DIKTAT

PEMROGRAMAN BERBASIS WEB

Oleh:

SUENDRI, M.Kom

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI (UIN) SUMATERA UTARA

DIKTAT

PEMROGRAMAN BERBASIS WEB

Oleh:

SUENDRI, M.Kom

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI (UIN) SUMATERA UTARA

KATA PENGANTAR

Puji dan syukur kepada Allah Ta'ala yang telah melimpahkan rahmat dan kasih sayangnya sehingga diktat ini selesai disusun dengan baik. Selawat dan salam juga teruntuk nabi junjungan alam, Muhammad Shollollohu 'Alaihi Wasallam sebagai panutan manusia hingga akhir zaman.

Diktat ini disusun sebagai kelengkapan matakuliah Pemrograman Berbasis Web pada Program Studi Sistem Informasi Universitas Islam Negeri (UIN) Sumatera Utara Medan. Diktat hanya untuk kepentingan perkuliahan, disadur dari berbagai sumber dengan memasukkan tulisan aslinya secara utuh selain tata letak sedangkan referensi sumber terkait dicantumkan di Daftar Pustaka.

Penulis menyadari masih banyak perbaikan yang diperlukan dalam penyusunan, aturan penulisan dan tata letak dari diktat ini, oleh kerena itu penulis mengharapkan kritik dan saran yang membangun. Penulis juga mengucapkan terimakasih kepada seluruh pihak yang dijadikan referensi dalam penulisan diktat ini.

Medan, Januari 2017

Suendri, M.Kom

DAFTAR ISI

BAB I
PENGANTAR PEMROGRAMAN WEB

1. Pendahuluan

Pemrograman web diambil dari dua suku kata yaitu pemrograman dan web. Pemrograman diartikan proses, cara, pembuatan program. Sedangkan Definisi Web adalah jaringan komputer yang terdiri dari kumpulan situs internet yang menawarkan teks, grafik, suara dan sumber daya animasi melalui *HyperText Transfer Protocol*. Orang banyak mengenal web dengan istilah WWW (*World Wide Web*), *World Wide Web* adalah layanan internet yang paling populer saat ini, internet mulai dikenal dan digunakan secara luas setelah adanya layanan WWW. WWW adalah halaman-halaman website yang dapat saling terkoneksi satu dengan lainnya (*hyperlink*) yang membentuk samudera belantara informasi. WWW berjalan dengan protokol *HyperText Transfer Protocol* (HTTP). Halaman Web merupakan *file* teks murni (*plain text*) yang berisi sintaks-sintaks HTML yang dapat dibuka, dilihat, diterjemahkan dengan *Internet Browser*. Sintaks HTML mampu memuat konten text, gambar, audio, video dan animasi. Kini internet identik dengan web, karena kepopuleran web sebagai standar *interface* pada lanyanan-layanan yang ada di internet, dari awalnya sebagai penyedia informasi, ini digunakan juga untuk komunikasi dari email sampai dengan chatting, sampai dengan melakukan transaksi bisnis (*commerce*).

2. HTTP

HTTP (*HyperTextTransfer Protocol*) adalah protokol yang dipergunakan untuk mentransfer dokumen dalam *World Wide Web* (WWW). Protokol ini adalah protokol ringan, tidak berstatus dan generik yang dapat dipergunakan berbagai macam tipe dokumen. Pengembangan HTTP dikoordinasi oleh Konsorsium *World Wide Web* (W3C) dan kerjasama Internet *Engineering Task Force* (IETF), bekerja dalam publikasi satu seri RFC, yang paling terkenal RFC 2616, yang menjelaskan HTTP/1.1, versti HTTP yang digunakan umum sekarang ini. HTTP adalah sebuah protokol meminta atau menjawab antara *client* dan *server*. Sebuh *client* HTTP seperti *web browser*, biasanya

memulai permintaan dengan membuat hubungan TCP/IP ke *port* tertentu di *workstation* yang jauh (biasanya port 80). Sebuah *server* HTTP yang menangkap sinyal di port tersebut menunggu *client* mengirim kode permintaan (*request*), seperti "GET / HTTP/1.1" (yang akan meminta halaman yang sudah ditentukan), diikuti dengan pesan MIME yang memiliki beberapa informasi kode utama yang menjelaskan aspek dari permintaan tersebut, diikut dengan badan dari data tertentu. Beberapa kepala (*header*) juga bebas ditulis atau tidak, sementara lainnya (seperti *workstation*) diperlukan oleh protokol HTTP/1,1. Begitu menerima kode permintaan (dan pesan, bila ada), *server* mengirim kembali kode jawaban, seperti "200 OK", dan sebuah pesan yang diminta, atau sebuah pesan *error* atau pesan lainnya.

Protokol HTTP pertama kali dipergunakan dalam WWW pada tahun 1990. Pada saat tersebut yang dipakai adalah protokol HTTP versi 0.9. Versi 0.9 ini adalah protokol transfer dokumen secara mentah, maksudnya adalah data dokumen dikirim sesuai dengan isi dari dokumen tersebut tanpa memandang tipe dari dokumen. Kemudian pada tahun 1996 protokol HTTP diperbaiki menjadi HTTP versi 1.0. Perubahan ini untuk mengakomodasi tipe-tipe dokumen yang hendak dikirim beserta *encoding* yang dipergunakan dalam pengiriman data dokumen. Sesuai dengan perkembangan infrastruktur internet maka pada tahun 1999 dikeluarkan HTTP versi 1.1 untuk mengakomodasi *proxy, cache* dan koneksi yang persisten.

3. Web

Web adalah suatu ruang informasi di mana sumber-sumber daya yang berguna diidentifikasi oleh pengenal global yang disebut *Uniform Resource Identifier* (URI). Secara umum, Web 1.0 dikembangkan untuk pengaksesan informasi dan memiliki sifat yang sedikit interaktif. Secara garis besar, sifat Web 1.0 adalah *Read*. Lalu, tak lama kemudian muncullah Web 2.0 yang merupakan revolusi bisnis di industri komputer yang disebabkan oleh penggunaan internet sebagai *platform*, juga merupakan suatu percobaan untuk memahami aturan untuk mencapai keberhasilan *platform* baru. Sifat Web 2.0 adalah *Read-Write*. Era Web 2.0 tidak membutuhkan orang jenius yang hanya berkutat sendiri di ruang tertutup atau laboratorium untuk membuat

teknologi baru yang dipatenkan agar membuat dirinya menjadi terkenal. Tapi era ini lebih membutuhkan orang untuk saling berbagi ilmu, pengalaman atau lainnya sehingga terbentuk komunitas online besar yang menghapuskan sifat-sifat individu. Sedangkan letak perbedaan Web 1.0 dan Web 2.0 yaitu :

a. Perilaku pengguna Membaca dan Menulis.
b. Pelaku utama Perusahaan Pengguna/Komunitas.
c. Hubungan dengan *server Client - server Peer to peer.*
d. Bahasa pemrograman penampil konten HTML XML.
e. Pola hubungan penerbit - pengguna Searah dan Dua arah / Interaktif.
f. Pengelolaan konten Taksonomi/*direktori* Folksonomi/penanda/*tag*.
g. Penayangan berbagai kanal informasi Portal RSS/Sindikasi.
h. Hubungan antar pengakses Tidak ada berhubungan.
i. Sumber konten Penerbit/pemilik situs Pengguna.

Yang menjadi kunci perbedaan dalam Web 2.0 dan Web 1.0 adalah keterbatasan pada Web 1.0 yang mengharuskan pengguna internet untuk datang ke dalam *website* tersebut dan melihat satu persatu konten di dalamnya. Sedangkan Web 2.0 memungkinkan pengguna internet dapat melihat konten suatu website tanpa harus berkunjung ke alamat situs yang bersangkutan. Kemampuan web 2.0 dalam melakukan aktivitas *drag and drop, auto complete, chat, voice* dapat dilakukan layaknya aplikasi *desktop*. Selanjutnya adalah Web 3.0, jika dunia seluler dikenal istilah 3G, maka di Internet ada yang namanya Web 3.0. Wow, apa pula ini? Apa bedanya dengan Web 2.0 yang sekarang sedang marak? Jangan salah, ternyata orang Indonesia juga sudah ada yang mengembangkannya. Konsep ini dapat diandaikan sebuah website sebagai sebuah intelektualitas buatan (*Artificial Intelegence*). Aplikasi-aplikasi *online* dalam website dapat saling berinteraksi, kemampuan interaksi ini dimulai dengan adanya *web service.*

Pada web 3.0 ini, sudah terjadi konvergensi yang sangat dekat antara dunia TI dengan dunia telekomunikasi. Dunia web dan telekomukasi berkembang pesat seiring dengan kebutuhan pengguna. Penggunaan perangkat TI dan telekomunikasi nantinya sudah seperti sama saja tidak ada bedanya. Saat ini saja pertanda seperti itu sudah

mulai bisa kita rasakan walaupun masih belum sempurna. Kita bisa menonton televisi di ponsel atau komputer, bisa mengakses internet di ponsel, bisa melakukan SMS dan telepon dari komputer. Ya karena konvergensi terhadap berbagai perangkat seperti hukum alam yang tidak bisa dielakkan. Semua mengalami evolusi menuju dunia yang lebih maju.

Permasalahan lain yang potensial muncul adalah, sebagai teknologi masa depan, Web 3.0 juga membutuhkan kecepatan akses Internet yang memadai dan spesifikasi komputer yang tidak enteng, hal ini disebabkan tak lain karena teknologi ini secara visual berbasis 3D. Sedangkan seperti yang kita tahu biaya akses Internet dengan kecepatan tinggi di Indonesia ini masih terbilang mahal bagi masyarakat umum. Belum lagi jika dihitung dari biaya spesifikasi perangkat komputer yang dibutuhkan, mungkin masyarakat Indonesia yang ingin menikmati kecanggihan layanan berbasis teknologi Web 3.0 masih harus menarik napas panjang. Namun karena Web 3.0 sendiri masih dalam pengembangan, seiring dengan berlalunya waktu sebagai masyarakat Indonesia kita masih bisa mengharapkan bahwa biaya komunikasi, dalam hal ini koneksi Internet kecepatan tinggi akan semakin murah nantinya, sehingga terjangkau bagi masyarakat luas. Saat ini adaptasi Web 3.0 mulai dikembangkan oleh beberapa perusahaan di dunia seperti secondlife, Google Co-Ops, bahkan di Indonesia sendiri juga sudah ada yang mulai mengembangkannya, yaitu Li'L Online (LILO) Community.

Permasalahan lain yang potensial muncul adalah, sebagai teknologi masa depan, Web 3.0 juga membutuhkan kecepatan akses Internet yang memadahi dan spesifikasi komputer yang tidak enteng, hal ini disebabkan tak lain karena teknologi ini secara visual berbasis 3D. Sedangkan seperti yang kita tahu biaya akses Internet dengan kecepatan tinggi di Indonesia ini masih terbilang mahal bagi masyarakat umum. Belum lagi jika dihitung dari biaya spesifikasi perangkat komputer yang dibutuhkan, mungkin masyarakat Indonesia yang ingin menikmati kecanggihan layanan berbasis teknologi Web 3.0 masih harus menarik napas penjang. Namun karena Web 3.0 sendiri masih dalam pengembangan, seiring dengan berlalunya waktu sebagai masyarakat Indonesia kita masih bisa mengharapkan bahwa biaya

komunikasi, dalam hal ini koneksi Internet kecepatan tinggi akan semakin murah nantinya, sehingga terjangkau bagi masyarakat luas.

4. Web Statis Dan Web Dinamis

Web statis adalah website yang mana pengguna tidak bisa mengubah konten dari web tersebut secara langsung menggunakan browser. Interaksi yang terjadi antara pengguna dan server hanyalah seputar pemrosesan link saja. Halaman-halaman web tersebut tidak memliki *database*, data dan informasi yang ada pada web statis tidak berubah-ubah kecuali diubah sintaksnya. Dokumen web yang dikirim kepada client akan sama isinya dengan apa yang ada di web server. Contoh dari web statis adalah web yang berisi profil perusahaan. Di sana hanya ada beberapa halaman saja dan kontennya hampir tidak pernah berubah karena konten langsung diletakan dalam *file* HTML saja.

Dalam web dinamis, interaksi yang terjadi antara pengguna dan server sangat kompleks. Seseorang bisa mengubah konten dari halaman tertentu dengan menggunakan browser. *Request* (permintaan) dari pengguna dapat diproses oleh server yang kemudian ditampilkan dalam isi yang berbeda-beda menurut alur programnya. Halaman-halaman web tersebut memiliki *database*. Web dinamis, memiliki data dan informasi yang berbedabeda tergantung input apa yang disampaikan client. Dokumen yang sampai di client akan berbeda dengan dokumen yang ada di *web server*. Contoh dari web dinamis adalah portal berita dan jejaring sosial. Lihat saja web tersebut, isinya sering diperbaharui (di-*update*) oleh pemilik atau penggunanya. Bahkan untuk jejaring sosial sangat sering di-*update* setiap harinya.

a. Interaksi antara pengunjung dan pemilik web

   Dalam web statis tidak dimungkinkan terjadinya interaksi antara pengunjung dengan pemilik web. Sementara dalam web dinamis terdapat interaksi antara pengunjung dengan pemilik web seperti memberikan komentar, transaksi online, forum dan sebagainya.

b. Bahasa *Script* yang digunakan

Web statis hanya menggunakan HTML saja, atau paling tidak bisa ditambah dengan CSS. Sedangkan web dinamis menggunakan bahasa pemrograman web yang lebih kompleks seperti PHP, ASP dan JavaScript atau bahasa pemrograman lainnya.

c. Penggunaan *Database*

Web statis tidak menggunakan *database* karena tidak ada data yang perlu disimpan dan diproses. Sedangkan web dinamis menggunakan *database* seperti MySQL, Oracle, MariaDB untuk menyimpan dan memproses data.

d. Konten

Konten dalam web statis hanya diberikan oleh pemilik web dan jarang di-*update*, sementara konten dalam web dinamis bisa berasal dari pengunjung dan lebih sering di-*update*. Konten dalam web dinamis bisa diambil dari *database* sehingga isinya pun bisa berbeda-beda walaupun kita membuka web yang sama.

5. Kebutuhan Pemrograman Web

Pembuatan halaman web membutuhkan persiapan tidak saja pengetahuan tentang bagaimana disain halaman web, namun juga perlu dukungan persiapan perangkat keras dan perangkat lunak.

a. Perangkat Keras

Perangkat keras yang dibutuhkan untuk pembuatan halaman web tidak berbeda jauh dengan kebutuhan komputasi biasa. Seperangkat komputer lengkap dengan CPU, monitor, *keyboard, mouse, printer* dan beberapa perangkat tambahan lain sudah dapat digunakan untuk membuat halaman web. Spesifikasi tergantung dari perangkat lunak yang akan diinstal pada perangkat komputer tersebut. Jika kita menginstal web server, pengolah gambar untuk disain halaman web, HTML editor yang komplek, tentu kita membutuhkan spesifikasi yang lebih tinggi.

b. Perangkat Lunak

Editor adalah perangkat lunak yang digunakan untuk membuat halaman-halaman web, baik yang bersifat statis maupun dinamis. Di pasar perangkat lunak, saat ini

tersedia banyak sekali jenis perangkat pengembang web, mulai dari yang sederhana sampai yang canggih dan kompleks. Namun sebenarnya untuk membuat halaman web baik statis maupun dinamis kita dapat menggunakan teks editor biasa seperti Notepad atau Vi. Hanya saja teks editor tidak menyediakan fasilitas-fasilitas yang memudahkan kita dalam membuat halaman web. Pada perangkat pengembang web yang lebih kompleks seperti Adobe Dreamweaver, Microsoft Visual Studio.Net, dan beberapa yang lainnya, kita akan mendapati fasilitas yang sangat membantu mempercepat pembuatan halaman web, antara lain: tampilan berbasis GUI, *automatic code completion* (melengkapi kode secara otomatis), WYSIWYG (*What You See Is What You Get*) HTML Editor, koneksi ke basis data yang lebih mudah, dan banyak lagi fasilitas. Tentu saja perangkat pengembang ini berharga relative mahal.

6. Web Browser

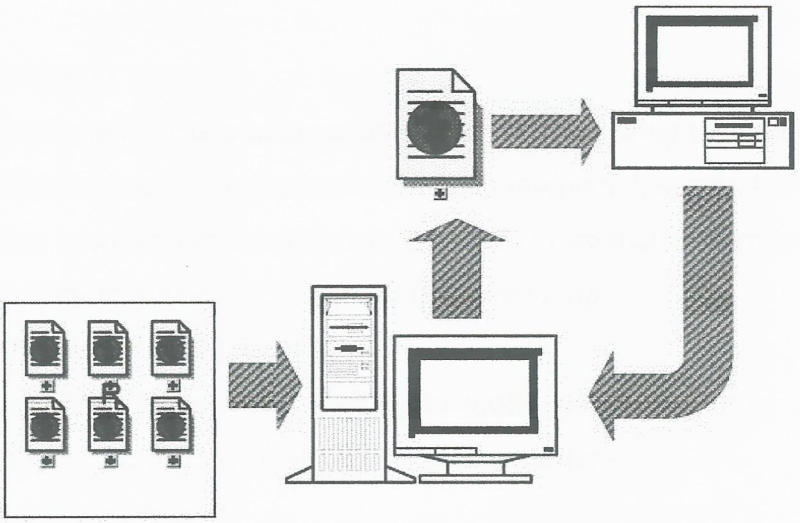
Penjelajah web atau Peramban web adalah perangkat lunak yang berfungsi untuk menerima dan menyajikan sumber informasi di Internet. Sebuah sumber informasi diidentifikasi dapat berupa halaman web, gambar, video, atau jenis konten lainnya. Meskipun penjelajah web terutama ditujukan untuk mengakses Internet, sebuah penjelajah juga dapat digunakan untuk mengakses informasi yang disediakan oleh *server web* dalam jaringan pribadi atau berkas pada sistem berkas. Beberapa penjelajah web yang populer adalah Google Chrome, Firefox, Internet Explorer, Opera, dan Safari.

Gambar 1. *Web Browser*

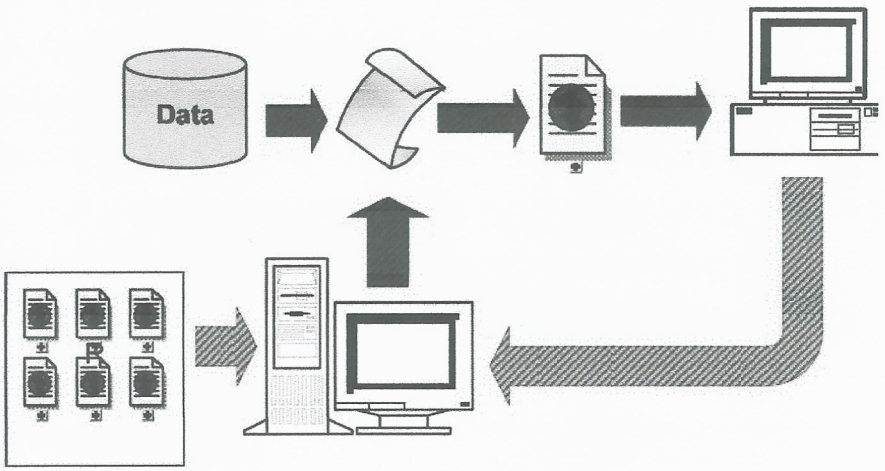
Sumber : http://www.thepropertyjungle.com/xml/cache/mceimages/what-is-a-web-browser.jpg

7. Web *Server*

Web *Server* merupakan sebuah perangkat lunak dalam server yang berfungsi sebagai penerima permintaan (*request*) berupa sebuah halaman web melalui HTTP atau HTTPS dari klien yang dikenal dengan browser web dan mengirimkan kembali (response) hasilnya dalam bentuk halaman-halaman web yang umumnya berbentuk dokumen HTML.



Gambar 2. Standar *Web Architecture*



Gambar 3. *Dynamic Web Architecture*

Adapun *Web Sever* yang banyak digunakan di internet sebagai berikut:

1. *Webserver Apache* (http://www.apache.org)
2. *Internet Information Service*, IIS (http://www.microsoft.com/iis)
3. *Xitami Web Server* (http://www.xitami.com)
4. *Sun Java System Web Server* (http://www.sun.com/software/products / web_srvr /home_web_srvr.xml)

B. *Server Side Scripting*

Server Side Scripting merupakan sebuah teknologi *scripting* atau pemrograman web dimana *script* (program) dikompilasi atau diterjemahkan di server. Dengan *server side scripting* dan memungkinkan untuk menghasilkan sebuah halaman web yang dinamis. Contoh-contoh *Server Side Scripting* (*Programming*):

a. ASP (*Active Server Page*) dan ASP.NET
b. ColdFusion (http://www.macromedia.com/software/coldfusion)
c. Java Server Pages (http://java.sun.com/products/jsp/)
d. Perl (http://www.perl.org)
e. Phyton (http://www.python.org)
f. PHP (http://www.php.net)

BAB II
DATABASE (DDL)

1. *Database*

Database atau **basis data** adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (*query*) basis data disebut sistem manajemen basis data (*database management system*, DBMS). Sistem basis data dipelajari dalam ilmu informasi.

Istilah "basis data" berawal dari ilmu komputer. Meskipun kemudian artinya semakin luas, memasukkan hal-hal di luar bidang elektronika, artikel ini mengenai basis data komputer. Catatan yang mirip dengan basis data sebenarnya sudah ada sebelum revolusi industri yaitu dalam bentuk buku besar, kuitansi dan kumpulan data yang berhubungan dengan bisnis.

Konsep dasar dari basis data adalah kumpulan dari catatan-catatan, atau potongan dari pengetahuan. Sebuah basis data memiliki penjelasan terstruktur dari jenis fakta yang tersimpan di dalamnya, penjelasan ini disebut skema. Skema menggambarkan objek yang diwakili suatu basis data, dan hubungan di antara objek tersebut. Ada banyak cara untuk mengorganisasi skema, atau memodelkan struktur basis data: ini dikenal sebagai model basis data atau model data. Model yang umum digunakan sekarang adalah model relasional, yang menurut istilah layman mewakili semua informasi dalam bentuk tabel-tabel yang saling berhubungan di mana setiap tabel terdiri dari baris dan kolom (definisi yang sebenarnya menggunakan terminologi matematika). Dalam model ini, hubungan antar tabel diwakili denga menggunakan nilai yang sama antar tabel. Model yang lain seperti model hierarkis dan model jaringan menggunakan cara yang lebih eksplisit untuk mewakili hubungan antar tabel.

Istilah *basis data* mengacu pada koleksi dari data-data yang saling berhubungan, dan perangkat lunaknya seharusnya mengacu sebagai *sistem manajemen basis data* (*database management system/DBMS*). Jika konteksnya sudah jelas, banyak administrator dan programer menggunakan istilah basis data untuk

kedua arti tersebut. Jadi secara konsep basis data atau *database* adalah kumpulan dari data-data yang membentuk suatu berkas (*file*) yang saling berhubungan (*relation*) dengan tatcara yang tertentu untuk membentuk data baru atau informasi. Atau basis data (*database*) merupakan kumpulan dari data yang saling berhubungan (relasi) antara satu dengan yang lainnya yang diorganisasikan berdasarkan skema atau struktur tertentu. Pada komputer, basis data disimpan dalam perangkat hardware penyimpan, dan dengan software tertentu dimanipulasi untuk kepentingan atau keguanaan tertentu. Hubungan atau relasi data biasanya ditunjukkan dengan kunci (key) dari tiap *file* yang ada. Data merupakan fakta atau nilai (*value*) yang tercatat atau merepresentasikan deskripsi dari suatu objek.

Data yang merupakan fakta yang tercatat dan selanjutnya dilakukan pengolahan (proses) menjadi bentuk yang berguna atau bermanfaat bagi pemakainya akan membentuk apa yang disebut informasi. Bentuk informasi yang kompleks dan terintegrasi dan pengolahan sebuah *database* dengan komputer akan digunakan untuk proses pengambilan keputusan pada manajemen akan membenuk Sistem Informasi Manajemen (SIM), data dalam basis data merupan item terkecil dan terpenting untuk membangun basis data yang baik dan valid. Data dalam basis data bersifat *integrated* dan *shared*:

- Terpadu (*integrated*), berkas-berkas data yang ada pada basis data saling terkait (terjadi dependensi data);
- Berbagi data (*shared*), data yang sama dapat dipakai oleh sejumlah pengguna dalam waktu yang bersamaan. Sering dinamakan sebagi sistem *multiuser*

Data merupakan suatu sumber yang sangat berguna bagi hampir disemua organisasi. Dengan tersedianya data yang melimpah, maka masalah pengaturan data secara efektif menjadi suatu hal yang sangat penting dalam pengembangan sistem informasi manajemen. Oleh karena itu, tujuan dari diadakannya pengaturan data adalah sebagai berikut:

- Menyediakan penyimpanan data untuk dapat digunakan oleh organisasi saat sekarang dan masa akan datang.

- Sebagai cara pemasukan data sehingga sehingga memudahkan tugas operator dan menyangkut pula waktu yang diperlukan oleh pemakai untuk mendapatkan data serta hak-hak yang dimiliki terhadap data yang ditangani
- Pengendalian data untuk setiap siklus agar data selalu *up to date* dan dapat mencerminkan perubahan spesifik yang terjadi di setiap sistem.
- Pengamanan data terhadap kemungkinan penambahan, modifikasi, pencurian, dan gangguan-gangguaan lain.

Suatu bangunan basis data memiliki jenjang sebagai berikut:

- Karakter, merupakan bagian data terkecil yang berupa angka, huruf, atau karakter khusus yang membentuk sebuah *item* data atau *field*. Contoh A,B,X,Y,2,1,2,9,0,=,<,> dan sebagainya.
- *Field*/item, merupakan representasi suatu atribut dan *record* (rekaman/tupel) yang sejenis yang menunjukkan suatu item dari data. Contoh *field* nama (berisi data nama-nama pegawai), *field* departemen (berisi data bagian atau spesifikasi pekerjaan), dan lain sebagainya.
- *Record*/rekaman/tupel: Kumpulan dari *field* membentuk suatu *record* atau rekaman. Record menggambarkan suatu unit data individu yang tertentu. Contoh: *file* pegawai, dimana tiap-tiap *recordnya* berisi kumpulan data nama, alamat, departemen, yang dapat mewakili tiap-tiap data.
- *File*, merupakan kumpulan dari *record-record* yang menggambarkan satu kesatuan data yang sejenis. Contoh *file* pegawai berisi data tentang semua yang berhubungan dengan pegawai seperti nama pegawai, alamat pegawai, departemen, yang dapat mewakili tiap-tiap data.
- *Database*, merupakan kumpulan dari *file* atau tabel yang membentuk suatu *database*. Contoh *database* pegawai PT Maju Terus terdiri atas *file* pegawai, *file* gaji, *file* golongan, dan sebagainya.

Dalam satu *file* terdapat *record-record* yang sejenis, sama besar, sama bentuk, yang merupakan satu kumpulan entitas yang seragam. Satu *record* terdiri dari *field* yang saling berhubungan menunjukkan bahwa *field* tersebut dalam satu pengertian yang lengkap dan direkam dalam satu *record*. Setiap nilai atau isi *field* memiliki

kapasitas ruang atau lebar yang sama. Jenis isi data sebuah *field* harus sesuai dengan tipe datanya. Nama sebuah *file* harus menggambarkan isi dari data *file* tersebut. Untuk melengkapi definisi tentang *file*, dalam *database* dikenal nama entitas (*entity*) dan atribut. Entitas adalah orang, tempat, kejadian, atau konsep yang informasinya direkam. setiap entitas memiliki atribut atau sebutan untuk mewakili suatu entitas. Sebagai contoh dalam sistem perkuliahan; mahasiswa, matakuliah, pembayaran, dosen adalah sebagai entitas. Sedangkan entitas mahasiswa memiliki atribut nomor induk, nama, jurusan, dan sebagainya. Atau dari contoh diatas entitasnya adalah pegawai, ang memilik atribut NIP, nama, alamat, tgl_lahir, jns_kel. Sistem basis data merupakan perpaduan antara basis data dan sistem manajemen basis data (SMBD). *Database* yang kompleks dan disertai dengan teknik pendokumentasian dan prosedur manipulasinya akan membentuk Sistem Manajemen Basis Data (*Database Management System-DBMS*). Singkatnya DBMS adalah *database* dan program untuk mengaksesnya.

2. MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi *GNU General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus di mana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak sama dengan proyek-proyek seperti *Apache*, di mana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, di mana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.

MySQL adalah sebuah implementasi dari sistem manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General*

Public License). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya; SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basisdata, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Kehandalan suatu sistem basisdata (DBMS) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, MySQL mendukung operasi basisdata transaksional maupun operasi basisdata non-transaksional. Pada modus operasi non-transaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak peladen basisdata kompetitor lainnya. Namun pada modus non-transaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi blogging berbasis web (wordpress), CMS, dan sejenisnya. Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basisdata transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus non-transaksional.

2.1 Sejarah MySQL

MySQL pada awalnya diciptakan pada tahun 1979, oleh Michael "Monty" Widenius, seorang programmer komputer asal Swedia. Monty mengembangkan sebuah sistem *database* sederhana yang dinamakan UNIREG yang menggunakan koneksi *low-level* ISAM *database engine* dengan *indexing*. Pada saat itu Monty bekerja pada perusahaan bernama TcX di Swedia.

TcX pada tahun 1994 mulai mengembangkan aplikasi berbasis web, dan berencana menggunakan UNIREG sebagai sistem *database*. Namun sayangnya,

UNIREG dianggagap tidak cocok untuk *database* yang dinamis seperti web. TcX kemudian mencoba mencari alternatif sistem *database* lainnya, salah satunya adalah mSQL (miniSQL). Namun mSQL versi 1 ini juga memiliki kekurangan, yaitu tidak mendukung *indexing*, sehingga performanya tidak terlalu bagus.

Dengan tujuan memperbaiki performa mSQL, Monty mencoba menghubungi David Hughes (*programmer* yang mengembangkan mSQL) untuk menanyakan apakah ia tertarik mengembangkan sebuah konektor di mSQL yang dapat dihubungkan dengan UNIREG ISAM sehingga mendukung indexing. Namun saat itu Hughes menolak, dengan alasan sedang mengembangkan teknologi indexing yang independen untuk mSQL versi 2. Dikarenakan penolakan tersebut, David Hughes, TcX (dan juga Monty) akhirnya memutuskan untuk merancang dan mengembangkan sendiri konsep sistem *database* baru. Sistem ini merupakan gabungan dari UNIREG dan mSQL (yang *source code*-nya dapat bebas digunakan). Sehingga pada May 1995, sebuah RDBMS baru, yang dinamakan MySQL dirilis.

David Axmark dari Detron HB, rekanan TcX mengusulkan agar MySQL di 'jual' dengan model bisnis baru. Ia mengusulkan agar MySQL dikembangkan dan dirilis dengan gratis. Pendapatan perusahaan selanjutnya di dapat dari menjual jasa "*support*" untuk perusahaan yang ingin mengimplementasikan MySQL. Konsep bisnis ini sekarang dikenal dengan istilah *Open Source*.

Pada tahun 1995 itu juga, TcX berubah nama menjadi MySQL AB, dengan Michael Widenius, David Axmark dan Allan Larsson sebagai pendirinya. Titel "AB" dibelakang MySQL, adalah singkatan dari "Aktiebolag", istilah PT (Perseroan Terbatas) bagi perusahaan Swedia.

2.2 Keistimewaan MySQL

MySQL memiliki beberapa keistimewaan, antara lain :

1. **Portabilitas**. MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.

2. **Perangkat lunak sumber terbuka**. MySQL didistribusikan sebagai perangkat lunak sumber terbuka, dibawah lisensi GPL sehingga dapat digunakan secara gratis.
3. **Multi-user**. MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. ***Performance tuning***, MySQL memiliki kecepatan yang menakjubkan dalam menangani query sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.
5. **Tipe data beragam**. MySQL memiliki ragam tipe data yang sangat kaya, seperti signed / unsigned integer, float, double, char, text, date, timestamp, dan lain-lain.
6. **Perintah dan Fungsi**. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah Select dan Where dalam perintah (*query*).
7. **Keamanan**. MySQL memiliki beberapa lapisan keamanan seperti level subnetmask, nama host, dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenkripsi.
8. **Skalabilitas dan Pembatasan**. MySQL mampu menangani basis data dalam skala besar, dengan jumlah rekaman (records) lebih dari 50 juta dan 60 ribu tabel serta 5 miliar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.
9. **Konektivitas**. MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, Unix soket (UNIX), atau Named Pipes (NT).
10. **Lokalisasi**. MySQL dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa. Meski pun demikian, bahasa Indonesia belum termasuk di dalamnya.
11. **Antar Muka**. MySQL memiliki antar muka (interface) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (Application Programming Interface).

12. **Klien dan Peralatan**. MySQL dilengkapi dengan berbagai peralatan (tool) yang dapat digunakan untuk administrasi basis data, dan pada setiap peralatan yang ada disertakan petunjuk online.
13. **Struktur tabel**. MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE, dibandingkan basis data lainnya semacam PostgreSQL ataupun Oracle.

2.3 Kemampuan MySQL

Berikut Fitur serta kapabilitas yang dimiliki oleh MySQL:

1. Unjuk kerja yang tinggi dalam memproses query sederhana, dalam arti dapat memproses lebih banyak SQL per satuan waktu.
2. Memiliki lebih banyak tipe data seperti : signed/unsigned integer yang memiliki panjang data sebesar 1,2,3,4 dan 8 byte, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET dan tipe ENUM.
3. Mendukung *field* yang dijadikan *Index,* dengan maksimal **32 index** dalam satu tabel.
4. MYSQL memiliki beberapa **lapisan keamanan**, seperti *subnetmask*, nama *host*, dan izin akses user dengan sistem perijinan yang mendetail serta sandi/password terenkripsi.
5. **Konektivitas** , MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP ,Unix soket (UNIX),atau Named Pipes(NT).
6. **Multi-user**. MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik
7. *Command and function*, MySQL memiliki fungsi dan operator secara penuh yang mendukung perintah *select* dan *where* dalam *query*.
8. *Structure Table,* MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani *ALTER TABLE* dibandingkan DBMS lainnya.
9. Mendukung penuh terhadap kalimat SQL GROUP BY dan ORDER BY. Mendukung terhadap fungsi penuh (COUNT(),COUNT(), DISTINCT() AVG(), STD(), SUM(), MAX() dan MIN()).

3. phpMyAdmin

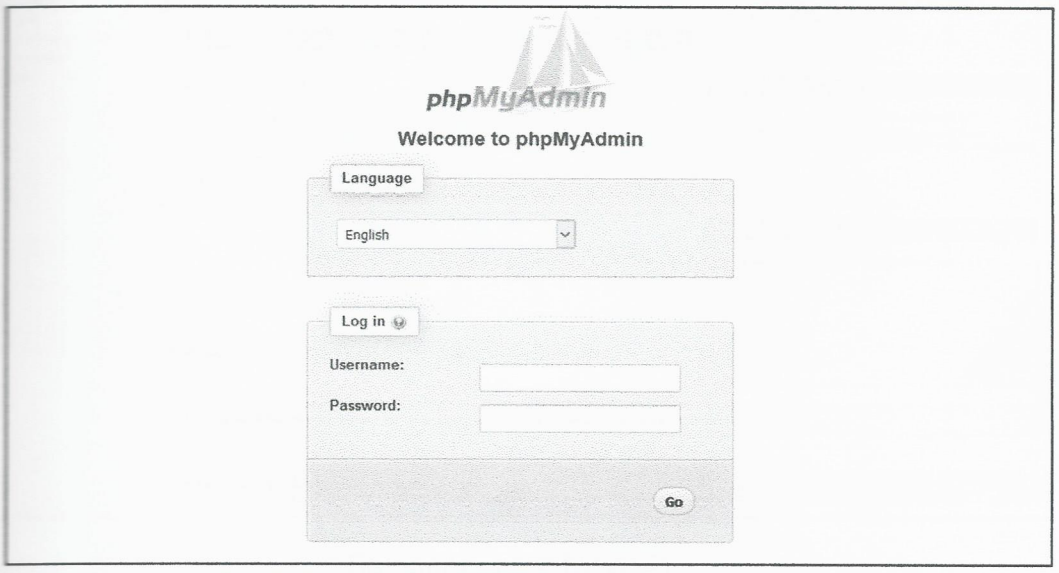
phpMyAdmin bukanlah sebuah *database*, namun hanya manajer dari MySQL untuk memudahkan *user* dalam mengatur *database* karena tampil dalam bentuk GUI yang bisa anda buka melalui *Browser* kesayangan anda.

a. Langkah awal, pastikan XAMPP anda telah aktif, Apache dan MySQL dalam keadaan *Start*.

b. Masukkan alamat berikut pada *address browser* anda. Gunakan *browser* terbaru seperti *Mozilla Firefox* atau *Google Chrome*.

```
http://localhost/phpmyadmin
```

c. Ketika muncul permintaan *password*, silakan anda inputkan *password* anda masing-masing, namun jika tidak ada *password*, anda bisa melewatkan langkah ini.



Gambar 4. *Login*

Secara *default*, *phpMyAdmin* tidak menggunakan *password* saat masuk kedalam sistemnya. Anda bisa membuat pengaturan *password* sendiri melalui *file* **config.inc.php** yang terdapat pada folder **xampp/phpMyAdmin/config.inc.php**

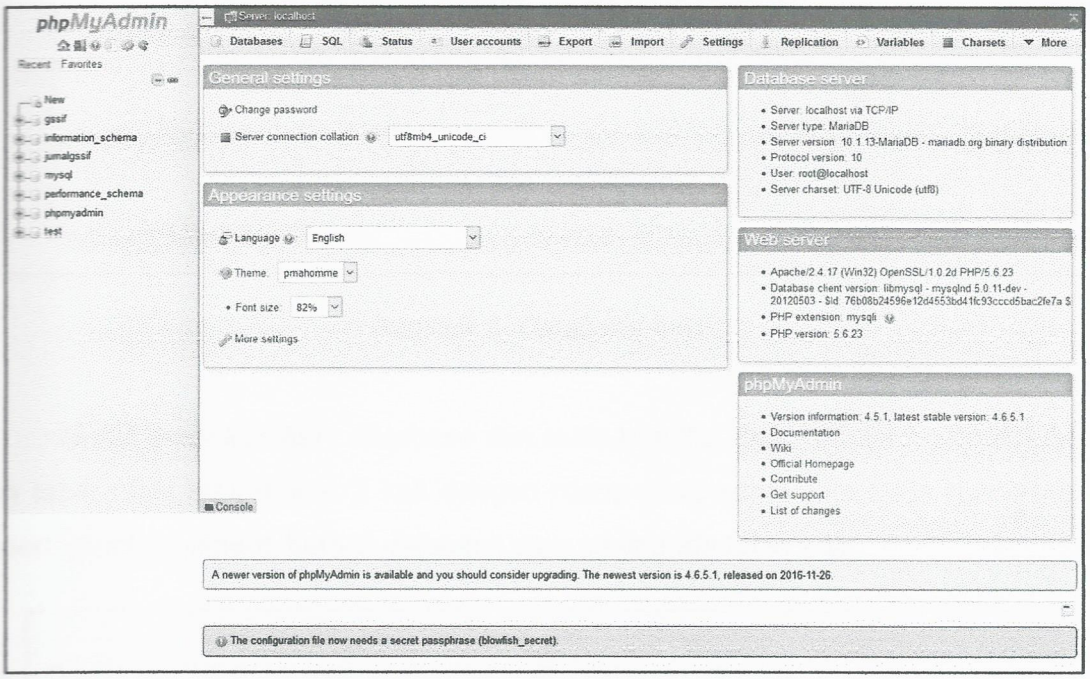
```
$cfg['Servers'][$i]['AllowNoPassword'] = true;
```

Ubah menjadi :

```
$cfg['Servers'][$i]['AllowNoPassword'] = false;
```

Atau jika *phpMyAdmin* sudah menyediakan *file* **config.sample.inc.php**, anda cukup mengganti **config.inc.php** yang lama dengan *file* tersebut.

d. Berikut *screenshoot* phpMyAdmin, yang menandakan *phpMyAdmin* anda telah siap untuk digunakan.

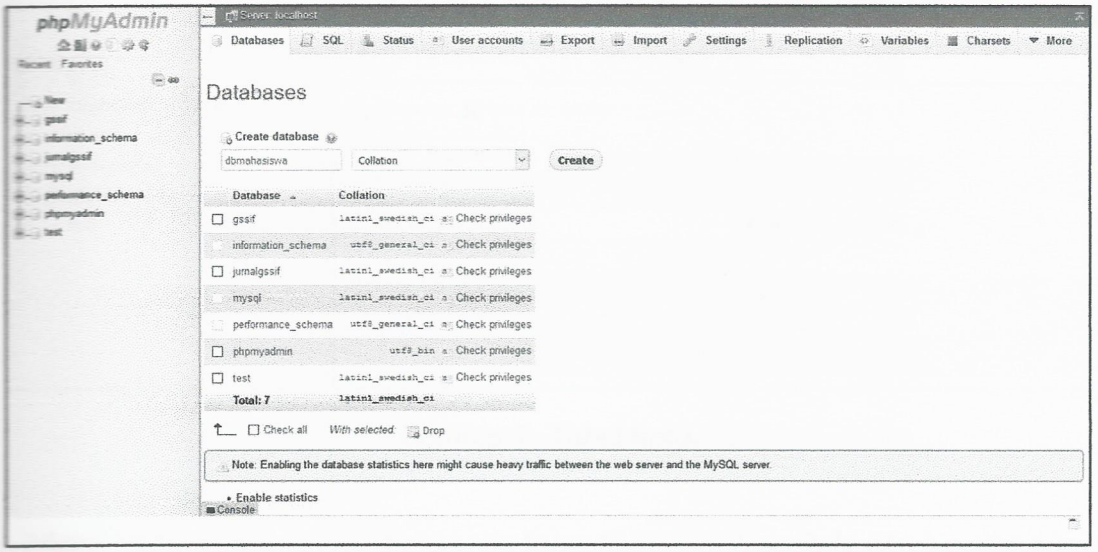


Gambar 5. *phpMyAdmin*

1) Menu bagian kiri merupakan daftar *database* yang telah dibuat sebelumnya.
2) Sedangkan dibagian tengah merupakan *form* untuk membuat pengaturan *database* dan sistem.

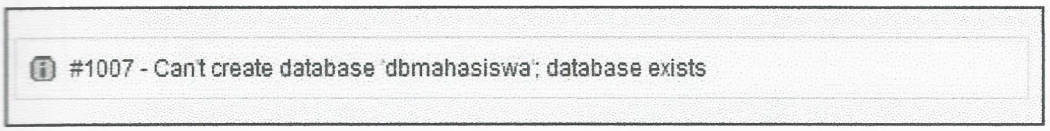
3) Menu lebih lanjut dapat dilihat pada bagian atas, terdapat pengaturan *Database, Form SQL*, Status, Pengaturan User, *Export*, *Import* dan sebagainya yang berfungsi untuk pengaturan sistem.

e. Untuk latihan pertama, buatlah sebuah *database* pada *form* **Create *database*** dengan nama **dbsisko** dan klik tombol ***Create*.**



Gambar 6. *Database* baru

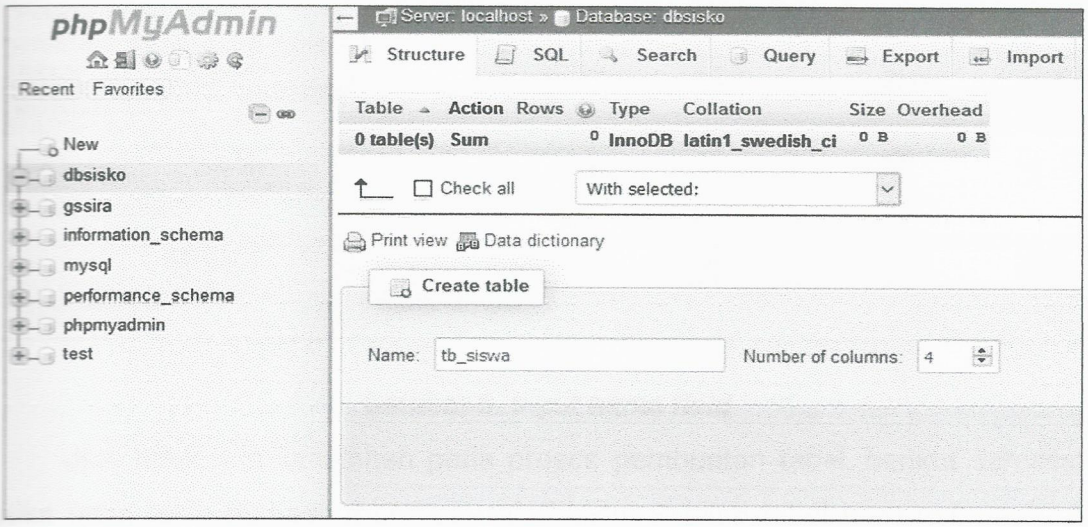
Untuk anda ketahui : *database* dan *table* bersifat unik, artinya nama *database* dan *table* tidak bisa dibuat 2 kali dengan nama yang sama, jika anda menemukan seperti gambar berikut, berarti *database* atau *table* anda telah ada.



Tips membuat *Database*

1. Gunakan Huruf Kecil
2. Jangan gunakan Spasi
3. Unik, tidak ada nama yang sama

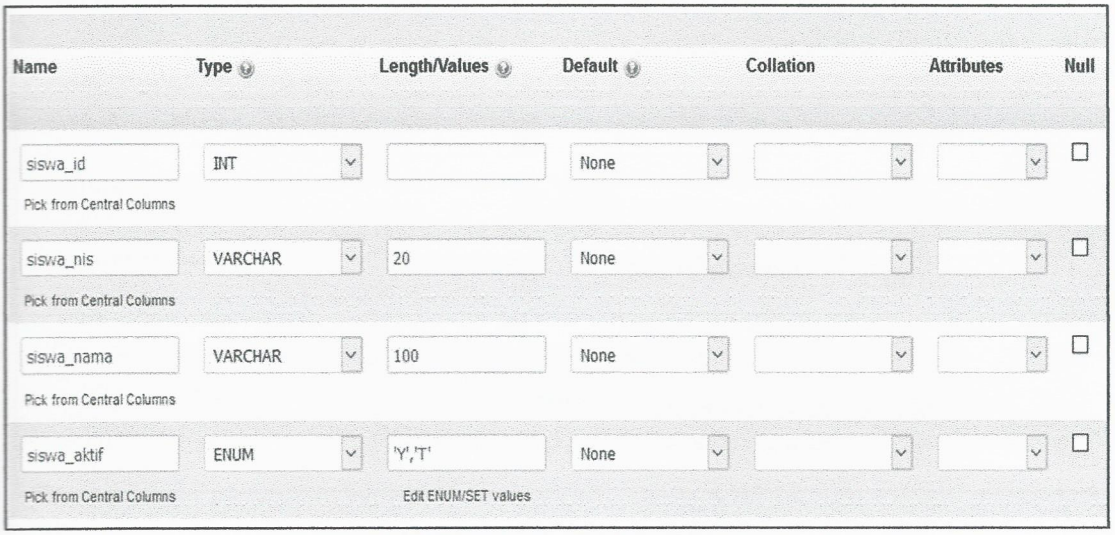
f. Jika *database* sudah berhasil anda buat, sekarang saatnya untuk membuat *table* dalam database anda. Silakan isikan **nama *table*** pada *form* ***Create table*** dan isikan jumlah ***field*** dari tabel yang ingin anda buat. Selanjutnya klik **Go.**



Gambar 7. Tabel baru.

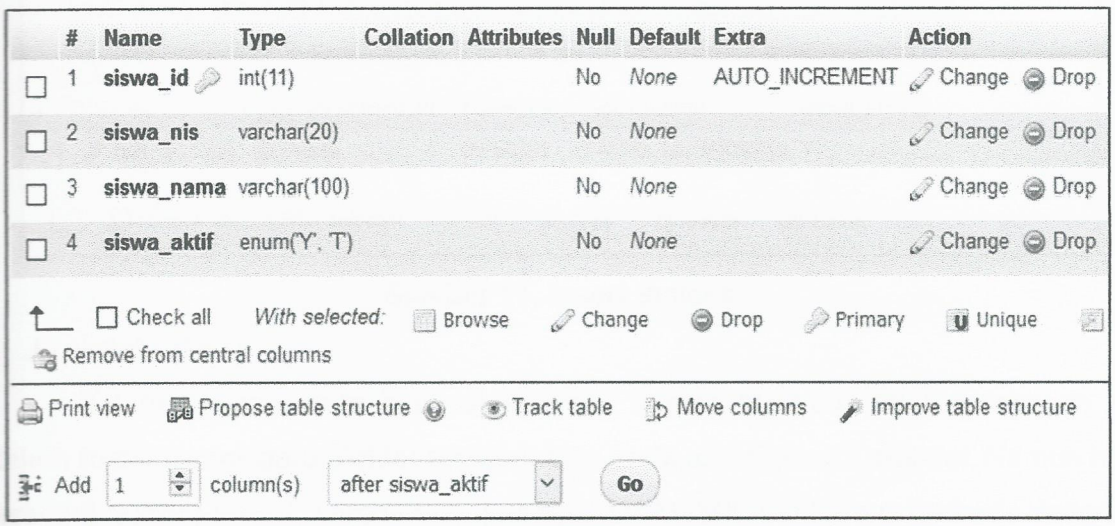
g. Silakan isikan nama-nama *field* pada form ***field***, pilih ***type field*** dan ***length/values*** (jumlah karakter) *field* tersebut.

NO	NAMA	TYPE	LENGTH/VALUES
1	siswa_id	INT	11 (PRIMARY KEY)
2	siswa_nis	VARCHAR	20
3	siswa_nama	VARCHAR	100
4	siswa_aktif	ENUM	'Y','T'



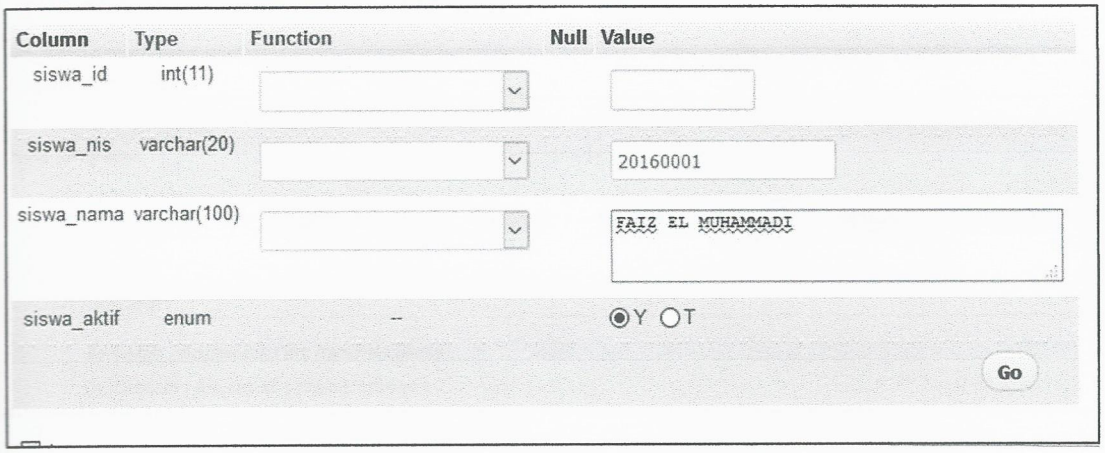
Gambar 8. Input nama *field*

Jika tidak ada kesalahan pada proses pembuatan tabel, berikut *screenshot* ketika anda telah berhasil membuat tabel baru. Setelah tabel baru terbentuk, Jika terjadi kesalahan atau ingin melakukan perbaikan, kita bisa menggunakan menu yang terdapat pada kolom *action*. Diantara menu *Change* untuk mengubah struktur *field* dan *Drop* untuk menghapus *field*



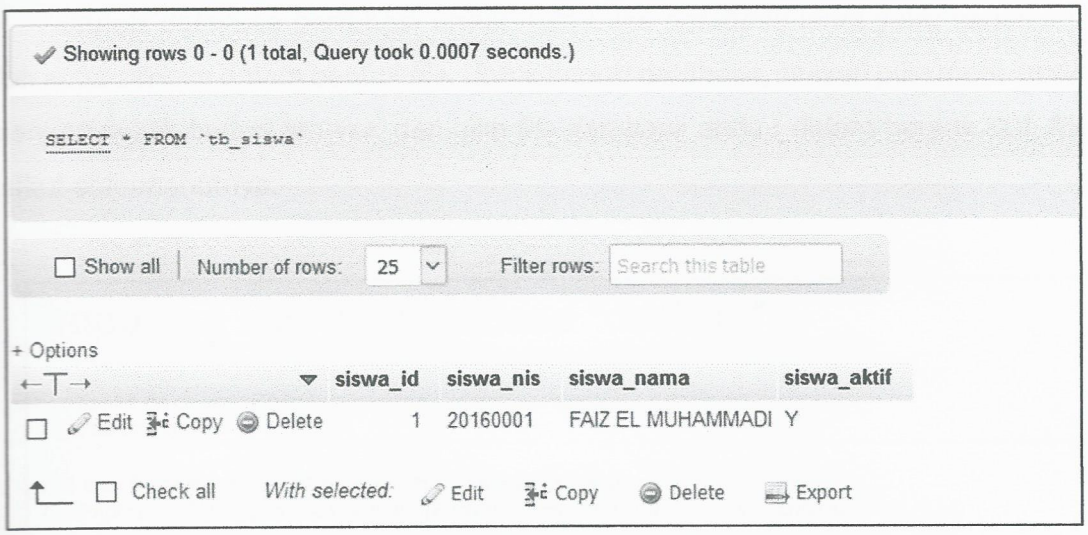
Gambar 9. Tabel baru sukses

h. Pilih menu ***Insert*** untuk memasukkan data kedalam *tabel* yang baru anda buat.



Gambar 10. Insert data

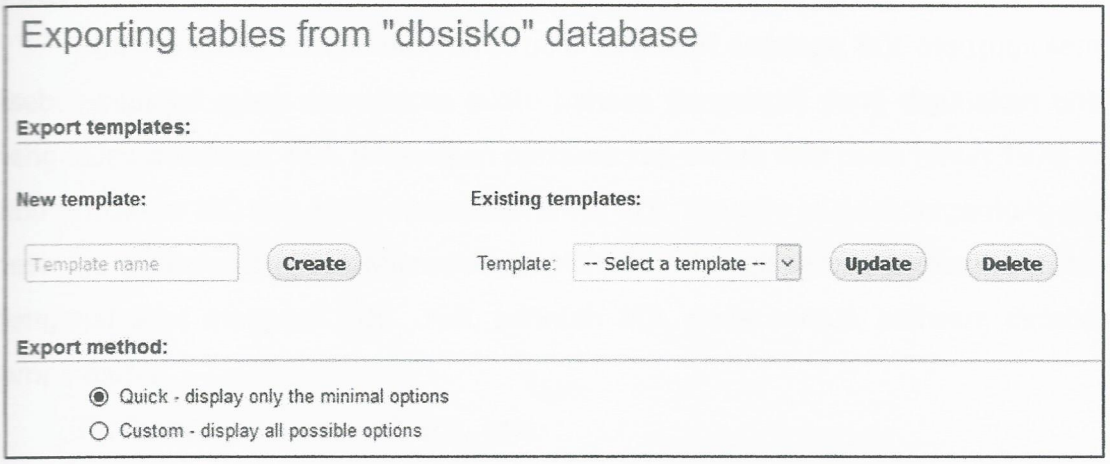
Jika tidak ada kesalahan, berikut *screenshoot* ketika anda telah berhasil menginput data ke dalam *table*.



Gambar 11. Insert Sukses

i. *Export database*

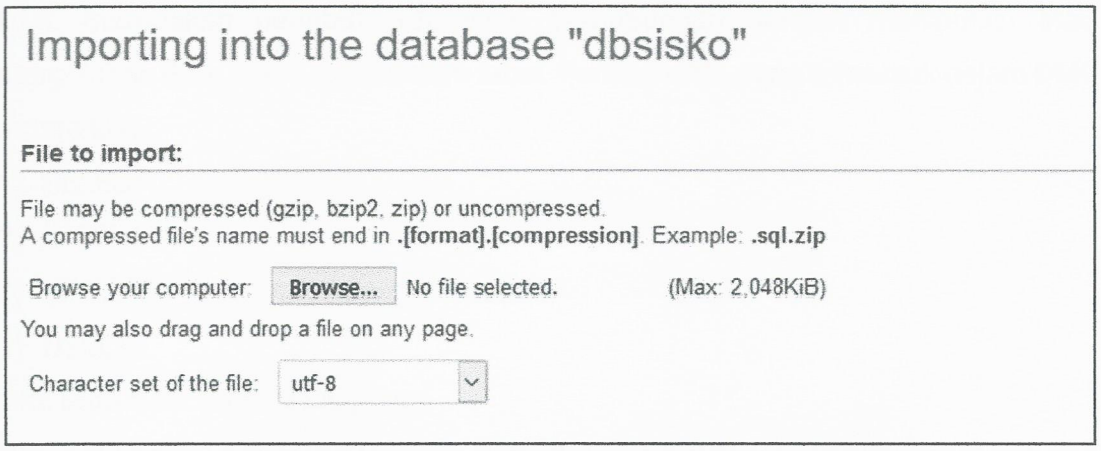
Database yang anda buat, sebenarnya secara otomatis telah tersimpan di dalam *folder* mysql/data. Folder tersebut bisa anda ambil secara manual. Namun hal tersebut kurang efektif, karena bisa jadi aplikasi MySQL berikutnya berbeda dengan yang anda miliki. Untuk menyimpan *database*, bisa anda klik bagian **export,** seperti pada screenshoot berikut.



Gambar 12. *Export database*

j. *Import database*

Import database berfungsi untuk membuka *database* lama yang sebelumnya telah anda simpan dalam bentuk *file*. Untuk *import* database, silakan pilih menu **import,** selanjutnya klik *button browse*, dan pilih *file database* anda (dalam bentuk SQL *file*). Berikut screenshotnya.



Gambar 13. *Import database*

4. SQL

SQL merupakan singkatan dari *Structured Query Language*. SQL atau juga sering disebut sebagai *query* merupakan suatu bahasa (*language*) yang digunakan untuk mengakses *database*. SQL dikenalkan pertama kali dalam IBM pada tahun 1970 dan sebuah standar ISO dan ANSII ditetapkan untuk SQL. Standar ini tidak tergantung pada mesin yang digunakan (IBM, Microsoft atau Oracle). Hampir semua *software database* mengenal atau mengerti SQL. Jadi, perintah SQL pada semua *software database* hampir sama.

Terdapat 3 jenis perintah SQL, yaitu :

a. **DDL atau *Data Definition Language***

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

1) CREATE
2) ALTER
3) RENAME
4) DROP

b. **DML atau *Data Manipulation Language***

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

1) SELECT
2) INSERT
3) UPDATE
4) DELETE

c. **DCL atau *Data Control Language***

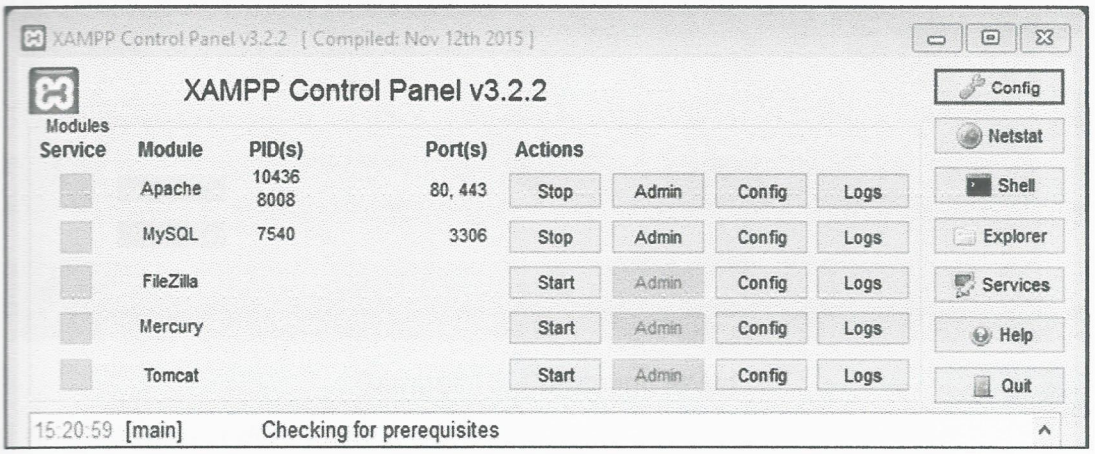
DCL merupakan perintah SQL yang berhubungan dengan manipulasi user dan hak akses (*priviledges*). Perintah SQL yang termasuk dalam DCL antara lain :

1) GRANT
2) REVOKE

Untuk Bab ini, kita akan membahas perintah SQL yang pertama yaitu DDL, sedangkan perintah kedua dan ketiga akan dibahas pada Bab berikutnya.

a. Langkah pertama, pastikan MySQL anda telah aktif.

Klik tombol *start* pada **Apache** dan **MySql** atau start pada tombol MySql saja.



Gambar 14. XAMPP

b. Aktifkan MySQL *Command Line*

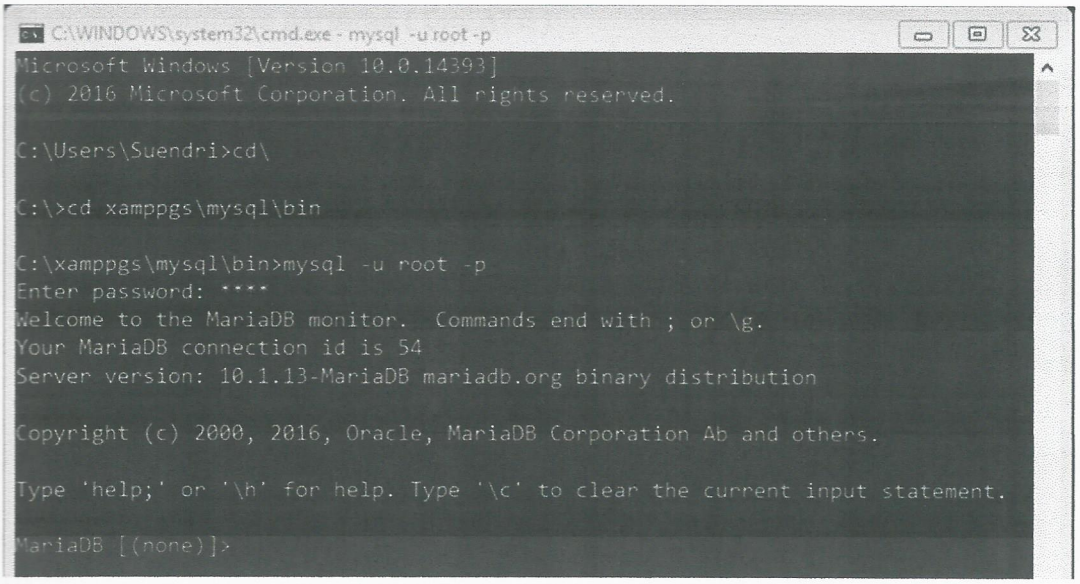
Buka *Command Pront* di windows anda atau dengan jalan pintas dengan menekan Logo **Windos+R** bersamaan pada *keyboard* anda dan mengetikkan perintah **cmd**. Selanjutnya pastikan anda pada partisi C dan masukkan perintah berikut.

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

Atau tambahkan perintah –p jika menggunakan *password.*

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

Berikut *screenshoot* mysql pada *command pront.*



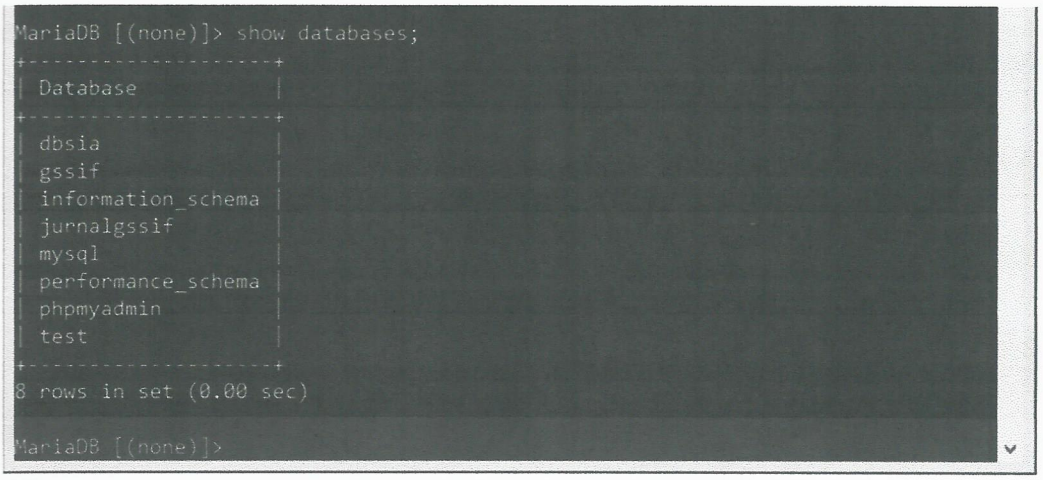
Gambar 15. *Command Pront*

3.1 Melihat Semua *database*

Untuk melihat semua *database* yang ada pada MySQL anda, anda dapat menggunakan perintah berikut:

```
SHOW databases;
```

Semua *database* yang ada akan ditampilkan pada *command pront* seperti pada gambar berikut ini.



Gambar 16. Melihat *Database*

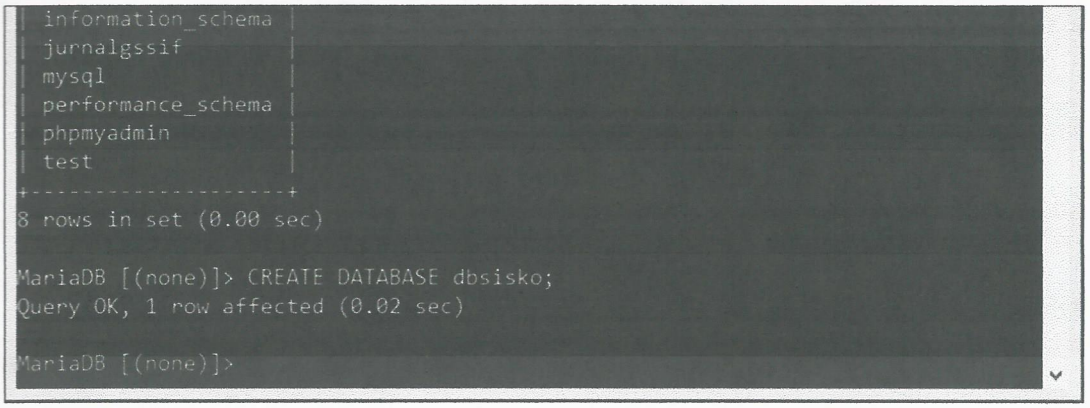
3.2 Membuat database pertama (CREATE)

Untuk membuat *database*, gunakan perintah berikut.

```
CREATE DATABASE nama_database;
```

```
CREATE DATABASE dbsisko;
```

Berikut gambar pembuatan database via *Command Pront* anda.



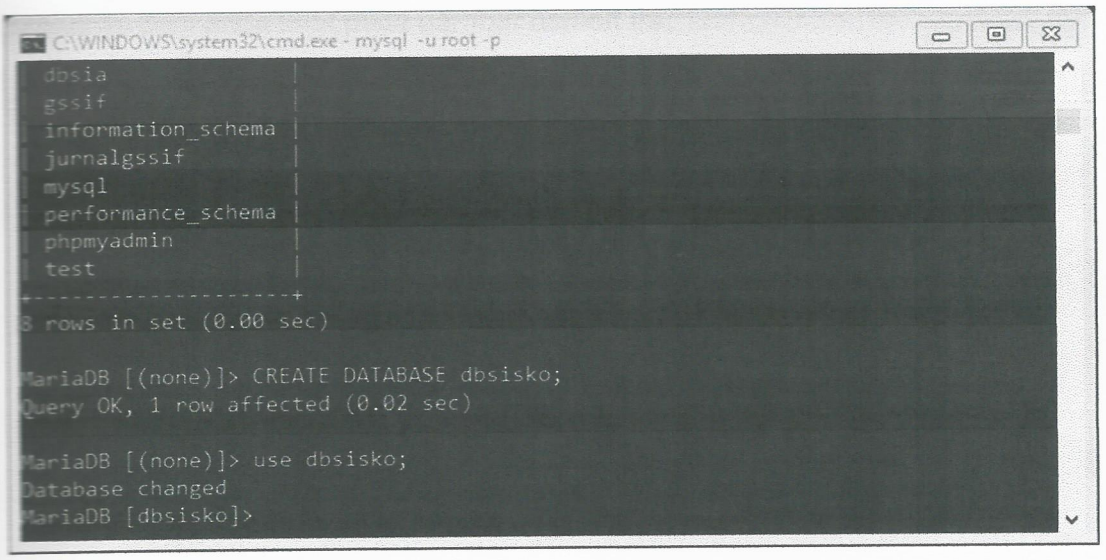
Gambar 17. *Database* baru

3.3 Menggunakan *database*

Sebelum kita melakukan penambahan, pengurangan atau operasi lain dalam *database* tersebut, kita mesti masuk terlebih dahulu ke dalam *database* yang kita buat sebelumnya.

```
USE nama_database;
```

```
USE dbsisko;
```



Gambar 18. Menggunakan *Database*

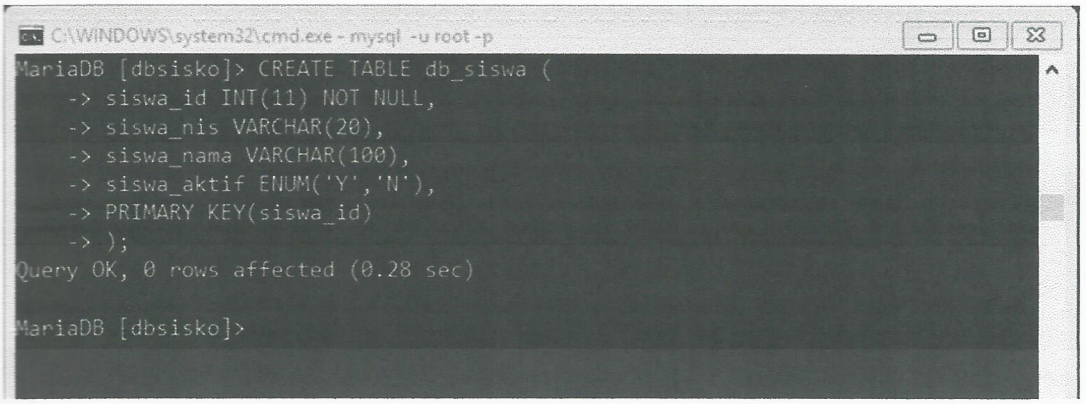
3.4 Membuat tabel

Berikut kode yang anda gunakan untuk membuat tabel.

```
CREATE TABLE nama_table (
field1 type,
field2 type,
...
PRIMARY KEY(fieldx)
);
```

```
CREATE TABLE tb_siswa (
siswa_id INT(11) NOT NULL,
siswa_nis VARCHAR(20),
siswa_nama VARCHAR(100),
siswa_aktif ENUM('Y','N'),
PRIMARY KEY(siswa_id)
);
```

Selanjutnya bisa anda lihat pada gambar berikut ini :



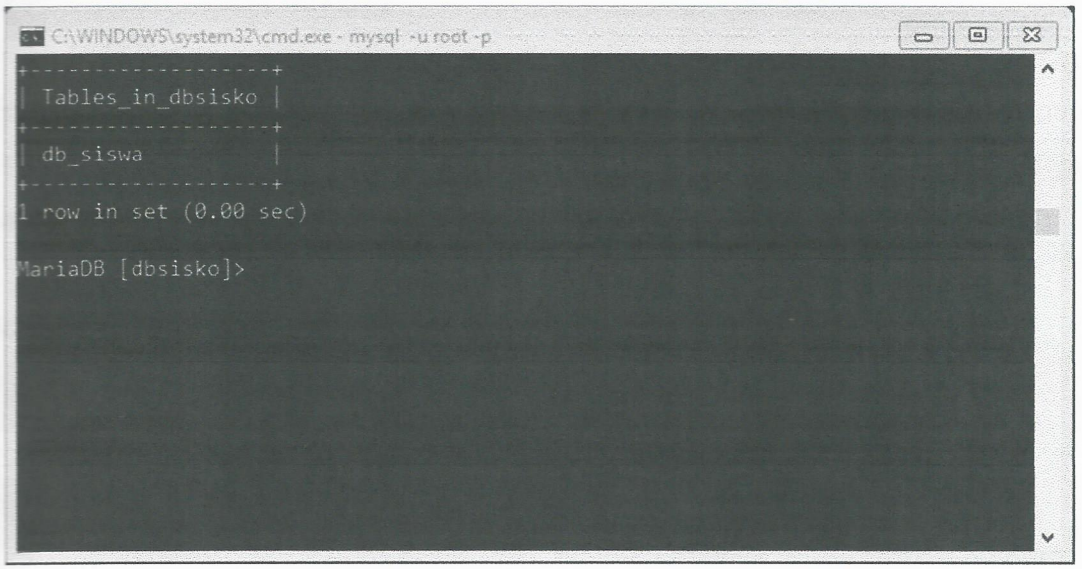
Gambar 19. Membuat tabel

3.5 Melihat tabel

Untuk melihat tabel-tabel yang telah kita buat sebelumnya, anda bisa menggunakan perintah berikut ini:

```
SHOW tables;
```

Berikut tampilan gambarnya.



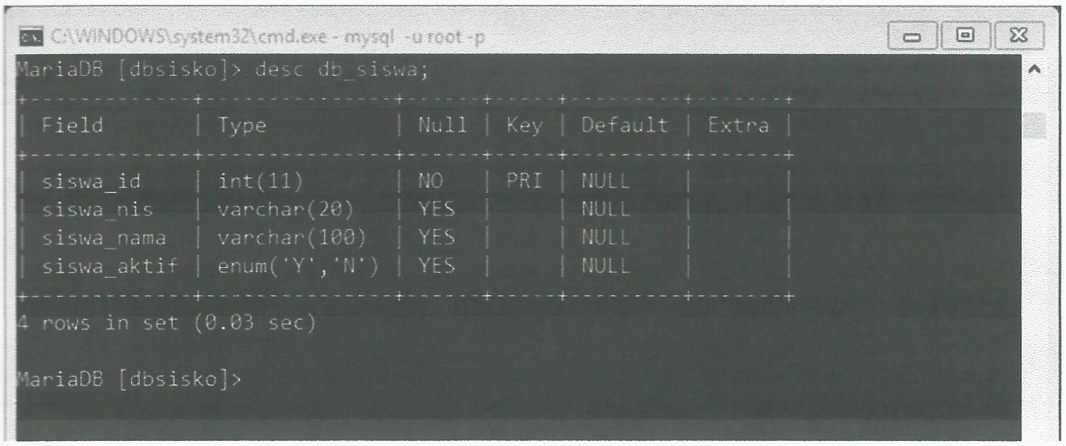
Gambar 20. Melihat tabel

3.6 Melihat Struktur tabel

Untuk melihat struktur dari tabel yang kita buat, ketikkan kode berikut:

```
DESC nama_table;
```

```
DESC db_siswa;
```



Gambar 21. Struktur Tabel

3.7 Perobahan Tabel (ALTER)

ALTER merupakan perintah yang digunakan untuk melakukan perobahan atau pengeditan terhadap tabel atau *field* yang anda dalam *database*, perobahan termasuk melakukan penambahan, pengeditan bahkan penghapusan dari tabel dan *field* yang telah ada sebelumnya.

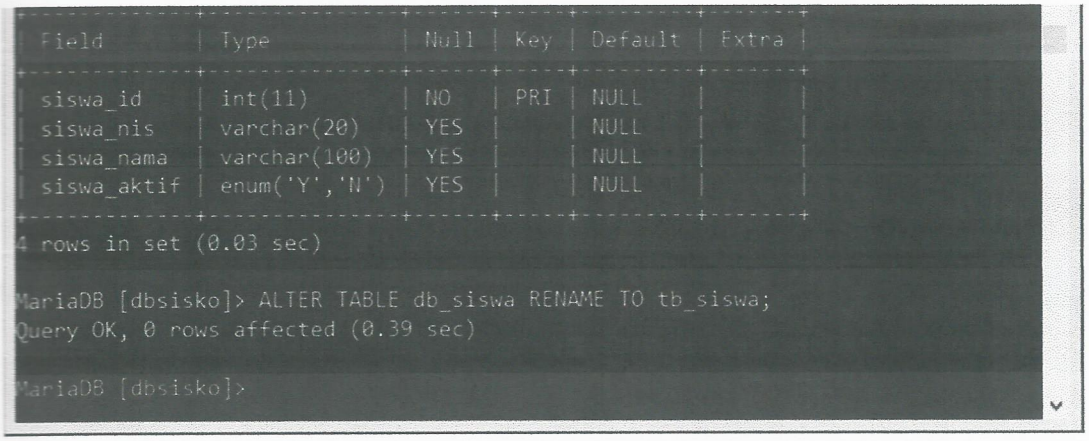
a. Merobah nama tabel

Jika anda ingin merobah atau mengganti nama *field,* gunakan perintah berikut.

```
ALTER TABLE nama_table RENAME TO nama_lain;
```

```
ALTER TABLE db_siswa RENAME TO tb_siswa;
```

Hasilnya dapat anda lihat pada gambar berikut ini.



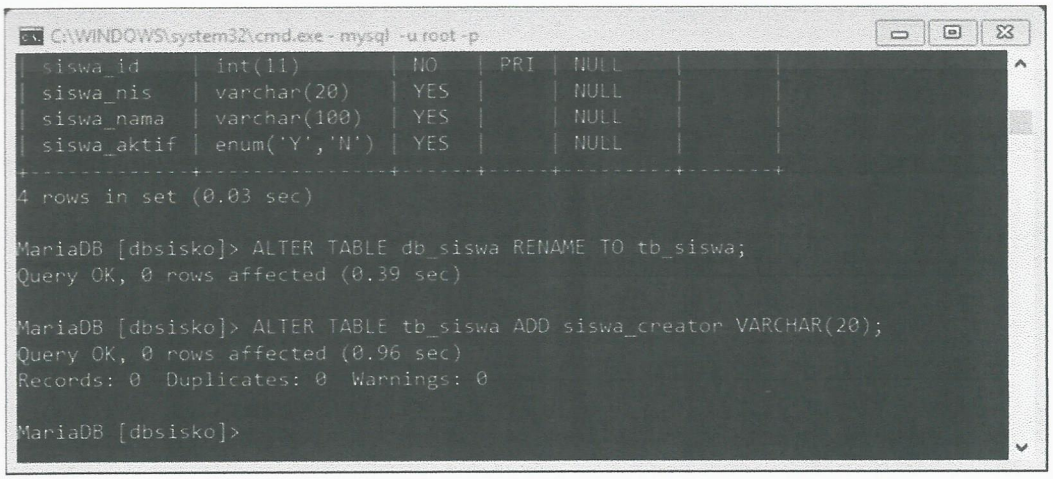
Gambar 22. Merobah tabel

b. Menambahkan / menyisipkan 1 *field* pada tabel.

```
ALTER TABLE nama_table alter_option;
```

```
ALTER TABLE tb_siswa ADD siswa_creator VARCHAR(20);
```

Berikut gambar tampilannya.



Gambar 23. Menyisip *Field*

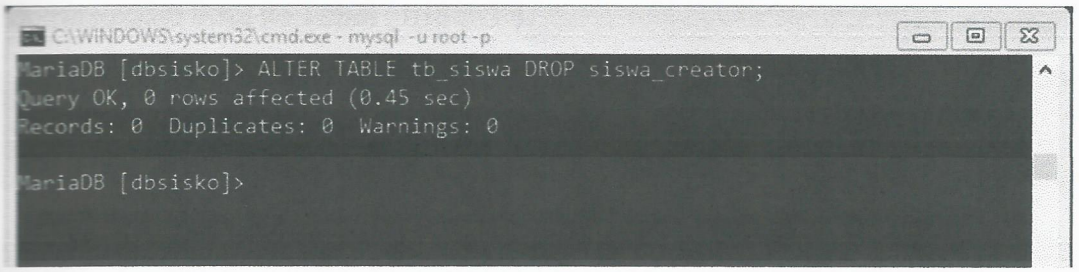
c. Menghapus 1 *field* pada tabel

Gunakan perintah berikut jika anda ingin menghapus 1 *field* dari tabel.

```
ALTER TABLE nama_table DROP nama_field;
```

```
ALTER TABLE tb_siswa DROP siswa_creator;
```

Hasilnya bisa anda lihat pada gambar berikut ini.

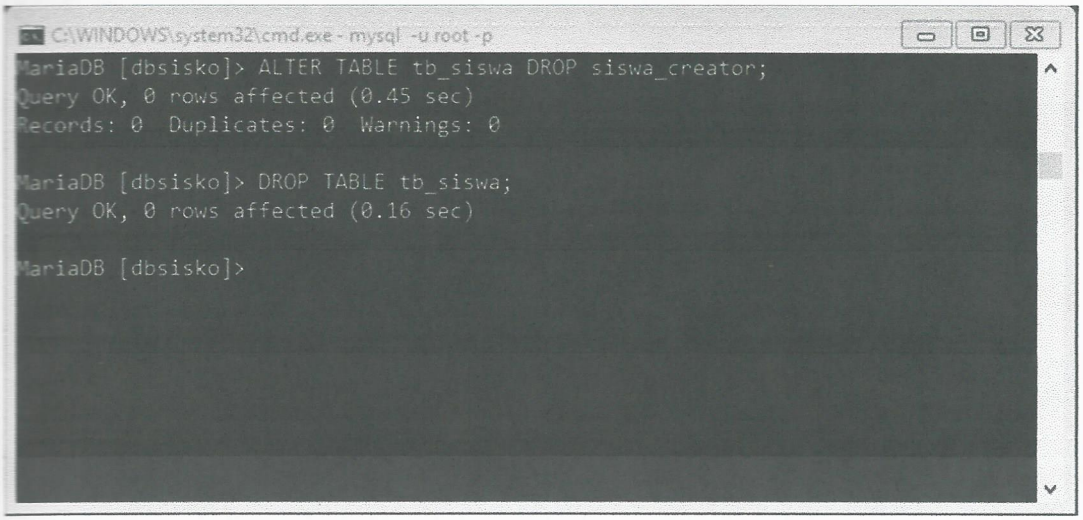


Gambar 24. Menghapus *Field*

3.8 Menghapus Tabel (DROP)

```
DROP TABLE nama_table;
```

```
DROP TABLE tb_siswa;
```



Gambar 25. Menghapus Tabel

BAB III
DATABASE (DML)

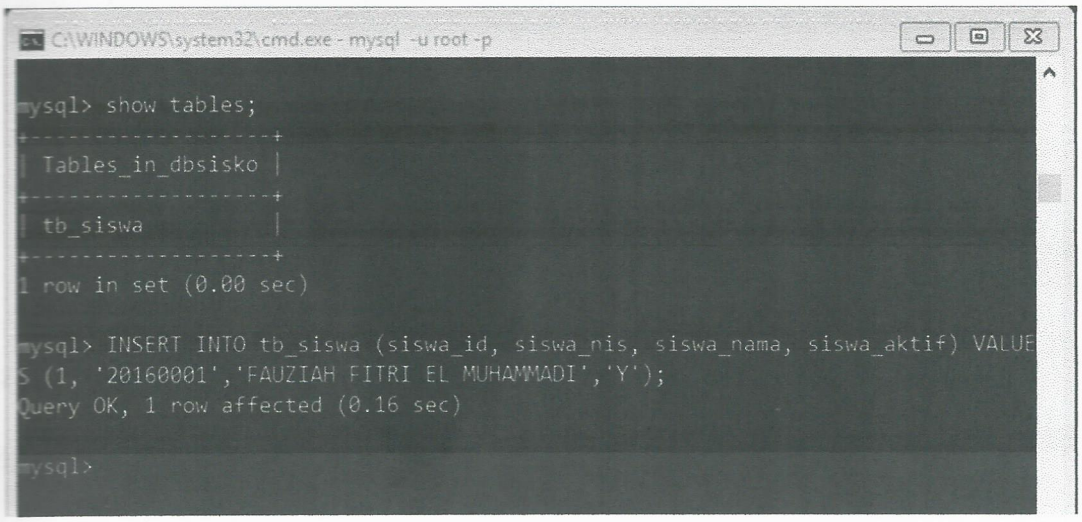
Pada BAB ini, kita akan membahas jenis perintah kedua pada MySQL yaitu DML *(Data Manipulation Language)*. Perintah tersebut meliputi SELECT, INSERT, UPDATE dan DELETE. Diasumsikan anda masih memiliki database *dbpertama* dan tabel *table1*. Bukalah *MySQL Command Line* anda.

1. Memasukkan Data ke Tabel (INSERT).

```
INSERT INTO nama_table (field1,field2,...,fieldx)  VALUES
(value1, value2, ..., valuex);
```

```
INSERT INTO tb_siswa (field1,field2,...,fieldx)  VALUES
(value1, value2, ..., valuex);
```

Anda bisa melihatnya pada gambar berikut ini.



Gambar 26. *Insert* Data

Perintah INSERT tersebut diatas berlaku untuk memasukkan data sesua dengan *field* yang diinginkan, jika data wajib dimasukkan kedalam semua *field,* maka

perintah INSERT tersebut bisa disingkat tanpa menuliskan nama *field* seperti dibawah ini:

2. Melihat isi tabel (SELECT).

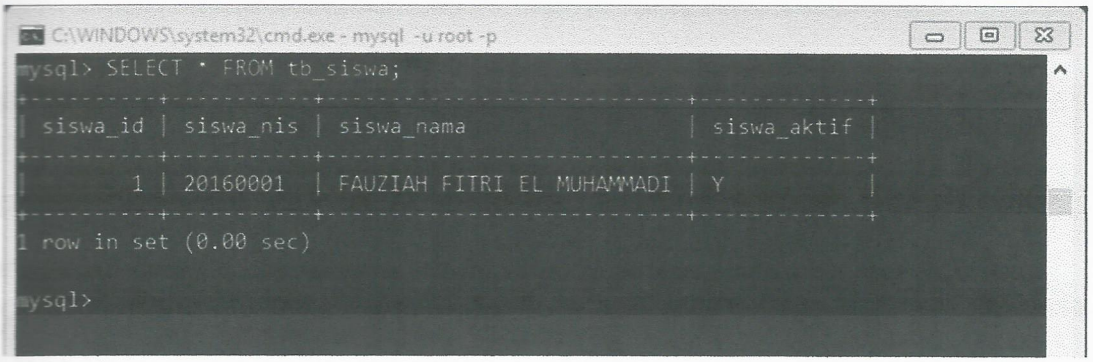
2.1 Melihat Seluruh isi tabel

Untuk melihat seluruh isi tabel bisa menggunakan tanda asterisk **"*"** atau tanda bintang.

```
SELECT * FROM nama_table;
```

```
SELECT * FROM tb_siswa;
```

Berikut *screenshot*nya :



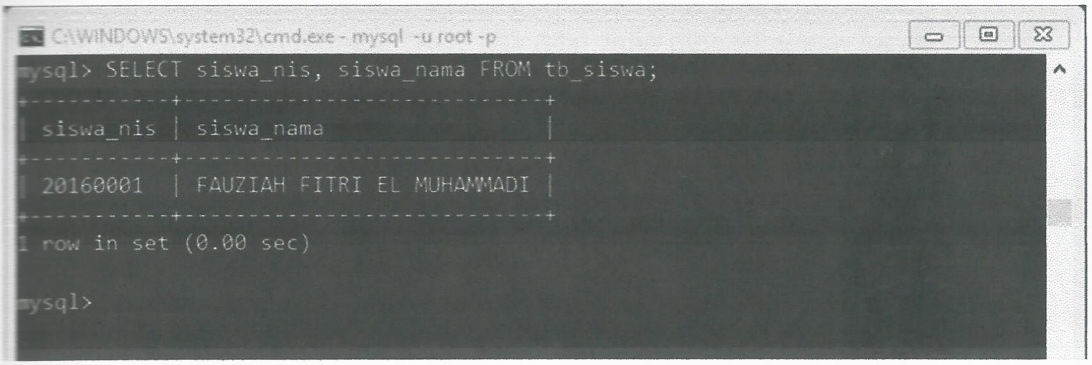
Gambar 27. Melihat isi tabel

2.2 Melihat *Field* tertentu pada tabel

```
SELECT nama_field1, nama_field2, ... FROM nama_table;
```

```
SELECT siswa_nis, siswa_nama FROM tb_siswa;
```

Berikut tampilan hasil dari perintah diatas.



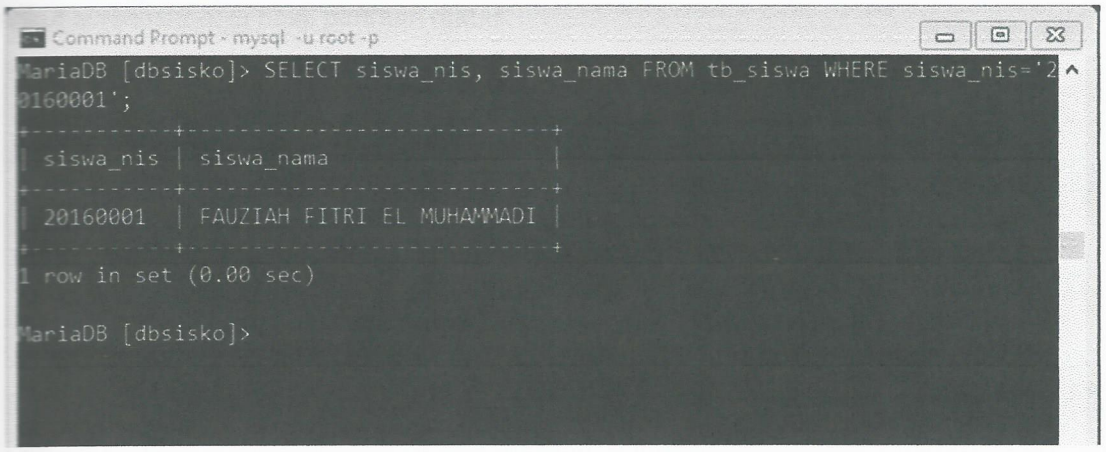
Gambar 28. Melihat field tertentu

2.3 Melihat *Field* tertentu pada tabel dengan kondisi tertentu

```
SELECT nama_field1, nama_field2, ... FROM nama_table WHERE
nama_field='kondisi';
```

```
SELECT siswa_nis, siswa_nama FROM tb_siswa WHERE
siswa_nis='20160001';
```

Berikut *screenshoot*nya :



Gambar 29. Melihat *field* tertentu dan kondisi tertentu

Berikut ini operator perbandingan yang dapat digunakan untuk membandingkan dua buah nilai dalam MySQL :

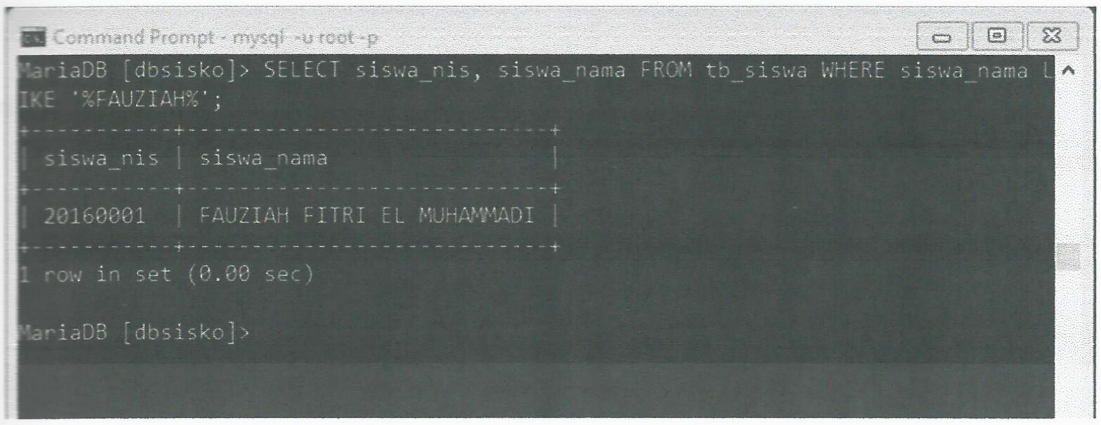
a. Operator **=**, akan bernilai TRUE jika nilai yang dibandingkan sama.
b. Operator **!=** atau **<>**, akan bernilai TRUE jika nilai yang dibandingkan TIDAK SAMA (berbeda).
c. Operator **>**, akan bernilai TRUE jika nilai yang pertama lebih besar dari nilai kedua.
d. Operator **>=**, akan bernilai TRUE jika nilai yang pertama lebih besar atau sama dengan nilai kedua.
e. Operator **<**, akan bernilai TRUE jika nilai yang pertama lebih kecil dari nilai kedua

2.4 Melihat *Field* tertentu pada tabel dengan isi *field* menyerupai

```
SELECT nama_field1, nama_field2, ... FROM nama_table WHERE
nama_field LIKE '%kondisi%';
```

```
SELECT siswa_nis, siswa_nama FROM tb_siswa WHERE siswa_nama
LIKE '%FAUZIAH%';
```

Berikut *screenshot*nya hasil perintah diatas.

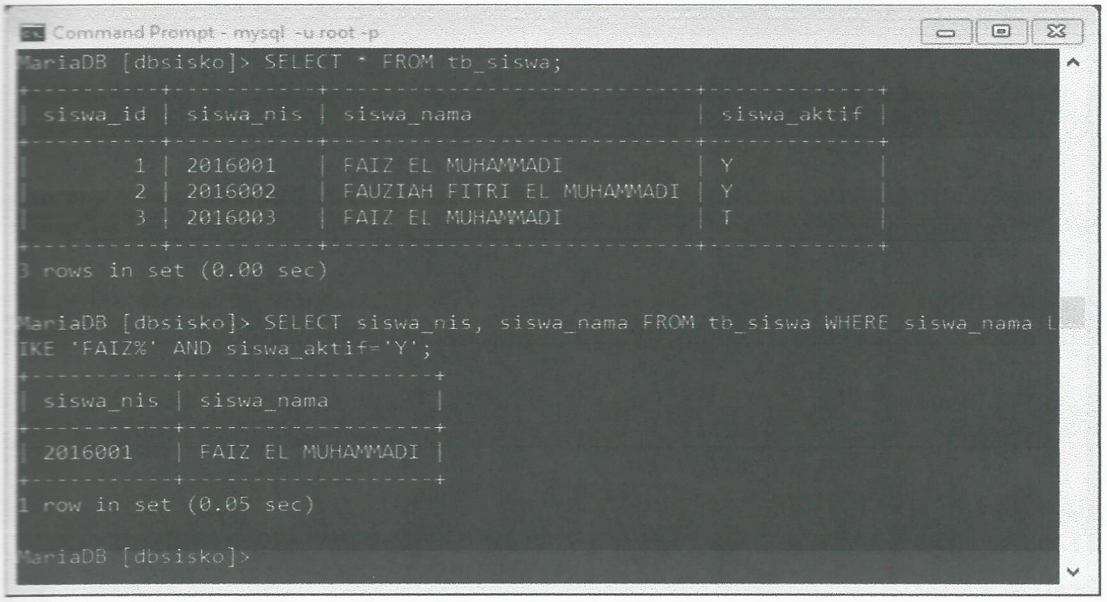


Gambar 30. Melihat isi dengan *field* menyerupai

2.5 Melihat *Field* tertentu pada tabel dengan beberapa syarat lebih dari satu

```
SELECT nama_field1, nama_field2, ... FROM nama_table WHERE
nama_field1='kondisi1' AND nama_field2='kondisi2';
```

```
SELECT siswa_nis, siswa_nama FROM tb_siswa WHERE siswa_nama
LIKE 'FAIZ%' AND siswa_aktif='Y';
```



Gambar 31. Melihat isi tabel dengan syarat tertentu

Berikut ini operator penghubung yang dapat digunakan untuk menghubungkan antara dua kondisi dalam MySQL :

a. Operator && atau AND, akan menghubungkan dua kondisi dimana akan bernilai TRUE jika kedua kondisi bernilai TRUE.
b. Operator || atau OR, akan menghubungkan dua kondisi dimana akan bernilai TRUE jika salah satu atau kedua kondisi bernilai TRUE.
c. Operator !, akan me-*reverse* nilai suatu kondisi logika.

2.5 Melihat *Field* tertentu pada tabel diurutkan berdasarkan field lainnya

Untuk mengurutkan *record* yang ditampilkan bisa menggunakan perintah ASC atau DESC. ASC (Ascending) merupakan perintah mengurutkan data dari terkecil hingga data terbesar sedangkan DESC (*Descending*) berfungsi untuk mengurutkan data dari terbesar hingga terkecil.

```
SELECT nama_field1, nama_field2, ... FROM nama_table ORDER
BY nama_field DESC;
```

```
SELECT siswa_nis, siswa_nama FROM tb_siswa ORDER BY
siswa_nama DESC;
```

2.5 Melihat *Field* tertentu pada tabel dengan jumlah record tertentu

```
SELECT nama_field1, nama_field2, ... FROM nama_table ORDER
BY nama_field LIMIT limit_awal,jumlah_record;
```

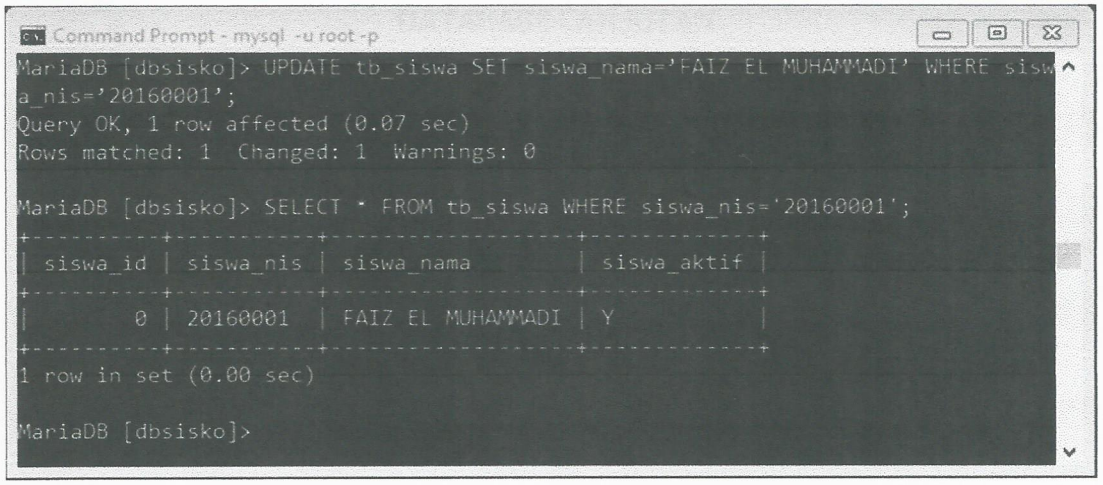
```
SELECT siswa_nis, siswa_nama FROM tb_siswa ORDER BY
siswa_nama LIMIT 0,2;
```

3. Memperbaharui *field* (UPDATE)

```
UPDATE nama_tabel SET nama_field='nilai_baru' WHERE
nama_field='nilai';
```

```
UPDATE tb_siswa SET siswa_nama='FAIZ EL MUHAMMADI' WHERE
siswa_nis='20160001';
```

Hasil perintah tersebut seperti pada gambar dibawah ini.



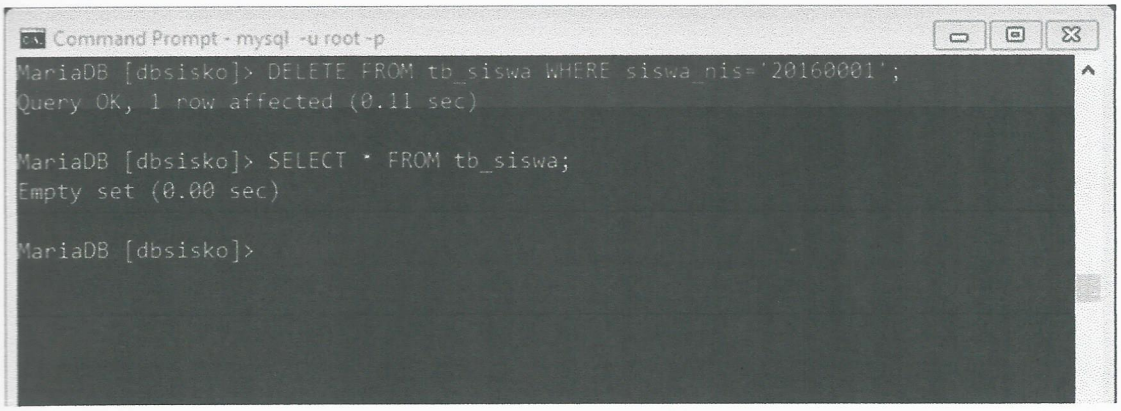
Gambar 32. *Update* Tabel

4. Menghapus *field* (DELETE)

```
DELETE FROM nama_tabel WHERE nama_field='nilai';
```

```
DELETE FROM tb_siswa WHERE siswa_nis='20160001';
```

Berikut tampilan hasilnya.



Gambar 33. *Delete field*

BAB IV
DATABASE LANJUTAN

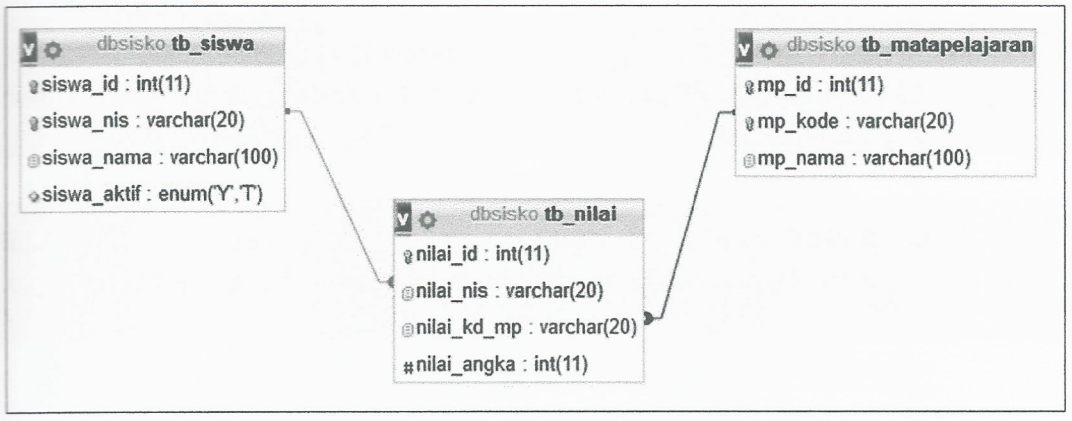
1. JOIN

JOIN adalah perintah SQL yang berfungsi untuk melakukan relasi antara kedua tabel atau lebih yang saling memiliki hubungan / relasi (ditandai dengan adanya primary key pada tabel master dan foregn key pada tabel transaksi). Ada beberap jenis join dimana diantaranya adalah :

a. **Where**
b. **Inner Join**
c. **Outer Join**
    1) **Right Join**
    2) **Left Join**

Tetapi pada dasarnya jika ingin menggabungkan kedua tabel atau lebih maka cukup menggunakan perintah JOIN saja maka tabel akan saling berelasi asalkan data yang direlasikan benar. Selain menggunakan Join cara lain untuk menghubungkan antara tabel dapat juga digunakan perintah WHERE.

Berikut contoh *database* dari latihan sebelumnya yang sudah dilengkapi dengan tabel matapelajaran dan nilai. Dimana antara siswa dengan nilai mempunyai hubungan *one-to-many* sedangkan antara matapelajaran dengan nilai mempunya hubungai *one-to-one*.



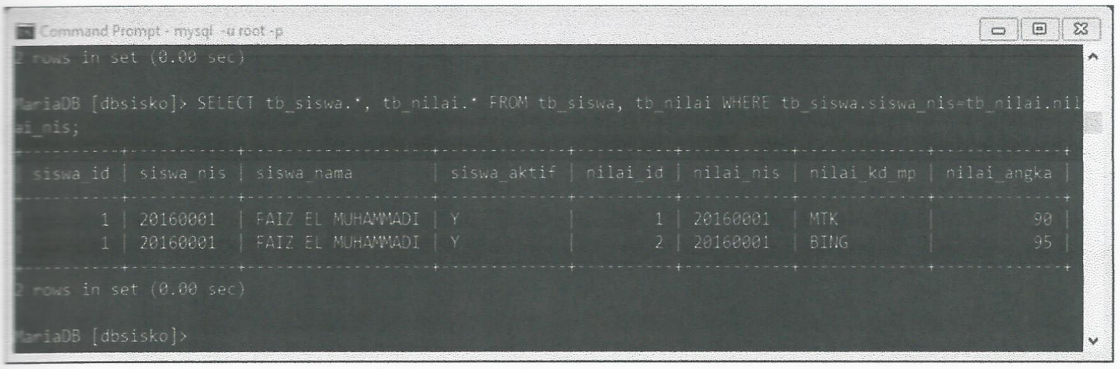
Gambar 34. Tabel *Relationship*

a. WHERE

Dengan WHERE, tabel akan digabungkan dua arah, sehingga tidak ada data yang NULL di satu sisi.

```
SELECT nama_table1.*, nama_tabel2.*, ... FROM nama_tabel1,
nama_tabel2 WHERE nama_tabel1.PK=nama_tabel2.PK;
```

```
SELECT tb_siswa.*, tb_nilai.* FROM tb_siswa, tb_nilai WHERE
tb_siswa.siswa_nis=tb_nilai.nilai_nis;
```



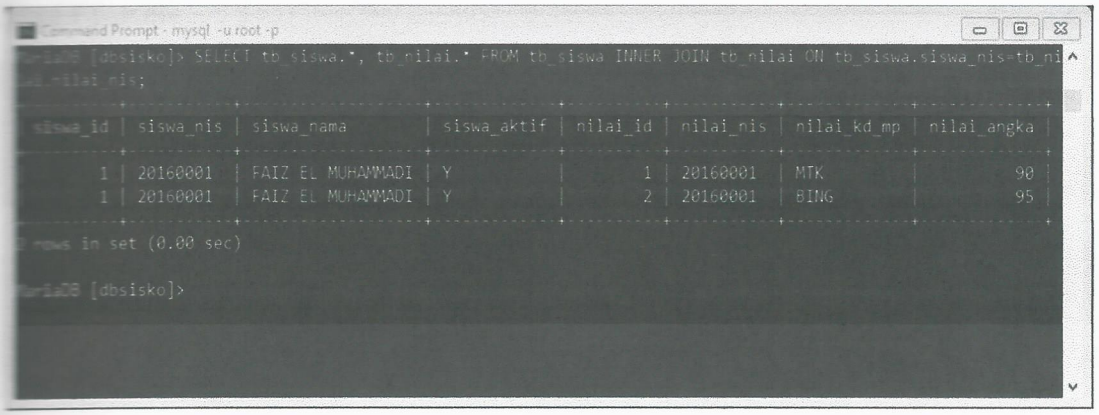
Gambar 35. *Select Where*

b. INNER JOIN

Seperti pada penggunaan perintah WHERE sebelumnya, dengan inner join tabel akan digabungkan dua arah, sehingga tidak ada data yang NULL di satu sisi.

```
SELECT nama_table1.*, nama_tabel2.*, ... FROM nama_tabel1
INNER JOIN nama_tabel2 ON nama_tabel1.PK=nama_tabel2.PK;
```

```
SELECT tb_siswa.*, tb_nilai.* FROM tb_siswa INNER JOIN
tb_nilai ON tb_siswa.siswa_nis=tb_nilai.nilai_nis;
```



Gambar 36. *Select Inner Join*

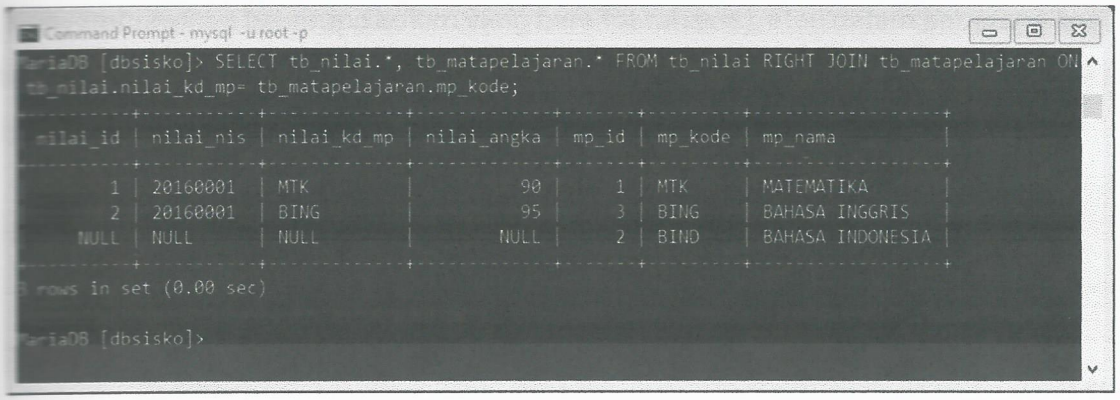
c. OUTER JOIN

Dengan outer join, tabel akan digabungkan satu arah, sehingga memungkinkan ada data yang NULL (kosong) di satu sisi.

1). RIGHT JOIN

```
SELECT nama_table1.*, nama_tabel2.*, ... FROM nama_tabel1
RIGHT JOIN nama_tabel2 ON nama_tabel1.PK=nama_tabel2.PK;
```

```
SELECT tb_nilai.*, tb_matapelajaran.* FROM tb_nilai RIGHT
JOIN tb_matapelajaran ON tb_nilai.nilai_kd_mp=
tb_matapelajaran.mp_kode;
```

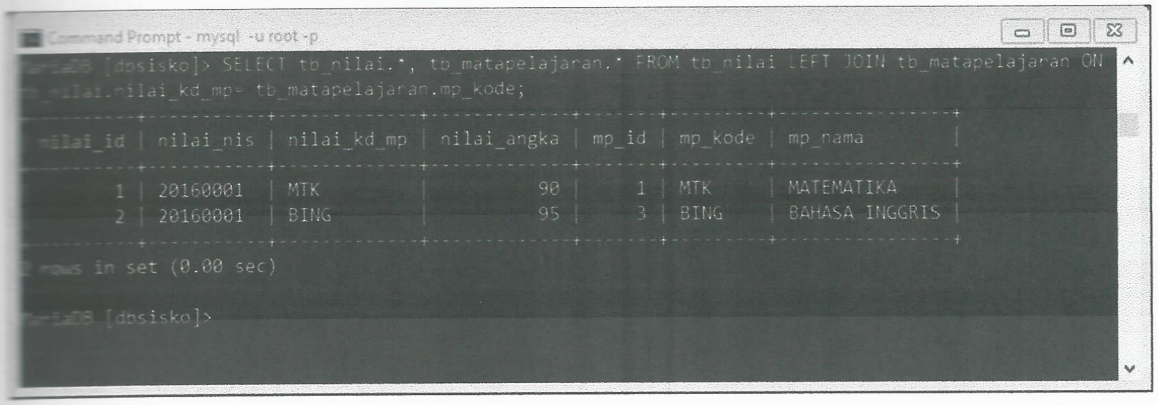


Gambar 37. *Select Outer Join*

2). LEFT JOIN

```
SELECT nama_table1.*, nama_tabel2.*, ... FROM nama_tabel1
LEFT JOIN nama_tabel2 ON nama_tabel1.PK=nama_tabel2.PK;
```

```
SELECT tb_nilai.*, tb_matapelajaran.* FROM tb_nilai LEFT
JOIN tb_matapelajaran ON tb_nilai.nilai_kd_mp=
tb_matapelajaran.mp_kode;
```



Gambar 38. *Select Left Join*

B VIEW

Di dalam MySQL, *View* dapat didefenisikan sebagai 'tabel virtual'. Tabel ini bisa berasal dari tabel lain, atau gabungan dari beberapa tabel. Tujuan dari pembuatan VIEW adalah untuk kenyamanan (mempermudah penulisan *query*), untuk keamanan (menyembunyikan beberapa kolom yang bersifat rahasia), atau dalam beberapa kasus bisa digunakan untuk mempercepat proses menampilkan data (terutama jika kita akan menjalankan *query* tersebut secara berulang).

```
CREATE VIEW nama_view AS SELECT nama_tabel1.*,
nama_tabel2.*, ... FROM nama_tabel1 INNER JOIN nama_tabel1
ON nama_tabel1.PK = nama_tabel2.PK;
```

BAB V
HTML

Hyper Text Markup Language (HTML) adalah sebuah *bahasa markah* yang digunakan untuk membuat sebuah halaman web, menampilkan berbagai informasi di dalam sebuah penjelajah web Internet dan pemformatan hiperteks sederhana yang ditulis dalam berkas format ASCII agar dapat menghasilkan tampilan wujud yang terintegerasi. Dengan kata lain, berkas yang dibuat dalam perangkat lunak pengolah kata dan disimpan dalam format ASCII normal sehingga menjadi halaman web dengan perintah-perintah HTML. Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan SGML (*Standard Generalized Markup Language*), HTML adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman web. HTML saat ini merupakan standar Internet yang didefinisikan dan dikendalikan penggunaannya oleh World Wide Web Consortium (W3C). HTML dibuat oleh kolaborasi Caillau TIM dengan Berners-lee Robert ketika mereka bekerja di CERN pada tahun 1989 (CERN adalah lembaga penelitian fisika energi tinggi di Jenewa).

1. Sejarah

Pada tahun 1980 seorang ahli fisika, Tim Berners-Lee, dan juga seorang kontraktor di CERN (Organisasi Eropa untuk Riset Nuklir) mengusulkan dan menyusun ENQUIRE, sebuah sistem untuk ilmuwan CERN dalam membagi dokumen. Sembilan tahun kemudian, Berners-Lee mengusulkan adanya sistem markah berbasis internet. Berners-Lee menspesifikasikan HTML dan menulis jaringan beserta perangkat lunaknya di akhir 1990. Pada tahun yang sama, Berners-Lee dan Robert Cailliau, insinyur sistem data CERN berkolaborasi dalam sebuah permintaan untuk pendanaan, namun tidak diterima secara resmi oleh CERN. Di catatan pribadinya sejak 1990 dia mendaftar "beberapa dari banyak daerah yang menggunakan hypertext" dan pertama-tama menempatkan sebuah ensiklopedia.

Penjelasan pertama yang dibagi untuk umum dari HTML adalah sebuah dokumen yang disebut "Tanda HTML", pertama kali disebutkan di Internet oleh Tim Berners-Lee pada akhir 1991. Tanda ini menggambarkan 18 elemen awal mula, versi sederhana dari HTML. Kecuali untuk *tag hyperlink*, yang sangat dipengaruhi oleh SGMLguid, in-house Standard Generalized Markup Language (SGML) berbasis format dokumen di CERN. Sebelas elemen ini masih ada di HTML 4.

HTML adalah bahasa markah yang digunakan peramban untuk menafsirkan dan menulis teks, gambar dan bahan lainnya ke dalam halaman web secara visual maupun suara. Karakteristik dasar untuk setiap item dari markah HTML didefinisikan di dalam peramban, dan karakteristik ini dapat diubah atau ditingkatkan dengan menggunakan tambahan halaman web desainer CSS. Banyak elemen teks ditemukan di laporan teknis ISO pada tahun 1988 TR 9537 *Teknik untuk menggunakan SGML*, yang pada gilirannya meliputi fitur bahasa format teks awal seperti yang digunakan oleh komandan *RUNOFF* dikembangkan pada awal 1960-an untuk sistem operasi: perintah-perintah format ini berasal dari perintah yang digunakan oleh pengetik untuk memformat dokumen CTSS secara manual. Namun, konsep SGML dari markah umum didasarkan pada unsur-unsur daripada hanya efek cetak, dengan pemisahan struktur dan markah juga; HTML telah semakin bergerak ke arah ini dengan CSS. Versi HTML bisa dilihat pada garis waktu berikut ini:

24 November 1995

HTML 2.0 dipublikasikan sebagai IETF RFC 1866. Penambahan RFC memperbanyak kemampuan untuk:

a. November 25, 1995: RFC 1867 (mengunggah *file* berdasarkan bentuk)
b. May 1996: RFC 1942 (tabel)
c. August 1996: RFC 1980 (peta gambar berbasis klien)
d. January 1997: RFC 2070 (internasionalisasi)

14 Januari 1997

HTML 3.2 dipublikasikan sebagai Konsorsium World Wide Web. Versi ini merupakan versi pertama yang dikembangkan dan distandarisasi secara khusus oleh Konsorsium World Wide Web, sebagaimana IETF sudah menutup kelompok kerja

HTMLnya pada 12 September, 1996. Pada awalnya disebut "Wilbur",[10] HTML 3.2 menghilangkan rumus matematika sama sekali yang sedang berkonsiliasi atas kasus tumpang tindih antara berbagai kepemilikan dan mengadopsi sebagian besar tanda markah visual dari Netscape. Elemen kedip dari Netscape dan elemen *marquee* besutan Microsoft dihilangkan karena kesepakatan bersama antara kedua perusahaan.[11] Sebuah markup untuk rumus matematika serupa dengan yang ada dalam HTML tidak memiliki standar sampai 14 bulan kemudian di MathML.

18 December 1997

HTML 4.0 telah dipublikasi sebagai sebuah *W3C Recommendation* yang menawarkan tiga variasi yang berbeda:

a. *Strict, in which deprecated elements are forbidden,*
b. *Transitional, in which deprecated elements are allowed,*
c. *Frameset, in which mostly only frame related elements are allowed.*

Initially code-named "Cougar", HTML 4.0 mengadopsi banyak tipe elemen dan atribut yang spesifik untuk peramban. HTML 4 adalah sebuah aplikasi SGML sesuai dengan ISO 8879 – SGML.

28 Oktober 2014

- HTML 5 (stabil) rekomendasi W3C.

2. HTML5

HTML5 adalah sebuah bahasa markah untuk menstrukturkan dan menampilkan isi dari Waring Wera Wanua, sebuah teknologi inti dari Internet. HTML5 adalah revisi kelima dari HTML (yang pertama kali diciptakan pada tahun 1990 dan versi keempatnya, HTML4, pada tahun 1997) dan hingga bulan Juni 2011 masih dalam pengembangan. Tujuan utama pengembangan HTML5 adalah untuk memperbaiki teknologi HTML agar mendukung teknologi multimedia terbaru, mudah dibaca oleh manusia dan juga mudah dimengerti oleh mesin.

HTML5 merupakan salah satu karya Konsortium Waring Wera Wanua (*World Wide Web Consortium*, W3C) untuk mendefinisikan sebuah bahasa markah tunggal yang dapat ditulis dengan cara HTML ataupun XHTML. HTML5 merupakan jawaban

atas pengembangan HTML 4.01 dan XHTML 1.1 yang selama ini berjalan terpisah, dan diimplementasikan secara berbeda-beda oleh banyak perangkat lunak pembuat web.

2.1 Proses Standarisasi W3C

Kelompok kerja untuk teknologi aplikasi *web hypertext* (WHATWG) mulai menspesifikasikan HTML5 pada bulan juni 2004 dengan nama Web Applications 1.0, hingga pada bulan maret 2010 spesifikasi ini masuk ke bagian *draft* standar di WHATWG, dan ke dalam bagian pengurusan *draft* di W3C. Ian Hickson mewakili Google ,Inc menjadi editor HTML5.

Pada tahun 2007 Spesifikasi HTML5 diadopsi sebagai pekerjaan permulaan untuk grup baru yang mengurus HTML di *World Wide Web Consorsium* (W3C). Grup ini pertama kali mempublikasikan hasil draft pekerjaan pertama mereka pada tanggal 22 januari 2008. Spesifikasi ini berstatus dalam tahap pengerjaan, dan diperkirakan akan tetap demikian selama bertahun-tahun, meskipun sebagian dari HTML5 sudah dalam tahap penyelesaian dan diimplementasikan pada penjelajah web sebelum keseluruhan spesifikasinya mencapai status rekomendasi final.

Berdasarkan pada jadwal kerja W3C, HTML5 diperkirakan menjadi kandidat rekomendasi pada akhir tahun 2010.Namun, publikasi pertama draft HTML 5 meleset selama 8 bulan.Permintaan dokumen terakhir dan tahap kandidat rekomendasi diharapkan dapat dicapai pada tahun 2008, tetapi hingga bulan Juli 2010 HTML5 masih dalam tahapan draft pengerjaan di W3C. WHATWG telah meminta penyelesaian terakhir untuk HTML5 sejak bulan oktober tahun 2009.

Editor HTML5, Ian Hickson, berharap spesifikasi HTML5 dapat mencapai tahap kandidat rekomendasi pada tahun 2012. Kriteria di W3C agar sebuah spesifikasi dapat berstatus - Direkomendasikan - adalah "yang kedua: 100% selesai dan penerapannya dapat dilakukan pada dua atau lebih sistem yang berbeda". Pada wawancaranya dengan TechRepublic, Hickson memperkirakan hal ini baru akan terjadi pada tahun 2022 atau setelahnya. Meski demikian, banyak bagian dari spesifikasi sudah stabil dan telah dapat diterapkan pada produk.

2.2 *Markup*

Pada HTML5 diperkenalkan beberapa elemen baru dan atribut yang merefleksikan tipikal penggunaan website modern. Beberapa diantaranya adalah pergantian yang bersifat semantik pada blok yang umum digunakan: yaitu elemen dan *inline* , sebagai contoh (sebagai blok navigasi website) dan <footer> (biasanya dikaitkan pada bagian bawah suatu website atau baris terakhir dari kode html). Banyak elemen lain yang memberikan kegunaan baru melalui antar muka yang telah distandarkan, seperti elemen multimedia <audio> dan <video>. Beberapa elemen yang telah ditinggalkan juga ditiadakan, termasuk elemen presentasi semata seperti <font> dan , yang sebenarnya dapat dikerjakan menggunakan Cascading Style Sheet (CSS).

2.3 API

Untuk menambah keluwesan pemformatan, pada HTML5 telah dispesifikasikan pengkodean application programming interfaces (APIs). Antarmuka document object model (DOM) yang ada dikembangkan dan fitur de facto didokumentasikan. Beberapa APIs terbaru pada HTML5 antara lain :

a. Elemen *canvas*, sebagai mode untuk menggambar objek dua dimensi (2D).
b. *Timed media playback*.
c. Media penyimpanan *offline*
d. Penyuntingan dokumen.
e. *Drag-and-Drop*.
f. *Cross-document messaging*
g. Manajemen sejarah kunjungan penjelajah web.
h. Tipe MIME dan penanggung jawab protokol registrasi.

Tidak semua teknologi di atas dimasukkan pada spesifikasi HTML5 W3C, meski teknologi tersebut telah terdapat dalam spesifikasi milik WHATWG HTML. Beberapa teknologi yang juga terkait namun tidak dijadikan bagian dalam spesifikasi HTML5 W3C dan WHATWG HTML5 adalah:

a. *Geolocation.*
b. *Web SQL Database*, media penyimpanan *database* lokal.
c. API *Database* terindeks, mode penyimpanan *hierarkis key-value (WebSimpleDB).*

2.4 Penangan Kesalahan

Sebuah peramban web HTML5 (text/html) akan fleksiblel dalam menangani kesalahan sintaks. HTML5 telah didesain agar peramban web lama dapat dengan aman mengabaikan konstruksi HTML5 yang baru. Perbedaan mendasar dengan HTML 4.01 adalah spesifikasi HTML5 memberikan aturan detail untuk meleksikalkan dan memparsing sebagai persyaratan agar berbagai peramban web tetap memberikan hasil yang sama saat terjadi kesalahan sintaks.Meskipun HTML5 telah memiliki perilaku konsisten untuk menangani dokumen-dokumen "Tag Soup", dokumen seperti ini tidak dapat dikatakan telah memenuhi standar HTML5.

2.5 Logo HTML5

Pada 18 Januari 2011, W3C memperkenalkan sebuah logo untuk representasi penggunaan dan tujuan HTML5. Tidak seperti logo lain yang sebelumnya telah diperkenalkan W3C, logo ini tidak mengisyaratkan validitas atau kesesuaian terhadap standar tertentu. Logo ini menjadi logo resmi sejak 1 April 2011. Saat logo ini pertama kali diperkenalkan ke muka publik, W3C menyatakan logo HTML5 ini sebagai sebuah "identitas visual secara umum bagi kumpulan berbagai teknologi open web, termasuk HTML5 CSS, SVG, WOFF, dan lainnya." Beberapa pendukung standar web, termasuk Proyek Standar Web (*The Web Standards Project*), mengkritik definisi "HTML5" sebagai istilah umum, terutama bahwa terjadi pengaburan terminologi dan potensi munculnya miskomunikasi. Tiga hari kemudian, W3C menanggapi umpan balik komunitas dengan mengubah definisi logo ini, yakni dengan menghapus bagian kesertaan berbagai teknologi terkait.

Gambar 39. Logo HTML5

Sumber: https://www.w3.org/html/logo/downloads/HTML5_Logo_512.png

2.6 *Element* HTML5

Tabel *Root Element*

Tag	*Description*
<html>	Merupakan root atau akar dari sebuah dokumen HTML, Semua Element lain harus keturunan dari element ini

Tabel *Document metadata*

Tag	*Description*
<head>	Merupakan koleksi metadata tentang dokumen, termasuk link atau definisi dari script dan style sheet.
<title>	Mendefinisikan judul dokumen, ditampilkan dalam judul bar browser atau pada tab halaman. Ini hanya dapat berisi teks dan setiap tag yang terkandung tidak ditafsirkan.
<base>	Mendefinisikan URL dasar untuk URL relative dalam halaman.
<link>	Digunakan untuk menghubungkan JavaScript dan CSS eksternal dengan dokumen HTML saatini.
<meta>	Mendefinisikan metadata yang tidak dapat didefinisikan dengan menggunakan elemen HTML lainnya.
<style>	tag Style digunakan untuk menulis inline CSS.

Tabel *Scripting*

Tag	*Description*
<script>	Mendefinisikan baik skrip internal atau link ke skrip eksternal. Bahasa script adalah JavaScript.
<noscript>	Mendefinisikan sebuah konten alternative untuk menampilkan bila browser tidak mendukung scripting.

Tabel *Sections*

Tag	*Description*
<body>	Merupakan konten utama dari sebuah dokumen HTML. Hanya ada satu <body> elemen dalam dokumen.
<section>	Element ini telah ditambahkan di HTML5 Mendefinisikan bagian dalam dokumen.
<nav>	Element ini telah ditambahkan di HTML5 Mendefinisikan bagian yang hanya berisi link navigasi.
<article>	Element ini telah ditambahkan di HTML5 Mendefinisikan mandiri konten yang bisa eksis secara independen dari sisa isi.
<aside>	Element ini telah ditambahkan di HTML5 Mendefinisikan beberapa konten disisihkan dari sisa konten halaman. Jika dihapus, isi yang tersisa masih membuat rasa.
<h1> - <h6>	Element Heading menerapkan enam tingkat dari post sebuah dokumen, <h1> adalah yang paling penting dan <h6> adalah yang lain-nya. Element Heading secara ringkas menjelaskan topic bagian itu memperkenalkan.
<hgroup>	Element ini telah ditambahkan di Grup HTML5 satu set <h1> unsur-unsur <h6> ketika heading memiliki beberapa tingkatan
<header>	Element ini telah ditambahkan di HTML5 Mendefinisikan header dari halaman atau bagian. Ini sering berisi logo, judul dari situs Web dan susunan navigasi dari konten.

<footer>	Element ini telah ditambahkan di HTML5 Mendefinisikan footer untuk halaman atau bagian. Ini sering berisi pemberitahuan hakcipta, beberapa link ke informasi hukum atau alamat untuk memberikan umpanbalik.
<address>	Mendefinisikan bagian yang berisi informasi kontak.

Tabel *Grouping content*

Tag	*Description*
<p>	Mendefinisikan porsi yang harus ditampilkan sebagai suatu paragrah.
<hr>	Merupakan pemisah antara paragraph dari bagian atau artikel.
<pre>	Menunjukkan bahwa isinya sudah terformat, dan bahwa format ini harus dilestarikan.
<blokquote>	Merupakankutipan.
	Mendefinisikan ordered list item, yaitu daftar yang berubah arti jika kita mengubah urutan unsur-unsurnya.
	Mendefinisikan daftar unordered item.
	Mendefinisikan sebuah item daftar ranting sering didahului oleh tanda bulatan seperti titik yang besar dalambahasalnggris.
<dl>	Mendefinisikan sebuah daftar definisi, yang merupakan daftar istilah dan definisi yang terkait.
<dt>	Merupakan suatu istilah yang didefinisikan oleh <dd> berikutnya.
<dd>	Merupakan defines istilah segera terdaftar<dt>.
<figure>	Elemeni ni telah ditambahkan di HTML5 Merupakan sosok yang ilustrasikan bagian dari dokumen.
<figcaption>	Elemen ini telah ditambahkan di HTML5 biasanya digunakan untuk keterangan pada gambar.
	Merupakan wadah generic tanpa makna khusus.

Tabel Pengolahan teks

Tag	*Description*
<a>	Merupakan suatu hyperlink, link kesumber lain..
	Merupakan teks ditekankan, hasilnya cetak miring.
	Merupakan teks sangatpenting. Hasilnya cetaktebal
<small>	Merupakan komentar sisi, yaitu teks seperti disclaimer, hak cipta yang tidak penting untuk pemahaman dari dokumen.
<s>	Merupakan konten yang tidak lagi akurat atau relevan.
<cite>	Merupakan judul karya.
<q>	Merupakan kutipan inline.
<dfn>	Merupakan istilah yang definisinya terkandung dalam isi terdekatnya.
<abbr>	Merupakan singkatan atau akronim, akhirnya dengan maknanya.
<time>	Elemen ini telah ditambahkan di HTML5 Merupakan nilai tanggal dan waktu, akhirnya dengan setara mesin-dibaca.
<code>	Merupakan beberapa kode komputer.
<var>	Merupakan sebuah variabel, yang merupakan ekspresi matematika yang sebenarnya atau konteks pemrograman, sebuah identifier mewakili sebuah konstanta, symbol mengidentifikasi kuantitas fisik, parameter fungsi, atau *placeholder* hanya dalam prosa.
<samp>	Merupakan output dari program atau komputer.hanya dalam prosa.
<sub><sup>	Merupakan subskrip, masing-masing superskrip.
<i>	Merupakan beberapa teks dengan suara tersendiri atau mood. Biasanya di tampilkan dalam bentuk italic.
<b>	Merupakan suatu teks yang mana perhatian diambil untuk tujuan pengucapan. Biasanya di tampilkan dalam bentuk Bold.
<u>	Ini tidak menyampaikan pentingnya ekstra dan tidak melibatkan suara alternatif. Biasanya di tampilkan dalam bentuk garis bawah.
<mark>	Elemen ini telah ditambahkan di HTML5 Merupakan teks yang disorot untuk tujuan referensi, yaitu untuk relevansinya dalam konteks lain.

<ruby>	Elemen ini telah ditambahkan di HTML5 Merupakan konten yang akan ditandai dengan penjelasan ruby, berjalan singkat dari teks yang disajikan bersama teks. Hal ini sering digunakan dalam hubungannya dengan bahasa Asia Timur di mana tindakan penjelasan sebagai panduan untuk pengucapan, seperti Furigana Jepang.
<rt>	Elemen ini telah ditambahkan di HTML5 Merupakan teks dari penjelasan ruby.
<rp>	Elemen ini telah ditambahkan di HTML5 Merupakan kurung sekitar penjelasan ruby, digunakan untuk menampilkan anotasi dengan cara alternative oleh browser tidak mendukung tampilan standar untuk penjelasan.
<bdi>	Elemen ini telah ditambahkan di HTML5 Merupakan teks yang harus terisolasi dari sekitarnya untuk format teks dua arah. Hal ini memungkinkan untuk menanamkan rentang teks dengan directionality, berbeda, atau tidak diketahui
<bdo>	Merupakan directionality anak-anak, dalam rangka untuk secara eksplisit menimpa algoritma bidirectional Unicode.
	Merupakan teks tanpa arti khusus. Ini harus digunakan ketika tidak ada elemen teks-semantik lainnya menyampaikan suatu makna yang memadai, yang, dalam hal ini, sering dibawa oleh atribut global sepertikelas, lang, atau dir.
 	Merupakan satu baris
<wbr>	Elemen ini telah ditambahkan di HTML5 Merupakan kesempatan istirahat garis, yang merupakan titik pembungkus disarankan dalam rangka meningkatkan pembacaan teks split pada beberapa baris.
<data>	Elemen ini telah ditambahkan di HTML5 Associates untuk isinya setara mesin-dibaca. (Elemen ini hanya dalam versi WHATWG dari standar HTML, dan bukan dalam versi W3C HTML5).

Tabel *Edits*

Tag	*Description*
<ins>	Mendefinisikan penambahan dokumen.
<del>	Mendefinisikan penghapusan dari dokumen.

Tabel *Embedded content*

Tag	*Description*
<img>	Menampilkan gambar.
<iframe>	Merupakan konteks browsing yang bersarang, yang merupakan dokumen HTML tertanam.
<embed>	Elemen ini telah ditambahkan di HTML5 Merupakan titik integrasi untuk konten, eksternal sering non_HTML, aplikasi atau interaktif.
<object>	Merupakan sebuah sumber eksternal, yang akan diperlakukan sebagai sebuah gambar, sebuah sub-dokumen HTML atau sumber daya eksternal untuk diproses oleh sebuah plugin.
<param>	Mendefinisikan parameter untuk digunakan oleh plugin dipanggi oleh elemen <object>.
<video>	Elemen ini telah ditambahkan di HTML5 Merupakan video, dan *file* terkait audio dan keterangan, dengan tampilan yang diperlukan untuk memainkannya.
<audio>	Elemen ini telah ditambahkan di HTML5 Merupakan suara, atau stream audio.
<source>	Elemen ini telah ditambahkan di HTML5 Memungkinkan penulis untuk menentukan sumber media alternative untuk elemen media seperti <video> atau <audio>
<track>	Elemen ini telah ditambahkan di HTML5 Memungkinkan penulis untuk menentukan jalur teks waktunya untuk elemen media seperti <video> atau <audio>

<canvas>	Elemen ini telah ditambahkan di HTML5 Merupakan daerah bitmap bahwa script dapat digunakan untuk membuat grafik, seperti grafik, permainan grafis, setiap gambar visual dengan cepat.
<map>	Dalam hubungannya dengan mendefinisikan peta gambar.
<area>	Dalam hubungannya dengan <map> , mendefinisikan peta gambar.
<svg>	Elemen ini telah ditambahkan di HTML5 Mendefinisikan gambar vectorial tertanam.
<math>	Elemen init elah ditambahkan di HTML5 Mendefinisikan sebuah rumus matematika.

Tabel *Table*

Tag	***Description***
<table>	Merupakan data dengan lebih dari satu dimensi.
<caption>	Merupakan judul dari tabel.
<colgroup>	Merupakan satu set dari satu atau lebih kolom tabel.
<col>	Merupakan suatukolom tabel.
<tbody>	Merupakan blokbaris yang menggambarkan data konkret dari tabel.
<thead>	Merupakan blokbaris yang menggambarkan label kolom tabel.
<tfood>	Merupakan blokbaris yang menggambarkan ringkasan kolom tabel.
<tr>	Merupakan deretan sel dalam sebuah tabel.
<td>	MMerupakan sebuah sel data dalam tabel.
<th>	Merupakan sebuah sel header di tabel

BAB VI
CSS

1. Pendahuluan

Cascading Style Sheet (CSS) merupakan aturan untuk mengatur beberapa komponen dalam sebuah web sehingga akan lebih terstruktur dan seragam. CSS bukan merupakan bahasa pemograman. Sama halnya styles dalam aplikasi pengolahan kata seperti Microsoft Word yang dapat mengatur beberapa *style*, misalnya *heading*, subbab, *bodytext, footer, images*, dan *style* lainnya untuk dapat digunakan bersama-sama dalam beberapa *file*. Pada umumnya CSS dipakai untuk memformat tampilan halaman web yang dibuat dengan bahasa HTML dan XHTML.

CSS dapat mengendalikan ukuran gambar, warna bagian tubuh pada teks, warna tabel, ukuran *border*, warna *border,* warna *hyperlink*, warna *mouse over*, spasi antar paragraf, spasi antar teks, margin kiri, kanan, atas, bawah, dan parameter lainnya. CSS adalah bahasa style sheet yang digunakan untuk mengatur tampilan dokumen. Dengan adanya CSS memungkinkan kita untuk menampilkan halaman yang sama dengan format yang berbeda.

2. Sejarah CSS

Nama CSS didapat dari fakta bahwa setiap deklarasi style yang berbeda dapat diletakkan secara berurutan, yang kemudian membentuk hubungan ayah-anak (*parent-child*) pada setiap *style*. CSS sendiri merupakan sebuah teknologi internet yang direkomendasikan oleh *World Wide Web Consortium* atau W3C pada tahun 1996. Setelah CSS distandarisasikan, Internet Explorer dan Netscape melepas *browser* terbaru mereka yang telah sesuai atau paling tidak hampir mendekati dengan standar CSS.

3. Versi

Untuk saat ini terdapat tiga versi CSS, yaitu CSS1, CSS2, dan CSS3. CSS1 dikembangkan berpusat pada pemformatan dokumen HTML, CSS2 dikembangkan

untuk memenuhi kebutuhan terhadap format dokumen agar bisa ditampilkan di printer, sedangkan CSS3 adalah versi terbaru dari CSS yang mampu melakukan banyak hal dalam desain website. CSS2 mendukung penentuan posisi konten, downloadable, huruf font, tampilan pada tabel layout dan media tipe untuk printer. Kehadiran versi CSS yang kedua diharapkan lebih baik dari versi pertama dan kedua.

CSS3 juga dapat melakukan animasi pada halaman website, di antaranya animasi warna hingga animasi 3D. Dengan CSS3 desainer lebih dimudahkan dalam hal kompatibilitas websitenya pada smartphone dengan dukungan fitur baru yakni media query. Selain itu, banyak fitur baru pada CSS3 seperti: multiple background, border-radius, drop-shadow, border-image, CSS Math, dan CSS Object Model.

3. Element CSS

3.1 *Selector*

Selector adalah elemen/tag HTML yang ingin diberi *style*. Anda dapat menuliskan langsung **nama tag** yang ingin diberi *style* tanpa perlu menambahkan tanda <>. Pada contoh kode CSS di atas, kita akan memberi *style* pada seluruh *tag* h1 yang terdapat dalam *file* HTML. Jika *tag* HTML yang ingin diberi *style* memiliki ID, anda dapat menuliskan nama ID tersebut dengan diawali tanda *kress* (#). Dan jika *tag* yang diberi *style* memiliki *class*, maka penulisan *selector* bisa dilakukan dengan tanda titik ".", diikuti dengan nama *class*.

```
.artikel
```

Jika anda hanya menuliskan satu *selector*, seperti contoh kode CSS di atas, maka seluruh *tag* h1 yang terdapat dalam *file* HTML akan memiliki *style* yang sama. Bagaimana jika kita hanya ingin memberi *style* pada *tag* h1 yang hanya terdapat di dalam *class* artikel. Maka penulisan *selector*-nya seperti berikut:

```
.artikel h1
```

Kode tersebut akan memerintahkan pada *browser* untuk memberi *style* pada *tag* h1 yang hanya terdapat di dalam ***class artikel*** (atau - h1 yang merupakan *child* dari *class* artikel). Andapun dapat memilih lebih dari satu *tag* untuk penghematan kode CSS. Misalnya ketika anda memiliki dua atau lebih *tag* dengan warna *background* yang sama,

```
h1{ background-color: #666666; }
p { background-color: #666666; }
a { background-color: #666666; }
```

Anda dapat menggabungkan selector dengan menambahkan tanda koma pada nama *tag* yang ingin diberi *style*.

```
h1, p, a { background-color: #66666; }
```

3.2 *Property* dan *Value*

Property adalah sifat-sifat yang ingin diterapkan pada *selector*, seperti warna text, warna *background*, jarak antar elemen, garis pinggir dan lain sebagainya. Untuk memberikan nilai/value pada *property* kita gunakan tanda titik dua ":". Setiap *property* diakhiri dengan titik koma ";", jika anda tidak mengakhirinya maka *browser* tidak akan mengetahui maksud dari *property* tersebut. *Property-property* pada CSS sangat mudah dimengerti karena lebih mirip bahasa kita sehari-hari. Misalnya untuk merubah warna text kita gunakan *property* color, untuk merubah warna *background* kita gunakan *property background-color*, untuk merubah ukuran huruf kita gunakan *property font-size*. Mudah dimengerti bukan?

```
.artikel h1 {
     color : red;
     background-color : blue;
     font-size : 20px;
}
```

3.3 Penulisan CSS

Ada tiga cara penulisan kode CSS, yaitu *inline, internal* dan *external*. Ketiganya bisa anda lakukan sesuai dengan kebutuhan. Berikut contoh penggunaan dari metode-metode tersebut:

a. *Inline*

Penulisan kode CSS dengan metode inline ini bisa dilakukan langsung pada *tag* yang ingin diberi *style* dengan menggunakan atribut *style*.

```
<h1 style="color : red;"> Judul Situs </h1>
```

Pada metode ini, anda tidak perlu menuliskan selector. Karena anda menuliskan CSS langsung pada *tag* yang ingin diberi *style*. Cara ini sangat tidak dianjurkan, karena Anda akan mencampurkan antara "Format" dan "Presentasi". Cara ini juga tidak efektif ketika anda akan melakukan perubahan pada CSS.

b.Internal

Metode CSS internal ditulis di dalam *tag style* yang ditempatkan pada *tag head*.

```
<HTML>
     <head>
          <title>Judul HTML</title>
          <style>
               h1 {
                    color : red;
               }
          </style>
     </head>
...
```

Metode kedua ini sangat dianjurkan untuk pengujian *style*, atau ketika anda hanya memiliki satu halaman web.

c. Eksternal

Metode yang terakhir adalah dengan membuat *file* CSS dan dipanggil di dalam *tag head*. *File* CSS memiliki ekstensi (akhiran) .CSS misalnya **namafile.css**. Pemanggilan *file* CSS dilakukan dengan menggunakan *tag link*:

```
<html>
     <head>
          <title>Judul HTML</title>
          <link rel=”stylesheet” href=”namafile.css” >
     </head>
...
```

Atribut *rel* adalah informasi hubungan (*relationship*) dari *tag link* tersebut, yaitu sebagai *stylesheet*. *Href* diisi dengan lokasi *file* CSS yang ingin dimuat. Pemanggilannya sama dengan pemanggilan gambar atau link.

BAB VII
PHP

1. Sejarah

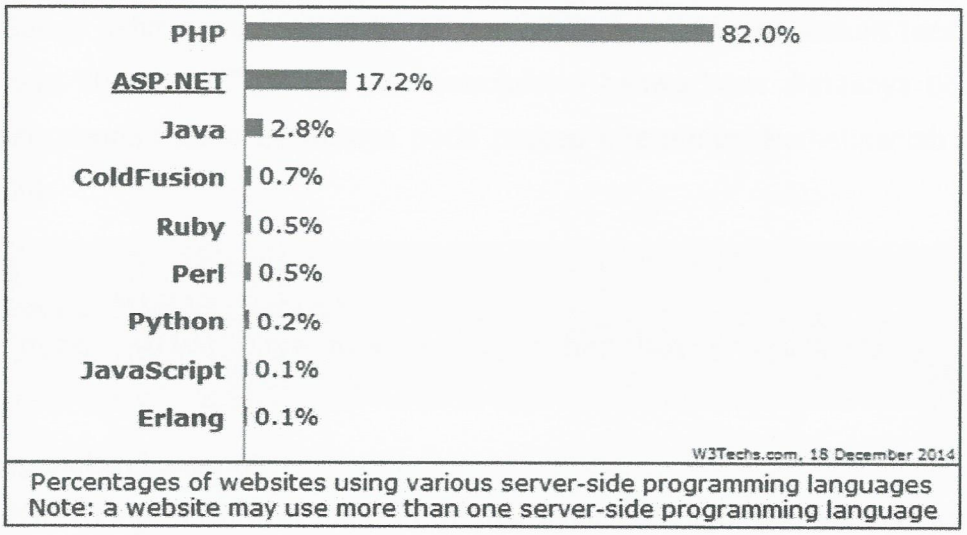
Pada awalnya PHP merupakan kependekan dari *Personal Home Page* (Situs personal). PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP masih bernama *Form Interpreted* (FI), yang wujudnya berupa sekumpulan skrip yang digunakan untuk mengolah data formulir dari web. Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan menamakannya PHP/FI. Dengan perilisan kode sumber ini menjadi sumber terbuka, maka banyak pemrogram yang tertarik untuk ikut mengembangkan PHP. Pada November 1997, dirilis PHP/FI 2.0. Pada rilis ini, interpreter PHP sudah diimplementasikan dalam program C. Dalam rilis ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan PHP/FI secara signifikan.

Pada tahun 1997, sebuah perusahaan bernama Zend menulis ulang interpreter PHP menjadi lebih bersih, lebih baik, dan lebih cepat. Kemudian pada Juni 1998, perusahaan tersebut merilis interpreter baru untuk PHP dan meresmikan rilis tersebut sebagai PHP 3.0 dan singkatan PHP diubah menjadi akronim berulang PHP: Hypertext Preprocessing. Pada pertengahan tahun 1999, Zend merilis interpreter PHP baru dan rilis tersebut dikenal dengan PHP 4.0. PHP 4.0 adalah versi PHP yang paling banyak dipakai pada awal abad ke-21. Versi ini banyak dipakai disebabkan kemampuannya untuk membangun aplikasi web kompleks tetapi tetap memiliki kecepatan dan stabilitas yang tinggi.

Pada Juni 2004, Zend merilis PHP 5.0. Dalam versi ini, inti dari interpreter PHP mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi objek ke dalam PHP untuk menjawab perkembangan bahasa pemrograman ke arah paradigma berorientasi objek.

Saat ini PHP adalah singkatan dari **PHP: Hypertext Preprocessor**, sebuah kepanjangan *rekursif*, yakni permainan kata dimana kepanjangannya terdiri dari singkatan itu sendiri. PHP dapat digunakan dengan gratis (free) dan bersifat *Open Source*. PHP dirilis dalam lisensi *PHP License*, sedikit berbeda dengan lisensi *GNU*

General Public License (GPL) yang biasa digunakan untuk proyek *Open Source*. Kemudahan dan kepopuleran **PHP** sudah menjadi standar bagi programmer web di seluruh dunia. Menurut wikipedia pada februari 2014, sekitar 82% dari web server di dunia menggunakan PHP. PHP juga menjadi dasar dari *aplikasi CMS (Content Management System)* populer seperti *Joomla, Drupal,* dan *WordPress*. Dikutip dari situs w3techs.com, (diakses pada 18 Desember 2014), berikut adalah market share penggunaan bahasa pemrograman server-side untuk mayoritas website di seluruh dunia :



Gambar 40. Bahasa Pemrograman

2. Penulisan PHP

PHP merupakan bahasa pemrograman yang disebut sebangai bahasa scripting, dalam arti PHP merupakan bahasa pemrograman yang ditempelkan/embedded pada bahasa atau aplikasi lain. Sebagai contoh, PHP ditempelkan ke dalam script HTML yang merupakan bahasa ibu untuk world wide web. Berikut ini beberapa hal yang berhubungan dengan cara meletakkan dan menuliskan

```
<?php
      .....
      .....
?>
```

a. Menampilkan data

Untuk menampilkan data kedalam standar output dapat menggunakan perintah *echo* atau *printf*.

```
<?php
     echo “Hello World”;
?>
```

b. Menggunakan semi kolon

Setiap akhir *statement* pada program PHP harus menggunakan tanda semi kolon yaitu titik kome “ ; “. Hal ini menunjukkan bahwa baris diatasnya tidak ada hubungan dengan baris berikutnya pada proses interpreter. Perhatikanlah contoh berikut ini:

```
<?php
     echo “Hello World”;
     echo “<div> Saya adalah baris berikutnya</div>”;
?>
```

c. Menggunakan komentar

PHP menyediakan fasilitas untuk membuat komentar pada *script* yang kita buat. Penggunaan komentar akan mempermudah proses perubahan pada waktu yang akan datang.

```
<?php

// Baris komentar baris tunggal
/*
   Baris komentar baris banyak
   Komentar baris banyak
*/
     echo “Hello World”;
?>
```

d. *Case sensitive*

PHP memiliki turunan seperti induknya yaitu C/C++ yang menganut sistem *case sensitive* yaitu sangat peka terhadap penulisan variabel. Huruf kapital berbeda dengan huruf kecil.

```
<?php

// Variabel 1
     $Nilai = 1;
// Variabel 2
     $nilai = 1;
?>
```

Antara Variabel 1 dengan Variabel 2 pada contoh diatas mempunyai nilai yang berbeda walapun dengan bacaan yang sama namun menggunakan **Huruf Kapital** dan **Huruf Kecil**.

e. Variabel

Variabel adalah suatu pengenal dalam program yang berfungsi untuk menyimpan nilai secara sementara dan dapat diubah-ubah nilainya. Untuk mendefinisikan variabel, diawali dengan simbol karakter dollar "$" dan diikuti oleh nama variabel.

```
<?php
     $namapengenal = nilai;
?>
```

Adapun aturan dalam menyusun variabel:

1) Tersusun dari karakter huruf, angka dan *underscore*(_).
2) Tidak boleh mengandung spasi.
3) Karakter pertama nama pengenal harus dari karakter huruf atau *underscore*.
4) Huruf kecil dan besar dibedakan.

Dalam PHP, tidak diperlukan pendeklarasian variabel dengan tipe datanya seperti bahasa pemrograman *pascal* dan C/C++. Setiap variabel yang terbentuk dalam program dianggap bertipe *variant*, dengan kata lain dapat menampung tipe data dengan jenis apapun.

Contoh penamaan variabel yang benar :

1) $nama_pemakai
2) $kota_3
3) $user1

Contoh penamaan variabel yang salah :

1) $nama pemakai
2) $5kota
3) $us\er1

f. Operator

Operator adalah suatu symbol yang berfungsi untuk menyusun sebuah ekspresi maupun operasi. Sedangkan yang dioperasikan operator disebut dengan operand. Adapun macam-macam operator yaitu :

1) Operator Aritmatika

   Operator ini berfungsi untuk melakukan proses matematika. Perhatikan tabel berikut ini:

Tabel Operator Aritmatika

Contoh	Nama	Hasil
$a + $b	Penjumlahan	Menjumlahkan $a dengan $b
$a - $b	Pengurangan	Selisih dari $a dengan $b
$a * $b	Perkalian	Perkalian antara $a dengan $b
$a / $b	Pembagian	Pembagian antara $a dengan $b
$a % $b	Modulus	Sisa pembagian dari $a dan $b

2) Operator Penugasan

Operator ini menggunakan tanda sama dengan "=", hal ini bukan berarti variabel sebelah kiri sama dengan sebelah kanan, tapi lebih spesifik lagi yaitu operand sebelah kiri akan berisi dengan nilai yang ada pada operand sebelah kanan.

$nilai = 10; artinya masukkan $nilai dengan 10;

3) Operator Pembanding

Operator pembanding berfungsi sesuai dengan namanya namanya yaitu untuk membandingkan dua angka.

Tabel Operator Pembanding

Contoh	Nama	Hasil
$a == $b	Sama dengan	Benar, jika $a sama dengan $b
$a === $b	Identik	Benar, jika $a sama dengan $b dan keduanya mempunyai tipe data yang sama
$a != $b	Tidak sama	Benar, jika $a tidak sama dengan $b
$a !== $b	Tidak identik	Benar, jika $a tidak sama dengan $b dan keduanya tidak mempunyai tipe data yang sama
$a < $b	Lebih kecil	Benar, jika $a lebih kecil dari $b
$a > $b	Lebih besar	Benar, jika $a lebih besar dari $b
$a <= $b	Lebih kecil atau sama dengan	Benar jika $a lebih kecil atau sama dengan $b
$a >== $b	Lebih besar atau sama dengan	Benar jika $a lebih besar atau sama dengan $b

4) Operator Ternary

Operator Ternary untuk membandingkan variabel dalam satu perintah/syntax saja. Fungsi yang digunakan sebagai berikut:

```
(Kondisi) ? (Ekspresi1) : (Ekspresi2);
```

Contoh:

```
($a < $b ) ? "Benar" : "Salah";
```

5) Operator Kenaikan/Penurunan

Operator ini digunakan untuk menaikkan nilai atau menurunkan nilai yang berpola sama.

Tabel Operator Kenaikan dan Penurunan

Contoh	Nama	Hasil
++$a	Pre-increment	Naikkan nilai $a dengan 1, kemudian kembalikan nilai $a
$a++	Post-increment	Kembalikan nilai $a, kemudian naikkan $a dengan nilai 1
–$a	Pre-decrement	Turunkan nilai $a dengan 1, kemudian kembalikan nilai $a
$a–	Post-decrement	Turunkan nilai $a, kemudian turunkan $a dengan nilai 1

6) Operator Logika

Operator logika digunakan untuk menentukan kondisi kebenaran dari dua operand.

Tabel Operator Logika

Contoh	Nama	Hasil
$a AND $b ($a && $b)	And (dan)	Benar, jika $a dan $b bernilai benar
$a OR $b ($a || $b)	Or (atau)	Benar, jika $a atau $b bernilai benar
$a XOR $b	Xor	Benar jika salah satunya benar, tapi tidak keduanya.
!$a	Not	Benar, jika $a tidak benar

7) Operator String

PHP Mengenal dua model operator string yaitu tanda titik "." Yang disebut dengan Concatenation Operator atau operator penyambung string dan ".=" sebagai operator penugasan penyambungan string.

```
<?php
     $string1 = "Hello";
     $string2 = "World";
     $hello = $string1 . " " . $string2;
```

BAB VIII
PHP KONTROL STRUKTUR

1. Tujuan

Setelah mempelajari bab ini, anda akan mampu:

a. menggunakan fungsi percabangan untuk melakukan proses pengontrolan pada saat proses *runtime*.
b. Dapat menggunakan dengan tepat fungsi-fungsi perulangan dalam melakukan pengrontrolan struktur program. Mulai dari For, Foreach, While dan Do While.
c. Dapat membuat dan menggunakan modul dalam mempermudah proses perangkat lunak dengan PHP.
d. Dapat menggunakan fungsi Built-in dari PHP yang berguna untuk melakukan pemanggilan fungsi yang kita buat dengan menggunakan require dan include.

2. Struktur Percabangan

2.1 Fungsi IF dan ELSEIF

Struktur percabangan adalah proses pengalihan program untuk mengeksekusi blok program lainnya berdasarkan pemeriksaan suatu kondisi atau ekspresi. Percabangan terdiri dari perintah IF dan ELSEIF

```
<?php
     if (kondisi) {
          Statement ...
     }
?>
```

```
<?php
     if (kondisi) {
          Statement ...
     } else {
          Statement ...
     }
?>
```

```
<?php
     if (kondisi) {
          Statement ...
     } elseif (kondisi)  {
          Statement ...
     } else {
          Statement ...
     }
?>
```

PHP juga menyediakan fasilitas alternatif untuk proses penulisan fungsi IF, menggunakan tanda titik dua ":" dan perintah *end* untuk menutup fungsi, perhatikan contoh berikut ini:

```
<?php
     if (kondisi):
          Statement ...
     elseif (kondisi):
          statement ...
     else :
          statement ...
     endif;
?>
```

2.2 Fungsi SWICTH ... CASE

Seperti pada bahasa pemrograman lainnya, PHP mendukung proses percabangan dengan menggunakan SWITCH ... CASE. Konsep utama dari fungsi ini sama dengan fungsi IF, yang akan menjalankan suatu eksekusi program berdasarkan kondisi yang diperiksa. Fungsi ini dapat melakukan proses pengontrolan untuk kondisi yang lebih dari 3 ekspresi, Sekalipun fungsi IF dapat melakukannya, namun menyebabkan script tersebut tidak mudah dikontrol.

```
<?php
     switch (kondisi)
     {
          Case $kondisi1:
               statement ...
          break;
          Case $kondisi2:
               statement ...
          break;
     }
?>
```

3. Struktur Pengulangan

Struktur pengulangan (*looping*) berfungsi mengontrol suatu proses yang dilakukan secara berulang-ulang didalam program. Dalam proses pengulangan, terdapat tiga kondisi yang harus terpenuhi sehingga *script* tersebut tidak menyebabkan *crash*, yaitu:

a. Pengulangan harus memiliki nilai awal.
b. Pengulangan harus memiliki batasan
c. Pengulangan harus memilik bentuk pengulangan *increment* atau *decrement*.

3.1 Fungsi FOR

FOR merupakan salah satu fungsi untuk melakukan proses pengulangan, dimana sintaks dan terminologinya mengikuti perilaku dari bahasa pemrograman C/C++. Adapun sintaks nya adalah:

```
for ($nilai_awal; $nilai_batas; $betuk_pengulangan) {
     Statement ...
}
```

```
for ($a=1; $a<=10; $a++) {
     Statement ...
}
```

`$nilai_awal` merupakan nilai awal dari sebuah proses pengulangan (*looping*). Kita selalu menggunakan operator assigment, yaitu tanda sama dengan "=". Pada contoh nilai `$a=1` artinya masukkan nilai 1 kedalam variabel `$a`. dengan demikian proses pengulangan akan dimulai dari 1. Selanjutnya $nilai_batas akan digunakan sebagai kondisi batas akhir dalam proses pengulangan. Nilai `$a<=10` artinya proses pengulangan akan terus dilakukan selama $a kecil dari sepuluh atau $a hingga mencapai nilai 10. Sedangkan `$bentuk_pengulangan` digunakan untuk menentukan kondisi pengulangan, apakah bertambah atau berkurang.

3.2 Fungsi FOREACH

Selain fungsi FOR, PHP juga menyediakan cara mengakses data dalam bentuk array yaitu menggunakan fungsi FOREACH. Secara konsep, FOREACH merupakan penggabungan antara For dan EACH dalam PHP. FOREACH lebih tepat digunakan untuk menampilkan pengulangan data dalam bentuk array

```
foreach ($data as $value) {
     Data yang digunakan adalah $value;
}
```

3.3 Fungsi WHILE

Fungsi WHILE melakukan proses pengulangan selama kondisi yang ditentukan bernilai benar. Sintaks dan terminologinya juga mengikuti perilaku dari bahasa pemrograman C/C++.

```
while ($kondisi) {
     Statement;
}
```

3.3 Fungsi DO ... WHILE

Berbeda dengan Fungsi WHILE sebelumnya, pada fungsi ini proses perulangan telah dilakukan terlebih dahulu selama kondisi yang ditentukan terpenuhi.

```
do {
     Statement;
} while ($kondisi)
```

4. Teknik Modulasi

Pada bagian ini kita akan memperlajari bagaimana menggunakan teknik modulasi yang sangat bermanfaat membuat sebuah aplikasi berbasis web. Konsep teknik modulasi sebenarnya proses pemisahan sebuah aplikasi menjadi bagian per bagian agar lebih mudah dalam proses pengembangan ataupun maintenance. Setelah program dipisah-pisah menjadi *file* yang lebih kecil, selanjutnya kita akan menyatukan kembali fungsinya kedalam program induk. Proses ini membutuhkan fungsi built-in yang disediakan oleh PHP untuk menyatukan modul-modul tersebut agar bisa digunakan bersamaan.

4.1 include dan include_once

Include digunakan untuk memanggil atau mengikutsertakan *file* lain kedalam halaman yang sedang kita buat.

```
Include “nama_file.php”;
```

```
Include_once “nama_file.php”;
```

4.2 require dan require_once

require digunakan untuk memanggil atau mengikutsertakan *file* lain kedalam halaman yang sedang kita buat. Berbeda dengan *include, require* akan menampilkan *Fatal Error* jika *file* yang dipanggil terdapat kesalahan dan program akan terhenti serta perintah-perintah dibawahnya tidak akan dieksekusi.

```
require “nama_file.php”;
```

```
require_once “nama_file.php”;
```

BAB IX
PHP FORM DAN ARRAY

1. Pendahuluan

Pada bab ini merupakan pengembangan dari struktur dasar bahasa pemrograman PHP, dimana pembahasan dimulai dari bagaimana menggunakan form yang merupakan standar bahasa HTML. Penggunaan form merupakan bagian yang mutlak untuk dipelajari agar anda dapat membuat interaksi antara user dan aplikasi dengan mudah. Fokus bahasan lebih ditekankan pada penggunaan form secara runtime. Sedangkan array merupakan salah satu tipe data dalam PHP yang sangat banyak digunakan untuk memudahkan proses input output pada koleksi data. Selanjutnya pembahasan akan difokuskan pada kolaborasi antara form dengan arraydalam koridor interaksi antara user dan aplikasi

2. Tujuan

Setelah mempelajari bab ini anda akan mampu untuk:

a. Membuat dan menggunakan elemen-elemen form pada halaman web.
b. Membuat form dinamis menggunakan script PHP.
c. Menggunakan informasi dari form HTML ke script PHP.
d. Menggunakan dan menangani data dalam bentuk array.
e. Dapat melakukan proses manipulasi terhadap data array baik menambahkan ataupun menghapus data yang ada dalam array.

3. Form

Tag form memiliki atribut sebagai berikut:

a. *Name*, nama dari *form* untuk digunakan pada saat kita melakukan proses data dari form.
b. *Method*, cara *form* melakukan proses pengiriman data. Terdapat dua cara yang digunakan pada method yaitu POST dan GET
c. *Action*, digunakan sebagai arah atau tujuan pengiriman data saat diproses.

```
<form name="form1" method="POST" action="proses.php">
```

3.1 Elemen Form

a. *TextField*, digunakan untuk menerima input dari user, sintaksnya adalah:

```
<input type="text" name="nama_lengkap">
```

b. *HiddenField,* fungsi yang sama dengan *TextField* namun tidak ditampilkan pada halaman user.

```
<input type="hidden" name="token">
```

c. *TextArea*, digunakan untuk menerima informasi dari user dengan area input lebih besar.

```
<textarea name="alamat"></textarea>
```

d. *RadioButton*, digunakan untuk pemilihan satu data.

```
<input type="radio" name="gender"> Laki-laki
<input type="radio" name="gender"> Perempuan
```

e. *CheckBox*, digunakan untuk pemilihan banyak data.

```
<input type="checkbox" name="hobby"> Membaca
<input type="checkbox" name="hobby"> Travelling
```

f. *List/Menu*, digunakan untuk memilih salah satu item dari daftar pilihan

```
<select name="agama">
     <option value="islam">Islam</option>
     <option value="hindu">Hindu</option>
     <option value="damai">Damai</option>
</select>
```

3.2 Parameter POST dan GET

Ketika elemen-elemen form dikirim ke halaman pengolahan data maka informasi yang ada pada setiap elemen form tersebut dapat dikenali oleh PHP. Terdapat dua cara untuk mengakses informasi tersebut yaitu dengan menggunakan parameter POST dan GET.

a. POST, dapat digunakan untuk menerima informasi yang dikirim lewat elemen-elemen form sebagai parameter posting.

```
$user = $_POST['user_name'];
```

b. GET, dapat digunakan untuk mengambil informasi yang dikirim lewat URL parameter.

```
$user = $_GET['user_name'];
```

Contoh program:

```
<?php
     if (isset($_POST['user_name'])) {
          $user = $_POST['user_name'];
          echo "Hai " . $user;
     }
?>

<form name="form1" method="POST" action="">
     <input type="text" name="user_name">
     <input type="submit" value="Simpan">
</form>
```

4. Membuat elemen Dinamis

Elemen-elemen *form* dinamis berfungsi agar tidak terjadi perulangan kode program yang panjang. Elemen tersebut cukup dipadukan dengan perintah PHP untuk mengulang perintah yang sama sehingga tidak membutuh ruang yang besar dalam kode programnya.

4.1 Elemen Menu

Pembuatan elemen *list* secara dinamis dapat dilakukan dengan perintah *for*. Perhatikan contoh berikut ini:

```
<form name="form1" method="POST" action="">
Pilih Tahun:
     <select name="tahun" id="tahun">
     <?php for($thn=1; $thn<50; $thn++) { ?>
          <option value="<?php echo (200+$thn); ?>">
               <?php echo (200+$thn); ?>
          </option>
     <?php } ?>
     </select>
</form>
```

4.2 Elemen Multi-Textfield

Elemen T*extField* dibangun menggunakan perulangan `For`. Dimana user dapat memilih berapa banyak jumlah data yang dimasukkan melalui sebuah list menu, kemudian dengan menggunakan `foreach()` data tersebut kita tampilkan.

```
<?php
     if(isset($_POST["button2"]) and $_POST["button2"] ==
     "Proses"){
          unset($_POST["jml"]);
     }
?>
<form id="form1" name="form1" method="post" action="">
<table width="100%" border="0" cellpadding="2">
     <tr>
```

```
      <td>Berapa Jumlah Text field
            <select name="jml" id="jml">
            <?php for ($jml=1; $jml < 30; $jml++){ ?>
                  <option value="<?php echo $jml; ?>">
                        <?php echo $jml; ?>
                  </option>
            <?php } ?>
            </select>
            <input type="submit" name="button" id="button" value="Add Text
Field">
            </td>
      </tr>
      <tr>
            <td><hr /></td>
      </tr>
      <?php for ($text=0; $text < $_POST["jml"]; $text++){ ?>
      <tr>
            <td>Text Field: <?php echo ($text+1); ?>
                  <input type="text" name="nama[]" id="nama[]"></td>
      </tr>
      <?php } ?>
      <?php if(isset($_POST["jml"])){ ?>
      <tr>
            <td>
                  <input type="submit" name="button2" id="button2"
value="Proses">
            </td>
      </tr>
      <?php } ?>
      <tr>
            <td>
            <?php
            if (isset($_POST["button2"]) and $_POST["button2"] ==
"Proses"){
                  if (is_array($_POST["nama"])){
                        foreach ($_POST["nama"] as $value){
                              if (!empty($value)){
                                    echo "Teman anda, ";
                                    echo $value;
                                    echo "<br />";
                              }
                        }
                  }
            }
            ?>
            </td>
      </tr>
</table>
</form>
```

5. *Array*

Array merupakan koleksi atau kumpulan data yang disimpan dalam satu variabel dan diletakkan pada memori. Data yang ada didalam *array* disebut dengan elemen *array*. Dalam PHP, *array* secara aktual akan melakukan proses pemesanan lokasi pada memori untuk melakukan penyimpanan data. Data dapat terdiri dari *Keys* dan *Values*. *Keys* berfungsi sebagai petunjuk lokasi dari elemen *array* atau *index* dan *Value* merupakan nilai yang disimpan didalam lokasi tersebut.

5.1 Membuat *Array*

Sintaks dari *array* sangatlah sederhana yaitu hanya menggunakan *keyword array()*. Didalam *array* dapat menggunakan sejumlah variabel yang dipisahkan dengan menggunakan tanda koma. Perhatikanlah pendefenisian array berikut ini:

```
$dataArray = array($key => $value);
```

Untuk mengakses elemen didalam array gunakan nama variabel dan simbol "[ ]", yang didalamnya berisi nomor elemen. Sebagai contoh:

```
$dataArray[0], $dataArray[1], ...
```

Elemen dalam *array* dimulai dari 0 sampai jumlahdata-1. Sedangkan untuk mengisi data didalam elemen *array* gunakan operator *assignment* "=".

5.2 Menggunakan *Key* dan *Value*

Selain itu kita dapat mendefenisikan *array* dengan menggunakan *key* dan *value*, secara *default* jika kita menggunakan *array* seperti pada bagian yang lalu maka otomatis *key* akan didefenisikan oleh *array* yang dimulai dari *index* 0. Bagian berikut ini akan dibuat contoh *array* dengan menggunakan pengaturan *key* yang didefenisikan sendiri.

```
<?php

$dataArray = array (1=>"senin", 2=>"selasa", "rabu",
"kamis", "jumat", "sabtu", "minggu");

$rabu = $dataArray[3];
echo $rabu . "<br>";
echo $dataArray[5];

foreach ($dataArray as $list=>$hari){
     echo $hari . "<br />";
}

?>
```

5.3 *Array* Ganda

Array ganda secara prinsip merupakan sebuah *array* yang di dalamnya terdapat *array* lainnya. Perhatikan penulisan *array* berikut ini:

```
$dataArray = array ("array1" => array(1=>"satu", 2=>"dua"),
                    "array2"=>array("warna"=>"merah",
                    "tinggi"=>200));
```

5.4 Manipulasi Elemen *Array*

Array dapat dibuat ataupun dimodifikasi secara *runtime*, artinya pembuatan dan pengisian *array* dilakukan pada saat aplikasi dijalankan. Untuk melakukannya kita harus memodifikasi melalui elemen-elemen *array*. Perhatikan kode berikut ini:

```
$dataArray = array();
```

Mengisi data dengan cara:

```
$dataArray[] = "Isi data";
$dataArray[20] = "Hallo";
```

Berikut contoh program memodifikasi *array*

```
<?php

$dataArray = array();
for ($i='A'; $i <= 'F'; $i++){
     $dataArray[] = $i;
}
echo "Ambil data ke empat " . $dataArray[3] . "<br />";

foreach ($dataArray as $value){
     echo $value . "  " ;
}

$dataArray[3] = 'Z';
echo "<br />";

foreach ($dataArray as $value){
     echo $value . " ";
}
?>
```

5.5 Menghapus *Array*

Selain dapat membuat dan mengisi data *array* secara *runtime*, kita juga dapat melakukan proses penghapusan terhadap *array* baik untuk elemen maupun seluruh *array*. Sintaks yang digunakan adalah:

`$dataArray = array();`

`Unset($dataArray[0]);` untuk menghapus elemen **`$dataArray[0];`**

`Unset($dataArray);` Untuk menghapus seluruh elemen *array*.

5.6 Menampilkan *Array* dengan *List*

Untuk menampikan data *array* agar lebih mudah bisa menggunakan perintah *list*. Perhatikan contoh berikut ini:

```
<?php

$server = array ("www.aliefweb.com", "stendy@aliefweb.com",
                    "aliefweb@yahoo.co.id");
list ($web, $email1, $email2) = $server;
echo "$web merupakan web saya, <br />";
echo "$email1 merupakan email komersil <br />";
echo "$email2 merupakan email free <br />";

?>
```

5.7 Menampilkan *array* dengan *Each*

Each merupakan sintaks yang digunakan untuk mengambil suatu data *array* menggunakan *key* dan *value*. Ini akan digunakan secara berpasangan dengan *list*. Perhatikan contoh berikut ini:

```
<?php

$gengWinny = array (1=>"winny", "ewin", "nando", "hoxy",
                        "ade", "via", "lia", "andrila");

while (list($anggota, $nama) = each ($gengWinny))
{
     echo $anggota . " = " . $nama;
     echo ", ";
}

?>
```

BAB X
PHP FUNGSI

1. Pendahuluan

Fungsi merupakan salah satu teknik yang digunakan untuk menyederhanakan sebuah aplikasi besar dengan memisahkan setiap kasus pada modul-modul yang lebih sederhana, dengan demikian akan mempermudah proses *maintenance* dan pengembangan *software*. Setelah mempelajari bab ini anda akan mampu untuk:

a. Menulis dan membuat fungsi untuk menyelesaikan beberapa kasus yang hampir sama dengan menggunakan satu modul saja.
b. Membuat dan menggunakan fungsi yang dapat melewatkan argumen ke dalam body fungsi
c. Membuat fungsi yang memiliki nilai kembalian, *return value* dan referensi.
d. Membuat fungsi yang dapat melewatkan sebuah argumen dalam bentuk referensi.

2. Fungsi

Fungsi atau yang banyak dikenal dengan nama modul, merupakan sebuah program yang terpisah dari program induk untuk memudahkan proses pengembangan perangkat lunak. Sebuah aplikasi yang besar akan lebih mudah diselesaikan jika kita membaginya ke dalam modul-modul yang lebih sederhana. Dengan demikian proses pengembangan sistem dan debuging akan lebih cepat dan mudah.

Fungsi dapat memanggil fungsi yag lain, tetapi hal ini dapat mempersulit proses debugin atau pelacakan kesalahan dalam program, hal ini akan diatasi dengan menggunakan teknik objek pada bab selanjutnya.

Untuk membuat fungsi, perhatikan kode berikut ini:

```
Function namaFungsi ($arg1, $arg2, ...) {
    ...
}
```

2.1 Menggunakan Fungsi

Setiap fungsi menggunakan nama yang unik, berbeda dengan nama fungsi yang lainnya, sehingga untuk menggunakan fungsi tersebut anda cukup memanggil nama pengenalnya dengan parameter atau argumen yang akan anda gunakan.

```
<?php
function viewInformasi()
{
     echo "*---------- saya dari fungsi ----------------*";
     echo "<br />Hallo... Mudahnya menggunakan fungsi ya <br
/>";
     echo "*---------- akhir dari fungsi ----------------*";
}

echo "Saya akan memanggil fungsi:";
echo "<br />";
viewInformasi();
?>
```

2.2 Fungsi dari dalam fungsi

Sekalipun tidak diharapkan ini terjadi karena dapat menyebabkan kesulitan dalam proses pelacakan kesalahan program atau *debuging*, namun PHP mengizinkan memanggil fungsi dari dalam fungsi. Perhatikan contoh berikut ini:

```
<?php
function defVariabel() {
     $nilaix = 100;
     $nilaiy = 120;
     echo $nilaix + $nilaiy;}

function printVar(){
     defVariabel();
}

printVar();

?>
```

2.3 Fungsi dalam Fungsi

Secara sederhana, fungsi dalam fungsi berarti kita membuat sebuah fungsi dimana didalamnya terdapat beberapa fungsi lagi. Perhatikan contoh berikut ini:

```
<?php
function viewFungsi()
{
     function hitungLuas()
     {
          $nilaix = 5;
          $nilaiy = 10;
          $luas = $nilaix * $nilaiy;
          echo $luas;
     }
}

viewFungsi();  // fungsi induk
hitungLuas();  // fungsi di dalamnya
?>
```

2.4 Fungsi dengan Argumen

Variabel yang digunakan pada fungsi dapat diinisialisasi terlebih pada kolom argumen. Berikut ini contoh sebuah fungsi yang akan melakukan pengolahan data dari nilai yang dilewatkan melalui daftar argumen.

```
<?php

function hitungLuasPersegiPanjang($panjang, $lebar)
{
     $luas = $panjang * $lebar;
     echo "Luas = " . $luas;
}

$x = 5;
$y = 10;
hitungLuasPersegiPanjang($x, $y);

?>
```

2.5 Melewatkan argumen array

Array yang merupakan kumpulan data juga dapat dikirim melalui argumen kedalam fungsi. Perhatikan contoh berikut ini:

```
<?php

function printArray($data)
{
     foreach ($data as $val){
          echo $val ."  ";
     }
}

// memanggil fungsi
$dArray = array ("senin", "selasa", "rabu");
printArray($dArray);

?>
```

2.6 Melewatkan fungsi dengan referensi

Kita dapat melewatkan suatu variabel yang merujuk kepada sebuah variabel – referensi. Adapun model penulisan argumen adalah sebagai berikut:

```
function namaFungsi(&$argumen) {
     ...
}
```

```
<?php

function viewString(&$rData)
{
     $rData .= " Saya dari dalam fungsi";
}

$teks = "halo...bos apakabar, ";
viewString($teks);
echo $teks;

?>
```

BAB XI
PHP OOP

1. Konsep OOP

OOP merupakan singkatan dari *Objek Oriented Programming* atau dalam singkatan bahasa indonesia disebut dengan PBO (Pemrograman Berorientasi Objek). Secara prinsip objek merupakan sebuah elemen yang dapat diselidiki atau dipahami. Jika kita merujuk kepada objek dari sudut pandang nyata berarti objek merupakan sesuatu yang memiliki bentuk dan massa, namun jika kita berbicara konsep maka objek bukan hanya sesuatu yang dapat dilihat dan dirasakan, tetapi termasuk sesuatu yang bersifat ide dan gagasan.

Dalam konsep pemrograman, objek akan ditinjau dari kedua hal tersebut yaitu elemen yang memiliki bentuk dan massa serta sesuatu yang masih dalam bentuk ide dan gagasan. Sehingga sudut pandang pemrograman melihat bahwa sesuatu yang abstrak dan kongkret merupakan objek yang dapat diimplementasikan ke dalam sebuah perangkat lunak.

Contoh dari objek seperti manusia, hewan, mobil, pohon, gagasan, ide dan sebagainya. Kesimpulannya, setiap objek memiliki sesuatu yang dapat dibedakan antara suatu objek dengan objek lainnya, dimana setiap objek secara umum memiliki kondisi tetap atau state dan *operation* maupun *method*, atau secara sederhana dikatakan bahwa objek memiliki pengenal atau properti/attributr dan *behavior/method*/fungsi.

Sebagai contoh sebuah mobil memiliki pengenal atau atribut yaitu memiliki ban, setir, pintu, engine dan sebagainya. *Behavior*nya dapat berupa kondisi mobil yang dapat dipercepat atau diperlambat, dapat mengeluarkan suara klakson, dapat menghidupkan lampu dan dapat bergerak mundur. Dari konsep ini makan berkembanglan sebuah teknik pemrograman baru yaitu pemrograman yang didasarkan pada dunia nyata atau objek yang sesungguhnya.

Dari uraian tersebut diatas maka dapat disimpulkan bahwa, pemrograman berorientasi objek berarti sebuah teknik pemrograman yang dalam proses pengembangannya menggunakan terminologi objek, dimana setiap objek memiliki

atribut beserta dengan fungsi yang dapat saling berinteraksi satu dengan lainnya seperti halnya sebuah objek.

2. *Class*

Dalam OOP, sebuah *class* merupakan *blueprint* dari suatu objek. Mungkin Anda bertanya, apa bedanya *class* dengan sebuah *function*? Sebuah *class* bisa berisi variabel dan *function*. Variabel yang terletak di dalam *class*, dinamakan *property* dan *function* yang ada di dalam sebuah *class* dinamakan *method*.

Untuk membuat sebuah *class*, strukturnya adalah sbb:

```
<?php
     class namakelas {
          var namavariabel;
          ...
     }
?>
```

Sebagai contoh misalkan kita membuat *class* bernama kendaraan

```
<?php
     class kendaraan {
          var $jumlahRoda;
          var $warna;
          var $bahanBakar;
          var $harga;
          var $merek;
     }
?>
```

Dalam contoh di atas, yang merupakan properti dari *class* kendaraan adalah: jumlahRoda, warna, bahanBakar dan harga. Sebuah *properties* dari suatu *class* dapat Anda bayangkan sebagai sifat atau informasi yang melekat dari suatu objek. Sebagai contoh misalkan kita pandang sebuah objek 'mahasiswa', maka properties dari

mahasiswa beberapa diantaranya adalah: nim, nama, alamat, nama orang tua, jurusan, fakultas dsb.

Latihan

a. Buatlah sebuah kelas bernama 'buku', kemudian deklarasikan beberapa properties dari buku tersebut, misalnya: judul buku, pengarang, penerbit, tahun tersebut dsb
b. Rancanglah sebuah kelas untuk menyatakan orang, kemudian tentukan sendiri properties nya dan selanjutnya tulis *class* tersebut ke dalam script PHP

3. *Function/Method*

Sebuah *function* dalam suatu *class* dinamakan *method*, dan sebuah *method* jika kita bayangkan adalah segala hal yang terkait dengan pekerjaan atau proses yang dapat diberikan pada suatu objek. Sebagai contoh method dalam kehidupan sehari-hari, adalah pada objek seorang 'mahasiswa'. Sebuah method kita bisa berikan pada mahasiswa tersebut misalnya: "kuliah". Di dalam method "kuliah" itu terdapat serangkaian proses mulai dari:

- ✓ registrasi kuliah
- ✓ ikuti kuliah
- ✓ ikuti ujian
- ✓ Jika ujian tidak lulus, maka ulangi ikuti kuliah

Itu sebagai contoh gambaran method dalam kehidupan sehari-hari.

Berikut ini contoh sebuah *function* yang dibuat dalam sebuah *class*. *Function* dalam contoh berikut ini digunakan untuk menentukan apakah sebuah kendaraan harganya mahal atau tidak. Di sini kendaraan dikatakan mahal jika harganya di atas 50 juta, dan jika di 50 juta ke bawah dikatakan murah.

```
<?php
     class kendaraan {
          var $jumlahRoda;
          var $warna;
          var $bahanBakar;
          var $harga;
          var $merek;

     function statusHarga() {
          if ($this->harga > 50000000) $status = 'Mahal';
          else $status = 'Murah';
          return $status;
     }
     }
?>
```

Perhatikan perintah **$this->harga** variabel **$this** merupakan *built in* variabel yang digunakan untuk mengakses properties atau *method* yang ada dalam *class* tersebut. Sehingga perintah **$this->harga** digunakan untuk mengakses atau membaca *property* dari **$harga** yang ada dalam *class* kendaraan.

Catatan:

Variabel **$status** dalam **function statusharga()** bukanlah termasuk *property* dari *class* kendaraan karena tidak didefinisikan dalam bentuk **var $status;**

Latihan

a. Dari kelas 'kendaraan' dalam contoh, tambahkan sebuah *property* 'tahun pembuatan'
b. Buatlah *function* dalam kelas 'kendaraan' dengan nama 'dapatSubsidi()' untuk menentukan apakah suatu kendaraan mendapat subsidi BBM atau tidak. Kendaraan yang mendapat subsidi adalah yang berbahan bakar 'Premium' dan tahun pembuatannya sebelum tahun 2005. *Function* ini me*return* 'Ya' jika mendapat subsidi, dan 'Tidak' jika tidak mendapat subsidi.

c. Buatlah *function* dalam kelas 'kendaraan' dengan nama 'hargaSecond()' untuk menentukan harga second dari kendaraan tersebut. *Function* ini me*return* harga *second* dari kendaraan dengan ketentuan:
   1) Jika tahun pembuatan di atas 2005, maka harga second nya turun 20% dari harga aslinya.
   2) Jika tahun pembuatan 2000 s/d 2005, maka harga second nya turun 30% dari harga aslinya.
   3) Jika tahun pembuatan di bawah 2000, maka harga second nya turun 40% dari harga aslinya.

4. Instantisasi Objek

Seperti yang telah dijelaskan sebelumnya bahwa sebuah *class* merupakan blueprint dari objek. Sebuah *class* belum menjadi objek sebelum kita lakukan sebuah proses instantisasi objek. Untuk melakukan instantisasi objek, perintahnya adalah sbb:

```
$handle = new namaclass();
```

Sebagai contoh, misalkan kita lakukan instantiasi pada *class* kendaraan :

```
<?php
     class kendaraan {
          var $jumlahRoda;
          var $warna;
          var $bahanBakar;
          var $harga;
          var $merek;
     function statusHarga() {
          if ($this->harga > 50000000) $status = 'Mahal';
          else $status = 'Murah';
          return $status;
     }
     }
     $kendaraan1 = new kendaraan();
?>
```

Jika script di atas dijalankan, maka di browser tidak muncul apa-apa. Hal ini terjadi karena kita belum menyuruh PHP untuk melakukan sesuatu pada objek $kendaraan1 tersebut.

Variabel **$kendaraan1** dalam hal ini dinamakan 'handle' karena kita akan gunakan $kendaraan1 untuk mengontrol dan menggunakan objek kendaraan. Kita juga bisa melakukan instantisasi objek tanpa menggunakan kurung, perhatikan contoh berikut ini yang menunjukkan proses instantisasi beberapa objek dari *class* kendaraan.

```
<?php
     class kendaraan {
          var $jumlahRoda;
          var $warna;
          var $bahanBakar;
          var $harga;
          var $merek;
     function statusHarga() {
          if ($this->harga > 50000000) $status = 'Mahal';
          else $status = 'Murah';
          return $status;
     }
     }

     $kendaraan1 = new kendaraan();
     $kendaraan2 = new kendaraan;
     $kendaraan3 = new kendaraan();
?>
```

5. Setting Properties

Setelah suatu objek kita lakukan instantitasi, selanjutnya kita bisa mensetting properties dari objek tersebut. Sebagai contoh, misalkan kita telah membuat objek $kendaraan1, kemudian bagaimana kita menset properti harga dan merek dari objek $kendaraan1 ini?

Kita dapat mensetting properties dari suatu objek dengan perintah:

```
$namaobyek->properti = value;
```

Perhatikan contoh berikut ini:

```
<?php
     class kendaraan     {
          var $jumlahRoda;
          var $warna;
          var $bahanBakar;
          var $harga;
          var $merek;
     function statusHarga() {
          if ($this->harga > 50000000) $status = 'Mahal';
          else $status = 'Murah';
          return $status;
     }
     }

     $kendaraan1 = new kendaraan();
     $kendaraan1->merek = 'Yamaha MIO';
     $kendaraan1->harga = 10000000;
?>
```

Perintah:

```
$kendaraan1->merek = 'Yamaha MIO';
```

Digunakan untuk mensetting properti merek 'Yamaha MIO' dari objek $kendaraan1. Kita juga bisa menggunakan *method* untuk proses setting properti ini, dan ini adalah cara yang lebih direkomendasikan dalam OOP.

```
<?php
     class kendaraan {
          var $jumlahRoda;
          var $warna;
          var $bahanBakar;
          var $harga;
          var $merek;
     function statusHarga() {
          if ($this->harga > 50000000) $status = 'Mahal';
          else $status = 'Murah';
          return $status;
     }
     function setMerek($x) {
          $this->merek = $x;
     }
     function setHarga($x) {
          $this->harga = $x;
     }
     }

     $kendaraan1 = new kendaraan();
     $kendaraan1->setMerek('Yamaha MIO');
     $kendaraan1->setHarga(10000000);
?>
```

Latihan

Dari *class* kendaraan diatas, buatlah objek dengan properti sebagai berikut:

Objek	Merek	Jumlah Roda	Harga	Warna	Bahan Bakar
$kendaraan2	Jupiter MX	2	18.000.000	Hijau	Premium
$kendaraan3	Vixion	2	23.000.000	Merah	Pertalite
$kendaraan4	Avanza	4	160.000.000	Hitam	Pertamax

6. Menjalankan *Method*

Dalam bagian ini, akan dijelaskan cara menjalankan sebuah method dari suatu objek. Ingat, bahwa menjalankan sebuah *method* dari suatu objek pada intinya adalah memanggil *function* yang dalam *class*.

Sebenarnya, dalam contoh sebelumnya sudah diberikan contoh untuk menjalankan method yaitu salah satunya melalui perintah:

```
$kendaraan1->setMerek('Yamaha MIO');
```

Perintah tersebut adalah menjalankan method setMerek() dari objek $kendaraan1, dan dalam hal ini setMerek() adalah sebuah *function* dalam *class* kendaraan. Contoh yang lain, misalkan kita akan menjalankan *method* statusHarga() yang digunakan untuk menampilkan status harganya apakah termasuk mahal atau murah.

```
<?php
      class kendaraan {
            var $jumlahRoda;
            var $warna;
            var $bahanBakar;
            var $harga;
            var $merek;
      function statusHarga() {
            if ($this->harga > 50000000) $status = 'Mahal';
            else $status = 'Murah';
            return $status;
      }
      function setMerek($x) {
            $this->merek = $x;
      }
      function setHarga($x) {
            $this->harga = $x;
      }
      }
      $kendaraan1 = new kendaraan();
      $kendaraan1->setMerek('Yamaha MIO');
      $kendaraan1->setHarga(10000000);
      echo $kendaraan1->statusHarga();
?>
```

Jika script di atas dijalankan, maka akan muncul 'Murah', karena harga nya kurang dari 50.000.000. Perhatikan dari beberapa contoh pemanggilan method di atas, bahwa setiap kali pemanggilan method jangan lupa memberi tanda kurung (), seperti pada:

```
$kendaraan1->setHarga(10000000);
```

Atau

```
$kendaraan1->statusHarga();
```

Karena kurung tersebut digunakan untuk meletakkan parameter bagi method tersebut.

Latihan

a. Perhatikan kembali soal latihan sebelumnya pada bab 5. Tampilkan status harga dari $kendaraan2, $kendaraan3 dan $kendaraan4.
b. Perhatikan kembali soal latihan pada bab 3 nomor 3. Tampilkan harga second dari $kendaraan2, $kendaraan3 dan $kendaraan4.

7. Mengakses *Properties*

Sekarang akan dijelaskan bagaimana cara mengakses *properties* dari suatu objek. Sebelumnya, pernah saya katakan bahwa properties dari suatu objek itu merupakan *value* dari variabel yang ada dalam *class*. Bagaimana cara mengakses *properties* dari suatu objek? Perhatikan contoh berikut ini:

```
<?php
    class kendaraan {
        var $jumlahRoda;
        var $warna;
        var $bahanBakar;
        var $harga;
        var $merek;
    function statusHarga() {
        if ($this->harga > 50000000) $status = 'Mahal';
        else $status = 'Murah';
        return $status;
    }
    function setMerek($x) {
        $this->merek = $x;
    }
    function setHarga($x) {
        $this->harga = $x;
    }
    }

    $kendaraan1 = new kendaraan();
    $kendaraan1->setMerek('Yamaha MIO');
    $kendaraan1->setHarga(10000000);
    echo 'Harga dari '.$kendaraan1->merek.' adalah Rp.
    '.$kendaraan1->harga;
?>
```

Perhatikan pada bagian perintah:

$kendaraan1->harga

Dan

$kendaraan1->merek

Kedua perintah di atas adalah digunakan untuk mengakses value dari *property* objek **$kendaraan1**, yaitu 'merek' dan 'harga'. Jika script di atas dijalankan, maka akan diperoleh output "Harga dari Yamaha MIO adalah Rp. 10000000"

Selain cara di atas, dapat pula menggunakan *method* dalam membaca *properties* dari suatu objek, dan cara inilah yang paling disarankan dalam OOP. Perhatikan contoh berikut ini:

```
<?php
     class kendaraan {
          var $jumlahRoda;
          var $warna;
          var $bahanBakar;
          var $harga;
          var $merek;
     function statusHarga() {
          if ($this->harga > 50000000) $status = 'Mahal';
          else $status = 'Murah';
          return $status;
     }
     function setMerek($x) {
          $this->merek = $x;
     }
     function setHarga($x) {
          $this->harga = $x;
     }
     function bacaMerek() {
          return $this->merek;
     }
     function bacaHarga() {
          return $this->harga;
     }
     }

     $kendaraan1 = new kendaraan();
     $kendaraan1->setMerek('Yamaha MIO');
     $kendaraan1->setHarga(10000000);
     echo 'Harga dari '.$kendaraan1->bacaMerek().' adalah
Rp.  '.$kendaraan1->bacaHarga();
?>
```

Dalam contoh di atas, untuk mengakses properti merek dibuat *function* sebagai berikut:

```
function bacaMerek() {
    return $this->merek;
}
```

Sedangkan *function* untuk mengakses *properti* harga kendaraan adalah:

```
function bacaHarga() {
    return $this->harga;
}
```

Selanjutnya untuk mengakses properti nama merek kendaraan, cukup dipanggil saja *method* bacaMerek() sbb:

```
$kendaraan1->bacaMerek()
```

Demikian pula untuk mengakses properti harga kendaraan melalui method bacaHarga();

```
$kendaraan1->bacaHarga()
```

Latihan

Perhatikan kembali soal latihan pada bab 5, berdasarkan objek yang telah dibuat, tampilkan properti setiap objek sedemikian hingga tampilan script apabila dijalankan di browser sebagai berikut:

a. Kendaraan Toyota Yaris, memiliki 4 roda, berbahan bakar Premium dan harganya Rp 160.000.000.
b. Kendaraan Honda Scoopy, memiliki 2 roda, berbahan bakar Premium dan harganya Rp 13.000.000.
c. Kendaraan Isuzu Panther, memiliki 4 roda, berbahan bakar Solar dan harganya Rp 170.000.000.

8. Modularitas *Class*

Pada contoh-contoh script di atas, *class* dan juga proses instantisasi dijadikan satu dalam sebuah *script*. Hal ini dirasa kurang efektif apabila *class* tersebut juga digunakan dalam *script* yang lain nantinya. Sehingga untuk alasan kemudahan penggunaan, biasanya sebuah *class* atau kumpulan *class* diletakkan dalam sebuah *script* tersendiri, yang selanjutnya tinggal di *include* kan dalam sebuah *script* apabila *class* tersebut akan digunakan. Dengan demikian kita tidak perlu menulis kembali isi *class* secara penuh dalam setiap *script*nya.

Sebagai contoh, perhatikan kembali contoh *script* pada poin 7 yang berbentuk sbb:

```
<?php
      class kendaraan {
            var $jumlahRoda;
            var $warna;
            var $bahanBakar;
            var $harga;
            var $merek;
      function statusHarga() {
            if ($this->harga > 50000000) $status = 'Mahal';
            else $status = 'Murah';
            return $status;
      }
      function setMerek($x) {
            $this->merek = $x;
      }
      function setHarga($x) {
            $this->harga = $x;
      }
      function bacaMerek() {
            return $this->merek;
      }
      function bacaHarga() {
            return $this->harga;
      }
      }

      $kendaraan1 = new kendaraan();
      $kendaraan1->setMerek('Yamaha MIO');
      $kendaraan1->setHarga(10000000);
      echo 'Harga dari '.$kendaraan1->bacaMerek().' adalah Rp.
      '.$kendaraan1->bacaHarga();
?>
```

Kita dapat memisahkan *class* 'kendaraan' ini dalam *file* tersendiri misalkan diberinama **class_kendaraan.php**

```
<?php
     class kendaraan {
          var $jumlahRoda;
          var $warna;
          var $bahanBakar;
          var $harga;
          var $merek;
     function statusHarga() {
          if ($this->harga > 50000000) $status = 'Mahal';
          else $status = 'Murah';
          return $status;
     }
     function setMerek($x) {
          $this->merek = $x;
     }
     function setHarga($x) {
          $this->harga = $x;
     }
     function bacaMerek() {
          return $this->merek;
     }
     function bacaHarga() {
          return $this->harga;
     }
     }
?>
```

Selanjutnya kita include kan *file* **class_kendaraan.php** ini ke dalam script lain apabila kita memerlukannya,

contoh.php

```
<?php
     include 'class-kendaraan.php';
     $kendaraan1 = new kendaraan();
     $kendaraan1->setMerek('Yamaha MIO');
     $kendaraan1->setHarga(10000000);
     echo 'Harga dari '.$kendaraan1->bacaMerek().' adalah
Rp.  '.$kendaraan1->bacaHarga();
?>
```

9. *Constructor*

Perhatikan kembali proses instantisasi yang ada di poin 4 dan setting *properties* di poin 5. Jika kita perhatikan, maka proses instantisasi dan *setting properties* ini dilakukan secara terpisah. Tentu saja proses ini agak terlalu bertele-tele. Ternyata kita bisa langsung melakukan instantisasi objek sekaligus melakukan setting propertiesnya. Proses ini dapat dilakukan dengan menggunakan '*constructor*'. Untuk membuat *constructor,* kita cukup membuat sebuah *function* dalam *class* dengan bentuk

```
<?php
     class namaClass {
          function __construct (parameter) {
          ...
     }
?>
```

Keterangan: Tanda __ merupakan tanda *underscore* (_) yang ditulis *double*.

Berikut ini contoh *constructor* untuk objek kendaraan, dimana sekaligus mensetting properti 'merek' dan 'harga' kendaraan.

class-kendaraan.php

```
<?php
     class kendaraan {
     var $jumlahRoda;
     var $warna;
     var $bahanBakar;
     var $harga;
     var $merek;
     function statusHarga() {
          if ($this->harga > 50000000) $status = 'Mahal';
          else $status = 'Murah';
          return $status;
     }
     function setMerek($x) {
          $this->merek = $x;
     }
     function setHarga($x) {
          $this->harga = $x;
     }
     function bacaMerek() {
          return $this->merek;
     }
     function bacaHarga() {
          return $this->harga;
     }
     function __construct($x, $y) {
          $this->merek = $x;
     $this->harga = $y;
     }
     }
?>
```

Perhatikan kode berikut:

```
function __construct($x, $y)
{
     $this->merek = $x;
     $this->harga = $y;
}
```

Function tersebut kita buat 2 parameter, dimana **$x** menyatakan merek kendaraan, dan **$y** adalah harganya. Selanjutnya, perintah:

$this->merek = $x;

Digunakan untuk *setting property* merek kendaraan berdasarkan nilai $x. Demikian juga perintah:

$this->harga = $y;

Untuk *setting property* harga kendaraan berdasarkan nilai **$y.** Selanjutnya, bagaimana cara melakukan instantisasi sekaligus setting propertiesnya? Perhatikan *script* berikut ini:

contoh.php

```
<?php
include 'class-kendaraan.php';
$kendaraan1 = new kendaraan('Yamaha MIO', 10000000);
echo 'Harga dari '.$kendaraan1->bacaMerek().' adalah Rp.
'.$kendaraan1->bacaHarga();
?>
```

10. *Encapsulation*

Di dalam dasar-dasar OOP, ada istilah *encapsulation*. Istilah ini terkait dengan aksesibilitas *properties* dalam suatu *class*. Dengan *encapsulation* ini, kita bisa mengatur sebuah properti apakah hanya bisa diakses dalam *class* tersebut saja, atau tidak. Aksesibilitas *properties* dalam *encapsulation* ini ada tiga sifat:

a. **Public** : Properti dapat diakses darimanapun
b. **Private** : Properti hanya dapat diakses dari dalam *class* saja
c. **Protected** : Properti hanya dapat diakses dari dalam *class* atau *class* turunan (*inheritanced class*)

Untuk membedakan ketiganya, perhatikan contoh berikut ini:

class-kendaraan.php

```
<?php
      class kendaraan {
            protected $jumlahRoda;
            public $warna;
            public $bahanBakar;
            public $harga;
            private $merek;
      function statusHarga() {
            if ($this->harga > 50000000) $status = 'Mahal';
            else $status = 'Murah';
            return $status;
      }
      function setMerek($x) {
            $this->merek = $x;
      }
      function setHarga($x) {
            $this->harga = $x;
      }
      function bacaMerek() {
            return $this->merek;
      }
      function bacaHarga(){
            return $this->harga;
      }
      function __construct($x, $y) {
            $this->merek = $x;
            $this->harga = $y;
      }
      }
?>
```

Perhatikan *class* di atas. Untuk properti 'warna', 'bahan bakar' dan 'harga' dibuat sebagai *public properties*. Sedangkan untuk properti 'jumlahRoda' dan 'merek', masing-masing sebagai *protected* dan *private properties*. Selanjutnya, perhatikan *script* contoh berikut ini:

contoh.php

```
<?php
include 'class-kendaraan.php';
$kendaraan1 = new kendaraan('Yamaha MIO', 10000000);
echo 'Nama merek : '.$kendaraan1->merek;
?>
```

Dalam script di atas, setelah proses instantisasi dan setting properti untuk objek $kendaraan1, akan dilakukan pengaksesan ke properti merek secara langsung (tanpa method), dengan memberikan perintah:

$kendaraan1->merek

Apa yang terjadi jika script di atas dijalankan? Ternyata akan muncul error.

Fatal error: Cannot access private property kendaraan::$merek

Hal ini terjadi karena properti merek bersifat *private*, sehingga properti ini tidak bisa diakses dari luar *class*. Bagaimana dengan akses ke properti harga secara langsung? Perhatikan script berikut ini:

contoh.php

```
<?php
include 'class-kendaraan.php';
$kendaraan1 = new kendaraan('Yamaha MIO', 10000000);
echo 'Harga : '.$kendaraan1->harga;
?>
```

Ternyata jika *script* di atas dijalankan, bisa memunculkan harga dari Yamaha Mio. Pertanyaannya, apakah bisa kita mengakses sebuah properti yang sifatnya *private* dalam *class* dari luar? Jawabnya adalah bisa, namun tidak dilakukan secara langsung dengan mengakses properti nya namun menggunakan method. Sebagai contoh, misalkan kita ingin mengakses properti merek yang sifatnya *private*, maka kita bisa menggunakan *method* bacaMerek().

contoh.php

```
<?php
include 'class-kendaraan.php';
$kendaraan1 = new kendaraan('Yamaha MIO', 10000000);
echo 'Harga : '.$kendaraan1->bacaMerek();
?>
```

Bagaimana dengan deklarasi properties menggunakan 'var' seperti pada contoh-contoh di awal, misalnya:

class_kendaraan.php

```
class kendaraan
{
var $jumlahRoda;
var $warna;
var $bahanBakar;
var $harga;
var $merek;
}
```

Penggunaan 'var' di depan nama *properties*, secara otomatis akan bersifat sebagai *public*. Berikutnya, muncul pertanyaan apakah yang bisa dibuat *encapsulation* dg sifat *private, protected* dan *public* ini hanya untuk *properties* saja? Jawabnya adalah TIDAK, sebuah *function* atau *method* pun bisa diterapkan hal ini. Sebagai contoh misalkan kita buat method statusHarga() sebagai *private method*.

class-kendaraan.php

```
<?php
      class kendaraan {
            protected $jumlahRoda;
            public $warna;
            public $bahanBakar;
            public $harga;
            private $merek;
      private function statusHarga() {
            if ($this->harga > 50000000) $status = 'Mahal';
            else $status = 'Murah';
            return $status;
      }
      function setMerek($x) {
            $this->merek = $x;
      }
      function setHarga($x) {
            $this->harga = $x;
      }
      function bacaMerek() {
            return $this->merek;
      }
      function bacaHarga() {
            return $this->harga;
      }
      function __construct($x, $y) {
            $this->merek = $x;
            $this->harga = $y;
      }
      }
?>
```

Kemudian kita cek, apakah efek jika sebuah *method* dibuat *private* dengan memanggil *method* statusHarga() di dalam *script*.

contoh.php

```
<?php
include 'class-kendaraan.php';
$kendaraan1 = new kendaraan('Yamaha MIO', 10000000);
echo 'Status harga : '.$kendaraan1->statusHarga();
?>
```

Jika *script* di atas dijalankan, maka akan muncul pesan *error* sbb:

Fatal error: Call to private method kendaraan::statusHarga() from context "

Yang menginformasikan bahwa method statusHarga() bersifat *private* sehingga tidak bisa diakses dari luar *class*.

11. Pewarisan (Inheritance)

Perhatikan kembali *class* 'kendaraan', selanjutnya bagaimana jika kita ingin membuat objek baru akan tetapi objek ini nanti berupa 'kereta api' ? Khusus kereta api ini nanti, ada properti yang digunakan untuk menyatakan jumlah gerbong. Sedangkan properti yang lain seperti merek, jumlah roda, harga dan bahan bakar sama seperti dalam *class* kendaraan. Oleh karena itu untuk objek kereta api ini kita perlu membuat *class* baru yang merupakan pengembangan dari *class* kendaraan. Dalam OOP, kita tidak perlu lagi membuat *class* baru ini, tapi cukup kita membuat *class* baru yang merupakan turunan atau warisan dari *class* sebelumnya. *Class* turunan ini, akan memiliki properti dan method yang sama seperti *class* pewarisnya, namun terdapat properti atau method tambahan khusus untuk *class* ini. Istilah pewarisan *class* ini dalam OOP dinamakan *inheritance*. Bagaimana cara membuat *class* turunan ini?

```
<?php
     class namaclassbaru extends namaclasslama
     {
          ...
     }
?>
```

Sebagai contoh perhatikan *script* **class-kendaraan.php** berikut ini:

```
<?php
     class kendaraan {
          protected $jumlahRoda;
          public $warna;
          public $bahanBakar;
          public $harga;
          private $merek;
     private function statusHarga() {
          if ($this->harga > 50000000) $status = 'Mahal';
          else $status = 'Murah';
          return $status;
     }
     function setMerek($x) {
          $this->merek = $x;
     }
     function setHarga($x) {
          $this->harga = $x;
     }
     function bacaMerek() {
          return $this->merek;
     }
     function bacaHarga() {
          return $this->harga;
     }
     function __construct($x, $y) {
          $this->merek = $x;
          $this->harga = $y;
     }
     }

     class keretaApi extends kendaraan {
          public $jumGerbong;
     function setGerbong($x) {
          $this->jumGerbong = $x;
     }
     function bacaGerbong() {
          return $this->jumGerbong;
     }
     }
?>
```

Perhatikan *class* 'keretaApi' yang merupakan turunan dari *class* 'kendaraan' dalam *script* di atas. Dalam *class* tersebut, dibuat properti bernama 'jumGerbong' (jumlah gerbong). Selain itu, khusus untuk *class* 'keretaApi' ini dibuat juga *method* untuk *setting* properti jumGerbong ini dengan nama setGerbong(), serta *method* bacaGerbong() untuk mengakses properti jumlah gerbong. Selanjutnya perhatikan *script* yang di dalamnya ada proses instantisasi objek kereta api ini, *setting properties* serta memanggil *method*.

```
<?php
     include 'class-kendaraan.php';

     $kereta1 = new keretaApi('KA Lokomotif', 15000000);
     $kereta1->setGerbong(20);
     echo 'Jumlah gerbong dari '.$kereta1->bacaMerek().
     ' yang seharga '.$kereta1->bacaHarga().
     ' adalah '.$kereta1->bacaGerbong();
?>
```

Jika script tersebut diperhatikan, maka terdapat constructor pada *class* keretaApi dimana dapat dilakukan instantisasi sekaligus setting properties untuk nama merek dan harganya. Mengapa kok bisa? Ya... karena *class* keretaApi adalah turunan dari *class* kendaraan, dimana di dalam *class* kendaraan terdapat constructor, sehingga untuk *class* keretaApi inipun dapat dilakukan hal yang sama.
Selanjutnya diberikan perintah

```
$kereta1->setGerbong(20);
```

Perintah tersebut setting *properties* jumlah gerbong pada objek $kereta1. Selain itu, perintah untuk memanggil *method* bacaMerek() dan bacaHarga() pun juga dapat dilakukan karena *class* keretaApi merupakan turunan dari *class* kendaraan. Adapun output di browser apabila script tersebut dijalankan adalah sbb:
"Jumlah gerbong dari KA Lokomotif yang seharga 15000000 adalah 20"

Latihan

a. Dalam *script* 'class-kendaraan.php', buatlah *class* baru bernama 'pesawat' yang merupakan turunan dari *class* kendaraan
b. Dalam *class* 'pesawat' yang telah dibuat, definisikan sebuah properti 'tinggiMaks' dengan sifat *private* untuk menyatakan ketinggian maksimum pesawat dan 'kecepatanMaks' dengan sifat *private* untuk menyatakan kecepatan maksimum pesawat
c. Dalam *class* 'pesawat', buatlah sebuah *method* bernama setTinggiMaks() untuk *setting* properti 'tinggiMaks' dan setKecepatanMaks() untuk *setting* properti kecepatan maksimum pesawat.
d. Dalam *class* 'pesawat', buatlah method bernama bacaTinggiMaks() untuk mengakses properti 'tinggiMaks'.
e. Dalam *class* 'pesawat', buatlah *method* bernama biayaOperasional() untuk menentukan biaya operasional pesawat, dimana untuk menghitung biaya ini tergantung dari harga pesawat yaitu dirumuskan:
    1) Jika tinggi maksimum pesawat lebih dari 5000 feet dan kecepatan maks lebih dari 800 km/jam, maka biaya operasional = 30% dari harga pesawat.
    2) Jika tinggi maksimum pesawat 3000-5000 feet dan kecepatan maks 500 – 800 km/jam, maka biaya operasional = 20% dari harga pesawat.
    3) Jika tinggi maksimum pesawat kurang dari 3000 feet dan kecepatan maks kurang dari 500 km/jam, maka biaya operasional = 10% dari harga pesawat.
    4) Selain itu, biaya operasionalnya = 5% dari harga pesawat.
f. Berdasarkan ketentuan pada nomor 1 s/d 5, tentukan biaya operasional dari pesawat-pesawat ini.

Merek Pesawat	Harga (Juta)	Tinggi Maks (Feet)	Kecepatan Maks (Km/Jam)
Boeing 737	2.000	7500	650
Boeing 747	3.500	8500	750
Cassa	750	3500	500

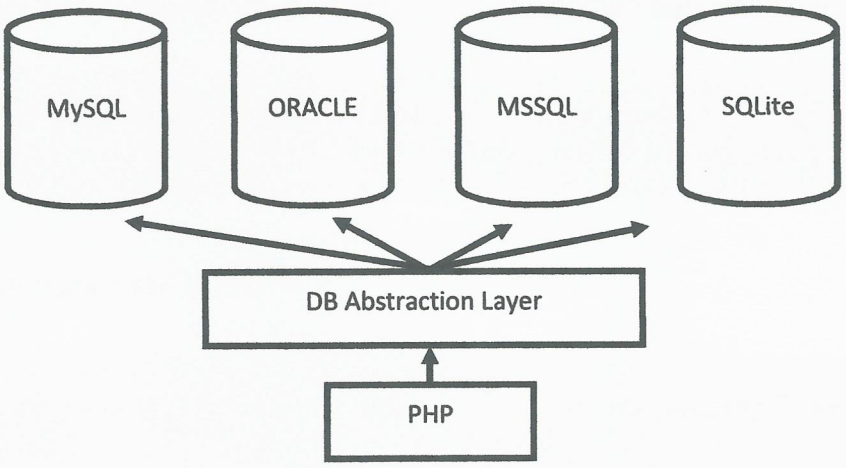
BAB XII

PHP PDO

1. Pendahuluan

PHP Data Objek atau PDO, merupakan salah satu API dari PHP yang disediakan untuk mengakses database dengan menggunakan teknik *objek-oriented* dan prosedural. Tujuan utama dari API PDO adalah menyediakan fasilitas untuk dapat mengakses berbagai macam *database* dengan menggunakan fungsi yang sama. PHP menyediakan abstraksi layer sebagai sebuah sarana untuk mentransformasikan seluruh perbedaan pada semua basis data.

Setiap basis data mempunyai cara tersendiri didalam melakukan pengaksesan datanya, perbedaan ini menyebabkan terlalu sulit untuk berpindah dari suatu database ke database yang lainnya. Akan tetapi terkadang kita membutuhkan beberapa tipe database untuk keperluan tertentu pada suatu aplikasi yang sama. Misalnya perusahaan anda secara resmi menggunakan MySQL, kemudian terdapat beberapa aplikasi yang secara ekslusif harus dijalan di ORACLE. Maka apakah perusahaan anda akan beralih menggunakan ORACLE dengan mengeluarkan anggaran yang besar? Dan harus membangun kembali sistem yang sudah ada dengan database yang baru? Untuk mengatasi masalah ini mulailah para developer mengembangkan abstraksi layer untuk melayani aplikasi didalam berkomunikasi dengan berbagai macam sistem basisdata.



Gambar 16. PDO

2. Menggunakan PDO

2.1 Prepare Statement

PDO atau *PHP Data Object* mendukung sepenuhnya penggunaan prepare statement. Prepare statement bertujuan menyediakan pernyataan SQL untuk dieksekusi oleh fungsi *execute()*. Selanjutnya proses *bindParam()* akan dilakukan untuk mengambil data dari hasil *query*.

Secara umum prosedur penggunaan *Prepare Statement* adalah sebagai berikut:

a. Defenisikan pernyataan SQL.
b. Lakukan proses *prepare()*, berdasarkan pernyataan SQL pada langkah pertama.
c. Kemudian lakukan proses binding dengan menggunakan *bindParam*, dimana langkah ini berguna untuk mencocokkan dengan posisi parameter pada SQL dengan posisi mendefenisikan variabel.
d. Selanjutnya lakukan proses pengumpulan hasil proses *query*/pengolahan ke daaalam *fetch()*.

Portotipe dari *prepare()* adalah sebagai berikut:

```
PDOStatement PDO::prepare (string statement[, array
driver_options])
```

Dimana string statement digunakan untuk mengisi pernyataan SQL. Proses pengisian pernyataan SQL dapat melewatkan dua model parameter yaitu menggunakan tanda ? atau menggunakan penamaan variabel. Perhatikan contoh penulisan berikut ini:

```
UPDATE tb_siswa SET siswa_nama = :siswa_nama,
siswa_aktif=:siswa_aktif WHERE siswa_nis=:siswa_nis;
```

Atau seperti pada kode berikut ini:

```
UPDATE tb_siswa SET siswa_nama = ? , siswa_aktif=? WHERE
siswa_nis=?;
```

```
<?php
try
{
     $db = new PDO ("mysql:host=localhost;
                              dbname=phpobject","root","");
     $sql = "SELECT kodebarang,jumlahorder
                         FROM barang_order
                         WHERE username = :username AND
                         jumlahorder > :jmlorder";
     $stmt = $db->prepare ($sql);
     $stmt->execute(array(":username"=>"winny12345",

     ":jmlorder"=>10));

     $db = null;
}
catch(PDOException $e)
{
          echo "Error: " . $e->getMessage();
}

?>
```

2.2 Fungsi *Execute()*

Fungsi *execute()* bertanggung jawab untuk melakukan eksekusi terhadap pernyataan SQL yang ada pada *prepare()*. Adapun prototipenya ada sebagai berikut:

```
Bool PDOStatement::execute ([array input_parameters]);
```

Dimana fungsi *execute()* dapat melewatkan parameter dalam bentuk array untuk parameter yang telah kita buar di dalam fungsi *prepare()*. Apabila anda tidak melewatkan paramater ke dalam fungsi tersebut maka *bindParam()*, bertanggung jawab untuk menyelesaikan kasus ini dengan menyediakan proses binding terhadap parameter-parameter yang dilewatkan pada proses *prepare()*.

```
<?php
try
{
     $db = new PDO ("mysql:host=localhost;
                              dbname=phpobject","root","");
     $sql = "SELECT kodebarang,jumlahorder
                         FROM barang_order
                         WHERE username = :username AND
                         jumlahorder > :jmlorder";
     $stmt = $db->prepare ($sql);
     $stmt->execute(array(":username"=>"winny12345",
                              ":jmlorder"=>10));
     $db = null;
}
catch(PDOException $e)
{
          echo "Error: " . $e->getMessage();
}
?>
```

2.3 *bindColumn()*

bindColumn() digunakan untuk mengambil nilai berdasarkan pada hasil pengolahan sesuia dengan pernyataan SQL pada fungsi *prepare()* dan menempatkan nilai tersebut pada variabel yang ada pada fungsi *bindColumn(),* sesuai dengan nomor *index* yang ada pada parameter pertama pada fungsi *bindColumn(). Index* kolom ini dimulai dari 1 dan bukan 0. Selain nomor urut pada kolom *field,* kita juga dapat menggunakan nama *field* dari kolom tersebut. Prototipenya sebagai berikut:

```
bool PDOStatement::bindColumn(mixed column, mixed &para [, int
type]);
```

Column terdiri atas nomor urut kolom pada *prepare* atau nama *field* yang dapat digunakan. Jika menggunakan nama *field* anda harus yakin bahwa kedua nama tersebut sama. Perhatikan contoh berikut ini:

```
<?php
try
{
     $db = new PDO ("mysql:host=localhost;
                              dbname=phpobject","root","");
     $sql = "SELECT kodebarang,jumlahorder
                         FROM barang_order
                         WHERE username = :username AND
                         jumlahorder > :jmlorder";
     $stmt = $db->prepare ($sql);
     $stmt->execute(array(":username"=>"winny12345",
                              ":jmlorder"=>10));
     $db = null;
}
catch(PDOException $e)
{
          echo "Error: " . $e->getMessage();
}
?>
```

2.4 *bindParam()*

bindParam() digunakan untuk menghubungkan variabel PHP berdasarkan pada tempat/lokasi nama ataupun tanda ? di dalam pernyataan SQL yang digunakan pada statement prepare dan variabel variabel ini akan dilewatkan kedalam pernyataan SQL tersebut. *bindParam()* berfungsi untuk mendefenisikan variabel yang akan dilewatkan kedalam pertanyataan SQL sedangkan pada *bindColumn()* digunakan untuk mengisi variabel yang didefenisikan dengan nilai hasil pengolahan dari pernyataan SQL. Fungsi akan dievaluasi ketika **PDOStatement::prepare** dieksekusi. Adapun prototipenya adalah:

```
bool PDOStatement::bindParam (mixed paramater, mixed &variable
[, int data_type [,int length [, mixed driver_options]])
```

Perhatikan contoh berikut ini:

```
<?php
try
{
     $db = new PDO
("mysql:host=localhost;dbname=phpobject","root","");
     $sql = "SELECT namamk FROM _tblsks
               WHERE kodemk = :kode AND sks = :sks";

     $kodemk = "oop";
     $sks = 3;

     $stmt = $db->prepare($sql);
     $stmt->bindParam(":kode", $kodemk, PDO::PARAM_STR, 12);
     $stmt->bindParam(":sks", $sks, PDO::PARAM_INT);
     $stmt->execute();
     $result = $stmt->fetch();
     echo $result[0];
}catch(PDOException $e)
{
     echo "Error: " . $e->getMessage();
}
?>
```

2.5 *fetch*

Setelah melakukan proses eksekusi, maka kita akan menampilkan data tersebut. Untuk melakukannya kita dapat menggunakan fungsi *fetch*. Secara umum fungsi *fecth* terdiri atas:

a. **PDOStatment::fetch()**
b. **PDOStatment::fetchAll()**
c. **PDOStatment::fetchColumn()**
d. **PDOStatment::fetchObject()**

Secara prinsip keempat model tersebut tidak terlalu jauh perbedaannya. **PDOStatment::fetch()** digunakan untuk mengumpulkan data/baris yang terasosiasi didalam sebuah variabel yang menyimpan hasil proses pengolahan *query,* yaitu melalui objek **PDOStatement**. Didalah **fetch()** tersebut, anda harus melewatkan

konstanta yang sudah disediakan oleh PDO. Misal **PDO::FETCH_ASSOC** atau **PDO::FETCH_NUM.**

```
<?php
try
{
     $db = new PDO ("mysql:host=localhost;
                     dbname=phpobject","root","");
     $ipk = 3.40;
     $sql = "SELECT * FROM _tblipk
                          WHERE ipk > ?";
     $stmt = $db->prepare($sql);
     $stmt->bindParam(1, $ipk, PDO::PARAM_INT);
     $stmt->execute();
     $stmt->bindColumn(1, $nama);
     $stmt->bindColumn(2, $email);
     $stmt->bindColumn(3, $ipk);
     while ($result = $stmt->fetch(PDO::FETCH_BOUND))
     {
          echo $ipk . "-->";
          echo $nama. ", " . $email;
          echo "<br />";
     }
     while ($result = $stmt->fetch(PDO::FETCH_BOTH))
     {
               echo $result["ipk"]. "-->";
               echo $result["nama"]. ", " . $result["nim"];
               echo "<br />";
     }

}catch(PDOException $e)
{
          echo "Error: " . $e->getMessage();
}
?>
```

DAFTAR PUSTAKA

Jubile Enterprise, 2012. *Buku Pintar HTML5+CSS3+Dreamweaver CS6.* Jakarta: PT Elex Media Komputindo.

Sakur, Stendy B. 2010. *PHP5 Pemrograman Berorientasi Objek Konsep dan Implementasi*. Yogyakarta. Penerbit Andi.

Solichin Achmad. 2010. *MySQL 5 Dari Pemula Hingga Mahir*. Tersedia di http://achmatim.net

Solichin Achmad. 2010. *Pemrograman Web dengan PHP dan MySQL*. Tersedia di http://achmatim.net

Yuana, Rosihan Ari. 2012. *Panduan Praktis OOP di PHP*. Tersedia di http://blog.rosihanari.net

https://id.wikipedia.org/wiki/Pemrograman_web diakses 17 Nopember 2016.

https://id.wikipedia.org/wiki/Basis_data di akses 11 Januari 2017

https://id.wikipedia.org/wiki/HTML diakses 11 Januari 2017

https://id.wikipedia.org/wiki/PHP diakses 11 Januari 2017