data antara kondisi1 dan kondisi2. Contoh penerapannya sebagai beriku : perintah untuk menampilkan data pada tabel prodi dimana kode prodi antara 10201 dan 10207 :

```
mysql>select * from prodi where kode_prodi
between ‘10202’and ‘10207’;
```

   Tampilan hasil :

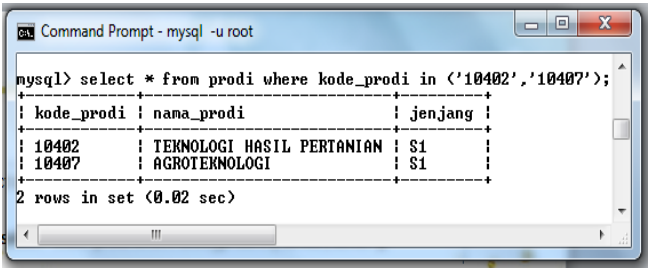


3. Operator In dan Not In

Operator In digunakan untuk menampilkan suatu data yang berada di dalam pilihan yang ada. Data yang ditampilkan hanya yang di cantumkan di dalam IN. Sedangkan NOT IN berarti data yang tidak ada di dalam pilihan. Contoh penggunaannya sebagai berikut :

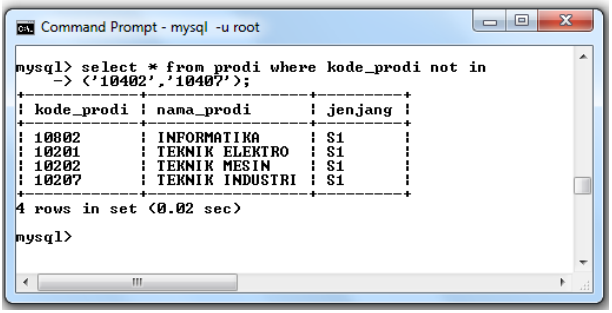
```
mysql> select * from  prodi where kode_prodi
    in ('10402','10407');
```

Tampilan hasil :




```
mysql> select * from  prodi where kode_prodi
    not in ('10402','10407');
```

Tampilan hasil :



4. Operator Like dan Not Like

   Operator LIKE digunakan untuk menampilkan data berdasarkan kondisi (*pattern*) yang di butuhkan. Pada operator ini data yang ditampilkan berdasarkan pada kriteria tertentu.

```
SELECT * FROM <nama_table> WHERE
<nama_kolom> LIKE <pattern> [ESCAPE
'escape_character'];
```

   Dimana :

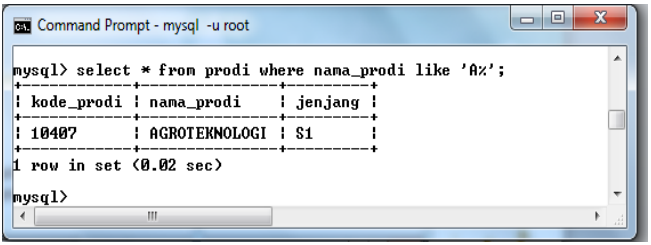
   a. Pattern atau pola adalah sebuah ekspresi karakter yang sesuai (matching) dengan yang diinginkan. Pattern ini bersifat mandatory (wajib ada).
   b. ESCAPE digunakan untuk escape character (karakter yang akan dicari). ESCAPE ini bersifat opsional.

   Pola (*pattern*) yang digunakan pada operator LIKE yaitu **%** (percentage) wildcard digunakan untuk pengecekan string yang sesuai (matching). Ada tiga jenis pattern yaitu '%ab%', '%ab', 'ab%'. Peggunaan percentage wildcard disesuaikan dengan kebutuhan data biasanya klausa ini digunakan untuk pencarian data. Berikut ini contoh penerapannya :

   1. Menampilkan nama prodi yang mempunyai nama dengan huruf pertama huruf '**A**‘.

```
Mysql> select * from prodi where
nama_prodi
       like ‘A%’;
```

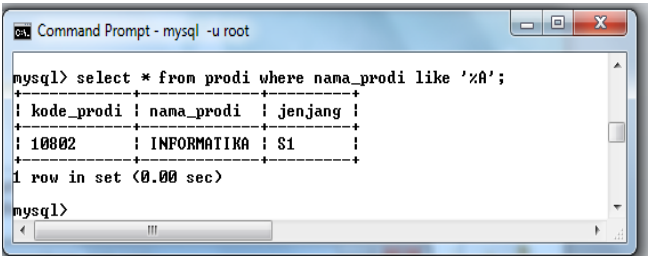
      Maka hasilnya sebagai berikut :



2. Menampilkan nama prodi yang mempunyai nama dengan huruf terakhir huruf '**A**'.

```
Mysql> select * from prodi where
nama_prodi like '%A';
```

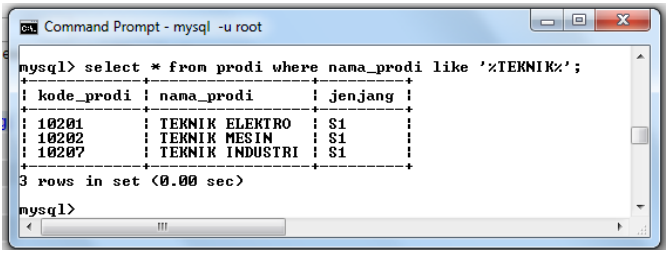
Maka hasilnya sebagai berikut :



3. Menampilkan nama prodi yang mempunyai nama sesuai dengan pencarian.

```
Mysql> select * from prodi where
nama_prodi like '%TEKNIK%';
```

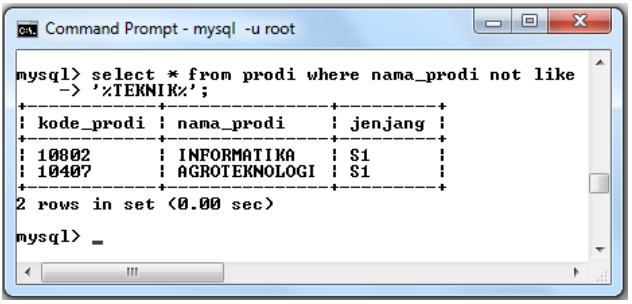
Maka hasilnya sebagai berikut :



Sedangkan operator NOT LIKE merupakan negasi atau kebalikan dari LIKE. Jadi menampilkan data selain yang dicari. Contoh penerapannya :

```
Mysql> select * from prodi where
nama_prodi Not like '%TEKNIK%';
```

Tampilan hasil :



9.10. Ringkasan

1. Data Manipulasi Language (DML) Data Manipulation Language (DDL) merupakan perintah-perintah yang berfungsi untuk melakukan manipulasi data-data yang ada didalam tabel.

2. Perintah-perintah DML antara lain insert, update, select dan delete.

9.11. Evaluasi

1. Berdasarkan database dan tabel yang sudah dibuat di bab sebelumnya, isikan data pada setiap tabel minimal 5.
2. Terapkan juga perintah DML update, select dan delete pada database tersebut.
3. Berilah contoh penggunaan klausa where, order by, group by, having, limit dan alias.
4. Berilah contoh penggunaaan logika AND, OR dan NOT
5. Berilah contoh peggunaan klausa LIKE dan NOT LIKE.

BAB 10
FUNGSI DI MYSQL

Pada bab ini akan membahasa tentag fungsi-fungsi yang ada di MySQL seperti fungsi sistem, fungsi agreagat dan fungsi string.

10.1. Definisi Fungsi

Fungsi adalah suatu rutin khusus yang disediakan oleh MySQL untuk melakukan manipulasi suatu data. Dengan fungsi memungkinkan pengguna melakukan pemrosesan terhadap data di dala server dengan 'server-side-processing'. Data langsung diproses dalam server dan hasilnya langsug di kirim ke host. Salah satu MySQL ceta dikenal dan 'user-friendly' adalah banyak dukungan fungsi build-in yang dimilikinya. MySQL sangat memanjakan penggunanya dengan banyak fungsi.

Fungsi terdiri atas nama fungsi dan diikuti parameter (*argument*) yang diapit tanda kurung. Berikut ini sintaks umum fungsi di MySQL :

```
Namafungsi([argumen1,[argument2,[....]]])
```

Fungsi digunakan sebagai bagian dari perintah SELEct (query). Secara umum fungsi yang digunakan dalam perinttah SELECT sintkasnya adalah sebagai berikut :

```
SELECT fungsi(ekspresi) [FROM namatabel];
```

FROM namatabel bersifat pilihan. Fungsi dalam MySQL bisa ditulis dalam bentuk huruf kecil atau huruf besar.

10.2. Fungsi Sistem

Fungsi sistem merupakan fungsi yang digunakan untuk memberikan informasi tentang server basis data kepada pengguna. Fungsi digunakan bersama SELECT. Fungsi-fungsi sistem adalah sebagai berikut :

1. Database()
   Fungsi ini digunakan memberikan informasi tentang server basis data pada pengguna.

   ```
   Mysql> select database();
   ```

2. Version()
   Fungsi ini digunakan untuk memberikan informasi tentang versi server yang digunakan pada pengguna.

   ```
   Mysql> select version();
   ```

3. Last_insert_id()
   Digunakan untuk menampilkan informasi data yang telah dihasilkan oleh MySQL pada field bertipe AUTO_INCREMENT.

   ```
   Mysql> select last_insert_id();
   ```

4. Session_user()
   Fungsi yang digunakan untuk menampilkan informasi pengguna yang sedang melakukan akses ke server basis data.

```
Mysql> select session_user();
```

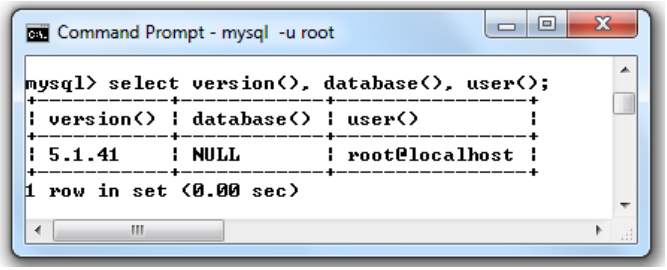
5. System_user()
   Fungsi yang digunakan untuk menampilkan informasi pengguna yang sedang melakukan akses ke server basis data.

```
Mysql> select system_user();
```

6. User()
   Fungsi yang digunakan untuk menampilkan informasi pengguna yang sedang melakukan akses ke server basis data.

```
Mysql> select user();
```

Berikut ini contoh penerapannya dalam basis data :



10.3. Fungsi Agregat

Fungsi agegrat merupakan fungsi standar SQL, yang digunakan untuk melakukan ringkasan, perhitungan statistik atau query pada suatu tabel. Fungsi agegrat adalah fungsi built-in yang selalu ada dalam tiap-tiap DBMS. Fungsi agegrat berbeda dengan fungsi aritmatika. Fungsi-fungsi Agegrat adalah sebagai berikut :

1. SUM(ekspresi)

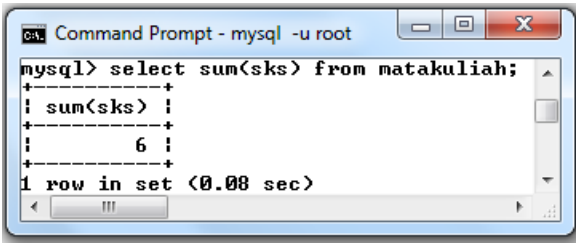
Fungsi ini digunakan untuk menbisakan nilai total dari suatu field. Berikut ini sintaks fungsi SUM :

```
select sum(nama_field) from nama_tabel;
```

Contoh menampilkan jumlah semua sks di tabel mata kuliah:

```
Mysql>select sum(sks) from matakuliah;
```

Tampilan hasil :



2. AVG(ekspresi)

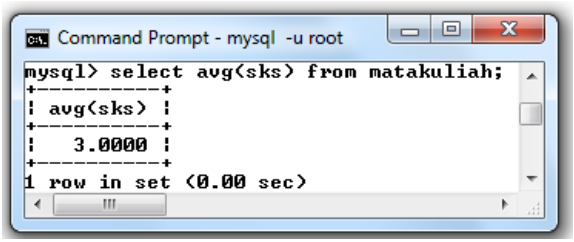
   Fungsi yang digunakan untuk mencari nilai rata-rata pada suatu field bertipe numerik atau integer. Nilai yang berisi NULL tidak dihitung oleh fungsi ini. Berikut ini sintaks fungsi AVG :

   ```
   select avg(nama_field) from nama_tabel;
   ```

   contoh :

   ```
   mysql>select avg(sks) from matakuliah;
   ```

   tampilan hasil :



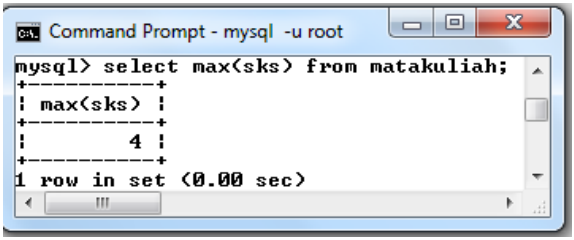
3. MAX(ekspresi)
   Fungsi yang digunakan untuk mencari nilai terbesar dari suatu field. Field yang dicari nilainya harus bertipe numerik atau integer. Berikut ini sintaks fungsi MAX :

```
select max(nama_field) from nama_tabel;
```

   Contoh :

```
Mysql>select max(sks) from matakuliah;
```

   Tampilan hasil :



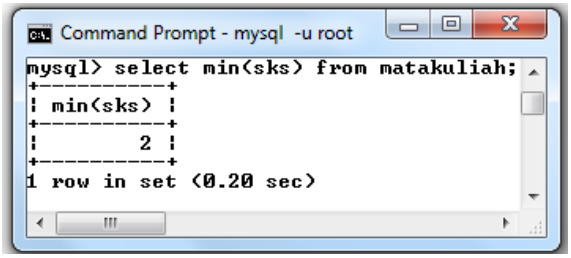
4. MIN(ekspresi)
   Fungsi yang digunakan untuk mencari nilai terkecil dari suatu field. Field yang dicari nilainya harus bertipe numerik atau integer. Berikut ini sintaks fungsi MIN :

```
select min(nama_field) from nama_tabel;
```

Contoh :

```
Mysql>select min(sks) from matakuliah;
```

Tampilan hasil :



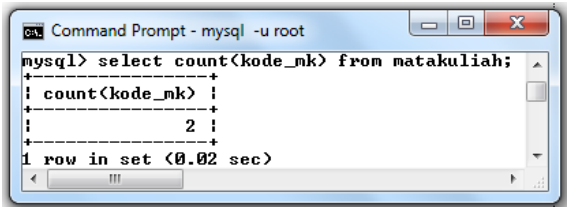
5. COUNT(x)
   Fungsi COUNT digunakan untuk menghitung jumlah record dari satu field atau tabel. Dimana x adaah nama field yang ingin dicari jumlah recordnya. Berikut ini sintaks fungsi COUNT :

```
select count(nama_field) from nama_tabel;
```

contoh :

mysql>select count(kode_mk) from matakuliah;

Tampilan hasil :



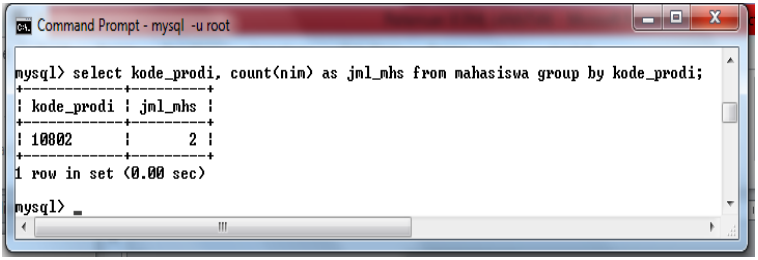
Fungsi COUNT juga bisa digabungkan dengan klausa GROUP BY untuk mencari jumlah data berdasarkan kelompok tertentu seperti sintaks berikut ini :

```
select nama_field, fungsi agregate from nama_tabel group by nama_field_kelompok;
```

contoh :

```
mysql> select kode_prodi, count(nim) as jml_mhs from mahasiswa group by kode_prodi;
```

Tampilan hasil :



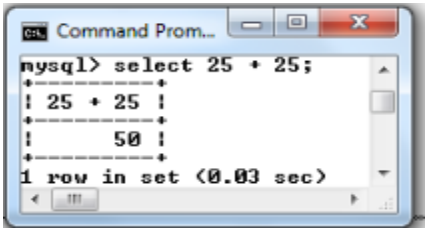
10.4. Fungsi Aritmatika

Fungsi ini untuk melakukan operasi perhitungan data bertipe numerik atau integer. Operasi aritmatika meliputi perkalian,

pembagian, penjumlahan dan pengurangan. Untuk mencoba fungsi ini bisa menggunakan klausa SELECT.

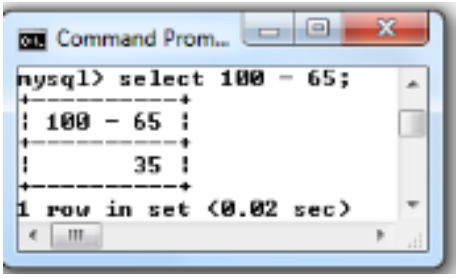
1. Penjumlahan (+);

   Digunakan untuk menjumlah dua nilai atau lebih suatu nilai bertipe numerik atau integer dan hasil perhitungan bertipe numerik atau integer. Contohnya sebagai berikut :



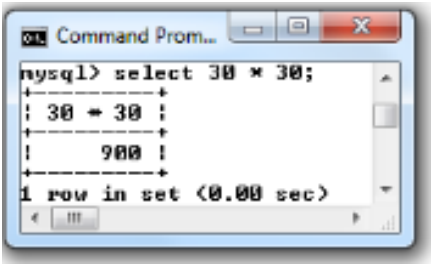
2. Pengurangan (-);

   Digunakan untuk mengurangi dua nilai atau lebih suatu nilai bertipe numerik atau integer. Hasil dari perhitungan juga bertipe numerik atau integer. Contoh penerapannya sebagai berikut :

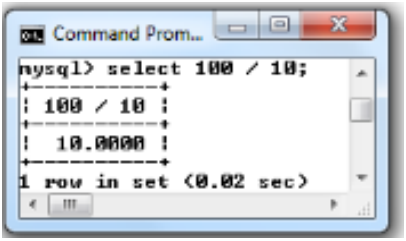


3. Perkalian (*);

   Digunakan untuk melakukan operasi perkalian dua nilai atau lebih suatu nilai bertipe numerik atau integer. Hasil dari perhitungan juga bertipe numerik atau integer. Contoh penerapannya sebagai berikut :



4. Pembagian (\);
   Digunakan untuk melakukan operasi pembagian dua nilai atau lebih suatu nilai bertipe numerik atau integer. Hasil dari perhitungan juga bertipe numerik atau integer. Contoh penerapannya sebagai berikut :

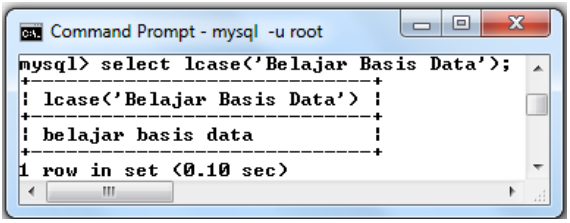


10.5. Fungsi String

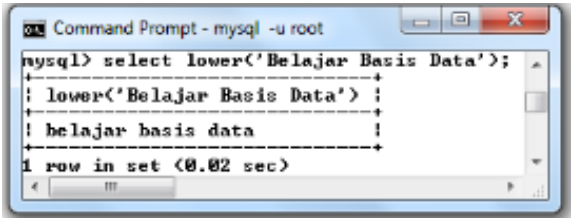
Fungsi string digunakan untuk melakukan manipulasi data teks (string). MySQL menyediakan banyak fungi bult-in untuk melakukan manipulasi teks. Berikut beberapa fungsi string yang disediakan oleh MySQL.

1. LCASE(x) atau LOWER()
   Fungsi ini digunakan untuk mengkonversi semua karakter dari nilai x ke huruf kecil semua. Contoh penerapan sebagai berikut :

```
Mysql> select lcase('Belajar Basis Data');
```



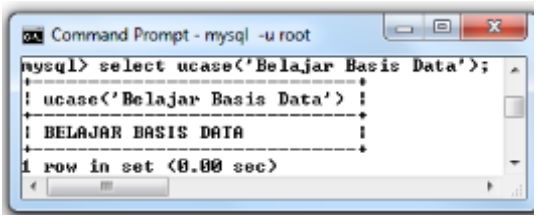

```
Mysql> select lower('Belajar basis Data');
```



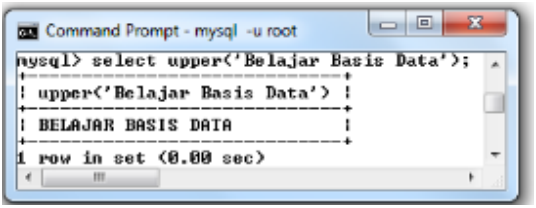
2. UCASE(x) atau UPPER()

   Fungsi ini digunakan untuk melakukan konversi suatu string sehingga karakter dalam string ditulis dalam huruf kapital semua. Contoh penerapan sebagai berikut :

```
Mysql> select ucase('Belajar Basis Data');
```




```
Mysql> select upper('Belajar Basis Data');
```



10.6. Fungsi Tanggal

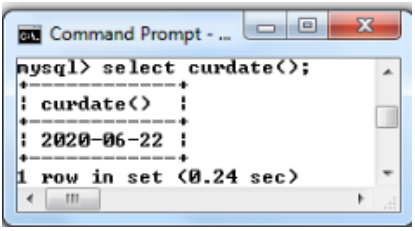
Tanggal pada MySQL menggunakan standar sistem operasi UNIX. Format tanggal dan jam menggunakan format : tahun, bulan, tanggal, jam, menit, detik. MySQL mempunyai fungsi-fungsi yang berkaitan dengan manipulasi tanggal dan jam sebagai berikut :

1. CURDATE() dan CURRENT_DATE()

   Fungsi yang digunakan untuk menampilkan tanggal sekarang dari sistem. Contoh penerapannya sebagai berikut :

   ```
   Mysql> select curdate();
   ```

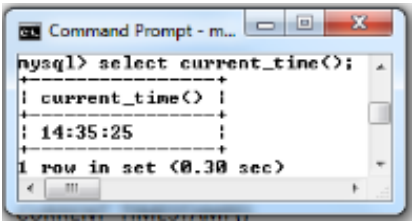
   Tampilan Hasil :



2. CURTIME() dan CURRENT_TIME()

   Fungsi yang digunakan untuk menampilkan jam sekarang dari sistem. Contoh penerapannya sebagai berikut :

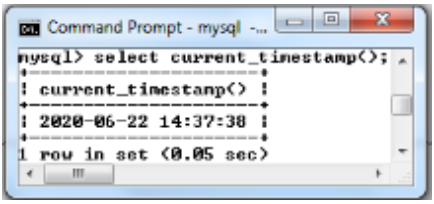
   ```
   Mysql> select current_time();
   ```



3. CURRENT_TIMESTAMP()

   Fungsi yang digunakan untuk menampilkan tanggal dan jam sekarang dari sistem. Contoh penerapannya sebagai berikut :

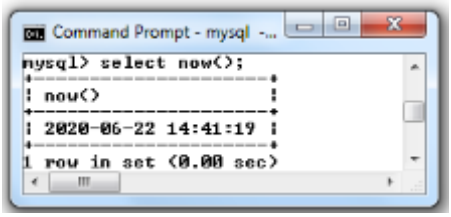
   ```
   Mysql> select current_timestamp();
   ```



4. NOW()

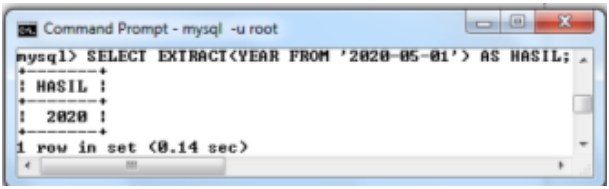
   Fungsi yang digunakan untuk menbisakan waktu sekarang. Ditampilkan dalam format lengkap. Contoh penerapannya sebagai berikut :

   ```
   Mysql> select now();
   ```



5. EXTRACT(value FROM date)
   Fungsi ini digunakan untuk menbisakan informasi tentang nilai dari tanggal berdasarkan tipe interval. Value diisi dengan tipe interval. Penerapannya sebagai berikut :

```
Mysql> SELECT EXTRACT(YEAR FROM '2020-05-01')
AS HASIL;
```



10.7. Ringkasan

1. Fungsi adalah suatu rutin khusus yang disediakan oleh MySQL untuk melakukan manipulasi suatu data.
2. Fungsi terdiri atas nama fungsi dan diikuti parameter (argument) yang diapit tanda kurung.
3. fungsi sistem dalam basis data terdiri atas database(), version(), last_insert_id(), session_user(), system_user(), user().
4. Fungsi agregrat dalam basis data terdiri atas sum, average, count, max dan min.
5. Fungsi String dalam basis data terdiri dari LCASE atau LOWER(), UCASE(x) atau UPPER().
6. Fungsi alam basis data Tanggal terdiri atas CURDATE() dan CURRENT_DATE(), CURTIME() dan CURRENT_TIME(), CURRENT_TIMESTAMP(), NOW(), EXTRACT(value FROM date).

10.8. Evaluasi

1. Berilah contoh penerapan fungsi sistem dalam basis data minimal 2 fungsi.
2. Berilah contoh penerapan fungsi agregat dalam basis data minimal 3 fungsi.
3. Berilah contoh penerapan fungsi String dalam basis data minimal 2 fungsi.
4. Berilah contoh penerapan fungsi Tanggal dalam basis data minimal 2 fungsi.

BAB 11
QUERY DAN VIEW

Bab ini membahas tentang query, join query, macam-macam join query dan view di MySQL.

11.1. Query

Query dalam basis data merupakan suatu proses pengolahan data yang digunakan untuk memberikan hasil dari basis data berdasarkan kriteria tertentu. Query tidak hanya membaca atau mengambil data. Query biasanya melibatkan beberapa tabel

yang direlasikan dengan menggunakan field kunci (*field key*) seperti *primary key* dan *foreign key*. Akan tetapi query juga bisa digunakan hanya pada satu tabel akan tetapi hasilnya terbatas dan kurang informatif.

Ada beberapa aturan yang harus diperhatikan dalam melakukan query antar tabel yaitu sebagai berikut :

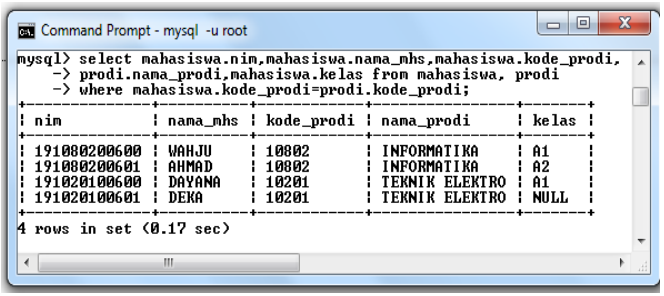
1. Tiap-tiap field disebutkan bersama dengan nama tabelnya, dipisahkan dengan tanda titik (.). Dimana sintaknya adalah namatabel.namafield. contohnya mahasiswa.nim artinya field nim dari tabel mahasiswa.
2. Tiap-tiap tabel yang terlibat dalam proses query harus disebutkan dalam klausa FORM, dengan pemisah tanda koma (,). Contohnya FROM mahasiswa, prodi. Pengurutan tabel tidak akan mempengaruhi proses query.
3. Kondisi pada klausa WHERE akan mempengaruhi jenis join yang terbentuk.

Berikut ini beberapa jenis query yang bisa digunakan untuk menampilkan data yang diinginkan antara lain :

1. Query dengan EQUIJOIN
   Equijoin adalah penggabungan antar tabel dengan menggunakan operator '=' pada kondisi klausa WHERE. Contoh penerapannya :

```
Select mahasiswa.nim,mahasiswa.nama_mhs,
mahasiswa.kode_prodi,prodi.nama_prodi,maha
siswa.kelas from mahasiswa, prodi where
mahasiswa.kode_prodi=prodi.kode_prodi;
```

   maka tampilan hasil script diatas adalah :

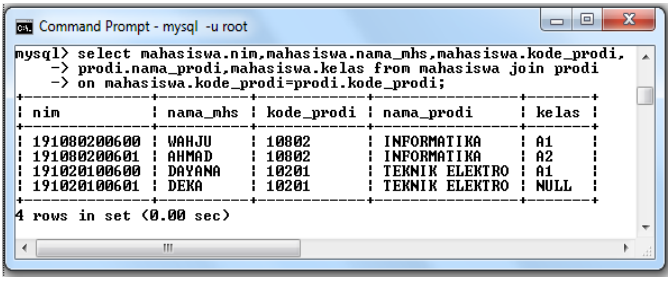


2. Query dengan operator JOIN.
   JOIN merupakan operasi yang digunakan untuk menggabungkan dua tabel atau lebih dengan hasil berupa gabungan dari kolom-kolom yang berasal dari tabel-tabel tersebut. Apabila penggabungan menggunakan JOIN maka klausa WHERE di ganti klausa ON. Sintaks perintah join adalah sebagai berikut :

   ```
   Select nama_tabel from tabel1 join tabel2
   on tabel1.nama_tabl=tabel2.nama_tabel
   ```

   Contoh penerapannya sebagai berikut :

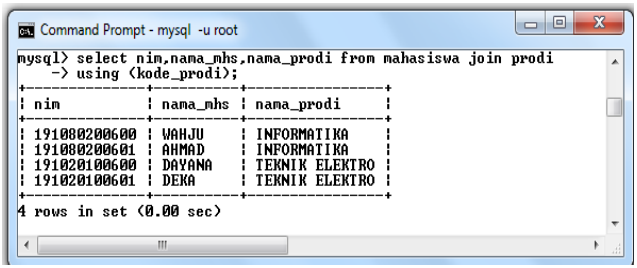
   Mysql> `Select mahasiswa.nim,mahasiswa.nama_mhs, mahasiswa.kode_prodi,prodi.nama_prodi,mahasiswa.kelas from mahasiswa join prodi on mahasiswa.kode_prodi=prodi.kode_prodi;`
   maka hasil tampilan dari script diatas adalah sebagai berikut :



3. Query dengan operator JOIN dan USING
   Klausa using di gunakan untuk menghubungkan relasi antar dua tabel dalam query. Contoh penerapannya sebagai berikut :

   ```
   Mysql>select nim,nama_mhs,nama_prodi from
   mahasiswa join prodi using (kode_prodi);
   ```

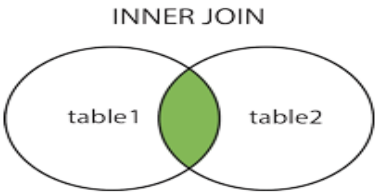
   Maka hasil tampilannya sebagai berikut :



4. Query dengan INNER JOIN
   Untuk menampilkan data yang mempunyai nilai sama pada dua tabel yang digabungkan serta sesuai dengan syarat dibelakang on (tidak boleh null). Dengan kata lain semua data dari tabel kiri menbisa pasangan data dari tabel sebelah kanan. Apabila ada data yang tidak sama maka tidak akan di tampilkan. Sintaks :

   ```
   SELECT nama_kolom FROM tabel1
   ```

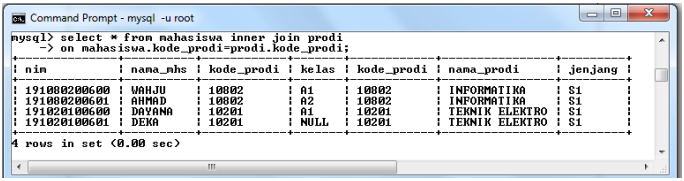
```
INNER JOIN tabel2 ON tabel1.nama_kolom=
tabel2.nama_kolom;
```



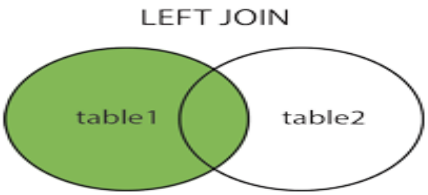
Contoh penerapannya :

```
Mysql>select * from mahasiswa inner join
      prodi on mahasiswa.kode_prodi=
      prodi.kode_prodi;
```

Tampilan hasil :



5. Query dengan LEFT JOIN
   untuk menampilkan data dari tabel sebelah kiri perintah left join beserta pasangannya dari tabel sebelah kanan. Meskipun terdapat data dari sebelah kiri tidak memiliki pasangan, tetap akan ditampilkan dengan pasangannya berupa nilai NULL.



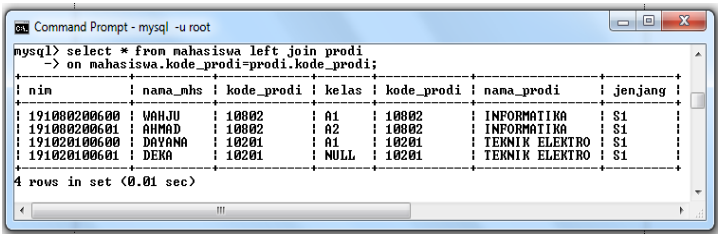
Sintaks :

```
SELECT nama_kolom FROM tabel1
LEFT JOIN tabel2 ON tabel1.nama_kolom=
tabel2.nama_kolom;
```

Contoh penerapannya :

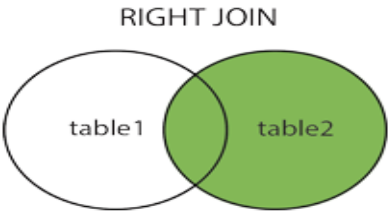
```
Mysql>select * from mahasiswa left join
      prodi on mahasiswa.kode_prodi=
      prodi.kode_prodi;
```

Tampilan hasil :



6. Query dengan RIGHT JOIN

untuk menampilkan data dari tabel sebelah kanan perintah right join beserta pasangannya dari tabel sebelah kiri. Meskipun terdapat data dari sebelah kanan tidak memiliki pasangan, tetap akan ditampilkan dengan pasangannya berupa nilai NULL.



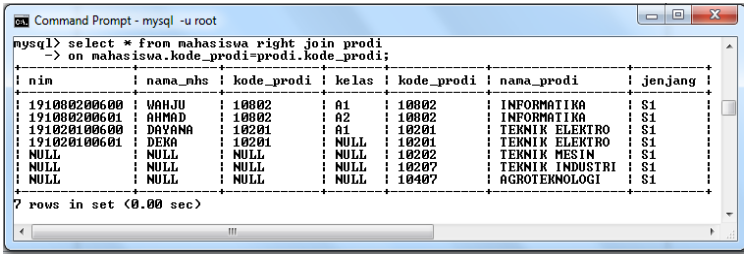

```
SELECT nama_kolom FROM tabel1
```

```
RIGHT JOIN tabel2 ON tabel1.nama_kolom=
tabel2.nama_kolom;
```

Contoh penerapannya :

```
Mysql>select * from mahasiswa right join
      prodi on mahasiswa.kode_prodi=
      prodi.kode_prodi;
```

Tampilan hasil :

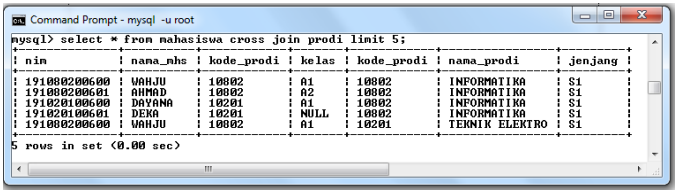


7. Query dengan CROSS JOIN

   Operator ini berguna untuk melakukan operasi penggabungan dengan perkalian kartesain. Namun penggabungan jenis ini jarang digunakan karena tidak menghasilkan nilai informasi yang efektif. Contoh :

```
select * from mahasiswa CROSS JOIN prodi
LIMIT 5;
```

Tampilan Hasil :

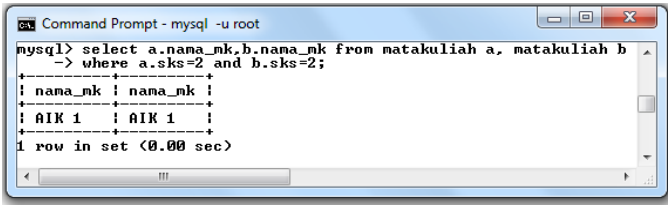


8. Query dengan SELF-JOIN

Self-join adalah jenis penggabungan antar field dari tabel yang sama. Untuk melakukan penggabungan self-join menggunakan alias. Misalkan untuk menampilkan matakuliah dengan sks yang sama , perintahnya :

```
Mysql>select a.nama_mk, b.nama_mk from matakuliah a, matakuliah b where a.sks=2 AND b.sks=2;
```

Hasil Tampilan :

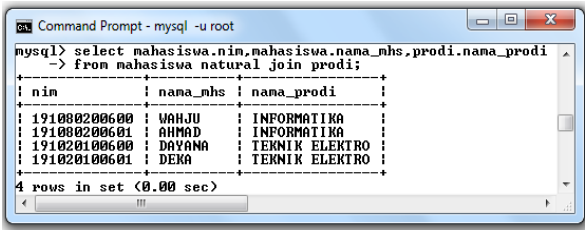


9. Query dengan NATURAL JOIN

   Operator ini digunakan untuk melakukan operasi equijoin dengan memperlakukan nama-nama kolom yang sama sebagai kolom penghubung. Contoh :

```
Mysql> select mahasiswa.nim, mahasiswa.nama_mhs, prodi.nama_prodi from mahasiswa natural join prodi;
```

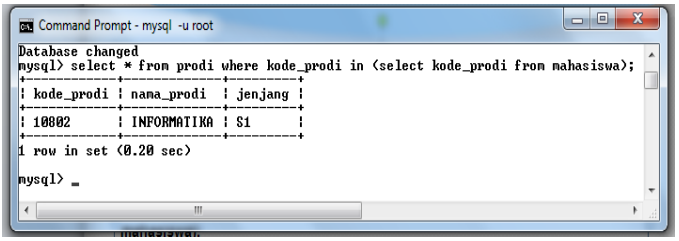
Tampilan hasil :



10. Nested Queries / Subquery (IN, NOT IN, EXISTS, NOT EXISTS)
    Subquery merupakan query di dalam query. Dengan menggunakan subquery, hasil dari query akan menjadi bagian dari query di atasnya. Subquery menggunakan klausa IN, NOT IN, EXISTS, NOT EXISTS yang terletak di dalam klausa WHERE. Pada klausa WHERE, subquery digunakan untuk memilih field-field tertentu yang kemudian digunakan oleh query.
    Contoh :
    - ❖ Perintah untuk menampilkan data prodi pada tabel prodi yang mana datanya tercantum pada tabel mahasiswa menggunakan klausa IN

      ```
      Mysql > select * from prodi where kode_prodi in (select kode_prodi from mahasiswa);
      ```

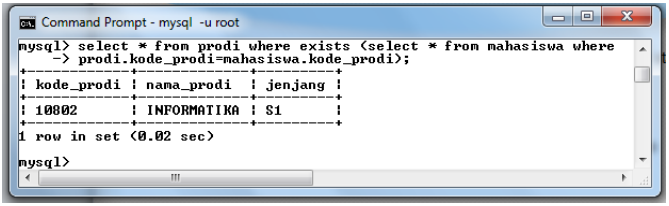
      Tampilan hasil :

    - ❖ Exists merupakan jenis operator Boolean, yang menghasilkan nilai benar atau salah. Perintah EXISTS berguna untuk mengatur hasil query. Contoh : Perintah untuk menampilkan data prodi pada tabel prodi yang mana datanya tercantum pada tabel mahasiswa menggunakan klausa EXISTS

```
Mysql > select * from prodi where exists
select * from    mahasiswa where
prodi.kode_prodi=mahasiswa.kode_prodi);
```

Tampilan hasil :

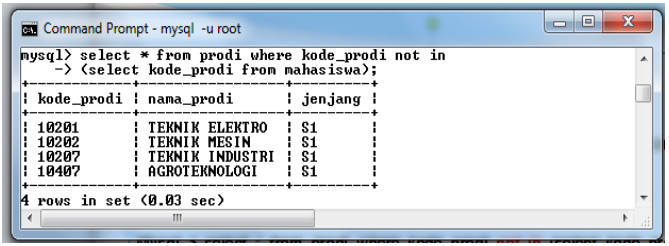


❖ Subquery dengan NOT IN

Contoh Perintah untuk menampilkan data prodi pada tabel prodi yang mana datanya tidak tercantum pada tabel mahasiswa menggunakan klausa NOT IN.

```
Mysql > select * from prodi where
kode_prodi not in (select kode_prodi from
mahasiswa);
```

Tampilan hasil :

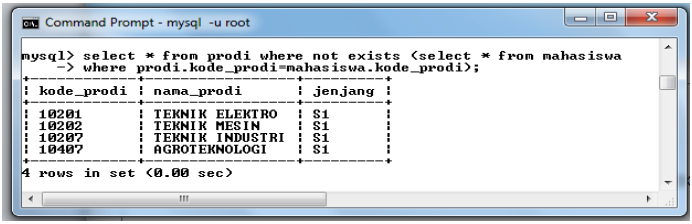


❖ Subquery dengan NOT EXISTS

Contoh Perintah untuk menampilkan data prodi pada tabel prodi yang mana datanya tidak tercantum pada tabel mahasiswa menggunakan klausa NOT EXISTS.

```
Mysql > select * from prodi where not
exists (select * from mahasiswa where
prodi.kode_prodi=mahasiswa.kode_prodi);
```

Tampilan hasil :



11.2. View di Basis Data

View merupakan perintah query yang disimpan pada basis data dengan suatu nama tertentu, sehingga bisa digunakan tiap-tiap saat untuk melihat atau menampilkan data tanpa menulis ulang sintak query tersebut. VIEW merupakan tabel virtual (*virtual table*) hasil dari sebuah statement Select Query. Mengapa View disebut virtual tabel karena View sama seperti tabel yaitu berisi kolom atau field dan record tetapi view bersifat read only (tidak bisa melakukan proses insert data ke view).

Data yang ada di View mengalami perubahan (*up to date*) Apabila data yang ada di tabel sumber berubah. Untuk membuat sebuah view di MySQL, bisa menggunakan perintah CREATE VIEW. Di bawah ini adalah sintak dasar untuk membuat view di MySQL:

```
CREATE [OR REPLACE] VIEW  nama_view AS
SELECT kolom_1, kolom_2, kolom_n
FROM nama_table
WHERE kondisi;
```

Dimana nama_view merupakan nama view yang akan dibuat, sedangkan kolom_1..kolom_n adalah Field-field pada tabel yang

digunakan dan pernyataan select _statement merupakan Perintah query gabungan dari tabel-tabel.

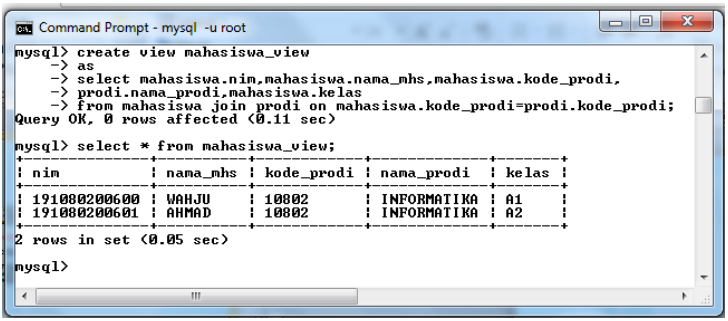
Perintah OR REPLACE digunakan Apabila ingin mengganti struktur view atau mengganti view dengan nama yang sama dengan perintah tersebut. Apabila tidak maka perintah CREATE VIEW akan menghasilkan error Apabila nama view yang ingin dibuat sudah ada sebelumnya. Berikut ini contoh penerapan view :

1. **View antar 2 tabel**

   View bisa di buat dengan menggabungkan dua tabel atau lebih seperti pada query. Misalkan akan membuat view dari relasi antara tabel “mahasiswa" dan “prodi" untuk menampilkan data program studi mahasiswa dari database akademika dengan nama “mahasiswa_view". Maka perintahnya adalah sebagai berikut :

```
Mysql > CREATE VIEW  mahasiswa_view
> AS
>
SELECT mahasiswa.nim, mahasiswa.nama_mhs,
mahasiswa.kode_prodi, prodi.nama_prodi,
mahasiswa.kelas
> FROM mahasiswa join prodi
on mahasiswa.kode_prodi=prodi.kode_prodi;
```

   Tampilan hasil :

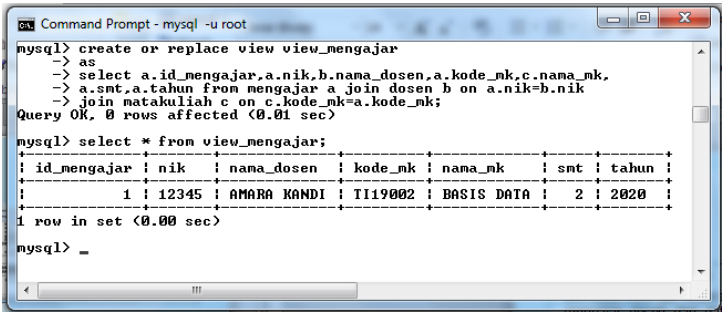


2. **View antar 3 tabel**

Membuat view dari relasi antara tabel “dosen”, “mengajar” dan “matakuliah” untuk menampilkan data mengajar dosen dari database akademika dengan nama "view_mengajar". Perintahnya adalah sebagai berikut :

```
Mysql > CREATE OR REPLACE
VIEW  view_mengajar
> AS
>
SELECT a.id_mengajar, a.nik,b.nama_dosen,a
.kode_mk,c.nama_mk, a.smt, a.kelas,a.tahun
FROM mengajar a join dosen b on
a.nik=b.nik join matakuliah c
on c.kode_mk=a.kode_mk;
```

Tampilan hasil :



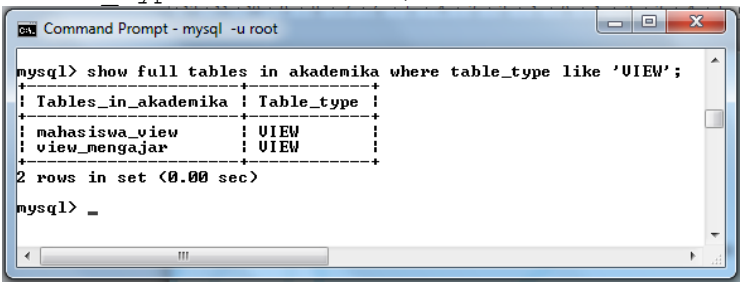
3. Melihat view yang ada di basis data aktif

Untuk melihat daftar view yang berada di basis data aktif maka menggunakan perintah :

```
SHOW FULL TABLES IN database_name WHERE
TABLE_TYPE LIKE 'VIEW';
```

Contoh :

```
show full tables in akademika where
table_type like 'VIEW';
```



4. Menghapus view

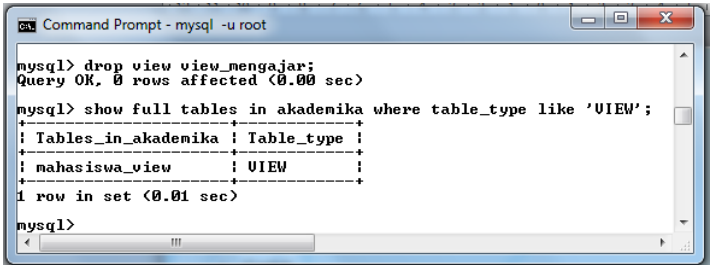
View juga bisa dihapus. Sintak untuk menghapus view adalah sebagai berikut :

```
DROP VIEW nama_view;
```

Contoh :

```
drop view view_mengajar;
```

tampilan hasil :



11.3. Ringkasan

1. Query dalam basis data merupakan suatu proses pengolahan data yang digunakan untuk memberikan hasil dari basis data berdasarkan kriteria tertentu.

2. Query biasanya melibatkan beberapa tabel yang direlasikan dengan menggunakan field kunci (field key) seperti primary key dan foreign key.
3. Jenis query yang digunakan untuk menampilkan data antara lain Query dengan EQUIJOIN, Query dengan operator JOIN dan USING, Query dengan operator JOIN, Query dengan LEFT JOIN, Query dengan RIGHT JOIN, Query dengan CROSS JOIN, Query dengan SELF-JOIN, Query dengan NATURAL JOIN, Nested Queries / Subquery (IN, NOT IN, EXISTS, NOT EXISTS).
4. View merupakan perintah query yang disimpan pada basis data dengan suatu nama tertentu, sehingga bisa digunakan tiap-tiap saat untuk melihat atau menampilkan data tanpa menulis ulang sintak query tersebut.

11.4. Evaluasi

1. Terapkan query join, left join, right join dan nested query yang sudah Anda pelajari ke dalam basis data yang sudah Anda buat minimal 2 tabel untuk menampilkan informasi yang berguna bagi user.
2. Buatlah sebuah view untuk menampilkan informasi yang diperlukan oleh user dari database yang telah Anda buat minimal dari 3 tabel.

BAB 12
HAK AKSES USER

Di bab ini akan membahas tentang definisi hak akses, hak akses superuser, mengatur hak akses, membatasi hak akses, hak akses penuh, hak akses kepada public dan pencabutan hak akses.

12.1. Definisi Hak Akses

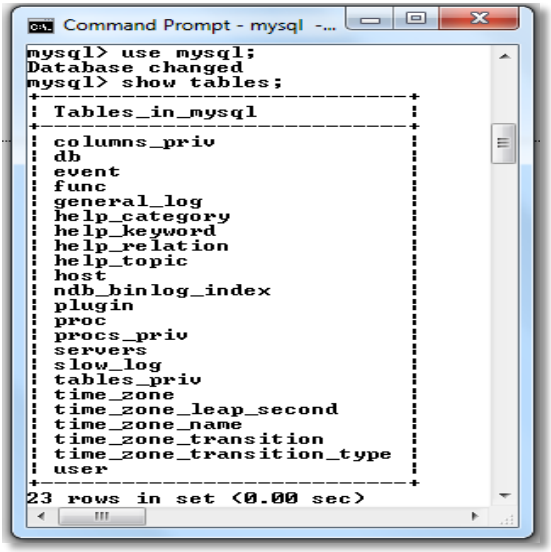
Basis data yang telah dibuat perlu adanya pengaturan agar data selalu dalam keadaan aman dari pengguna yang tidak berhak. Pengaturan hak akses berguna dalam pengaturan eran pengguna dalam melihat atau melakukan aksi di basis data misalkan hanya pengguna tertentu yang bisa membaca atau pengguna lain bisa melakukan perubahan dan penghapusan data. Hak akses perlu di lakukan pengaturan dengan tujuan supaya data yang bersifat rahasia bisa terlindungi. Macam-macam perintah yang terkait dengan hak akses adalah SELECT, INSERT, UPDATE, DELETE, REFERENCES, INDEX, CREATE, ALTER dan DROP.

selama ini pengguna hanya menggunakan 1 user dalam MySQL, yaitu user 'root'. User root ini otomatis dibuat pada saat instalasi MySQL Server pertama kali. User 'root' dalam istilah keamanan komputer sering disebut sebagai 'superuser'. Superuser merupakan tingkatan user tertinggi dimana user ini bisa membuat, melihat, mengubah, bahkan menghapus seluruh database dan menjalankan perintah apapun yang terdapat dalam MySQL.

Seluruh user dan hak aksesnya (*privileges*), disimpan oleh mysql pada sebuah database khusus, yakni database mysql. Tabel khusus ini langsung dibuat secara otomatis pada saat instalasi MySQL. Untuk mengakses user root maka tahapannya sebagai berikut :

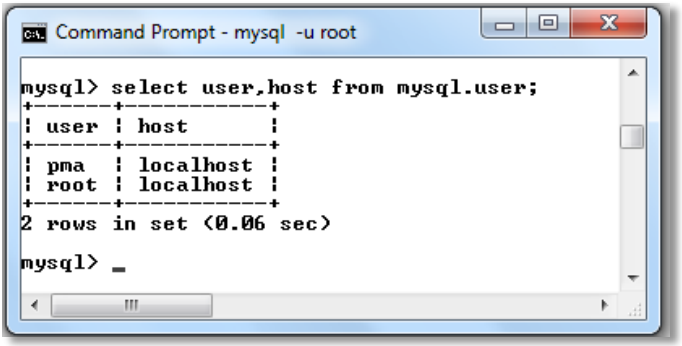
1. Login sebagai user root
2. Mysql>use mysql;
3. Mysql>Show tables;

Tampilan hasil :



Sedangkan perintah untuk melihat user yang ada di basis data MySQL adalah sebagai berikut :

1. Login dengan user root kemudian ketikkan perintah :
2. Mysql>use mysql;
3. mysql>select user,host from mysql.user;
4. Untuk keluar dari mysql :
5. Mysql>quit;



Untuk melihat apa saja hak akses yang dimiliki oleh masing-masing user tersebut dengan menggunakan query SHOW GRANTS FOR. Format dasar query SHOW GRANTS FOR adalah sebagai berikut:

```
mysql>SHOW GRANTS FOR 'nama_user'@'lokasi_user';
```

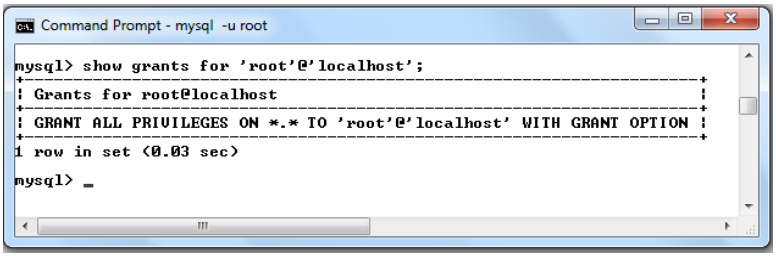
Dimana :

a. nama_user adalah nama dari user yang akan diquery.
b. lokasi_user adalah alamat IP dari user nama_user, bisa berupa: localhost, 192.168.0.5, atau '%'.

Misalkan ingin melihat hak akses user root maka sintaknya adalah sebagai berikut :

```
Mysql>show grants for 'root'@'localhost';
```

Tampilan hasil :



Namun pada aplikasi real dunia nyata, menggunakan user root untuk mengakses basis data dalam operasional sehari-hari sangat tidak disarankan. Memberikan kemampuan dan hak akses untuk menghapus seluruh basis data akan berdampak fatal pada operasional aplikasi.

12.3 Membuat User

Sebelum memberikan hak akses maka harus membuat user baru terlebih dahulu.

1. Create User

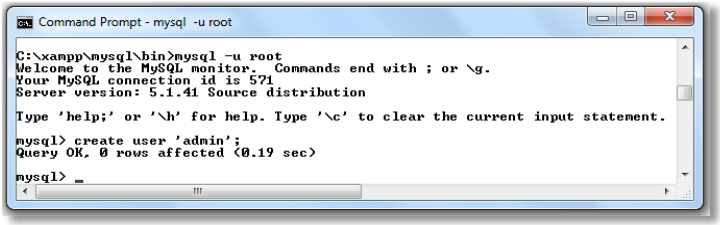
   Untuk membuat user baru, MySQL menyediakan query CREATE USER, berikut format dasar perintah:

   ```
   CREATE USER 'nama_user';
   ```

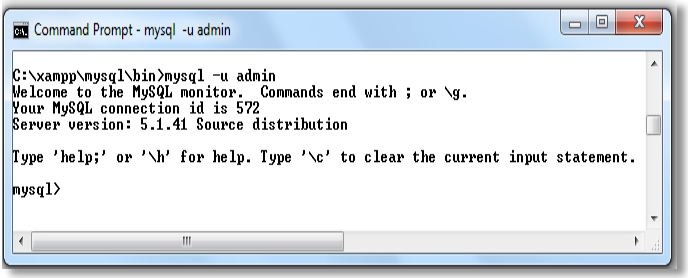
   Dimana nama_user adalah nama dari user yang akan dibuat, maksimal 16 karakter. Misalkan akan membuat user admin, berikut contoh querynya sebagai berikut :

   ```
   mysql> CREATE USER 'admin';
   ```

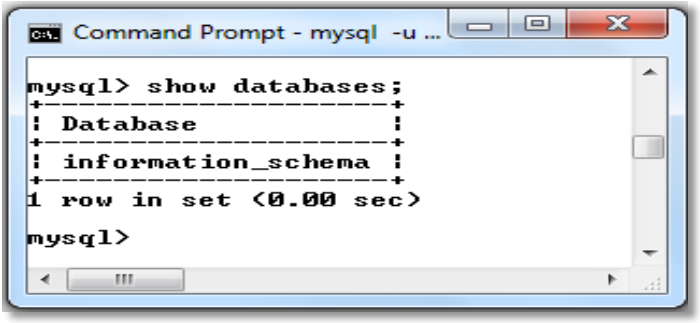
   Tampilan hasil SQL :



Untuk mencoba menggunakan user tersebut, harus keluar dari user root yang digunakan saat ini, dan login sebagai user baru yaitu admin.



User ini belum mempunyai basis data, karena user admin belum memiliki hak akses untuk basis data apapun. Untuk mengetahui basis data apa saja yang ada di user ini bisa meggunakan perintah SHOW DATABASES.



Terlihat bahwa hasil SHOW DATABASES dari user admin hanya berisi database information_schema. Database information_schema sendiri bukan merupakan database 'asli'. information_schema akan ada untuk tiap-tiap user MySQL dan hanya database 'virtual' yang digunakan untuk menyimpan metadata (data keterangan) tentang database. User hanya bisa menggunakan query SELECT untuk database ini, tetapi tidak untuk query DELETE, INSERT, maupun UPDATE.

2. Create User dengan Password

   Sedangkan untuk membuat user baru dengan password dengan perintah:

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

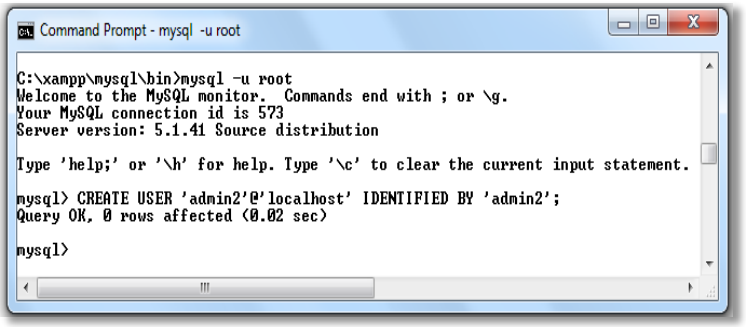
   Dimana :

   a. nama_user merupakan nama dari user yang akan dibuat, maksimal 16 karakter.
   b. lokasi_user adalah lokasi tempat user yang diperbolehkan mengakses. Apabila berada di komputer yang sama dengan MySQL Server, lokasi_user ditulis sebagai 'localhost', namun Apabila berada di komputer tertentu, bisa mengisinya dengan alamat IP seperti '192.168.0.1', atau alamat host domain seperti user.websiteku
   c. password merupakan password yang harus dituliskan pada saat nama_user mengakses MySQL server.

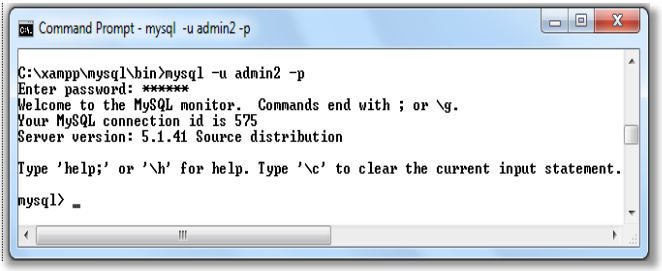
   Contoh :

```
Mysql>CREATE USER 'admin2'@'localhost'
    IDENTIFIED BY 'admin2';
```

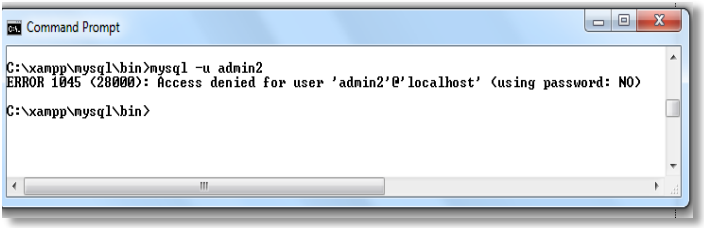
   Tampilan hasil SQL adalah sebagai berikut :



Dan Apabila mengakses user tersebut, haruslah menggunakan password.



Apabila Tidak maka akan di tolak



3. Mengubah User

   User dan password yang sudah dibuat bisa di ubah (*update*) dengan menggunakan query CREATE OR REPLACE. Berikut format penggunaan:

```
CREATE OR REPLACE USER
nama_user@lokasi_user IDENTIFIED BY
'password';
```

Contoh :

```
Myql> create or replace user
admin2@localhost IDENTIFIED BY 'admin2';
```

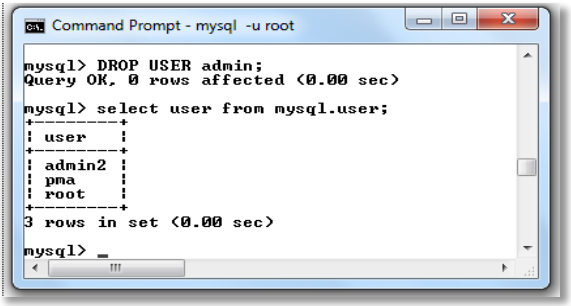
4. Menghapus User

MySQL menyediakan query untuk menghapus user yaitu dengan menggunakan perintah DROP USER. Berikut format penggunaan:

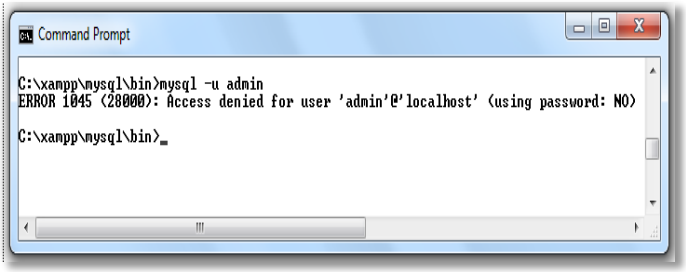
```
DROP user nama_user;
```

Dimana nama_user adalah nama dari user yang akan dihapus. Contoh menghapus user admin :

```
Mysql>DROP USER admin;
```



Apabila user ingin masuk mysql dengan user admin maka akan ada peringatan error.



12.2. Mengatur Hak Akses User

Hak akses dalam MySQL selain dibatasi dengan query apa saja yang dibolehkan, juga bisa dibatasi pada level dimana query tersebut akan dijalankan, misalkan pada level basis data, level tabel atau level kolom. Dalam perancangan aplikasi yang membutuhkan basis data, tiap-tiap user yang akan mengakses basis data seharusnya memiliki batasan masing-masing sesuai dengan fungsinya.

Sebagai contoh, pada database akademi, terdapat 3 buah tabel, yakni mahasiswa, jurusan dan user. Misalkan dibuatkan user admin yang hanya diberikan hak akses untuk melihat data tabel di basis data, namun tidak bisa merubah apapun di dalam tabel tersebut. Di dalam MySQL, user admin tersebut hanya kita berikan hak akses SELECT.

1. Hak Akses GRANT

Untuk memberikan hak akses kepada sebuah user, MySQL menyediakan query GRANT.

Ada 4 Level hak akses di MySQL:

a. Global (keseluruhan)

User bisa memiliki hak akses untuk seluruh database yang terdapat di dalam MySQL.

Penulisan query GRANT untuk level ini adalah:

```
GRANT SELECT ON *.* TO
'user'@'localhost';
```

Menggunakan *.*, sehingga user tersebut bisa mengakses seluruh tabel pada seluruh database.

b. Level Database
Hak akses ini yaitu user memiliki hak akses penuh untuk sebuah database.
Query GRANT untuk level database ini adalah:

```
GRANT SELECT ON nama_database.* TO
'user'@'localhost';
```

Untuk hak akses untuk seluruh tabel, penulisannya adalah nama_database.*

c. Level tabel
Hak akses ini berarti user memiliki hak akses untuk sebuah tabel yang berada pada sebuah database.
Query GRANT untuk level ini adalah:

```
GRANT SELECT ON nama_database.nama_tabel
TO 'user'@'localhost';
```

Hak akses yang dimiliki user hanya terbatas pada level sebuah tabel saja.

d. Level Kolom
Hak akses ini adalah hak akses paling kecil yang bisa diberikan kepada sebuah user. Dengan hak akses level kolom, user hanya memiliki hak akses untuk beberapa kolom pada sebuah tabel. Query GRANT untuk level kolom ini adalah:

```
GRANT SELECT (field1,field2) ON
nama_database.nama_tabel TO
'user'@'localhost';
```

Contoh hak akses GRANT untuk level Tabel serta format dasar query GRANT:

```
GRANT hak_akses ON nama_database.
nama_tabel TO 'nama_user'@'lokasi_user';
```

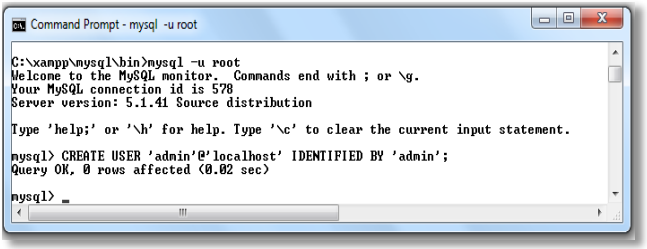
Dimana :

1. hak_akses : privileges yang akan berikan kepada user tersebut. Hak akses disini berisi query yang diperbolehkan, seperti: SELECT, INSERT, UPDATE, DELETE, atau query lainnya. Apabila ingin memberikan hak penuh untuk semua query dasar tersebut, hak_akses ini bisa diisi dengan ALL.
2. nama_database : nama database yang ingin diberikan hak akses. Apabila mengizinkan user tersebut bisa mengakses semua database yang ada, nama_database bisa ditulis dengan tanda bintang (*).
3. nama_tabel : nama tabel yang ingin diberikan hak akses. Apabila mengizinkan user bisa menggunakan semua tabel, nama_tabel bisa ditulis dengan tanda bintang (*).
4. nama_user : nama dari user yang akan diberikan hak akses.
5. lokasi_user : alamat IP dari user yang ingin diberikan hak akses.

Contoh :

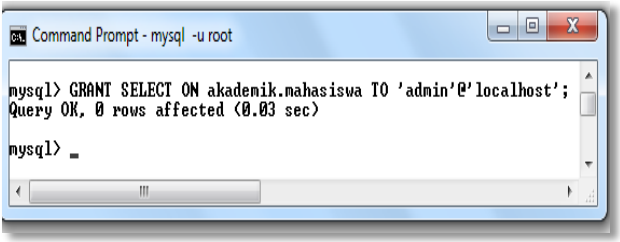
misalkan ingin memberikan privileges kepada admin untuk bisa melihat (melakukan query SELECT) pada tabel mahasiswa yang berada pada database akademik, maka query yang digunakan adalah sbb :

a. Login ke user root
b. Buat user baru dengan nama ‘admin’@’localhost’ dengan password admin.

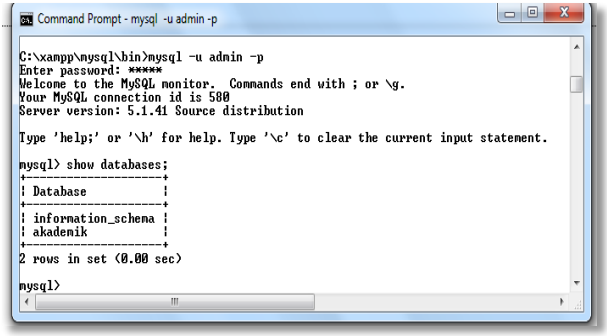


c. Kemudian beri hak akses untuk bisa melihat data yang ada di tabel maahsiswa di database akademik :

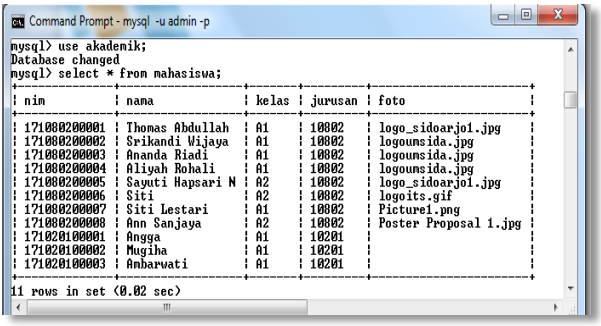
```
mysql> GRANT SELECT ON
akademik.mahasiswa TO
'admin'@'localhost';
```



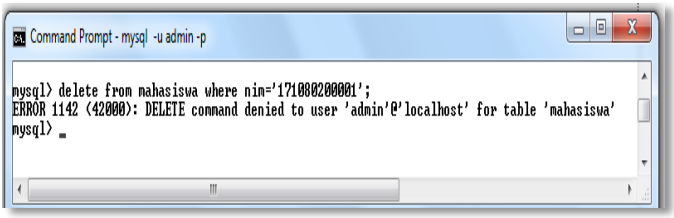
Untuk mencoba user admin, maka kita keluar dari root, dan masuk sebagai admin :



d. Kemudian untuk melihat data memakai perintah select ;



e. Mencoba menghapus data akan ditolak karena tidak mempunyai hak akses :



f. Untuk Memberikan Hak Akses Seluruh Tabel Untuk memberikan hak akses penuh kepada pemakai. Misalkan membuat user mahasiswa yang diberikan hak akses untuk bisa melihat seluruh tabel yang ada pada database akademik :

```
mysql> CREATE USER
'mahasiswa'@'localhost' IDENTIFIED BY
‘mahasiswa';

mysql> GRANT SELECT ON akademik.* TO
'mahasiswa'@'localhost';
```

g. Untuk Memberikan seluruh Hak Akses dengan GRANT ALL Misalkan membuat user mhs yang diberikan hak akses keseluruhan untuk bisa melakukan CRUD tabel mahasiswa yang ada pada database akademik :

```
mysql> CREATE USER 'mhs'@'localhost'
IDENTIFIED BY 'mhs';

mysql> GRANT ALL ON akademik.mahasiswa
TO 'mhs'@'localhost';
```

Melihat hak akses user mhs@localhost :

```
Mysql>show grants for 'mhs'@'localhost';
```

h. Untuk keperluan yang lebih spesifik, hak akses bisa juga dibatasi hanya untuk kolom tertentu. Dimana datanya tidak boleh diketahui oleh user tersebut. Misal :

```
mysql> CREATE USER
'tamu'@'localhost';
mysql> GRANT SELECT (nim,nama) ON
akademik.mahasiswa  TO
'tamu'@'localhost';
```

Perhatikan cara penulisan kolom yang diberikan hak aksesnya, setelah hak akses (dalam contoh adalah SELECT), penulisan nama kolom harus berada dalam tanda kurung.

2. Menghapus Hak Akses (REVOKE)
Hak akses yang diberikan ke seorang user, kadang perlu dilakukan perubahan, tergantung kondisi dan kebijakan pengguna. Untuk menghapus user, pengguna bisa menggunakan query DROP USER. Tapi terkadang hanya

diperlukan penghapusan hak aksesnya saja tanpa harus menghapus user yang bersangkutan maka bisa menggunakan perintah REVOKE.

a. REVOKE

Perintah ini digunakan untuk melakukan pencabutan hak akses sebagian pemakai atau secara keseluruhan.

Bentuk umum :

```
REVOKE hak_akses ON
nama_database.nama_tabel FROM nama_user;
```

atau

```
REVOKE hak_akses ON namatabel FROM
nama_user;
```

Contoh :

Admistrator ingin mencabut hak akses user admin, maka perintahnya :

```
mysql> REVOKE SELECT ON
akademik.mahasiswa FROM
'admin'@'localhost';
```

b. REVOKE ALL

Perintah ini digunakan untuk melakukan pencabutan hak akses secara keseluruhan

Bentuk umum :

```
REVOKE ALL ON nama_database.nama_tabel
FROM nama_user;
```

Contoh :

```
mysql> REVOKE ALL ON akademik.* FROM
‘mahasiswa'@'localhost';
```

c. DROP USER

Untuk menghapus user secara permanen dari basis data.

```
DROP USER nama_user@lokasi_user;
```

12.3. Ringkasan

1. Pengaturan hak akses berguna dalam pengaturan eran pengguna dalam melihat atau melakukan aksi di basis data misalkan hanya pengguna tertentu yang bisa membaca atau pengguna lain bisa melakukan perubahan dan penghapusan data.
2. Perintah yang digunakan dalam pengaturan hak askes antara lain CREATE USER, DRP USER, GRANT dan REVOKE.
3. Macam-macam perintah yang terkait dengan hak akses adalah SELECT, INSERT, UPDATE, DELETE, REFERENCES, INDEX, CREATE, ALTER dan DROP.

12.4. Evaluasi

1. Buatlah beberapa user dalam database yag suda Anda buat yang memiliki hak akses berbeda, minimal 2 user.
2. Terapkan perintah GRANT dan REVOKE pada user tersebut.

DAFTAR PUSTAKA

Gehani Narain . 2011. *The Database Book: Principles & Practice Using MySQL*. Silicon Press. USA.

Indrajani. 2015. *Database Design (Case Study All In One*). Elex Media Komputindo. Jakarta.

Jayanti Ari Dwi Ketut Ni.dkk. 2018. *Teori Basis Data*.Penerbit ANDI. Yogyakarta.

Kristanto Harianto. 1994. Konsep & Perancangan Database. Penerbit ANDI, Yogyakarta.

Pamungkas Ajika Canggih. 2017. *Pengantar Dan Implementasi Basis Data*. Deepublis Publisher. Yoyakarta.

Qalbi, Iilyan Ainun. 2014. *Konsep Dasar Basis Data*. Universitas Mulawarman. Samarinda.

Rob Peter. Coronel Carlos. Crockeett Keeley . 2008. *Database Sistem Design, Implementation & Management*. Cengage Learning EMEA. UK

Saputro T Wahju. 2005. MySQL Untuk Pemula. Pena Media. Yogyakarta

Subandi, Syahidi Akhrian Aulia, 2018, *Basis Data : Teori dan Praktik Menggunakan Microsoft Office Access*, Poliban Press, Yogyakarta.

Widodo Wahyu Agus.Kurnianingtyas Diva. 2017. *Sistem Basis Data*.UB Press. Malang.

Yanto Robi. 2016. Manajemen Basis data Menggunakan *MySQL*. Deepublis Publisher. Yoyakarta

W3Schools Online SQL Tutorials. Domain **https://www.w3schools.com/**

MySQL Tutoral. Domain https://www.mysqltutorial.org

BIODATA PENULIS

Ika Ratna Indra Astutik, S.Kom., M.T. dilahirkan di Sidoarjo, 13 Mei 1981. Pada tahun 2005, penulis mendapat gelar Sarjana Teknik Informatika dari Universitas Muhammadiyah Sidoarjo. Penulis melanjutkan magister Jaringan Cerdas Multimedia di ITS dengan program beasiswa dari DIKTI. Tahun 2015, penulis secara resmi mendakatkan gelar M.T. Penulis mengawali karirnya sebagai Dosen di prodi Teknik Informatika Universitas Muhammadiyah Sidoarjo. Penulis juga aktif terlibat dalam kegiatan penelitian dan pengabdian kepada masyarakat. Penelitian yang pernah dilakukan oleh penulis adalah tentang sistem informasi berbasis web, basis data dan sistem pengambilan keputusan. Penulis juga dipercaya mengelola Basis Data Universitas Muhammadiyah Sidoarjo sampai sekarang.

Mochamad Alfan Rosid, S.Kom., M.Kom. lahir di Sidoarjo, 25 April 1986. Lulus Sarjana Komputer Universitas Muhammadiyah Sidoarjo tahun 2010. Penulis melanjutkan studi S2 di Prodi Teknologi Informasi Program Pascasarjana Sekolah Tinggi Teknik Surabaya lulus tahun 2014 dengan mendakatkan gelar M.Kom. Penulis mengawali karirnya sebagai Dosen di prodi Teknik Informatika Universitas Muhammadiyah Sidoarjo. Penulis juga aktif terlibat dalam kegiatan penelitian dan pengabdian kepada masyarakat. Penelitian yang pernah dilakukan oleh penulis adalah tentang sistem informasi berbasis, basis data dan sistem pengambilan keputusan. Selain pendidikan dan pengajaran penulis juga terlibat dalam kegiatan penelitian dan pengabdian kepada masyarakat baik didanai oleh Ristekdikti maupun dana mandiri.