# Elemen Dasar PHP

- karakter
- pengenal
- tipe data
- variabel
- konstanta
- operator

1. Karakter: berupa huruf, sebuah angka tunggal, sebuah spasi, tanda kontrol seperti carriage return (\r), atau simbol (+,?,& ...)
2. Pengenal (identifier): digunakan untuk memberi nama variabel, fungsi, atau kelas. Aturan penamaan pengenal sebagai berikut.
   - karakter yang dapat digunakan adalah huruf, angka, atau garis bawah (_)
   - karakter pertama harus berupa huruf atau garis bawah
   - panjang pengenal bisa berapa saja.
   - huruf kecil dan besar dibedakan, kecuali fungsi-fungsi yang tersedia pada PHP (seperti print, Print, PRINT dianggap sama oleh PHP)

   Contoh benar:
   ```
   nama
   nama_pemakai
   _nama
   I
   kuartal13
   NamaBarang
   ```

   Contoh salah:
   ```
   3bulan -> diawali angka
   nama barang -> ada spasi
   nama-barang -> ada tanda minus
   ```

3. Tipe data
   ada 3 macam:
   - integer: bilangan bulat antara -2 milyar hingga +2 milyar
   - double: tipe data bilangan real atau titik mengambang, bilangan yang mempunyai bagian pecahan
   - string: tipe data teks (sederetan karakter yang tidak menyatakan bilangan seperti nama barang atau nama orang, dsb)

4. Konstanta (literal): menyatakan nilai yang tetap di dalam program
   Contoh:
   print("Halo!");
   Halo adalah konstanta string

5. Variabel: untuk menyimpan nilai yang berubah-ubah, ditulis dengan awalan $.
   Contoh:
   ```
   $gaji = 2000000;
   printf("Gaji pertama = %d <br> \n", $gaji);
   $gaji = 1.5*$gaji;
   printf("Gaji sekarang = %d <br>\n", $gaji);
   ```
   Berbeda dengan bahasa C dan Pascal, PHP tidak memerlukan pendeklarasian variabel, jadi bisa diubah kapan saja sewaktu-waktu.

6. Konversi Nilai: digunakan untuk konversi dari suatu tipe data ke tipe data yang lain.
   Contoh:
   ```
   settype($suhu, "double");
   ```
   -> membuat isi variabel suhu diubah menjadi bertipe double.

   Contoh untuk diuji di browser:
   ```
   <?php
   $suhu = "30 derajat celcius";
   print("Tipe data string : $suhu <br>\n");

   settype($suhu, "double");
   print("Tipe data double: $suhu <br>\n");

   settype($suhu, "integer");
   print("Tipe data integer: $suhu <br>\n");

   settype($suhu, "string");
   print("Tipe data string: $suhu <br>\n");
   ?>
   ```

7. Operator: simbol yang digunakan untuk melakukan operasi, seperti penjumlahan, perkalian, pembandingan kesamaan dua buah nilai, atau memberi nilai ke variabel. Nilai yang dioperasikan operator (disebut operand atau argumen) bersama-sama operator membentuk ekspresi (ungkapan).
   Contoh:
   2+3*4
   disebut ekspresi, karena tanda + dan * disebut operator, sedangkan 2,3,4 adalah operand atau argumen.

**Operator Aritmetika**

Operator yang digunakan untuk operasi matematika. Daftar operator aritmetika:

| Operator | Fungsi | Prioritas |
|---|---|---|
| + | Penjumlahan | Ketiga |
| - | Pengurangan | Ketiga |
| * | Perkalian | Kedua |
| / | Pembagian | Kedua |
| % | Sisa Pembagian | Kedua |
| ++ | Penaikan | Pertama |
| -- | Penurunan | Pertama |

Contoh:

```
<?php
print("Penjulahan dan pengurangan: <br>\n");
printf("6+1 = %d <br>\n", 6+1);
printf("6-1 = %d <br>\n", 6-1);
print("<br><br>");

print("Perkalian: <br>\n");
printf("6*3 = %d <br>\n", 6 * 3);
printf("6*3.14 = %d <br>\n", 6*3.14);
print("<br><br>");

print("Pembagian: <br>\n");
print("6/4 ="); print(6/4); print("<br>");
print("6/3.14 ="); print(6/3.14); print("<br>");
print("<br><br>");

print("Sisa pembagian: <br>\n");
print("6%5 ="); print(6%5); print("<br>");
print("6%4 ="); print(6%4); print("<br>");
print("6%3 ="); print(6%3); print("<br>");
print("<br><br>");

print("Penaikan ++ : <br>\n");
$x = 1;
print("x = $x <br>\n");
$x++;
print("x++ = $x <br>\n");
print("<br><br>");

print("Penurunan -- : <br>\n");
$x = 10;
print("x = $x <br>\n");
$x--;
```

```
print("x-- = $x <br>\n");
?>
```

Operator aritmetika juga berlaku pada tipe data string

**Prioritas Operator**
Contoh:
2+3*4 = ? (20 atau 14?)
Lihat tabel operator aritmetika dan buktikan.

**Operator penugasan (=)**
Digunakan untuk memberikan nilai pada suatu variabel.
Contoh:

```
<?php
$x = 100;
print("Variabel x adalah = $x <br>\n");

$x += 2;
print("Variabel  x adalah = $x <br>\n");

$x -= 2;
print("Variabel x adalah = $x <br>\n");

$x /= 2;
print("Variabel x adalah = $x <br>\n");

$x %= 40;
print("Variabel x adalah = $x <br>\n");

$x &= 2;
print("Variabel x adalah = $x <br>\n");

$x |= 2;
print("Variabel x adalah = $x <br>\n");

$x ^= 2;
print("Variabel x adalah = $x <br>\n");

$x = "Seratus";
$x .= " Lima";
print("Variabel x adalah = $x <br>\n");
?>
```

**Operator pembandingan**
Operator pembandingan (relasional) digunakan untuk pembandingan dua buah operand dan menghasilkan nilai benar (true) (1) atau salah (false) (0).
Daftar operator pembandingan

| Operator | Makna |
|---|---|
| '== | sama dengan |
| < | kurang dari |
| > | lebih kurang |
| <= | kurang dari atau sama dengan |
| >= | lebih dari atau sama dengan |
| != | tidak sama dengan |
| <> | tidak sama dengan |

Contoh:

```
<?php
$a = 1;
$b = 2;
$c = 1;
$me1 = "Saya";
$me2 = "SAYA";

printf("$a > $b => %d <br>\n", $a > $b);
printf("$a < $b => %d <br>\n", $a < $b);
printf("$a == $c => %d <br>\n", $a == $c);
printf("$a != $b => %d <br>\n", $a != $b);
printf("$a <> $c => %d <br>\n", $a <> $c);
printf("$me1 <> $me2 => %d <br>\n", $me1 <> $me2);
printf("$me1 == $me2 => %d <br>\n", $me1 == $me2);
?>
```

**Operator logika**

Digunakan untuk menggabungkan kondisi berganda dan menghasilkan sebuah ekpresi yang bernilai benar (nilai 1) atau salah (nilai 0). Operator logika antara lain:

- and atau && => menghasilkan benar bila kedua operand benar.
- or atau || => menghasilkan benar bila ada salah satu operand benar.
- xor => menghasilkan benar jika hanya salah satu di antara operand bernilai benar.
- ! => contoh: !$x => menghasilkan nilai salah bila $x bernilai benar (kebalikan nilai aslinya)

| Operand1 | Operand2 | and (&&) | or ( \|\| ) | xor |
|---|---|---|---|---|
| benar | benar | true | true | false |
| benar | salah | false | true | true |
| salah | benar | false | true | true |
| salah | salah | false | false | false |

8. Variabel variabel

   Memungkinkan nama variabel ditentukan dan digunakan secara dinamis.
   Contoh:
   $kota = "Balikpapan";
   $$kota = 1000000;
   Pernyataan di atas identik dengan:
   $Balikpapan = 1000000;
   Penulisan $$kota dapat ditulis menjadi: ${$kota} (lebih baik dipakai dalam perintah print)

# Pernyataan Kontrol

***- if***
Bentuknya:

```
if(ekspresi)
      pernyataan;
```

Bagian pernyataan dijalankan hanya bila bagian ekspresi bernilai benar.

Contoh:

```
<?php
$total = 200000;
$keterangan = "Tak ada diskon";

if ($total >= 100000)
      $keterangan = "Dapat Diskon";

print("$keterangan <br>\n");
?>
```

***- if else***
Bentuknya:

```
if(ekspresi)
      pernyataan_1
else
      pernyataan_2
```

Bagian pernyataan_1 dijalankan bila ekspresi bernilai benar, pernyataan_2 dijalankan bila ekspresi bernilai salah.

Contoh:

```
<html>
<head>
<title>Menentukan Hari ini</title>
</head>
<body>
<form method=GET action="lat8.php">
Masukkan nama hari (Inggris):
<input type="text" name="hari"><br>
<input type="submit" value="Submit"><br>
</form>
</body>
</html>

<?php
```

```
$sekarang = date(l);

if($_GET[hari]==$sekarang)
      print("Benar, hari ini hari ".$_GET[hari]);
else
      print("Salah, hari ini bukan ".$_GET[hari]);

?>
```

***- if elseif***

Bentuknya:

```
if(ekspresi)
      pernyataan_1
elseif
      pernyataan_2
else
      pernyataan_3
```

Digunakan untuk pengambilan keputusan yang melibatkan banyak alternatif.

Contoh:

```
<html>
<head>
<title>Menentukan Hari ini</title>
</head>
<body>
<form method=GET action="lat9.php">
Masukkan nama hari (Inggris):
<input type="text" name="hari"><br>
<input type="submit" value="Submit"><br>
</form>
</body>
</html>

<?php
if($_GET[hari]=="Sunday")
      echo "Minggu";
else if ($_GET[hari]=="Monday")
      echo "Senin";
else if ($_GET[hari]=="Tuesday")
      echo "Selasa";
else if ($_GET[hari]=="Wednesday")
      echo "Rabu";
else if ($_GET[hari]=="Thursday")
      echo "Kamis";
else if ($_GET[hari]=="Friday")
      echo "Jumat";
```

```
else if ($_GET[hari]=="Saturday")
      echo "Sabtu";
else
      echo "";

?>
```

***- switch***

Mirip ***if elseif***, ***switch*** juga menyelesaikan persoalan yang melibatkan banyak alternatif. Bentuknya sbb:

```
switch (ekspresi)
{
      case ekspresi_case_1 :
            pernyataan_1;
      break;
      case ekspresi_case_2 :
            pernyataan_2;
      break;
      case ekspresi_case_3 :
            pernyataan_3;
      ...
      default:
            pernyataan_n;
}
```

**Contoh:**

```
<html>
<head>
<title>Menentukan Hari</title>
</head>
<body>
<form method=GET action="lat10.php">
Masukkan nama hari (Inggris):
<input type="text" name="hari"><br>
<input type="submit" value="Submit"><br>
</form>
</body>
</html>
```

```
<?php
switch($_GET[hari])
{
      case "Sunday" :
            echo "Minggu";
      break;
      case "Monday" :
            echo "Senin";
      break;
      case "Tuesday" :
            echo "Selasa" ;
      break;
      case "Wednesday" :
            echo "Rabu" ;
      break;
      case "Thursday" :
            echo "Kamis" ;
      break;
      case "Friday" :
            echo "Jumat" ;
      break;
      case "Saturday" :
            echo "Sabtu" ;
      break;
      default :
            echo "Masukkan nama hari yang benar.";
}
?>
```

Bagian default bisa tidak dimasukkan.

**- Operator ? :**

Dikenal sebagai operator tertiary, disebabkan operator ini melibatkan tiga buah operand. Operand ini dapat digunakan untuk melakukan pengambilan keputusan tetapi dalam bentuk ekspresi dalam bentuk ekspresi. Bentuk penggunaannya:

```
ekspresi_berkondisi ? nilai_1 : nilai_2
```

ekspresi di atas memberikan hasil sesuai dengan `nilai_1` jika ekspresi di depan tanda `?` bernilai benar. Apabila `ekspresi_berkondisi` bernilai salah, maka hasil ekspresi berupa `nilai_2`.

**- while**
pernyataan untuk pengulangan.
bentuknya:

```
while (ekspresi)
{
      pernyataan_pernyataan
}
```

pernyataan `while` akan memeriksa nilai ekspresi terlebih dahulu. Jika bernilai benar maka pernyataan-pernyataan yang terdapat dalam {} akan dijalankan dan ekspresi dievaluasi lagi. Proses ini diulang terus menerus sampai ekspresi bernilai salah.

Contoh:

```
<?php
$bilangan = 1;

while ($bilangan <= 25)
      {
      print($bilangan."<br>\n");
      $bilangan++;
      }
?>
```

**- do-while**
serupa dengan pernyataan while.
Bentuknya:

```
do
      {
      pernyataan_pernyataan
      } while (ekspresi);
```

pengulangan akan berakhir jika ekspresi (yang diuji sesudah pernyataan-pernyataan dijalankan) bernilai salah.

**Contoh:**

```
<?php
$bilangan = 1;

do
      {
      print($bilangan. "<br>\n");
      $bilangan++;
      } while ($bilangan < 26);
?>
```

**- for**

pernyataan yang biasa digunakan untuk menangani pengulangan proses.
bentuknya:

```
for (ekspresi_1; ekspresi_2; ekspresi_3)
      {
      pernyataan_pernyataan
      }
```

apabila yang terletak antara tanda { dan } hanya berupa sebuah pernyataan, maka tanda { dan } boleh tidak ditulis.

bentuk pernyataan **for** identik dengan pernyataan berikut:

```
ekspresi_1;
while (ekspresi_2)
      {
      pernyataan_pernyataan
      ekspresi_3;
      }
```

Jadi:
- ekspresi_1 = ekspresi untuk memberi nilai awal terhadap variabel yang akan digunakan untuk melakukan pencacahan pengulangan.
- ekspresi_2 berlaku sebagai kondisi untuk menentukan pengulangan terhadap pernyataan yanga da di dalam {} akan dilakukan atau tidak.
- ekspresi_3 digunakan untuk mengatur nilai variabel yang digunakan dalam ekspresi_1.

Contoh:

```
<?php
      for ($bilangan =1; $bilangan <=25; $bilangan++)
      print($bilangan. "<br>\n");
?>
```

contoh pernyataan di atas sama dengan contoh **while** di bawah ini:

```
<?php
$bil = 1 ;
while ($bil < 20)
      {
      print($bil ."<br>\n");
      $bil++;
      }
?>
```

**- break**
digunakan untuk keluar dari suatu kalang (proses yang berulang).

Contoh:
```
<?php
 for ($i = 1; $i <= 25; $i++)
      {
      print ($i ."<br>\n");
            if ($i == 10)
            break;
      }
      print ("Selesai <br>\n");
?>
```

hasilnya, bilangan 11 s.d 25 tidak tampak, karena **break** mengakhiri pernyataan **for**.

Pemberian angka di belakang break terjadi di dalam pernyataan **switch**, yang dimaksudkan untuk keluar dari kalang (proses yang berulang).

Contoh:
```
<?php
for ($i = 1; $i <=25; $i++)
 {
      switch ($i)
      {
       case 5 :
            print("Nomor lima tanda break 1.<br>");
       break 1;
       case 10 :
            print("Nomor sepuluh kena tanda break 2.<br>");
       break 2;
       default:
            print($i ."<br>\n");
       break;
      }
 }
print("Selesai <br>\n");
?>
```

# Penanganan Berkas

Konsep:
Sebuah berkas mirip sebuah buku. Bila kita membaca isi buku, hal pertama yang kita lakukan adalah membuka buku. Begitu juga berkas, sebelum melakukan operasi pada berkas (operasi seperti pembacaan, pemanipulasian, perekaman data), maka hal pertama adalah membuka berkas lebih dulu, setelah itu baru operasi berkas, kalau sudah selesai penutupan berkas.

Setiap berkas memiliki penunjuk (mirip kursor pada Word). Bila kita membaca/merekam berkas, maka data yang ditunjuk oleh penunjuk inilah yang akan dibaca/direkam.

Fungsi-fungsi operasi berkas pada PHP

1. Untuk membuka berkas: `fopen(nama_berkas, mode)`

2. Untuk menutup berkas: `fclose(pegangan)`
   argumen pegangan berkas diperoleh saat memanggil fopen.

3. Untuk merekam data ke berkas: `fputs(pegangan, data)`

4. Untuk membaca data berkas; `fgets(pegangan, panjang)`
   argumen pegangan berkas diperoleh ketika memanggil fopen, argumen panjang menyatakan jumlah karakter yang akan dibaca.

5. Untuk membaca sebuah karakter pada berkas: `fgetc(pegangan)`

6. Untuk memeriksa apakah penunjuk berkas sedang menunjuk akhir berkas atau tidak: `feof(pegangan)`

7. Untuk memindahkan penunjuk berkas ke posisi dalam berkas:
   `fseek(pegangan, ofset [,acuan])`

8. Untuk meletakkan penunjukan berkas ke awal berkas: `rewind(pegangan)`

9. Untuk memperoleh posisi penunjuk berkas: `ftell(pegangan)`

10. Untuk menampilkan seluruh data dimulai dari posisi penunjuk berkas hingga akhir berkas: `fpassthru(pegangan)`

Mode pembukaan berkas

| Mode | Keterangan |
|---|---|
| r | Berkas hanya bisa dibaca. Penunjuk berkas akan diletakkan di awal berkas. |
| r+ | Berkas dibuka dengan mode baca dan tulis (artinya, bisa merekam data atau membaca data). Penunjuk berkas akan diletakkan di awal berkas. |
| w | Mode perekaman data. Berkas akan diciptakan kalau belum ada. Kalau sudah ada, isinya akan dihapus. |
| w+ | Berkas dibuka dengan mode baca dan tulis (artinya, bisa merekam data atau membaca data). Berkas akan diciptakan kalau belum ada. Kalau sudah ada, isinya akan dihapus. |
| a | Mode untuk penambahan data. Pada saat berkas dibuka, penunjuk berkas diletakkan pada akhir berkas. Bila berkas belum ada, berkas akan diciptakan. |
| a+ | Berkas dibuka dengan mode baca dan tulis (artinya, bisa merekam data atau membaca data). Penunjuk berkas akan diletakkan di akhir berkas. Bila berkas belum ada, berkas akan diciptakan. |

Contoh:
Aplikasi Pengisian Buku Tamu Sederhana

```
<html>
<head>
<title> Buku Tamu</title>
</head>
<body>
<form action="simpan.php" method=GET>
Nama: <input type=text name=nama><br><p>
Email: <input type=text name=email><br><p>
Komentar: <textarea rows=4 cols=40
name=komentar></textarea><br><p>
<input type=submit value="Submit"><p>
</form>
</body>
</html>
```

**simpan.php**

```
<?php

if(empty($_GET[nama]) || empty($_GET[email]) ||
empty($_GET[komentar]))
      {
      print("Kolom nama, email, dan komentar harus diisi");
      exit;
      }

//proses penyimpanan ke dalam berkas

$pegangan = fopen("data/bukutamu.txt", a);
fputs($pegangan, $_GET[nama] . "\n");
fputs($pegangan, $_GET[email] . "\n");
fputs($pegangan, $_GET[komentar] . "\n");

fclose($pegangan);

print("Terima kasih, komentar Anda sudah disimpan.<br> Anda
bisa komentar lagi deh, tekan tombol BACK browser.");
?>
```

**bacabukutm.php**

```
<?php
$no_data = 1; // untuk nomor urut;
$pegangan = fopen("data/bukutamu.txt", r);

while (!feof($pegangan))
      {
            $nama = trim(fgets($pegangan, 255));
            if($nama == false)
            break;
            $email = trim(fgets($pegangan, 255));
            $komentar = trim(fgets($pegangan, 255));

      // tampilkan;

      echo "<br>Data ke-".$no_data." : <br>\n";
      printf ("Nama : %s <br>\n", $nama);
      printf ("Email : %s <br>\n", $email);
      printf ("Komentar %s <br>\n", $komentar);

      $no_data++; // nomor urut naik bertambah-incremental;
      }

?>
```

##########################################################################
**Tugas**:
Selain penggunaan file teks sebagai tempat penyimpanan data buku tamu di atas, berikan contoh lain pemanfaatan file teks. Tulis kode programnya pada lembar jawaban, dikumpulkan minggu depan, Senin, 17/05, pada pertemuan berikutnya (masing-masing mahasiswa tidak boleh sama kode programnya).
##########################################################################

Bila menggunakan file teks (*flat file*) untuk menyimpan data, maka:

1. data mudah dibaca oleh manusia dan mudah dimanipulasi atau bahkan dihapus secara manual menggunakan teks editor.
2. data yang disimpan sangat terbatas, kadang dipengaruhi kemampuan file-system dalam mengelola kapasitas file, seperti FAT-32 (Windows) yang hanya mampu menyimpan file sebesar 4G.
3. informasi yang terdapat dalam file teks dapat digali secara sederhana dengan utilitas teks, di Unix/Linux seperti dengan perintah `cat`, `head`, `tail`, dan `grep`.
4. membutuhkan program (yang terdiri atas kode-kode yang panjang) dan rumit untuk mem-parse (memecah) tiap baris atau paragraf file teks ke dalam struktur data variabel sebelum melakukan pengolahan terhadap variabel-variabel tersebut.

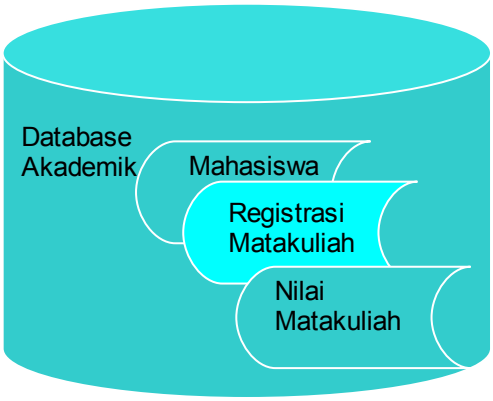
Untuk itu diperlukan perangkat lunak database. Menyimpan data di database memiliki banyak keuntungan yaitu kemudahan dan kecepatan. Dengan database relasional, tidak perlu menulis program berpuluh atau ratusan ribu baris hanya sekedar untuk query atau reporting, tapi cukup dengan menulis sintaks/perintah/bahasa SQL yang cukup pendek, yang akan dibahas berikutnya.

# Pengenalan Database

*Database* (basis data) adalah kumpulan data yang saling terkait yang disusun agar mudah untuk diakses.

Manfaat *database*:

1. Mudah untuk memperoleh informasi tertentu.
2. Dalam aplikasi, mampu untuk mendapatkan jawaban pertanyaan-pertanyaan seperti:
    - Berapa jumlah mahasiswa yang mengikuti kuliah “Pengantar Basis Data”?
    - Siapa saja yang lulus pada periode Agustus tahun ini?
    - Berapa persentase mahasiswa yang tidak melakukan registrasi pada semester lalu?
    - Berapa jumlah SKS yang diperoleh oleh mahasiswa dengan NIM 12345?




Berbagai contoh aplikasi basis data:

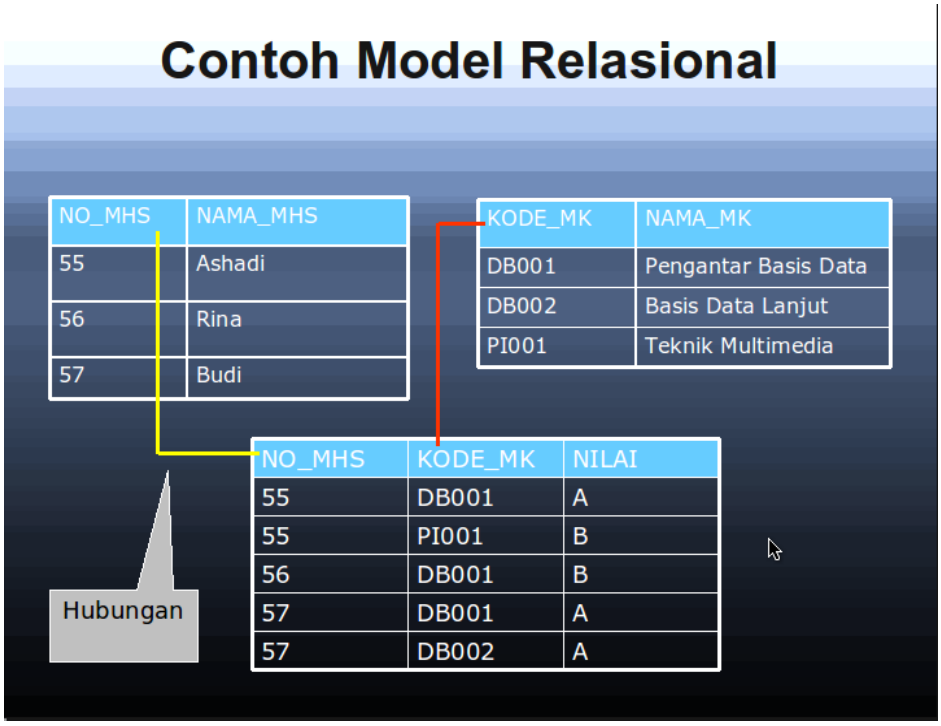
1. ATM (*Automatic Teller Machine*)
2. e-Mobile Banking dan Internet Banking
3. Tracking paket barang
4. Sistem Informasi Akademik
5. Pembelajaran jarak jauh (*Distance Learning*)
6. Reservasi tiket pesawat dan kereta api
7. Sistem perpustakaan
8. *Social Network* (Facebook, Twitter, Plurk, dsb)

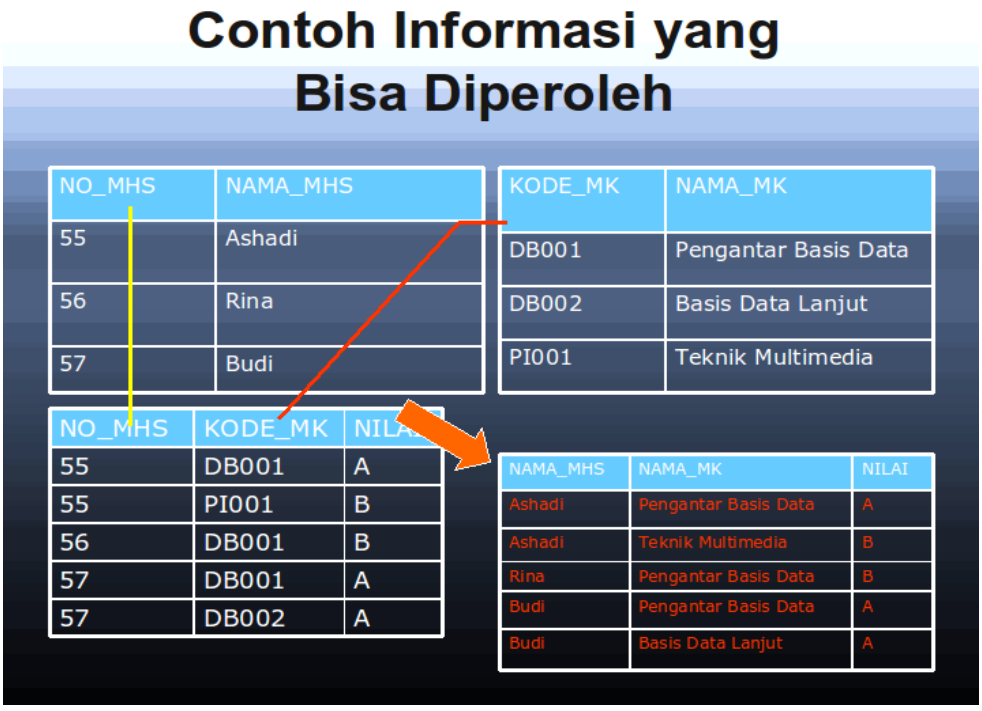
Keuntungan menggunakan database:

1. Independensi program-data
2. Meminimalkan redundansi data
3. Meningkatkan konsitensi data
4. Meningkatkan kemampuan berbagi data
5. Meningkatkan produktivitas pengembangan aplikasi
6. Meningkatkan pencapaian standarisasi
7. Meningkatkan kuantitas dan kualitas data
8. Meningkatkan tanggapan dan kemudahan akses terhadap data
9. Mengurangi pemeliharaan program

Salah satu model database yang terkenal  adalah model relasional, dimana data disusun dalam bentuk **tabel-tabel**, dan antar dua tabel bisa punya **hubungan**.

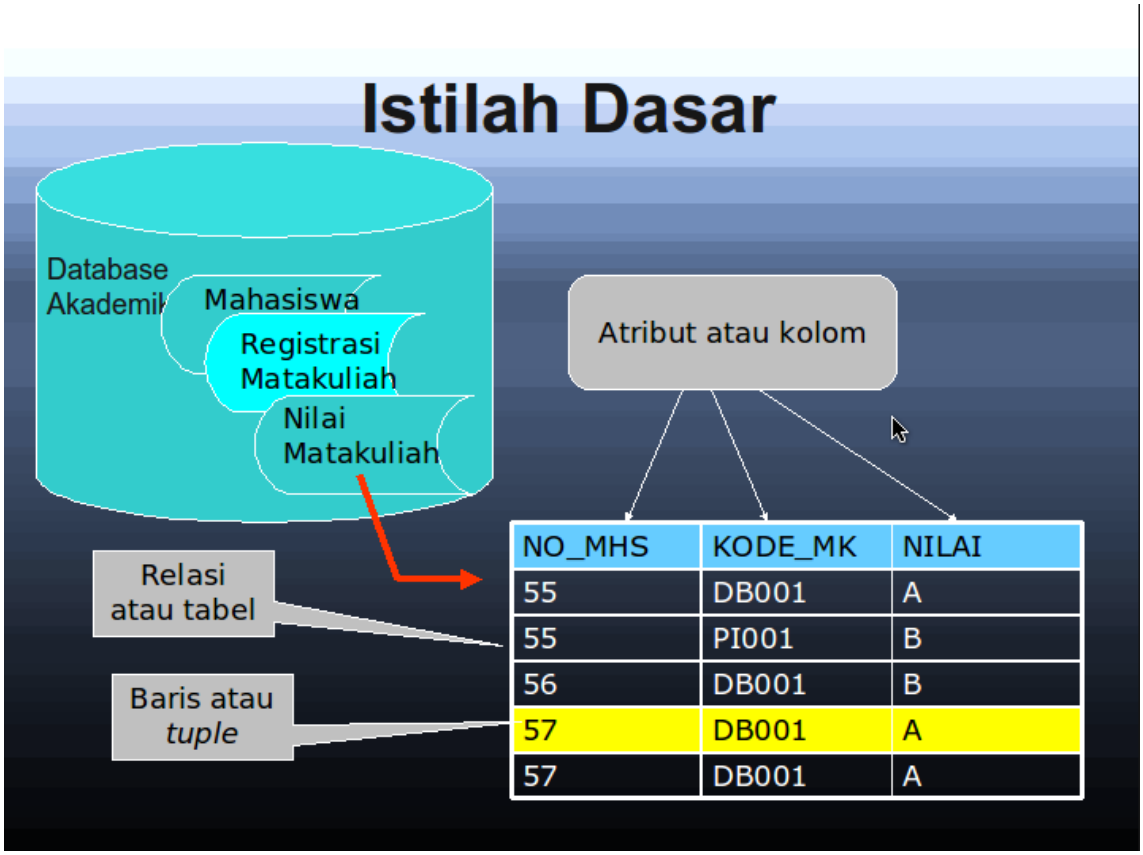
Contoh:


Contoh Model Relasional
NO_MHS
NAMA_MHS
55
Ashadi
56
Rina
57
Budi
KODE_MK
NAMA_MK
DB001
Pengantar Basis Data
DB002
Basis Data Lanjut
PI001
Teknik Multimedia
NO_MHS
KODE_MK
NILAI
55
DB001
A
55
PI001
B
56
DB001
B
57
DB001
A
57
DB002
A
Hubungan


Contoh Informasi yang Bisa Diperoleh
NO_MHS
NAMA_MHS
55
Ashadi
56
Rina
57
Budi
KODE_MK
NAMA_MK
DB001
Pengantar Basis Data
DB002
Basis Data Lanjut
PI001
Teknik Multimedia
NO_MHS
KODE_MK
NILAI
55
DB001
A
55
PI001
B
56
DB001
B
57
DB001
A
57
DB002
A
NAMA_MHS
NAMA_MK
NILAI
Ashadi
Pengantar Basis Data
A
Ashadi
Teknik Multimedia
B
Rina
Pengantar Basis Data
B
Budi
Pengantar Basis Data
A
Budi
Basis Data Lanjut
A

Karakteristik dalam relasional

1. tidak ada baris yang kembar (tidak ada redundansi)
2. urutan baris tidak penting
3. setiap atribut memiliki nama yang unik
4. letak atribut bebas
5. setiap atribut memiliki nilai yang tunggal dan jenisnya sama untuk semua baris



# *SQL (Structured Query Language)*

SQL (baca: si-kwel, awalnya adalah SEQUEL) merupakan salah satu cara untuk berinteraksi dengan database relasional, dibuat pertama kali oleh IBM sebagai bahasanya database relasional. Konsep di SQL secara langsung merupakan terjemahan lansung akan apa yang ada pada model relasional. Misalnya, tabel dan view di SQL adalah relasi, tupel adalah row (baris), deretan atribut adalah kolom, ada juga dikenal dengan sebutan domain, relational integrity, key, dan sebagainya.

Bahasa SQL cenderung lebih mendekati bahasa Inggris dan tidak terlalu banyak mengandung simbol atau operator yang sangat sulit diingat. Bahasa SQL adalah bahasa yang bersifat deklaratif, tidak prosedural. Sebuah query SQL yang kompleks pun dinyatakan dalam sebuah statetment saja.

Bahasa SQL tidak bersifat case-sensitive (membedakan karakter huruf besar dan kecil). Nama-nama tabel dan kolom juga umumnya tidak bersifat case-sensitive untuk lingkungan tertentu seperti di Windows, sedangkan di Unix/Linux sensitive. Meski demikian, agar memudahkan untuk digunakan di segala *platform*/lingkungan sebaiknya gunakan case-sensitive.

Bahasa SQL (Sintaks SQL) terdiri atas statement atau kalimat atau perintah. Statement adalah unit dasar dalam bahasa SQL. Antara satu statement dengan yang lain dipisahkan tanda titik koma (;). Sebuah statetement dapat ditulis dalam beberapa baris -- dipotong-potong dengan *newline* -- tanpa mengubah artinya, dengan tujuan memudahkan pembacaan saat menulis query panjang.

Statement SQL digolongkan menjadi 3 kelompok: DML, DDL, dan DCL

1. DML (*data manipulation language*), adalah perintah/kalimat SQL untuk query database (SELECT), memasukkan data (baris) (INSERT), memperbarui (UPDATE), atau menghapus data (DELETE), dan mengosongkan sebuah tabel (TRUNCATE).
2. DDL (*data definition language*), adalah perintah yang berkaitan dengan pembuatan tabel (CREATE TABLE), mengubah tabel (ALTER TABLE), dsb.
3. DCL (*data control language*), berhubungan dengan pengaturan akses ke data, seperti GRANT (untuk memberi user akses terhadap sejumlah perintah) dan REVOKE (untuk mencabut akses yang sebelumnya di-GRANT).

Selain 3 kelompok tersebut, ada juga perintah COMMIT dan ROLLBACK yang berhubungan dengan transaksi, DECLARE/FETCH/MOVE/CLOSE yang berhubungan dengan cursor, dan lain sebagainya.

##########################################################################

Soal Latihan (mengingat kembali materi basis data):

1. Buatlah perintah SQL untuk membuat database baru dengan nama: Sekolah.
2. Buatlah perintah SQL untuk membuat tabel baru dengan nama: SISWA yang terdiri atas 4 kolom, yaitu NIM, NAMA, JK, JURUSAN.
3. Buatlah perintah SQL untuk mengisi 3 baris tabel SISWA.

##########################################################################

# *Database* dalam PHP

Vendor perangkat lunak *database* yang didukung oleh PHP antara lain:

1. dBase
2. DB++
3. FrontBase
4. filePro
5. Firebird/InterBase
6. Informix
7. IBM DB2 — IBM DB2, Cloudscape and Apache Derby
8. Ingres — Ingres DBMS, EDBC, and Enterprise Access Gateways
9. MaxDB
10. Mongo
11. mSQL
12. Mssql — Microsoft SQL Server
13. MySQL
14. Mysqli — MySQL Improved Extension
15. Mysqlnd — MySQL Native Driver
16. OCI8 — Oracle OCI8
17. Ovrimos
18. SQLParadox — Paradox File Access
19. PostgreSQL
20. SQLite
21. SQLite3
22. Sybase
23. tokyo_tyrant

Seluruh perangkat lunak di atas mendukung bahasa SQL yang digunakan dalam PHP.

# Mengenal MySQL

MySQL adalah database relasional (RDBMS, *Relational Database Management System*) gratis dan open-source (awalnya memiliki lisensi GPL, *General Public License,* namun kini memiliki lisensi dari Oracle) yang mula-mula tersedia di Unix/Linux, namun kini tersedia di sistem operasi lain seperti Windows. MySQL mudah diinstall, dipakai, dan dapat dihubungkan dengan berbagai bahasa permograman, sangat aksesibel.

Kelebihan MySQL pada kecepatannya, baik kecepatan koneksi maupun kecepatan query-query sederhana. Fitur-fitur yang disediakan cukup menbantu dalam pembuatan aplikasi web, seperti klausa LIMIT dalam SELECT, full text index, dan recovery database yang mudah.

Kelemahan MySQL kadang tidak sesuai standar, seperti terlalu mudah menerima masukan data. Belum ada fitur view, trigger, stored procedure (baru ada di MySQL 5), atau check constraint yang sudah diadopsi banyak database lain seperti MSSQL, Oracle, dan PostgreSQL.

**Utilitas MySQL**
Sebelum melangkah ke perintah-perintah SQL, perlu diketahui lebih dulu beberapa utilitas yang akan biasa digunakan dalam mengoperasikan MySQL. Pada pembahasan ini MySQL dijalankan pada sistem operasi Linux. Untuk pengguna Windows bisa menyesuaikan dengan perintah yang sama. (Dengan memahami perintah-perintah dasar ini diharapkan nantinya akan bisa menggunakan perintah-perintah yang ada di antarmuka program seperti MySQLFront, phpmyadmin, MySQL Query Browser dan sebagainya).

**mysql**
mysql merupakan program yang disediakan MySQL untuk berdialog dengan server MySQL. Pada sistem operasi Windows dapat ditemukan pada direktori bin (mysql.exe). Pada Unix/Linux cukup ketik pada terminal dengan perintah:

```
$ mysql [options] [database] [<inpputfile] [>outputfile]
```

Untuk mengetahui [options], ketik `mysql --help`

Untuk masuk ke mysql, ketik:

```
$ mysql -u username -ppassword
```

Setelah perintah di atas dijalankan maka dapat memberikan perintah SQL dan beberapa perintah lain yang berkaitan dengan operasi database.

Contoh:

```
subura@localhost:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 161
Server version: 5.0.75-0ubuntu10.3 (Ubuntu)

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

mysql>
```

untuk keluar ketikkan perintah `exit, quit` atau `\q`

**mysqladmin**

mysqladmin adalah utilitas yang digunakan untuk memenuhi tugas-tugas administrasi MySQL. Sintak perintahnya adalah:

```
mysqladmin [options] command command
```

Contoh melihat versi MySQL:

```
subura@localhost:~$ mysqladmin -V
mysqladmin  Ver 8.41 Distrib 5.0.75, for debian-linux-gnu
on i486
```

Untuk mengetahui options, ketik `mysqladmin --help` atau `mysqladmin -?`

mysqldump

Digunakan untuk men-dump skema informasi dan data dari database, dengan maksud untuk mem-backup database agar dapat diimpor dan digunakan pada server lain. Sintaknya adalah:

```
mysqldump [options] database [tables]
```

**mysqlimport**

Digunakan untukmengambil data dari DBMS lain, atau dari file teks seperti .txt, atau dari file spreadsheet seperti Excell. Sintaknya sebagai berikut:

```
mysqldump [options] database textfile1 [textfile2 ...]
```

Beberapa command mysqladmin:

| **Command** | **Deskripsi** |
|---|---|
| create databasename | Create a new database |
| debug | Instruct server to write debug information to log |
| drop databasename | Delete a database and all its tables |
| extended-status | Gives an extended status message from the server |
| flush-hosts | Flush all cached hosts |
| flush-logs | Flush all logs |
| flush-status | Clear status variables |
| flush-tables | Flush all tables |
| flush-threads | Flush the thread cache |
| flush-privileges | Reload grant tables (same as reload) |
| kill id,id,... | Kill mysql threads |
| password new-password | Change old password to new-password, MySQL 4.1 hashing. |
| old-password new-password | Change old password to new-password in old format. |
| ping | Check if mysqld is alive |
| processlist | Show list of active threads in server |
| reload | Reload grant tables |
| refresh | Flush all tables and close and open logfiles |
| shutdown | Take server down |
| status | Gives a short status message from the server |
| start-slave | Start slave |
| stop-slave | Stop slave |
| variables | Prints variables available |
| version | Get version info from server |

**Perintah SQL pada MySQL**

Membuat database:

```
mysql> CREATE DATABASE kuliah;
```

Contoh:

```
mysql> create database kuliah;
Query OK, 1 row affected (0.00 sec)
```

Catatan: nama database tidak boleh sama dengan kosakata SQL seperti SELECT, INSERT, UPDATE dan semacamnya, untuk menghindari kerancuan.

Untuk melihat database yang sudah ada:

```
mysql> SHOW databases;
```

Contoh:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| db_baak            |
| db_inklusi         |
| db_rizali          |
| db_satria          |
| db_stikom          |
| kuliah             |
| mysql              |
| phpmyadmin         |
| rumahaisya         |
+--------------------+
10 rows in set (0.01 sec)
```

Mengaktifkan/memilih database yang akan digunakan;

```
mysql> USE kuliah;
```

Contoh:

```
mysql> use kuliah;
Database changed
```

Membuat tabel baru pada database `kuliah`:

```
mysql> create table table_name
(
column_1 column_type column_attributes,
column_2 column_type column_attributes,
primary key(column_name),
index index_name(column_name)
);
```

Contoh:

```
mysql> create table siswa (
    -> nim varchar(10),
    -> nama varchar(30),
    -> primary key(nim)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

Melihat tabel yang telah dibuat pada database `kuliah`:

Contoh:

```
mysql> show tables;
+------------------+
| Tables_in_kuliah |
+------------------+
| siswa            |
+------------------+
1 row in set (0.00 sec)
```

Menyisipkan data pada tabel siswa:

```
mysql> INSERT INTO kuliah.siswa (
    -> nim,
    -> nama
    -> )
    -> VALUES (
    -> '1001', 'Budiman'
    -> ), (
    -> '1002', 'Jono'
    -> );
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

Melihat data pada tabel siswa:

```
select * from nama_tabel;
```

Contoh:

```
mysql> select * from kuliah.siswa;
+------+---------+
| nim  | nama    |
+------+---------+
| 1002 | Jono    |
| 1001 | Budiman |
+------+---------+
2 rows in set (0.00 sec)
```

Mengubah data siswa atas nama Jono menjadi Jony:

```
update nama_tabel set
nama_kolom1='nilai1',
nama_kolom2='nilai2',
nama_kolom3='nilai3',
...
where <klausa where>;
```

Contoh:

```
mysql> update kuliah.siswa set
->nama='Jony'
->where nim='1002';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Melihat apakah data sudah berubah:

```
select * from nama_tabel;
```

Contoh:

```
mysql> select * from kuliah.siswa;
+------+---------+
| nim  | nama    |
+------+---------+
| 1002 | Jony    |
| 1001 | Budiman |
+------+---------+
2 rows in set (0.00 sec)
```

Menghapus data 'Budiman':

```
delete from nama_tabel where <klausa where>;
```

Contoh:

```
mysql> delete from kuliah.siswa where nim='1001';
Query OK, 1 row affected (0.00 sec)
```