

MATERI PEMBEKALAN

JUNIOR WEB PROGRAMMER



HARI KE-2

J.620100.004.02	Menggunakan Struktur Data
J.620100.016.01	Menulis kode dengan prinsip sesuai guidelines dan best practices



JUNIOR WEB
PROGRAMMER

J.620100.004.02

Menggunakan Struktur Data

1

MENGGUNAKAN STRUKTUR DATA

Objektif:

1. Mengidentifikasi Konsep Data dan Struktur Data
 2. Menggunakan Tools Perangkat Lunak
-

Struktur data adalah sebuah bagian dari ilmu pemrograman dasar yang mempunyai karakteristik yang terkait dengan sifat dan cara penyimpanan sekaligus penggunaan atau pengaksesan data. Struktur data adalah suatu koleksi atau kelompok data yang dapat dikarakterisasikan oleh organisasi serta operasi yang didefinisikan terhadapnya. Pengertian struktur data adalah kumpulan elemen data (mulai dari *byte*) yang ditentukan tipe datanya, diorganisasi (dibentuk, disusun, atau dikelompokkan) dan akan diproses sesuai dengan tipe datanya.

Struktur data bertujuan agar cara merepresentasikan data dalam membuat program dapat dilakukan secara efisien dalam pengolahan di memori dan pengolahan penyimpanan dari program ke *storage* juga lebih mudah dilakukan. Struktur data sangat penting dalam sistem komputer. Terhadap setiap variabel di dalam program, secara eksplisit ataupun implisit, didefinisikan struktur data yang akan menentukan operasi yang berlaku terhadap variabel tersebut.

Pada garis besarnya, terdapat beberapa jenis berdasarkan elemen datanya, yaitu:

1. Struktur data sederhana, yang terdiri atas:
 - a. Array (Larik)
 - b. Record (Catatan)

2. Struktur data majemuk, meliputi:
 - a. Linier (*Stack*/Tumpukan, *Queue*/Antrian, *List* dan *Multi-List*/Daftar)
 - b. Non-Linier (*Binary Tree*/Pohon Biner, *Graph*/Graf)

1. Mengidentifikasi Konsep Data dan Struktur Data

Konsep data dapat dijelaskan dengan membahas basis data. Basis data adalah inti sistem informasi di dalam komputer. Wajar jika pengolahan data sering dipelajari dan dikembangkan dengan teknologi modern masa kini supaya lebih menjanjikan dan selaras dengan tujuan organisasi.

Basis Data

Basis data juga mempunyai ciri-ciri penting yang menjadi karakteristik pribadinya dibanding komponen lain di dalam sistem informasi. Pertama, basis data merupakan sistem yang bisa menyimpan data ke media penyimpanan seperti *floppy disk* ataupun *harddisk*, sehingga ada media khusus yang benar-benar dipakai untuk menyimpan segala bentuk data di dalamnya. Basis data juga dikenal sebagai sistem yang bekerja dengan pengolahan data yang bisa diubah, ditambahkan hingga dihapus sesuai keinginan pengguna. Perintah kerja tersebut bisa dilakukan dengan mudah dan juga terkontrol dengan media *software*, di dalamnya ada semacam *tools* yang bisa digunakan untuk perintah tersebut. Ciri utama basis data terakhir adalah data yang tersimpan di dalam sistem bisa terpisah dari program utama sesuai keinginan pengguna, sehingga setiap data yang tersedia tidak akan ribet ketika diakses oleh pengguna. Ini memungkinkan pengguna mendapat data dengan cepat dan mudah kapanpun.

Basis data memiliki tujuan tertentu demi kemudahan akses informasi yang diperlukan pengguna. Untuk lebih jelasnya, berikut beberapa tujuan penerapan basis data di dalam organisasi:

1. Membuat para *user* mudah mendapat data yang diperlukan
2. Menyediakan tempat khusus untuk menyimpan data yang relevan
3. Menghapus berbagai data yang dianggap tidak penting atau terlalu berlebihan di dalam sistem
4. Melindungi data pribadi atau organisasi jika terjadi kerusakan fisik di dalamnya
5. Memungkinkan terjadinya perkembangan lanjutan sistem database yang lebih modern dan terorganisir di dalamnya.

Adapun manfaat yang dari basis data adalah:

1. Menjadi komponen penting di dalam sistem informasi yang diperlukan individu atau perusahaan. Basis data menjadi dasar informasi yang dibutuhkan pengguna.
2. Menambah kualitas informasi dengan akurat, relevan dan cepat. Akhirnya setiap informasi yang diberikan basis data tidak basi atau tidak *upto_date*. Informasi dianggap berguna apabila manfaatnya lebih baik dibanding biaya untuk mendapatkannya.
3. Mengatasi masalah *redundancy* data atau rangkap data di dalam sistem.
4. Menghindari masalah inkonsistensi data
5. Basis data mengatasi kesulitan jika memerlukan data yang diperlukan.
6. Menambah kemudahan ketika menyusun data.
7. Database bisa digunakan banyak pengguna dalam waktu yang bersamaan.
8. Database digunakan untuk perlindungan sekaligus keamanan data. Setiap data yang tersedia bisa diakses dan dimanipulasi pihak yang

mendapat otoritas dengan login menggunakan akun dan *password* yang telah disiapkan.

9. Supaya pemakai bisa menyusun pandangan abstraksi data tersedia. Tujuannya agar menyederhanakan komunikasi antar pengguna dengan database dan sistem yang akhirnya bisa mempresentasikan pandangannya masing-masing.

Basis data adalah sekumpulan data yang telah terorganisir dengan rapi dan baik oleh sistem. Semua data tersebut bisa disimpan, dimanipulasi dan bisa dipanggil kapan saja oleh penggunanya.

Data merupakan kumpulan kejadian dan fakta yang bisa dipakai untuk penyelesaian masalah berbentuk informasi khusus di dalam database. Data bisa tercantum dalam bentuk bunyi, gambar, teks, simbol, angka, huruf atau kombinasi beberapa di dalamnya.

File menunjukkan kumpulan beberapa *record* yang bisa menggambarkan informasi data tertentu dengan baik dalam sebuah database. Contoh file yang ada didalam database adalah informasi berisi data yang berisikan nama barang tertentu di dalamnya.

Tabel disebut sebagai kumpulan *record* dan *field* yang sudah lengkap di sistem database.

Record adalah kumpulan *field* yang sudah lengkap di dalam basis data. Kumpulan tersebut biasanya dihitung di satuan baris yang telah tersedia di database.

Istilah *field* merujuk pada kumpulan berbagai karakter di dalam database yang mempunyai arti di dalamnya. *Field* adalah kolom di dalam tabel yang bisa diisi nama tertentu.

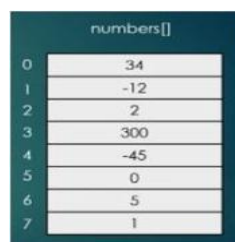
Struktur Data

Secara garis besar, struktur data adalah model logika untuk menyimpan, merepresentasikan data dan cara untuk menyediakan tempat yang terorganisir agar data yang disimpan dapat dibaca dengan lebih mudah. Dalam struktur data, terdapat beberapa jenis berdasarkan elemen datanya, yaitu:

1. Array (Larik)

Larik adalah struktur data statik yang menyimpan sekumpulan elemen yang bertipe sama. Setiap elemen diakses langsung melalui indeksinya. Indeks larik harus tipe data yang menyatakan keterurutan misalnya integer atau karakter. Banyaknya elemen larik harus sudah diketahui sebelum program dieksekusi. Tipe elemen larik dapat berupa tipe sederhana, tipe terstruktur, atau tipe larik lain. Nama lain *array* adalah larik, tabel, atau vektor. Di dalam *array* terdapat beberapa operasi yang perlu diketahui, yaitu:

- *Insert* = Data pada *array* dapat ditambahkan baik di awal, tengah, maupun akhir.
- *Search/Read* = Mengambil data pada *array* dilakukan dengan menggunakan *index* dari data tersebut.
- *Remove* = Menghapus data pada *array* dapat dilakukan berdasarkan index ataupun langsung tertuju pada data apa yang akan dihapus.



numbers[]	
0	34
1	-12
2	2
3	300
4	-45
5	0
6	5
7	1

Gambar 1.1 Array Sederhana

2. Stack (Tumpukan)

Stack (tumpukan) adalah *list linier* yang dikenal elemen puncaknya (*top*), aturan penyisipan dan penghapusan elemen tertentu (penyisipan selalu dilakukan "di atas" (*top*), penghapusan selalu dilakukan pada *top*). Aturan penyisipan dan penghapusan, *top* adalah satu-satunya alamat tempat terjadi operasi. Elemen yang ditambahkan paling akhir akan menjadi elemen yang akan dihapus. *Stack* atau tumpukan merupakan sebuah koleksi objek yang menggunakan prinsip LIFO (*Last in First Out*), yaitu data yang terakhir kali dimasukkan akan pertama kali keluar dari *stack* tersebut.

Ciri Stack:

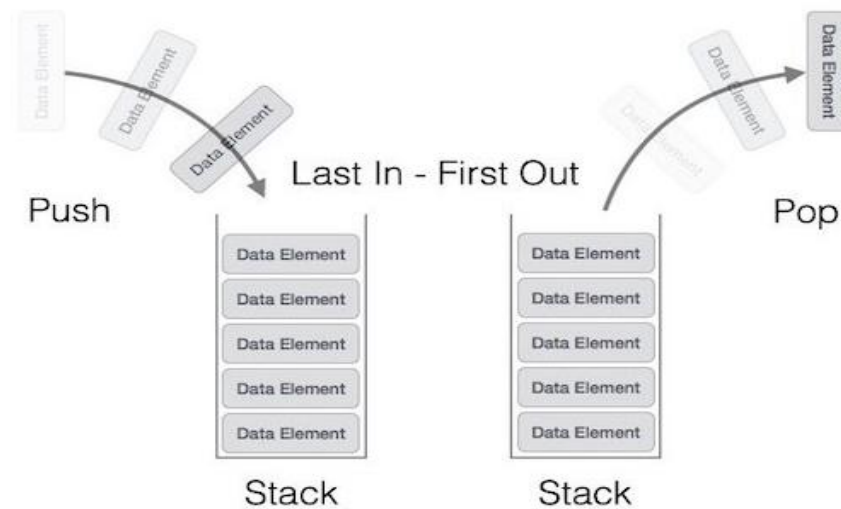
- Elemen *TOP* (puncak) diketahui
- Penyisipan dan penghapusan elemen selalu dilakukan di *TOP*
- *Last in First Out*

Pemanfaatan Stack:

- Perhitungan ekspresi aritmatika (*posfix*)
- Algoritma *backtracking* (runut balik)
- Algoritma rekursif

Stack memiliki beberapa operasi, yaitu:

- `push(item);` menambahkan suatu item baru ke atas (*top*) dari *stack*. Perlu *item* dan tidak mengembalikan apapun.
- `pop();` menghapus *item* teratas dari *stack*. Tidak perlu parameter dan mengembalikan *item*, *stack* berubah.
- `peek();` mengintip *top item* dari *stack* tetapi tidak menghapusnya. Tidak memerlukan parameter dan *stack* tidak berubah.



Gambar 1.2 Ilustrasi Stack

3. Queue (Antrian)

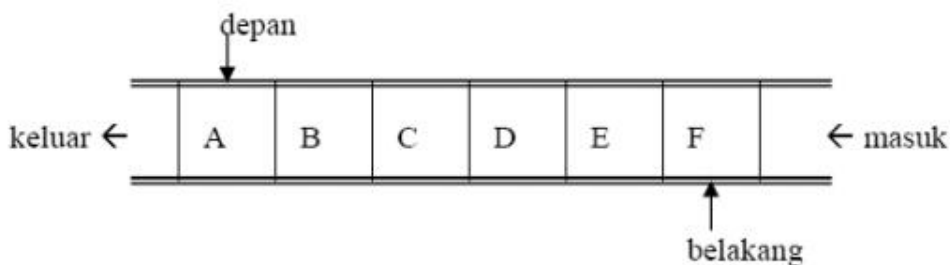
Queue (antrian) adalah *list linier* yang dikenali elemen pertama (*head*) dan elemen terakhirnya (*tail*). *Queue* juga berarti salah satu bentuk struktur data yang juga merepresentasikan *linked list*, di mana yang berbeda dalam *queue* tersebut adalah cara menambah data dan mengambil data. Sesuai dengan namanya yaitu *queue* atau antrian, data yang dimasukan dari belakang (*insert at back*), sehingga data yang pertama kali dimasukan berada pada node pertama, dan data yang dimasukan terakhir juga akan berada pada node yang terakhir. Di mana untuk pengambilan proses pengambilan datanya, yang diambil adalah data pertama, dan setelah data pertama diambil maka *node* yang berisi data pertama tersebut akan di-*null*-kan, sehingga posisi *node* pertama akan berpindah pada *node* setelah *node* pertama tersebut. Proses ini biasanya disebut dengan *FIFO*, atau *First in First Out*.

Metode yang digunakan untuk memasukan data ke dalam *queue* tersebut dinamakan *enqueue* dan yang untuk mengambil data dinamakan *dequeue*, di mana untuk pengambilan proses pengambilan

datanya, yang diambil adalah data pertama, dan setelah data pertama diambil maka *node* yang berisi data pertama tersebut akan di-*null*-kan, sehingga posisi *node* pertama akan berpindah pada *node* setelah *node* pertama tersebut.

Operasi pada *queue* adalah sebagai berikut:

- **enqueue(item);** menambahkan suatu *item* baru ke ujung satu antrian. Perlu *item* dan tidak mengembalikan sesuatu.
- **dequeue();** menghapus *item* depan dari antrian. Tidak memerlukan parameter dan mengembalikan *item*-nya. Antrian termodifikasi.



Gambar 1.3 Ilustrasi Queue

4. List dan Multi - List (Daftar)

List linier adalah sekumpulan elemen bertipe sama, yang mempunyai keterurutan tertentu, yang setiap elemennya terdiri dari 2 bagian. Sebuah *list linier* dikenali dengan (1) elemen pertamanya, biasanya melalui alamat elemen pertama yang disebut (*first*); (2) Alamat elemen berikutnya (*sukesor*), jika diketahui alamat sebuah elemen, yang dapat diakses melalui *fieldnext*; (3) Setiap elemen mempunyai alamat, yaitu tempat elemen disimpan dapat diacu. Sebuah elemen akan megacu pada alamat yang terdefinisi, informasi yang tersimpan pada elemen *list* dapat diakses; (4) Elemen terakhirnya.

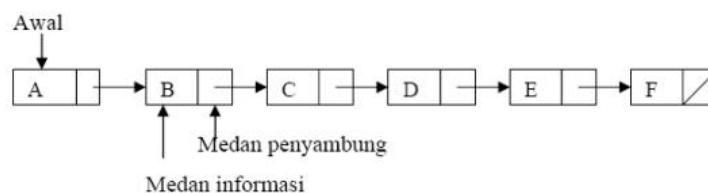
Double Linked List merupakan suatu *linked list* yang memiliki dua *variabel pointer* yaitu *pointer* yang menunjuk ke *node* selanjutnya dan *pointer* yang menunjuk ke *node* sebelumnya.

Di dalam *Linked List* ada beberapa operasi yang perlu diketahui yaitu:

- *Insert* di awal
- *Insert* di akhir
- *Remove* sebuah nilai
- *Remove node* pertama
- *Traversing forward and reverse*

Jika Tail = Null, maka kondisi *Linked List* adalah kosong.

Perintah yang tepat untuk menyatakan *Linked list* berada dalam kondisi kosong, adalah Head=Tail=Null.



Gambar 1.4 Ilustrasi Linked List

1.1 Konsep Data dan Struktur Data Diidentifikasi sesuai dengan Konteks Permasalahan

Dalam pemrograman, variabel adalah suatu lokasi penyimpanan (di dalam memori komputer) yang berisikan nilai atau informasi yang nilainya tidak diketahui maupun telah diketahui. Dalam definisi bebasnya, variabel adalah kode program yang digunakan untuk menampung nilai tertentu. Nilai yang disimpan di dalam variabel selanjutnya dapat dipindahkan ke dalam database, atau ditampilkan kembali ke pengguna.

Nilai dari variabel dapat diisi dengan informasi yang diinginkan dan dapat diubah nilainya pada saat kode program sedang berjalan. Sebuah variabel memiliki nama yang digunakan untuk mengakses nilai dari variabel itu. Variabel digunakan untuk menampung nilai inputan dari *user*, atau nilai yang didefinisikan sendiri.

Mendeklarasikan variabel adalah memberikan nama variabel sebagai identitas pengenalan dan menentukan tipe data variabel. Beberapa *identifier* yang sejenis bisa dideklarasikan bersamaan. Mendeklarasikan konstanta adalah memberikan nama konstanta sebagai identitas pengenalan dan menentukan nilai konstanta.

Tipe Data Sederhana

Tipe data sederhana merupakan tipe data dasar yang sering dipakai oleh program, meliputi: *integer* (bilangan bulat), *real* (bilangan pecahan), *boolean* (logika), dan *char* (alfanumerik dan tanda baca).

1. Bilangan Integer

merupakan tipe data berupa bilangan bulat, terbagi atas beberapa kategori. Tidak mengandung pecahan dan biasanya disajikan dalam memori komputer sebagai angka bulat. Di dalam aritmetika komputer, *integer* mudah untuk disajikan dan diproses. Tabel 1.1 menunjukkan jenis data, ukuran dalam memori dan rentang nilainya.

Tabel 1.1 Kategori Bilangan Integer

Tipe Data	Ukuran Tempat	Rentang Nilai
Byte	1 byte	0 s/d +255
Shortint	1 byte	-28 s/d +127
Integer	2 byte	-32768 s/d 32767
Word	2 byte	0 s/d 65535
Longint	4 byte	2147483648 s/d 2147483647

2. Bilangan Real

Bilangan *real* atau nyata merupakan jenis bilangan pecahan, dapat dituliskan secara biasa atau model *scientific*. ditulis menggunakan titik (atau koma) desimal. Contoh bilangan *real*: 34.265 -3.55 0.0 35.997E+11, di mana E merupakan simbol perpangkatan 10, sehingga

452.13 mempunyai nilai sama dengan 4.5213e2. Penggolongan tipe data bilangan *real* dapat dilihat pada tabel 1.2.

Tabel 1.2 Kategori Bilangan Real

Tipe Data	Ukuran Tempat	Rentang Nilai
Real	6 byte	2.9×10^{-39} s/d 1.7×10^{38}
Single	4 byte	1.5×10^{45} s/d 3.4×10^{38}
Double	8 byte	5.0×10^{-324} s/d 1.7×10^{308}
Extended	10 byte	3.4×10^{-4932} s/d 1.1×10^{4932}
Comp	8 byte	-9.2×10^{18} s/d 9.2×10^{18}

3. Char

Tipe lain dari data adalah karakter, yang elemennya merupakan aksara (simbol): (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, ..., X, Y, Z, ?, ,*...) meliputi digit numerik, karakter alfabetik, dan spesial karakter (simbol-simbol lain). Karakter merupakan komponen dari *string* karakter (atau disebut *string* saja), yang merupakan tipe data yang teramat penting.

Karakter yang dikenal dalam komputer biasanya mencakup:

- 26 huruf besar dan kecil latin (atau sesuai dengan huruf yang digunakan di suatu negara, seperti Cina, Jepang, Arab, dan sebagainya)
- 10 digit Latin, Arab atau Romawi
- Karakter khusus yang dapat dicetak, seperti tanda aritmatik, tanda baca, dan sebagainya
- Ruang kosong atau *blank*
- Karakter pengendalian (tidak dapat dicetak), seperti pengendalian kursor, dan sebagainya.

Tipe data ini menyimpan karakter yang diketikkan dari *keyboard*, memiliki 266 macam yang terdapat dalam tabel ASCII (*American Standard Code for Information Interchange*). Contoh: 'a' 'B' '+', dan sebagainya. Yang perlu diingat bahwa dalam menuliskannya harus dengan memakai tanda kutip tunggal. Jenis data ini memerlukan alokasi memori sebesar 1(satu) *byte* untuk masing-masing data.

4. Tipe Data Boolean

merupakan tipe data logika, yang berisi dua kemungkinan nilai: *TRUE* (benar) atau *FALSE* (salah), Yang kerap kali dinyatakan pula sebagai 1 dan 0. Hal tersebut membuat sebuah satuan data dapat cukup berisi satu bit saja.

Tipe Data Terstruktur

Tipe ini terdiri atas *string*, *array*, *record*, dan *pointer*. *String* adalah tipe data jenis *array*, namun *string* memiliki kekhasan tersendiri sebagai *array* dari karakter.

1. Tipe Data String

Merupakan suatu data yang menyimpan *array* (larik), sebagai contoh 'ABCDEF' merupakan sebuah konstanta *string* yang berisikan 6 *byte* karakter. Ukuran Tempat untuk tipe data ini adalah 2 s/d 256 *byte*, dengan jumlah elemen 1 s/d 255. *String* dideklarasikan dengan *string* (konstanta) atau *string*. Bila ukuran *string* tidak didefinisikan maka akan banyak memakan ruang, karena ukuran *string* menyesuaikan dengan *default*-nya. Sebagai contoh:

```
var kata: string [20];          atau          var kata: string;
```

String merupakan *array* dari karakter. Kesimpulannya adalah kata[1] merupakan karakter pertama dari *string*, kemudian kata[2], merupakan elemen kedua, dan seterusnya.

2. Tipe Data Array (Larik)

Suatu *array* adalah sebuah struktur data yang terdiri atas banyak variabel dengan tipe data sama, di mana masing-masing elemen variabel mempunyai nilai indeks. Setiap elemen *array* mampu untuk menyimpan satu jenis data (yaitu: variabel). Suatu *array* dinyatakan dengan *type*. *Array* merupakan struktur data yang statis, yaitu jumlah elemen yang ada harus ditentukan terlebih dahulu dan tak bisa diubah saat program berjalan. Sifat masing-masing elemen *array* mengikuti jenis data yang dimilikinya, untuk *array* dengan tipe bilangan *integer* atau *real* bisa melakukan berbagai standar operasi aritmatika seperti penjumlahan, perkalian, pengurangan, dan sebagainya. Bahwa sifat dari *array* dimanfaatkan untuk operasi matrik.

a. Array Satu Dimensi

Sebuah *array* dimensi satu, yang misalnya diberi nama NILAI, dapat dibayangkan berbentuk seperti Gambar 1.5.



Nilai(1)	Nilai(2)	Nilai(3)	- - -	Nilai(n)
----------	----------	----------	-------	----------

Gambar 1.5 Array Dimensi Satu

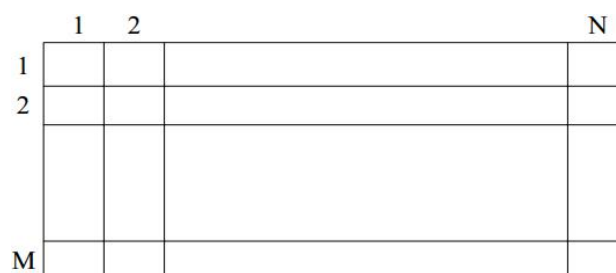
Subscript atau indeks dari elemen *array* menyatakan posisi, elemen pada urutan dalam *array* tersebut. Notasi yang digunakan bagi elemen *array*, biasanya adalah nama *array* dilengkapi dengan *subscript*. Secara umum, suatu *array* dimensi satu A dengan tipe data T dan *subscript* bergerak dari L sampai dengan U, ditulis sebagai $A(L:U) = (A(I))$, $I = L, L+1, L+2, \dots, U$, dan setiap elemen $A(I)$ bertipe data T.

Sebagai contoh, dapat menuliskan data hasil pencatatan suhu suatu ruangan setiap satu jam selama periode 24 jam, dalam sebuah *array* dimensi satu. Harga minimum dari *subscript* dari array disebut batas bawah atau *lower bound*, sedangkan harga maksimumnya

disebut batas atas atau *upper bound*. Jadi pada array tersebut, L merupakan batas bawah, dan U batas atas, sedangkan untuk *array* "suhu" yang elemennya dapat ditulis sebagai $SUHU(I)$, batas bawahnya adalah 1 dan batas atasnya 24. $SUHU(I)$ menyatakan suhu pada jam ke-1, dan I memenuhi $1 \leq I \leq 24$, I merupakan *integer*. Batas bawah dari *array*, pada beberapa aplikasi, tidak selalu diambil 1. Kadang-kadang diambil batas bawah nol, bahkan juga negatif. Banyaknya elemen sebuah array disebut rentang atau *range*. Sehingga *array* $A(L:U)$ mempunyai *range* sebesar $U-L+1$. Secara khusus bila $L=1$ dan $U=N$, maka *range* dari *array* $A(1:N)$ adalah $N-1+1 = N$.

b. Array Multidimensi

Dalam *array* multidimensi terdiri atas baris (*row*) dan kolom (*column*). *Index* pertama adalah baris dan yang kedua adalah kolom. Sebuah *array* dimensi banyak atau multi-dimensional *array* didefinisikan sebagai sebuah *array* yang elemennya berupa *array* pula. Misal *array* B mempunyai M elemen berupa *array* pula, yang terdiri dari N elemen. Bila hal tersebut digambarkan, akan terbentuk baris dan kolom seperti terlihat pada gambar berikut.



	1	2		N
1				
2				
M				

Gambar 1.6 Array Dimensi Dua

3. Record

Sebuah *record* rekaman disusun oleh beberapa *field*. Tiap *field* berisi data dari tipe dasar/bentukan tertentu. *Record* mempunyai kelebihan untuk menyimpan suatu sekumpulan elemen data yang berbeda-beda

tipe (dibanding *array*). *Record* (Catatan) adalah definisi tipe dan sekumpulan primitif (operasi dasar) terhadap tipe tersebut. Tipe diterjemahkan menjadi tipe terdefinisi dalam bahasa pemrograman yang bersangkutan.

Contoh, sebuah *record* dengan empat buah field.

Field 1	Field 2	Field 3	Field 4
---------	---------	---------	---------

Gambar 1.7 Record

Cara pendeklarasian dari *record* adalah sebagai berikut:

- Mendefinisikan tipe dari *record* (jumlah *field*, jenis tipe data yang dipakai),
- Mendefinisikan variabel untuk dilakukan operasi.

4. Pointer

Tipe data *pointer* bersifat dinamis, variabel akan dialokasikan hanya pada saat dibutuhkan dan sesudah tidak dibutuhkan dapat dialokasikan kembali.

Kelebihan dari *pointer* adalah memiliki sifat dinamis, ukurannya disesuaikan berdasarkan kebutuhan, dan alokasi variabel dapat diatur sesuai kebutuhan. Bentuk umum:

```
Var <namavar> : <^tipe data>
```

1.2 Alternatif Struktur Data dibandingkan Kelebihan dan Kekurangannya untuk Konteks Permasalahan yang Diselesaikan

Dalam penggunaan struktur data, perlu diketahui terlebih dahulu bagaimana data tersebut akan dapat diproses dalam komputer sehingga struktur data apa yang paling tepat untuk digunakan. Perlu diketahui secara sekilas gambaran mengenai struktur data yang dijabarkan pada tabel berikut.

Tabel 1.3 Kelebihan dan Kekurangan Struktur Data

Struktur Data	Kelebihan	Kekurangan
<i>Array</i>	Penambahan data di belakang mudah dilakukan	Ukuran tetap, Penghapusan lambat, pencarian lama
<i>Array Terstruktur</i>	Pencarian jauh lebih cepat dibandingkan <i>array</i> yang tidakurut	Ukuran tetap, penghapusan lambat, penyisipan lama
<i>Stack/Tumpukan</i>	Penambahan dilaksanakan dengan cepat, akses terhadap data yang terakhir kali dimasukkan bisa dilakukan dengan cepat	Pencarian dan penghapusan lambat
<i>Queue/Antrian</i>	Data yang pertama kali masuk mudah untuk diakses	Akses data yang lain lambat
<i>Linked List/Berantai</i>	Penyisipan dan penghapusan data mudah	Pencarian lama
<i>Binary Tree/Pohon Biner</i>	Penyisipan dan penghapusan data mudah	Penghapusan kompleks

2. Menggunakan Tools Perangkat Lunak

Ada beberapa aplikasi database yang bisa digunakan. Beberapa aplikasi tersebut antara lain:

1. MySQL

Merupakan salah satu aplikasi yang cukup populer diantara aplikasi lain database. Aplikasi yang satu ini memiliki sifat *open source* serta bisa digunakan untuk membuka *platform* dengan baik tanpa mengalami gangguan. MySQL sendiri bisa digunakan sebagai sistem operasi dari Linux, Windows, mac OS X, dan masih banyak lagi.

Terlebih, aplikasi ini juga bersifat network, sehingga bisa menggunakan dengan multi *user*.



Gambar 1.8 Logo MySQL

2. MariaDB

Merupakan sebuah aplikasi yang dikembangkan oleh sistem basis data yang sama dengan aplikasi MySQL. Disebutkan alasan mengapa mengembangkan aplikasi ini karena sebagai bentuk kekhawatiran pada Oracle Corporation. Adapun tujuan dikembangkannya MariaDB adalah mempertahankan dan menjaga kompatibilitas tinggi terhadap MySQL.

3. Microsoft SQL Server

Aplikasi database yang satu ini tentu sudah diketahui. Sesuai dengan namanya, aplikasi ini merupakan sebuah aplikasi basis data yang berasal dari Microsoft. Bahasa database yang digunakan tersebut adalah Transact-SQL yang merupakan gabungan dari SQL standar ISO/ANSI. Memang bahasa ini merupakan bahasa yang populer dan banyak digunakan oleh Sybase ataupun Microsoft.

Pada mulanya, Ms. SQL Server merupakan data yang digunakan untuk database skala menengah atau kecil. Saat ini bisa menggunakan aplikasi ini untuk database dengan skala yang lebih besar. Adapun beberapa kelebihan jika menggunakan aplikasi yang satu ini adalah dapat digunakan oleh banyak platform, mempunyai tipe data banyak, database cukup banyak, dan lain sebagainya.



Gambar 1.9 Logo Microsoft SQL Server

4. Oracle Database

Aplikasi ini merupakan aplikasi database terbaik yang bisa digunakan untuk menyimpan data dengan skala besar. Bahkan bisa menggunakannya untuk menyimpan data hingga ukuran *terabyte*. Saat ini pun ada banyak orang yang menggunakan aplikasi ini. Alasannya karena dapat menyimpan dalam jumlah besar selain itu juga mudah di-*download*. Pada aplikasi ini juga memiliki versi gratis serta tampilan yang sesuai dengan yang baru saja terjun ke dunia database.

5. PostgreSQL

Aplikasi database selanjutnya adalah PostgreSQL. atau yang lebih sering dikenal dengan nama PostgreSQL. Aplikasi ini merupakan aplikasi basis data yang bersifat data relasional sehingga bisa digunakan untuk menyimpan hingga mengembalikan data dengan aman. Aplikasi ini sendiri mampu merespon perintah dari perangkat lunak atau aplikasi yang lain. Bisa menggunakan aplikasi ini dengan lebih ringan, meskipun digunakan oleh banyak pengguna. Perlu diketahui juga bahwa pengguna yang mempunyai macOS Server, aplikasi PostgreSQL adalah database *default*. Hal ini berarti aplikasi basis data yang satu ini memang sudah terinstal dan bawaan dari OS tersebut.

6. SQLite

Aplikasi database lainnya adalah SQLite. Aplikasi yang satu ini cocok bagi yang ingin mempunyai data yang terstruktur. Bahkan bisa memanfaatkannya sebagai *caching* serta menyediakan data dari *cloud*. Bagi yang juga suka dengan data dalam bentuk kolom dan baris maka gunakan saja aplikasi yang satu ini.

7. DBeaver

Merupakan sebuah aplikasi yang mempunyai *mode graphical* atau GUI. Bisa mendapatkan versi *Community Edition* ataupun versi yang berbayar. Adapun beberapa fitur menarik yang disediakan adalah dukungan banyaknya *platform* serta kemampuan menulis berbagai file ekstensi (*plugins*).



Gambar 1.10 Logo DBeaver

8. MongoDB

Aplikasi basis data yang satu ini memiliki cara kerja yang berorientasi pada *open source* serta *cross platform*. Tak hanya itu, aplikasi ini juga digolongkan sebagai aplikasi dengan basis data NoSQL sehingga cara kerjanya pun memakai prinsip yang hampir sama dengan JSON (aplikasi database lain).

9. Apache Cassandra

Aplikasi yang satu ini adalah aplikasi yang bersifat *cluster*. Aplikasi ini sendiri memiliki *benchmark* yang bisa digunakan dengan lebih baik jika dibandingkan dengan NoSQL lainnya. Aplikasi database yang satu ini juga dapat menjamin keamanan data saat perangkat mati. Tak

hanya itu, aplikasi ini juga sudah dilengkapi dengan tim pendukung yang sangat profesional dalam keperluan enterprise.



Gambar 1.11 Logo Apache Cassandra

2.1 Struktur Data Diimplementasikan sesuai dengan Bahasa Pemrograman yang akan Dipergunakan

Sintaks penulisan variabel dalam bahasa pemrograman berbasis web pada umumnya adalah seperti berikut:

```
$namaVariabel = [nilai variabel];
```

Bagian `$namaVariabel` adalah nama yang diberikan untuk variabel tersebut, sedangkan `[nilai variabel]` adalah nilai yang akan dimasukkan ke dalam variabel tersebut.

1. Integer

Nilai *integer* dapat bernilai positif (+) maupun negatif (-). Jika tidak diberi tanda, maka diasumsikan nilai tersebut adalah positif.

```
$b = 5;  
$c = $a + 5;
```

2. Real

Sama saja dengan *integer*, hanya saja, tipe data ini menerima data desimal dengan angka "." sebagai pembaginya.

```
$nilaiBahasaIndonesia = 9.3;  
$rataRata = ($nilaiMatematika + $nilaiBahasaIndonesia) / 2;
```

3. Boolean

Tipe data ini adalah tipe data yang paling simpel, akan tetapi butuh logika yang kuat untuk bisa memanfaatkannya dengan benar.

```
$apakahSiswaLulus = true;  
$apakahSiswaSudahUjian = false;
```

4. String

Semua teks bertipe data *string* diapit oleh tanda petik satu (' ') maupun tanda petik dua (" ").

```
$namaDepan = "Wiwied";  
$namaBelakang = 'Widiastuti';
```

5. Array

Tipe data *array* berfungsi untuk menyimpan himpunan data. Himpunan data tersebut diapit oleh tanda kurung siku ([]).

```
$listMahasiswa = ["Wied", "Rini", "Julia", "Kuwat"];
```

6. Pointer

Memiliki sifat dinamis, ukurannya disesuaikan berdasarkan kebutuhan, dan alokasi variabel dapat diatur sesuai kebutuhan. Pendeklarasian seperti variabel biasa, namun ditambahkan simbol topi (^) sebelum tipe datanya.

```
Var <namavar> : <^tipe>
```

2.2 Akses terhadap Data Dinyatakan dalam Algoritma yang Efisiensi sesuai Bahasa Pemrograman yang akan Dipakai

Beberapa aturan tentang cara penggunaan dan penulisan variabel atau konstanta yang perlu diperhatikan adalah:

1. Apakah penulisan variabel harus diawali dengan tanda khusus/tidak
2. Apakah variabel dalam bahasa tertentu bersifat case sensitif/tidak
3. Bagaimana cara memberikan nilai kepada variabel
4. Apakah variabel dalam bahasa tertentu tidak memerlukan deklarasi terlebih dahulu
5. Apakah variabel dalam bahasa tertentu memiliki tipe/tidak
6. Apakah terdapat variabel sistem dalam bahasa pemrograman tertentu (*predefined variables*)

Terdapat empat jenis pengaksesan satuan data, yaitu *Sequential Access*, *Direct Access*, *Random Access* dan *Associative Access*. Penjelasan dari keempat pengaksesan tersebut adalah:

1. Sequential Access

Memori diorganisasikan menjadi unit-unit data, yang disebut *record*. Akses dibuat dalam bentuk urutan linier yang spesifik. Informasi pengalamatan dipakai untuk memisahkan *record-record* dan untuk membantu proses pencarian. Mekanisme baca/tulis digunakan secara bersama (*shared read/write mechanism*), dengan cara berjalan menuju lokasi yang diinginkan untuk mengeluarkan *record*. Waktu *access record* sangat bervariasi.

Contoh *sequential access* adalah akses pada pita magnetik

2. Direct Access

Seperti *sequential access*, *direct access* juga menggunakan *shared read/write mechanism*, tetapi setiap blok dan *record* memiliki alamat yang unik berdasarkan lokasi fisik. Akses dilakukan secara langsung terhadap kisaran umum (*general vicinity*) untuk mencapai lokasi akhir. Waktu aksesnya bervariasi.

Contoh *direct access* adalah akses pada disk.

3. Random Access

Setiap lokasi dapat dipilih secara *random* dan diakses serta dialamati secara langsung. Waktu untuk mengakses lokasi tertentu tidak tergantung pada urutan akses sebelumnya dan bersifat konstan.

Contoh *random access* adalah sistem memori utama.

4. Associative Access

Setiap *word* dapat dicari berdasarkan pada isinya dan bukan berdasarkan alamatnya. Seperti pada RAM, setiap lokasi memiliki mekanisme pengalamatannya sendiri. Waktu pencariannya tidak

bergantung secara konstan terhadap lokasi atau pola *access* sebelumnya.

Contoh *associative access* adalah memori *cache*.

Tahapan Pembuatan Struktur Data

Pembuatan struktur data dalam bahasa pemrograman harus melalui tahap berikut:

1. Tahap Pertama

Spesifikasi atau pendeskripsian struktur data menyatakan apa yang dapat dilakukan struktur data, bukan cara penempatannya. Pendeskripsian ini melibatkan level *logic* sehingga dapat digunakan konvensi matematika untuk menyatakan sifat-sifat struktur data yang dikehendaki.

2. Tahap Kedua

Implementasi menyatakan cara penerapan struktur data dengan struktur data yang telah ada. Implementasi struktur data adalah proses pendefinisian tipe data abstrak sehingga semua operasi dapat dieksekusi komputer. Implementasi berisi deklarasi struktur penyimpanan item data serta algoritma untuk implementasi operasi, sehingga menjamin terpenuhinya karakteristik struktur data, relasi item data atau invariant pada struktur data tersebut.

3. Tahap Ketiga

Pemrograman struktur data adalah penterjemahan menjadi pernyataan dalam bahasa pemrograman tertentu. Struktur data dibangun menggunakan fasilitas pembentuk atau pembuatan struktur data yang disediakan bahasa seperti *array*, *record* dan lain-lain.

Implementasi Rancangan Struktur Data

Database adalah sebuah media utama yang harus dibuat dalam membangun sebuah basis data, karena dapat meletakkan beberapa tabel dengan *field-field*-nya. Perintah yang digunakan untuk menciptakan database adalah:

```
CREATE DATABASE nama_database;
```

Setelah membuat database pada MySQL akan menampilkan database yang dibuat, dengan menggunakan perintah:

```
SHOW DATABASES;
```

Setelah membuat database, akan dicoba menghapus database yang baru dibuat dengan perintah berikut ini:

```
DROP DATABASE nama_database;
```

Tabel adalah objek utama yang harus dibuat dalam database karena akan melakukan input data dalam tabel tersebut, kalau tidak membuat tabel di dalam database maka tidak akan bisa mengakses databasenya. Tabel dibuat setelah membuat database, di dalam tabel terdapat baris dan kolom. Baris diistilahkan dengan *recordset* dan kolom dengan *field*. Pertama harus menggunakan database yang dibuat, berikut perintahnya:

```
USE nama_database;
```

Setelah database digunakan, maka dapat membuat tabel dengan perintah berikut ini:

```
CREATE TABLE nama_tabel(  
    Field-1 type(length),  
    Field-2 type(length),  
    Filed-2 type(length));
```

Length adalah panjang karakter yang diinginkan untuk setiap field. Setelah membuat tabel, langkah selanjutnya menampilkan tabel yang baru dibuat di MySQL dengan perintah berikut ini:

```
SHOW TABLES;
```

Setelah tabel dibuat, langkah selanjutnya melihat isi struktur tabel tersebut dengan menggunakan perintah berikut ini:

```
DESC nama_tabel;            atau            DESCRIBE nama_tabel;
```

Langkah selanjutnya yaitu menghapus tabel pada database, dengan perintah berikut ini:

```
DROP TABLE nama_tabel;
```





JUNIOR WEB
PROGRAMMER

J.620100.016.01

**Menulis Kode dengan Prinsip
sesuai Guidelines dan Best
Practices**

4

MENULIS KODE DENGAN PRINSIP SESUAI GUIDELINES DAN BEST PRACTICE

Objektif:

1. Menerapkan coding guidelines dan best practices dalam penulisan program (kode sumber).
 2. Menggunakan ukuran performansi dalam menuliskan kode sumber.
-

1. Coding Guide Lines dan Best Practices

Cara menuliskan kode mempengaruhi kemudahan kode untuk dimengerti baik oleh penulis kode tersebut (terutama setelah kode tersebut ditulis pada waktu yang telah lalu (lama)). Meski pada mayoritas bahasa pemrograman tidak ada keharusan (aturan) dalam penulisan kode, sebagai perkecualian adalah Phyton) namun pada prakteknya sebaiknya guideline/best practice penulisan kode diperhatikan.

1.1 Komentar

Komentar berguna untuk menjelaskan kode. Terdapat dua jenis komentar, block comment dan line comment (disebut juga sebagai inline comment). Block comment adalah komentar yang lebih dari satu baris (biasanya diberi tanda awal dan akhir), sementara Line comment adalah komentar satu baris (sering juga di bagian kanan suatu instruksi).

```
/*  
  
Ini isi komentar blok .  
Variasi 1  
  
*/
```

```
/******\  
* *  
* Ini isi komentar blok . Variasi2 *  
* *  
*****/
```

Block comment adalah komentar yang lebih dari satu baris (biasanya diberi tanda awal dan akhir), sementara Line comment adalah komentar satu baris (sering juga di bagian kanan suatu instruksi). Kegunaan komentar:

- a. Sebagai pseudocode (algoritma) pada tahap awal pembuatan kode
- b. Sebagai deskripsi untuk menunjukkan tujuan dari pembuat kode
- c. Untuk menggabungkan informasi tambahan seperti logo, diagram, flowchart dan lainnya.
- d. Meta data. Komentar blok bisa diberikan di awal modul/subroutine/class yang menyatakan secara singkat deskripsi kode tersebut termasuk informasi parameter masukan/keluaran, library yang dipakai, dan lainnya.
- e. Debugging. Komentar untuk membantu dalam proses debugging dengan menjadikan Sebagian kode sebagai komentar agar tidak dieksekusi.

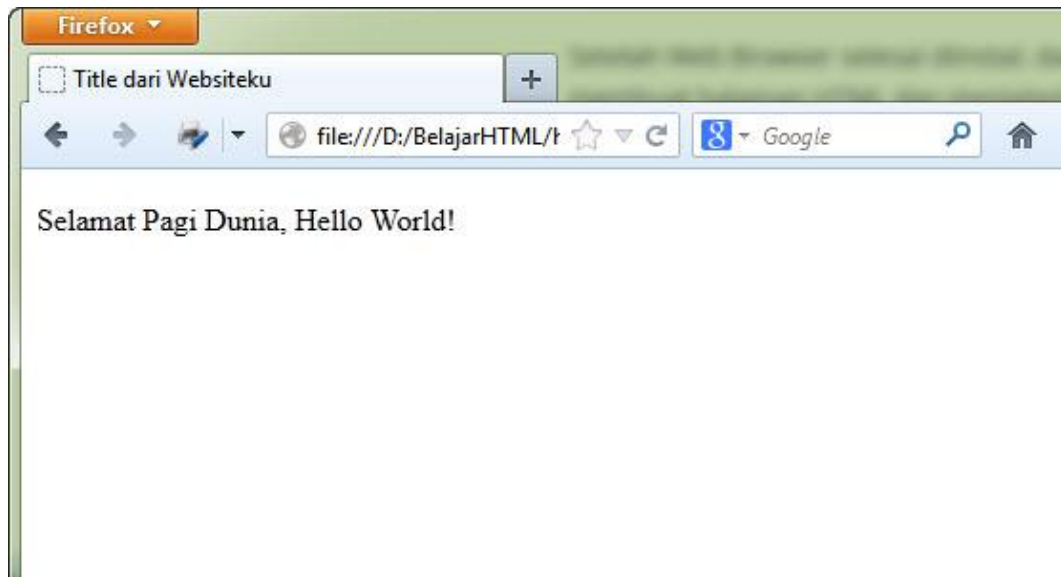
1.2 Struktur HTML

Setiap halaman HTML setidaknya memiliki struktur dasar yang terdiri dari: Tag DTD atau DOCTYPE, tag html, tag head, dan tag body.

Contoh:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>Title dari Websiteku</title>
5  </head>
6  <body>
7    <p>Selamat Pagi Dunia, Hello World!</p>
8  </body>
9  </html>
```

Save sebagai *halaman.html* dan jalankan file dengan cara double klik file tersebut, atau **klik kanan → Open With → (Browser)**.



Gambar 1. Tampilan HTML pada Browser

1.2.1 Tag pada HTML

Sebagai sebuah bahasa markup, HTML membutuhkan cara untuk memberitahu web browser untuk apa fungsi sebuah text. Apakah text itu ditulis sebagai sebuah paragraf, list, atau sebagai link. Dalam HTML, tanda ini dikenal dengan istilah tag.

Hampir semua tag di dalam HTML ditulis secara berpasangan, yakni tag pembuka dan tag penutup, dimana objek yang dikenai perintah tag berada di antara tag pembuka dan tag penutup. Objek disini dapat berupa text, gambar, maupun video. Penulisan tag berada di antara dua kurung siku: "<" dan ">". Berikut adalah format dasar penulisan tag HTML:

```
<tag_pembuka>objek yang dikenai perintah tag</tag_penutup>
```

Beberapa tag tersebut adalah:

1. Tag <html>

Merupakan tag pembuka dari keseluruhan halaman web. Semua kode HTML harus berada di dalam tag ini. Tag html dimulai dengan <html> dan diakhiri dengan </html>

2. Tag <head>

Elemen pada tag **<head>** umumnya akan berisi berbagai definisi halaman, seperti kode **CSS**, **JavaScript**, dan kode-kode lainnya yang tidak_tampil di browser. Tag **<title>** dalam contoh kita sebelumnya digunakan untuk menampilkan title dari sebuah halaman web. Title ini biasanya ditampilkan pada bagian paling atas web browser.

3. Tag <body>

Tag <body> berisi semua elemen yang akan tampil dalam halaman web, seperti paragraf, tabel, link, gambar, dll. Tag body ini ditutup dengan </body>.

4. Tag <p>

HTML menyediakan tag khusus untuk membuat paragraf. Penulisannya menggunakan huruf p, sebagai berikut: <p>.

5. Tag

Digunakan untuk memisahkan baris dalam paragraph.

6. Tag dan

Tag digunakan untuk menghasilkan garis miring, dan untuk menebalkan huruf.

7. Tag heading

HTML menyediakan tag khusus untuk membuat judul atau di dalam HTML lebih di kenal dengan istilah: heading. *Heading* dirancang terpisah dari paragraf. Tag Heading biasanya digunakan untuk judul dari paragraf, atau bagian dari text yang merupakan judul. Tag heading di dalam HTML terdiri dari 6 tingkatan, yaitu <h1>, <h2>, <h3>, <h4>, <h5>, dan <h6>. Tag heading secara *default* akan ditampilkan oleh web browser dengan huruf tebal (bold). Tampilan dari tag header juga dibuat bertingkat. Tag header <h1> memiliki ukuran huruf paling besar, sedangkan <h6> terkecil.

8. Tag , , dan

Dalam HTML, tag list terdiri dari 2 jenis, ordered list (berurutan) dan unordered list (tidak berurutan). Ordered list akan ditampilkan dengan angka atau huruf, sedangkan unordered list dengan bulatan atau kotak. Ordered list menggunakan tag , dan unordered list menggunakan tag , sedangkan untuk list sendiri menggunakan tag . Untuk membuat unordered list, tinggal ganti tag menjadi . Penggunaan tag list pada HTML tidak hanya untuk membuat daftar saja. Dengan CSS, tag list dapat digunakan untuk membuat menu navigasi di dalam halaman web, seperti menu *home*, *contact us*, dll. Tag list juga dapat digunakan untuk *nested list*, atau *list bersarang*, yang artinya sebuah list yang berada di dalam list lainnya.

9. Tag Image

Tag Image digunakan untuk menampilkan gambar kedalam halaman web, menggunakan . Terdapat atribut *src* dalam tag yang merupakan atribut yang berisi alamat dari gambar yang akan ditampilkan. Alamat ini bisa relatif atau absolute. Atribut lainnya membolehkan kita untuk menentukan besar dari gambar yang ditampilkan, yaitu *width* dan *height*. Untuk mempertahankan proporsi gambar, namun tetap membuat gambar menjadi besar/kecil, cantumkan hanya salah satu atribut saja (*width* saja atau *height* saja, namun tidak keduanya). Misalkan jika kita menetapkan atribut *width=300px* (tanpa mencantumkan *height*), maka web browser akan menampilkan gambar dengan *lebar 300px*, dan menghitung secara otomatis tinggi gambar agar gambar tetap proporsional.

10. Tag Tabel

Dalam menampilkan data yang terstruktur atau tampilan dari *database*, kita biasanya akan membuatnya dalam bentuk tabel. HTML juga menyediakan Tabel tag digunakan untuk menampilkan data dalam bentuk tabel. Untuk membuat tabel di HTML, kita membutuhkan setidaknya 3 tag, yaitu tag `<table>`, tag `<tr>`, dan tag `<td>`:

- Tag `<table>` digunakan untuk memulai tabel
- Tag `<tr>` adalah singkatan dari *table row*, digunakan untuk membuat baris dari tabel.
- Tag `<td>` adalah singkatan dari *table data*, digunakan untuk menginput data ke tabel. Dalam pembuatan tabel terdapat atribut `border` yang digunakan untuk memberikan nilai garis tepi dari tabel. Nilai ini dalam ukuran **pixel**. **`border="1"`**, berarti kita menginstruksikan kepada web browser bahwa tabel tersebut akan memiliki garis tepi sebesar 1 pixel. Jika tidak ditambahkan, secara default tabel tidak memiliki garis tepi.

11. Tag input

Bentuk-bentuk dari keluarga tag input ini dibedakan berdasarkan atribut `type`:

- `<input type="text" />` atau bisa juga `<input />` adalah *textbox* inputan biasa yang menerima input berupa text, contohnya digunakan untuk inputan *nama*, *username*, dan inputan yang berupa text pendek. Input type text ini juga bisa memiliki atribut `value` yang bisa diisi nilai tampilan awal dari text.
- `<input type="password" />` dalam tampilannya sama dengan type text, namun teks yang diinput tidak akan terlihat, akan berupa bintang atau bulatan. Biasanya hanya digunakan untuk inputan yang sensitif seperti *password*.

- `<input type="checkbox" />` adalah inputan berupa *checkbox* yang dapat diceklist atau di centang oleh user. User dapat memilih atau tidak memilih checkbox ini. Type checkbox memiliki atribut `checked` yang jika ditulis atau diisi dengan nilai `checked`, akan membuat checkbox langsung terpilih pada saat pertama kali halaman ditampilkan.
- `<input type="radio" />` mirip dengan checkbox, namun user hanya bisa memilih satu diantara pilihan group radio. Type radio ini berada dalam suatu grup dan user hanya bisa memilih salah satunya.
- `<input type="submit" />` akan menampilkan tombol untuk memproses form. Biasanya diletakkan pada baris terakhir dari form. Atribut `value` jika diisi akan membuat text tombol submit berubah sesuai inputan nilai `value`.

12. Tag `<textarea>`

Tag `textarea` pada dasarnya sama dengan `input type="text"`, namun lebih besar dan dapat berisi banyak baris. Panjang dan banyak baris untuk text area diatur melalui atribut `rows` dan `cols`, atau melalui CSS. Elemen yang berada diantara tag `textarea` akan ditampilkan sebagai text awal dari form.

13. Tag `<select>`

Tag `select` digunakan untuk inputan yang telah tersedia nilainya, dan user hanya dapat memilih dari nilai yang ada. Tag `select` digunakan bersama-sama dengan tag `option` untuk membuat box pilihan. Ketika form dikirim untuk diproses, nilai dari tag `<option>` akan dikirimkan. Nilai ini adalah berupa text diantara tag `option`, kecuali jika kita memberikan atribut `value`. Jika atribut `value` berisi nilai, maka nilai `value`-lah yang akan dikirim. Ada atau tidaknya atribut `value` ini tidak

akan tampak dalam tampilan form. Tag select memiliki atribut selected yang dapat ditambahkan agar tag select berisi nilai awal.

Contoh penggunaan tag pada HTML:

```
<!DOCTYPE html>
<html>
<head>
  <title>Belajar Membuat Form </title>
</head>
<body>
<form action=" formulir.html" method="get">
```

```
Nama: <input type="text" name="nama" value="Nama Kamu" />
<br />
```

```
Password: <input type="password" name="password" />
<br />
```

Jenis Kelamin :

```
<input type="radio" name="jenis_kelamin" value="laki-laki" checked />
Laki - Laki
<input type="radio" name="jenis_kelamin" value="perempuan" />
Perempuan
<br />
```

```
Hobi: <input type="checkbox" name="hobi_baca" /> Membaca Buku
      <input type="checkbox" name="hobi_nulis" checked /> Menulis
      <input type="checkbox" name="hobi_mancing" /> Memancing
<br />
```

Asal Kota:

```
<select name="asal_kota" >
  <option value="Kota Jakarta"> Jakarta</option>
  <option value="Kota Bandung">Bandung</option>
  <option value="Kota Semarang" selected>Semarang</option>
</select>
<br />
```

Komentar Anda:

```
<textarea name="komentar" rows="5" cols="20">
Silahkan katakan isi hati anda
</textarea>
<br />
<input type="submit" value="Mulai Proses!" >
</form>
</body>
</html>
```

1.2.2 Membuat komentar di HTML

Umumnya akan sering lupa tentang cara kerja kode yang dibuat sendiri setelah beberapa waktu. Programmer yang baik harus menyisipkan penjelasan pada sebuah komentar di sekitar kode program. tag komentar pada HTML juga berguna untuk membuat Sebagian isi web tidak tampil untuk sementara. Misalnya saat ingin mencoba tampilan kode HTML yang akan dibuat tanpa menghapus kode sebelumnya. Untuk membuat komentar di HTML, menggunakan awalan `<!--` dan penutup `-->`. Contoh:

```
1
2  <!DOCTYPE html >
3  <html>
4  <head>
5    <title>Belajar Tag Komentar (comment)</title>
6  </head>
7  <body>
8    <!-- <p>Ini berada di dalam tag komentar,
9      dan tidak akan tampil di browser</p> -->
10    <p>Ini bukan komentar, dan akan tampil di browser</p>
11  </body>
12  </html>
```



1.3 Aturan Penulisan Skrip PHP

Seperti bahasa pemrograman lain, PHP juga memiliki aturan penulisan tertentu. Berikut aturan penulisan file PHP :

1. Penamaan file PHP

File php harus diakhiri dengan ekstensi `.php`, selain ekstensi tersebut, web server tidak akan menganggapnya sebagai file PHP sehingga tidak akan memprosesnya.

2. Selalu gunakan open tag `<?php` atau `<?=`

3. Tidak perlu menggunakan close tag `?>` pada script full PHP

Penggunaan close tag `?>` hanya jika script PHP yang kita buat bergabung dengan script lain seperti HTML, closing tag tersebut sebagai akhir dari script PHP.

4. Penulisan perintah (statement) pada PHP

Setiap statement ini harus diakhiri dengan tanda semicolon atau ; kecuali perintah yang menggunakan kurung kurawa {}, contoh seperti statement class, if else, do while, dan lainnya.

5. Membedakan huruf besar dan huruf kecil (case sensitive)

Secara umum semua perintah pada PHP tidak membedakan huruf besar dan huruf kecil (case insensitive) KECUALI pada penamaan variabel. Contoh perintah yang tidak membedakan penggunaan huruf besar dan huruf kecil adalah pemanggilan fungsi -baik fungsi bawaan seperti echo maupun fungsi yang kita buat sendiri-, pemanggilan class, perulangan, dan lainnya.

6. Aturan penulisan script spasi, tab, dan enter

Pada eksekusi program, PHP akan mengabaikan spasi, tab dan enter. Hal ini akan memudahkan kita untuk menulis script dengan baik, karena dapat memecah statemen menjadi beberapa baris. Ketika menulis script PHP, usahakan selalu memformatnya dengan baik agar mudah dibaca dan di pahami, baik oleh diri kita sendiri maupun oleh orang lain.

7. Penulisan komentar pada PHP

Seperti bahasa pemrograman lain, pada PHP, kita juga dapat menulis komentar. Komentar ini penting karena selain dapat memudahkan kita mengingat script yang telah kita buat, juga dapat memudahkan programmer lain memahami program yang kita buat. Pada PHP, kita dapat menulis komentar dengan menggunakan tanda // untuk satu baris komentar, dan /* ... */ untuk komentar yang terdiri dari satu baris atau lebih.

1.4 Penulisan Skrip PHP

Pada dasarnya penulisan skrip PHP hampir sama seperti penulisan skrip HTML, skrip PHP ditambahkan ke dalam HTML dengan menggunakan delimiter khusus. Delimiter merupakan karakter atau kumpulan karakter yang membedakan antara skrip atau tag dengan teks biasa dalam HTML. Seperti kita ketahui, delimiter untuk tag HTML adalah karakter < dan >, dan semua skrip tersebut dinamakan dengan *.php. Untuk PHP, delimiter yang digunakan adalah sebagai berikut:

1. Untuk dokumen SGML/HTML biasa:

```
<? Skrip PHP ?> atau <?php skrip php ?>
```

2. Untuk dokumen XML:

```
<?php skrip PHP ?>
```

3. Untuk editor yang tidak mendukung PHP:

```
<script language="php">  
skrip php  
</script>
```

4. Dapat juga menggunakan delimiter ASP:

```
<% skrip PHP %>
```

Baris-baris pada skrip PHP dipisahkan dengan cara yang sama dengan C atau Perl, yaitu dengan menambahkan karakter titik koma (;). Contoh:

```
<?  
Skrip php;  
Skrip php;  
?>
```

Jika terdapat skrip yang hanya terdiri dari satu baris, ada dua gaya penulisan yang dapat digunakan:

```
<?  
Skrip php;  
?>
```

Atau:

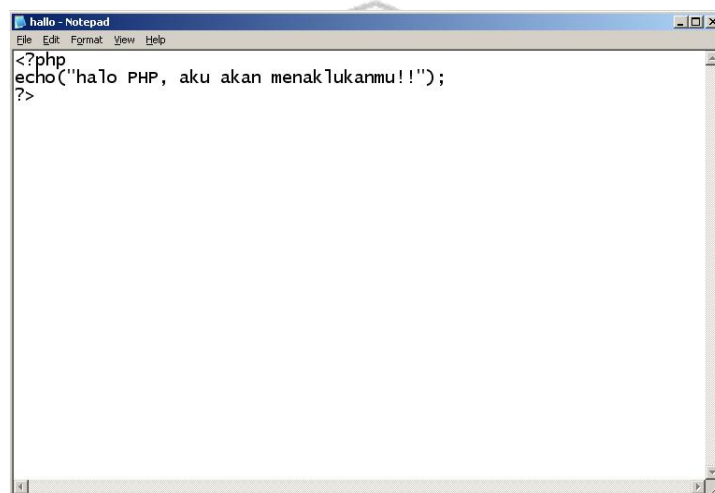
```
<? Skrip php ?>
```

Jika digunakan gaya penulisan yang kedua, tanda titik koma tidak perlu digunakan, karena tag penutup `?>` sudah menandakan akhir dari skrip tersebut.

1.5 Membuat Program PHP

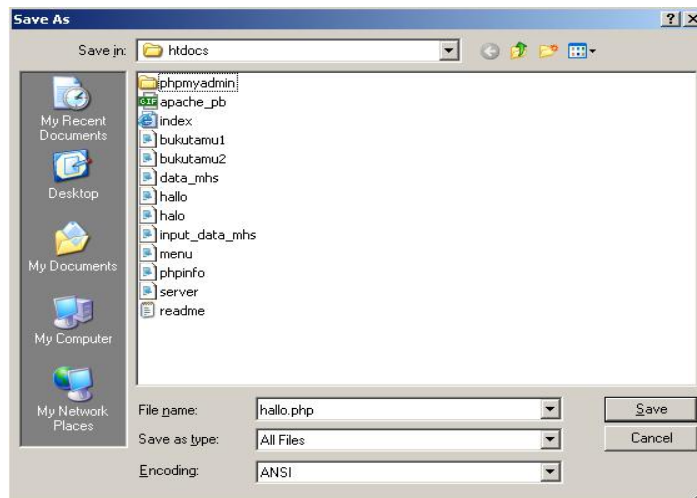
Membuat program PHP cukup sediakan saja sebuah program editor untuk menuliskan programnya, seperti Notepad (Windows) atau vi editor (Linux). Tahapannya adalah sebagai berikut:

4. Buka program **Notepad**, lalu tulislah skrip PHP seperti pada gambar.



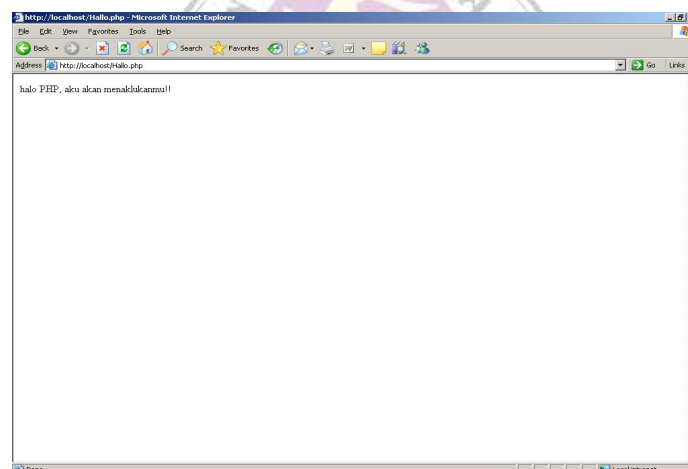
Gambar 1. Menulis Skrip PHP di Notepad

5. Kemudian, klik menu **File > Save As**, maka akan tampil kotak dialognya, pilih pada bagian **Save in**: htdoc (C:\Apache\htdocs), lalu isikan pada bagian **Save As type** : All files dan **File name** : Hallo.php. Lihat gambar 2.



Gambar 2. Kotak Dialog Save As

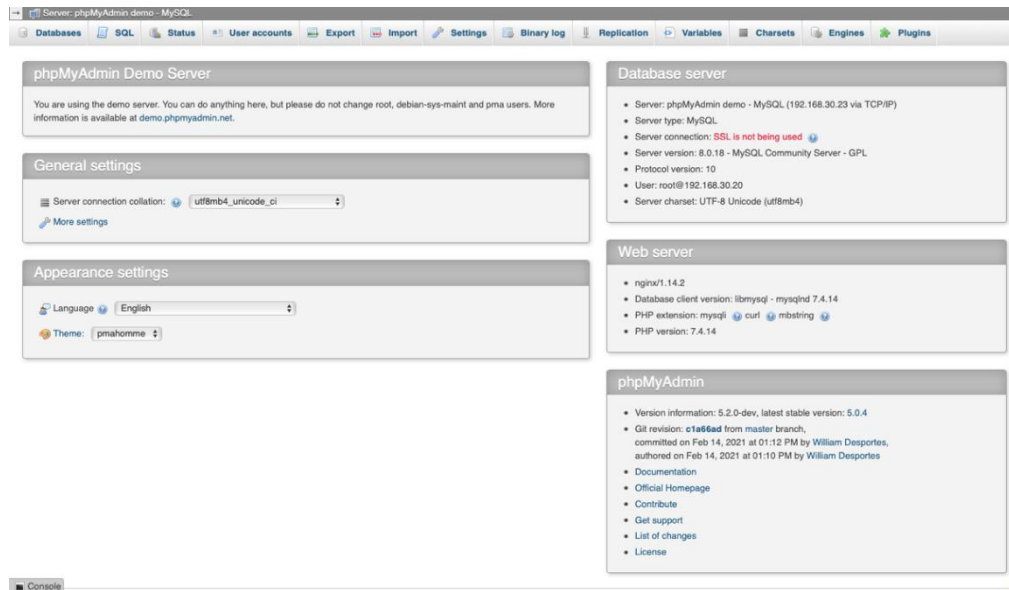
6. Hidupkan server **Apache**, lalu buka **Internet Explorer** dan isikan pada bagian **address** : <http://localhost/Hallo.php>, maka akan tampil hasilnya. Lihat gambar 3.



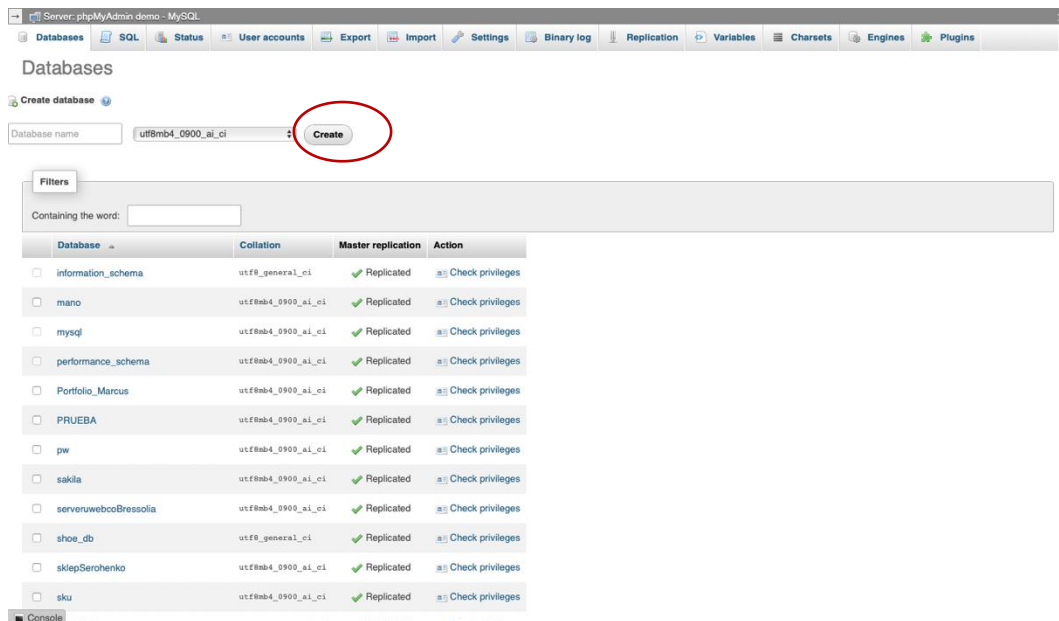
Gambar 3. Menjalankan Skrip PHP di Browser

1.6 Membuat Database MySQL

Untuk membuat database di MySQL dapat menggunakan MySQL under shell atau PHPMyAdmin. Modul ini menggunakan PHPMyAdmin sebagai tool administrasi MySQL. Untuk penggunaannya adalah pada browser ketik <http://localhost/phpmyadmin/>.

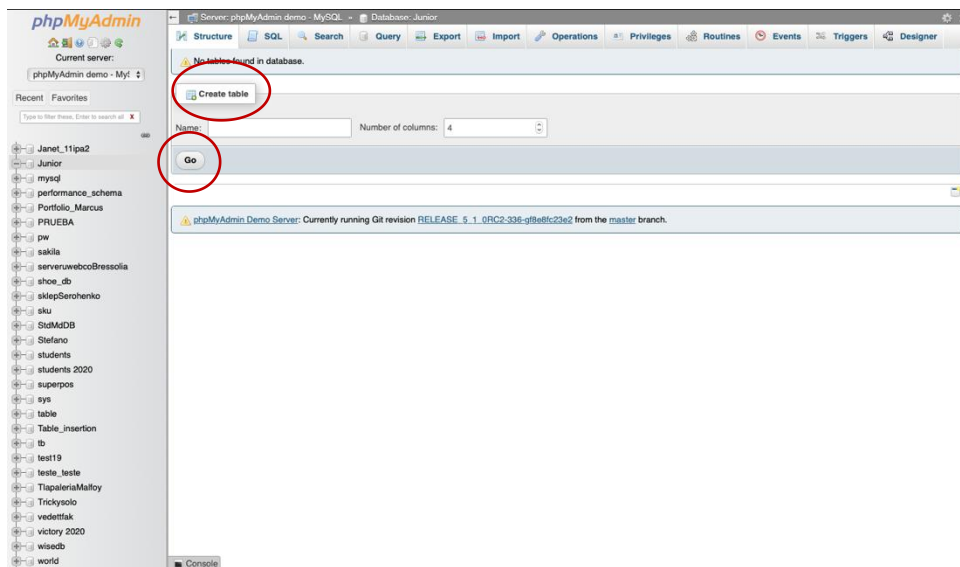


Gambar 10. PHPMyAdmin pada Browser



Gambar 11. Membuat Database

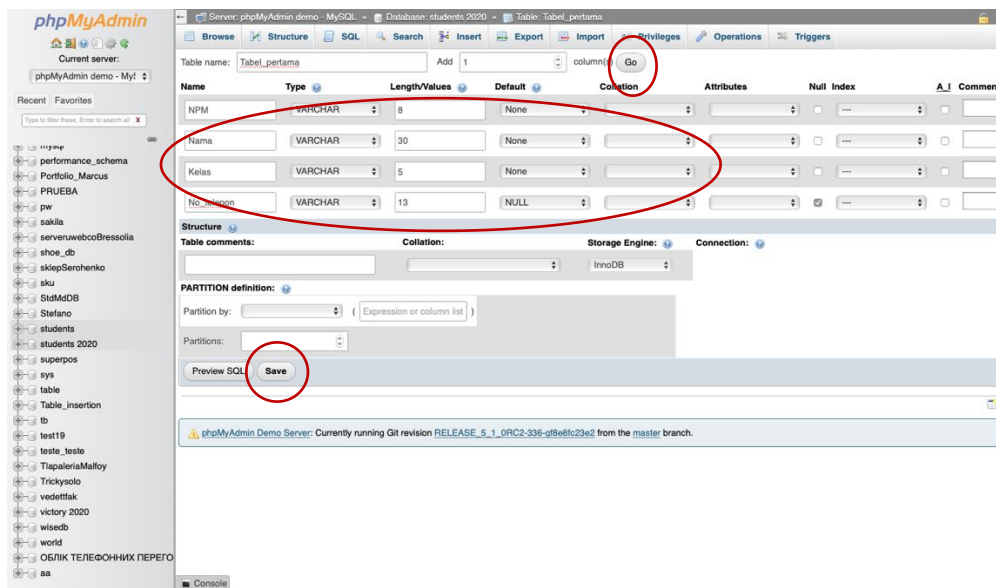
Dalam membuat database, pertama kali yang dilakukan adalah memberi nama database, contohnya: students 2020. Ketik PHP lalu tekan create. Lalu akan muncul gambar berikut.



Gambar 12. Tampilan Database yang Sudah Dibuat

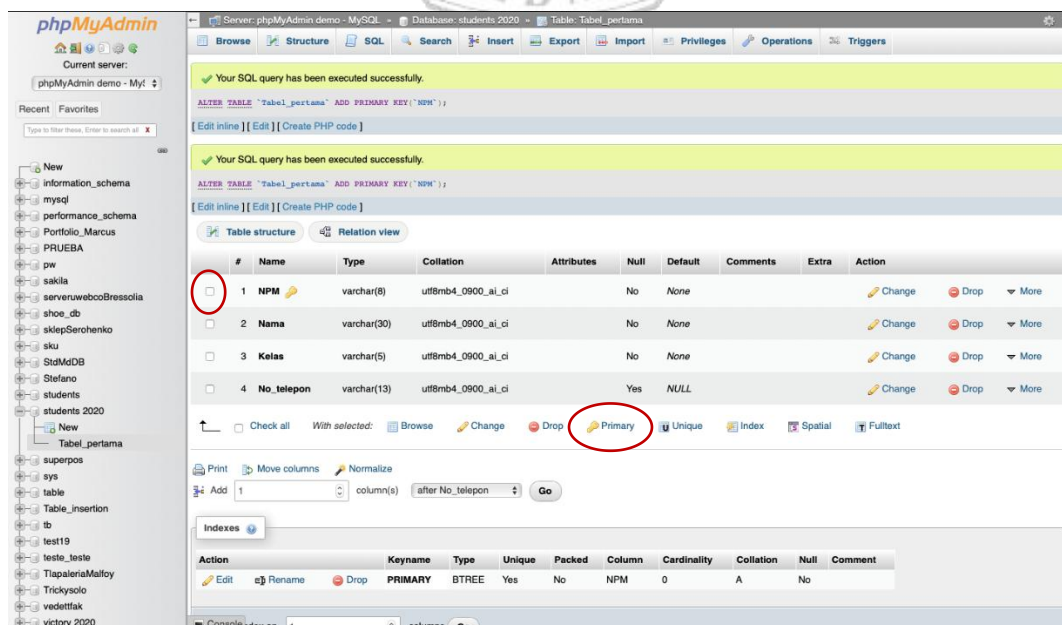
Database yang telah berhasil dibuat akan tampil pada kolom sebelah kiri. Berikutnya adalah membuat tabel, berikan nama tabel dan jumlah kolom pada tabel, lalu tekan GO.

Setelah tabel berhasil dibuat, akan muncul tampilan seperti pada gambar 13. Kemudian kita dapat memberikan nama field/atribut tabel pada kolom Name, menentukan tipe data pada kolom Type, menentukan panjang nilai pada kolom Length/Value, dan dapat mengatur nilai awal pada kolom Default. Untuk menambahkan kolom baru, dapat mengisi pada kolom Add column, atur jumlah kolom tambahan yang diinginkan lalu tekan GO. Jika tabel sudah sesuai, Langkah berikutnya adalah menekan tombol Save.



Gambar 13. Tampilan Membuat Tabel

Jika tabel sudah berhasil dibuat maka akan muncul tampilan seperti pada gambar 14. Kemudian atur salah satu field/atribut sebagai primary key dengan memberikan ceklist, pada box yang terdapat pada bagian kiri nama field/atribut kemudian tekan tombol primary. Maka akan muncul icon key (kunci) pada atribut/field yang kita set sebagai primary key.



Gambar 14. Tampilan Tabel

1.7 Membuat Database MySQL


Setelah database dan tabel sudah dibuat, dapat melakukan koneksi PHP dengan MySQL menggunakan skrip:

```
<?php
//buat koneksi MySQL untuk user: root, tanpa password, alamat: localhost
$link=mysql_connect('localhost','root','');

//cek apakah koneksi dengan MySQL berhasil
if ($link)
{
    //koneksi berhasil
    echo "Koneksi dengan MySQL berhasil";
}
else
{
    //koneksi gagal
    echo "Koneksi dengan MySQL gagal";
}

//memeriksa nilai dari $link
echo "<br />";
echo 'hasil var_dump variabel $link : ';
var_dump($link);

?>
```



1.8. Penanganan Kesalahan (galat/error)

Default error handling di PHP sangat sederhana. Pesan kesalahan dengan nama file, nomor baris dan pesan yang menjelaskan kesalahan tersebut dikirim ke browser.

1.8.1 Dasar Penanganan Kesalahan: Menggunakan die() function

Contoh error yang muncul, jika tidak dapat menemukan file yang diinginkan:

```
Warning : fopen(welcome.txt) [function.fopen]: failed to open stream:  
No such file or directory in C:\webfolder\test.php on line 2
```

Untuk mencegah pengguna dari mendapatkan pesan kesalahan seperti di atas, kita menguji apakah file tersebut ada sebelum kita mencoba untuk mengaksesnya menggunakan:

```
<?php  
if(!file_exists("welcome.txt")) {  
    die("File not found");  
} else {  
    $file=fopen("welcome.txt","r");  
}  
?>
```

1.8.2 Membuat Kesalahan Handle Kustom

Membuat handler kesalahan kustom cukup sederhana. Kami hanya membuat fungsi khusus yang dapat dipanggil ketika terjadi kesalahan di PHP. Fungsi ini harus mampu menangani minimal dua parameter (tingkat kesalahan dan pesan kesalahan) tapi dapat menerima hingga lima parameter (optionally: file, line-number, and the error context):

Sintaksis

```
error_function(error_level,error_message,  
error_file,error_line,error_context)
```

Parameter	Deskripsi
error_level	Wajib. Menentukan tingkat laporan kesalahan untuk kesalahan yang ditetapkan pengguna. Harus angka nilai. Lihat tabel di bawah untuk kemungkinan tingkat laporan kesalahan
error_message	Wajib. Menentukan pesan kesalahan untuk kesalahan yang ditetapkan pengguna
error_file	Pilihan. Menentukan nama file di mana kesalahan terjadi

error_line	Pilihan. Menentukan nomor baris di mana kesalahan terjadi
error_context	Pilihan. Menentukan array yang berisi setiap variabel, dan nilai-nilai mereka, digunakan ketika terjadi kesalahan

1.6.3 Kesalahan Tingkat Report

Tingkat laporan kesalahan ini adalah jenis yang berbeda dari kesalahan kesalahan handler yang ditetapkan pengguna dapat digunakan untuk:

Nilai	Konstan	Deskripsi
2	E_WARNING	Non-fatal kesalahan run-time. Pelaksanaan script tidak dihentikan
8	E_NOTICE	pemberitahuan run-time. Script menemukan sesuatu yang mungkin kesalahan, tetapi juga bisa terjadi saat menjalankan script biasanya
256	E_USER_ERROR	Fatal user-generated error. Ini seperti E_ERROR ditetapkan oleh programmer menggunakan fungsi PHP trigger_error()
512	E_USER_WARNING	Non-fatal peringatan user-generated. Ini seperti E_WARNING ditetapkan oleh programmer menggunakan fungsi PHP trigger_error()
1024	E_USER_NOTICE	Pengguna dihasilkan pemberitahuan. Ini seperti E_NOTICE ditetapkan oleh programmer menggunakan fungsi PHP trigger_error()
4096	E_RECOVERABLE_ERROR	kesalahan fatal catchable. Ini seperti E_ERROR tapi bisa ditangkap oleh pegangan ditetapkan pengguna (see also set_error_handler())
8191	E_ALL	Semua kesalahan dan peringatan (E_STRICT became a part of E_ALL in PHP 5.4)

2. Ukuran Performansi dalam Menuliskan Kode Sumber

Setiap program atau kode ketika dijalankan akan menggunakan sumber daya komputasi, seperti memori (space) dan waktu eksekusi. Kode yang baik selain bekerja sesuai dengan spesifikasinya juga harus efisien dalam menggunakan sumber daya komputasi tersebut.

Resources meliputi penggunaan memori dan lama eksekusi. Efisiensi dalam kode sumber terkait dengan efisiensi langkah dan proses (kecepatan) dan efisiensi penggunaan memori. Setiap kode memiliki kompleksitas yang menggambarkan bagaimana resources yang diperlukan.

2.1 Manajemen Variabel

Setiap mendefinisikan sebuah variabel maka kita memerintahkan komputer untuk menyediakan ruang dimemory yang nantinya akan diisi dengan nilai. Gunakan variable untuk menyimpan nilai yang akan dipanggil lagi nantinya dan usahakan tidak ada variabel yang tidak terpakai karna akan membuang - buang memory.

Nilai dari variabel dapat di isi dengan informasi yang diinginkan dan dapat diubah nilainya pada saat kode program sedang berjalan. Sebuah variabel memiliki *nama* yang digunakan untuk mengakses nilai dari variabel itu. Variabel dalam PHP digunakan untuk menampung nilai inputan dari user, atau nilai yang kita definisikan sendiri. Namun PHP memiliki beberapa aturan tentang cara penggunaan dan penulisan variabel. Aturan penulisan variable dalam PHP:

- Variabel di dalam PHP harus diawali dengan dollar sign atau tanda dollar (\$).
- Setelah tanda \$, sebuah variabel PHP harus diikuti dengan karakter pertama berupa huruf atau *underscore* (_), kemudian untuk karakter kedua dan seterusnya bisa menggunakan huruf, angka atau

underscore (_). Dengan aturan tersebut, variabel di dalam PHP *tidak bisa* diawali dengan angka.

- Minimal panjang variabel adalah 1 karakter setelah tanda \$.

```
1  <?php
2  $i;
3  $nama;
4  $Umur;
5  $_lokasi_memori;
6  $ANGKA_MAKSIMUM;
7  ?>
```

- Variable dalam PHP bersifat case sensitive. PHP membedakan variabel yang ditulis dengan huruf besar dan kecil (bersifat case sensitif), sehingga \$belajar tidak sama dengan \$Belajar dan \$BELAJAR, ketiganya akan dianggap sebagai variabel yang berbeda.

2.2 Cara Memberikan Nilai kepada variable

Sama seperti sebagian besar bahasa pemrograman lainnya, untuk memberikan nilai kepada sebuah variabel, PHP menggunakan tanda sama dengan (=). Operator '*sama dengan*' ini dikenal dengan istilah *Assignment Operators*. Perintah pemberian nilai kepada sebuah variabel disebut dengan *assignment*. Jika *variabel* tersebut belum pernah digunakan, dan langsung diberikan nilai awal, maka disebut juga dengan proses *inisialisasi*. Variabel tidak perlu mendeklarasikan terlebih dahulu. Anda bebas membuat variabel baru di tengah-tengah kode program, dan langsung menggunakannya tanpa dideklarasikan terlebih dahulu. Dalam kelompok bahasa pemrograman, PHP termasuk **Loosely Type Language**, yaitu jenis bahasa pemrograman yang variabelnya tidak terikat pada sebuah tipe tertentu. Di dalam PHP, setiap variabel bebas diisi dengan nilai apa saja.

```
1  <?php
2  $a = 17; // nilai variabel a berisi angka (integer)
3  $a = "aku"; // nilai variabel a diubah menjadi kata (string)
4  $a = 17.42; // nilai variabel a diubah menjadi desimal (float)
5  ?>
```

2.3 Simpan Hasil Perhitungan ke dalam Variabel

Ketika membuat sebuah program atau sistem pasti didalam sistem tersebut ada proses aritmatika setidaknya untuk sistem yang sedikit kompleks. Ketika menghitung suatu operasi aritmatika maka simpanlah hasil operasi tersebut kedalam variabel dengan catatan jika operasi tersebut dilakukan lebih dari sekali. Tujuannya adalah untuk menghindari CPU time yang besar karena setiap operasi aritmatika diprogram kita akan dihandle oleh CPU maka akan sangat efisien ketika hasil dari aritmatika tersebut disimpan didalam variabel maka CPU tidak akan mengulang proses yang sama dan akan mengambilnya dimemory. Ini juga berlaku ketika kalian memanggil fungsi dengan return value.

2.4 Minimalisir perulangan bertingkat

Perulangan bertingkat akan memakan waktu eksekusi jauh lebih lama dibandingkan dengan perulangan yang tidak bertingkat.

