

Pengenalan Basis Data+Data base

Definisi Basis Data

- Basis Data adalah **kumpulan informasi** yang di simpan di dalam komputer secara sistematis sehingga dapat di periksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Basis Data juga terdiri dari 2 kata yaitu **Basis** dan **Data** dari dua kata tersebut memiliki definisi tersendiri.

Basis

- Basis dapat di artikan sebagai **markas** atau gedung tempat bersarang atau berkumpul

Data

- Data yaitu kumpulan **fakta dunia nyata** yang mewakili suatu objek, seperti manusia, barang, dan lain-lain yang direkam ke dalam bentuk angka, bentuk, teks, bunyi, gambar, atau, juga kombinasinya
-

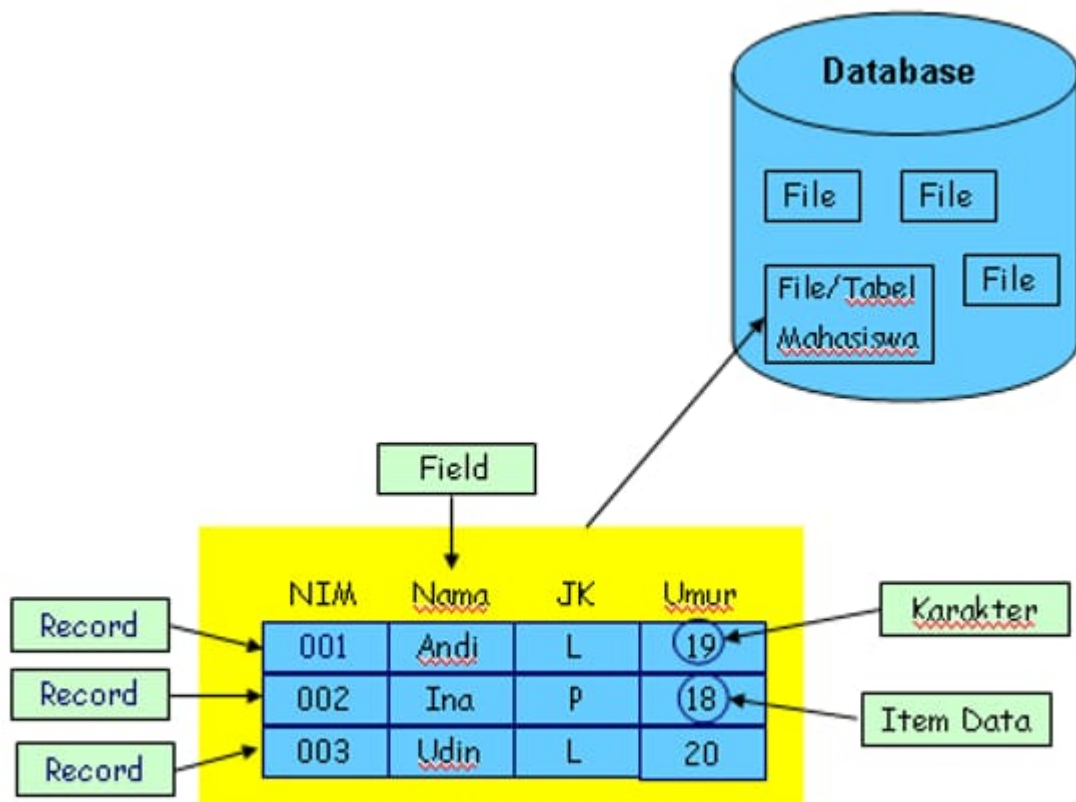
? Peranan Basis Data

Peranan Basis Data di Ponsel

Peranan basis data terdapat pada ponsel genggam yang sering kita gunakan. Pada ponsel genggam, kita biasanya menyimpan nomor-nomor contact relasi kita. Nomor-nomor yang telah kita simpan secara otomatis dapat terurut sesuai abjad. Apabila kita melakukan panggilan masuk atau panggilan keluar semua akan terekam seperti nomor HP yang telah kita hubungi, kapan kita menghubungi nomor tersebut, siapa nama pemilik nomor tersebut, semua akan muncul

Struktur Data Base

Contoh Gambar



Struktur database melibatkan tabel sebagai entitas utama penyimpanan data, dalam konteks tabel mahasiswa, setiap table memiliki **field(kolom)** seperti NIM, nama, jk, umur.

Record adalah **satu baris dalam tabel yang memuat nilai spesifik untuk setiap field.**

Misalnya, record dengan NIM 001, nama "Andi", jk "Laki-laki", dan umur 19.

Dalam konteks penyimpanan fisik, tabel-tabel ini disimpan dalam **tabung** database. Database ini **merupakan wadah** yang dapat berisi satu atau lebih tabel, setiap tabel dalam data base mahasiswa bisa di anggap sebagai file terkait dengan informasi mahasiswa ini **menciptakan struktur** heararki di mana **database adalah wadah utama** tabel adalah file, dan record adalah entitas individual yang menyimpan informasi mahasiswa

Installing *XAMPP*

- Untuk memulai membuat data base kalian harus mendownload aplikasi **XAMPP**
 1. Pergi ke Chrome dan search **download XAMPP**
 2. Pilih website **apachefriends.org**
 3. Kemudian klik download, kemudian tunggu hasil download
 4. Lalu intall aplikasi **XAMPP**
 5. Tekan **next** hingga menginstall **XAMPP**

6. dan klik finis

Databases

Create Database

- membuat data database kita menggunakan query `CREATE_DATABASE [nama_database]` setelah langkah ini *database* akan terbuat, sebagai berikut;

```
CREATE DATABASE nama_database;
```

```
MariaDB [(none)]> CREATE DATABASE bombom_kecil;  
Query OK, 1 row affected (0.014 sec)  
  
MariaDB [(none)]>
```

Show Database

- Untuk menampilkan *data base* kita bisa menggunakan `SHOW_DATABASES;` untuk menampilkan data base yang telah dibuat;

```
SHOW DATABASES;
```

```
MariaDB [(none)]> SHOW DATABASES;  
+-----+  
| Database |  
+-----+  
| bombom_kecil |  
| information_schema |  
| mysql |  
| performance_schema |  
| phpmyadmin |  
| rahmat_rental2 |  
| rental_rahmat |  
| test |  
+-----+  
8 rows in set (0.022 sec)
```

Use Database

- Untuk menggunakan *database* kita menggunakan query `USE [nama_database]`.

```
USE nama_database;
```


```
MariaDB [(none)]> USE bombom_kecil;  
Database changed  
MariaDB [bombom_kecil]>
```

Delete Database

- Lalu untuk menghapus *data base* kita menggunakan query `DROP DATABASE [nama_database]` dan *database* akan terhapus setelah menuliskan query ini.

```
DROP DATABASE nama_database;
```

```
MariaDB [(none)]> DROP DATABASE bombom kecil;  
Query OK, 0 rows affected (0.115 sec)  
  
MariaDB [(none)]> SHOW DATABASES;  
ERROR 1064 (42000): You have an error in your  
  at line 1  
MariaDB [(none)]> SHOW DATABASES;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| phpmyadmin |  
| rahmat_rental2 |  
| rental_rahmat |  
| test |  
+-----+
```



Tipe Data

Angka

- **INT:** Untuk menyimpan nilai bilangan bulat (integer). Misalnya, INT dapat digunakan untuk menyimpan angka seperti 1, 100, -10, dan sebagainya.
- **DECIMAL:** Digunakan untuk menyimpan nilai desimal presisi tinggi, cocok untuk perhitungan finansial atau keuangan.
- **FLOAT dan DOUBLE:** Digunakan untuk menyimpan nilai desimal dengan presisi floating-point. DOUBLE memiliki presisi lebih tinggi dibandingkan FLOAT.
- **TINYINT, SMALLINT, MEDIUMINT, dan BIGINT:** Tipe data ini menyimpan bilangan bulat dengan ukuran yang berbeda-beda.

Contoh :

```
CREATE TABLE contoh_tabel (  
    id INT,  
    harga DECIMAL(10, 2),  
    jumlah_barang TINYINT  
);
```

Dalam contoh tersebut, id menggunakan tipe data INT, harga menggunakan tipe data **DECIMAL** dengan presisi 10 digit dan 2 angka di belakang koma, dan jumlah_barang menggunakan tipe data **TINYINT**.

Teks

- **CHAR(N)** Menyimpan string karakter tetap dengan panjang N. Contoh: **CHAR(10)** akan menyimpan string dengan panjang tepat 10 karakter.
- **VARCHAR(N)**: Menyimpan string karakter dengan panjang variabel maksimal N. Misalnya, **VARCHAR(255)** dapat menyimpan string hingga 255 karakter, tetapi sebenarnya hanya menyimpan panjang yang diperlukan plus beberapa overhead.
- **TEXT**: Digunakan untuk menyimpan teks dengan panjang variabel, tanpa batasan panjang tertentu. Cocok untuk data teks yang panjangnya tidak terduga.
- **ENUM**: Memungkinkan Anda mendefinisikan set nilai yang mungkin dan membatasi kolom hanya dapat mengambil salah satu dari nilai tersebut.
- **SET**: Mirip dengan ENUM, namun dapat menyimpan satu atau lebih nilai dari himpunan yang telah ditentukan.

Contoh :

```
CREATE TABLE contoh_tabel (
    nama CHAR(50),
    alamat VARCHAR(100),
    catatan TEXT,
    status ENUM('Aktif', 'Non-Aktif')
);
```

Tanggal

- **DATE** : Menyimpan nilai tanggal dengan format YYYY-MM-DD.
- **TIME**: Menyimpan nilai waktu dengan format HH:MM:SS.
- **DATETIME**: Menggabungkan nilai tanggal dan waktu dengan format YYYY-MM-DD HH:MM:SS.
- **TIMESTAMP**: Sama seperti DATETIME, tetapi dengan kelebihan diatur secara otomatis saat data dimasukkan atau diubah.

Contoh :

```
CREATE TABLE ContohTabel (
    tanggal DATE,
    waktu TIME,
    datetimekolom DATETIME,
    timestampkolom TIMESTAMP
);
```

Dalam contoh ini, kolom **tanggal** akan menyimpan nilai tanggal, **waktu** menyimpan nilai waktu, **datetimekolom** menyimpan kombinasi tanggal dan waktu, dan **timestampkolom** akan secara otomatis diatur saat data dimasukkan atau diubah.

Boolean

- **BOOL / BOOLEAN / TINYINT(1)**: Digunakan untuk menyimpan nilai boolean, yang dapat mewakili kebenaran atau kesalahan. Representasi nilai benar adalah **1**, sedangkan nilai salah direpresentasikan sebagai 0. Meskipun nilai selain 0 dianggap benar, secara umum, ketiganya seringkali digunakan secara bergantian. Seringkali, ketika Anda mendeklarasikan kolom sebagai **BOOL** atau **BOOLEAN**, MySQL mengonversinya secara otomatis menjadi **TINYINT(1)**, yang juga dapat digunakan untuk menyimpan nilai boolean dengan 0 untuk false dan 1 untuk true.

Contoh :

1. Menggunakan `BOOLEAN`

```
CREATE TABLE contohTabel (  
    title VARCHAR(255),  
    completed BOOLEAN  
);
```

Dalam contoh diatas, kita mendefinisikan kolom `completed` sebagai tipe data `BOOLEAN`. Ini merupakan cara yang sah dan umum digunakan di MySQL. Nilai yang dapat disimpan dalam kolom ini adalah `TRUE` atau `FALSE`, atau dalam representasi angka, 1 atau 0.

2. Menggunakan `BOOL`

```
CREATE TABLE contohTabel (  
    title VARCHAR(255),  
    completed BOOL  
);
```

Dalam contoh ini, kita menggunakan `BOOL` sebagai tipe data untuk kolom `completed`. Perlu dicatat bahwa MySQL secara otomatis mengonversi `BOOL` menjadi `TINYINT(1)`. Oleh karena itu, pada dasarnya, ini setara dengan contoh pertama. Namun, beberapa pengembang lebih suka menggunakan `BOOLEAN` untuk kejelasan.

3. Menggunakan `TINYINT(1)`

```
CREATE TABLE contohTabel (  
    title VARCHAR(255),  
    completed TINYINT(1)  
);
```

Dalam contoh ini, kita menggunakan `TINYINT(1)` sebagai tipe data untuk kolom `completed`. Ini adalah pendekatan yang valid karena MySQL mengonversi `BOOL` menjadi `TINYINT(1)` secara otomatis. Dalam hal ini, nilai yang dapat disimpan adalah `1` untuk `TRUE` dan `0` untuk `FALSE`.

Tabel

Created Table

- Jika kalian ingin membuat tabel Ketikkan

```
mysql CREATE TABLE [nama_table] ( [nama_kolom1 tipe_data(ukuran) [tipe_constraint]  
[nama_kolom2 tipe_data(ukuran) [tipe_constraint] nama_kolom3 tipe_data(ukuran)  
[tipe_constraint] );
```

kemudian enter. Nanti akan muncul seperti di bawah.

Field	Type	Null	Key	Default	Extra
id_pelanggan	int(4)	NO	PRI	NULL	
nama_depan	varchar(25)	NO		NULL	
nama_belakang	varchar(25)	NO		NULL	
no_telp	char(12)	YES	UNI	NULL	
4 rows in set (0.106 sec)					

Struktur Table

- Lalu setelah membuat tabel kita dapat menampilkan struktur dari tabel yang kita buat dengan cara mengetik `DESC (nama_tabel)` dan hasilnya akan seperti di bawah

```
DESC nama_tabel;
```

```
MariaDB [Rental_Rahmat]> Desc pelanggan;  
+-----+-----+-----+-----+-----+-----+  
| Field      | Type      | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| id_pelanggan | int(4)     | NO   | PRI | NULL    |       |  
| nama_depan   | varchar(25) | NO   |     | NULL    |       |  
| nama_belakang | varchar(25) | NO   |     | NULL    |       |  
| no_telp      | char(12)   | YES  | UNI | NULL    |       |  
+-----+-----+-----+-----+-----+-----+  
4 rows in set (0.224 sec)
```

Show Tables

- Untuk menampilkan daftar tabel yang ada dalam *database* kita menggunakan query `SHOW TABLES;` dan hasil yang akan tampil ialah seperti berikut

```
SHOW TABLES;
```

```
MariaDB [RENTAL_RAHMAT]> SHOW TABLES;  
+-----+  
| Tables_in_rental_rahmat |  
+-----+  
| ini                      |  
| pelanggan                |  
+-----+  
2 rows in set (0.047 sec)
```

Q&A

Pertanyaan

1. Mengapa hanya kolom `id_pelanggan` yang menggunakan constraint `PRIMARY KEY`?
2. Mengapa pada kolom `no_telp` yang menggunakan data `CHAR` bukan `VARCHAR`?
3. Mengapa hanya kolom `no_telp` yang menggunakan constraint `UNIQUE`?
4. Mengapa kolom `no_telp` tidak memakai constraint `NOT NULL` sementara kolom lainnya menggunakan constraint tersebut?
5. Tuliskan perbedaan antara `PRIMARY KEY` dengan `UNIQUE`?

Jawaban:

1. Untuk membedakan `id Pelanggan` yang sama, mencegah duplikasi, dan mempermudah pencarian data.
2. Tipe data `char` menyimpan data dalam karakter panjang lebih efisien. pencarian pada kolom tipe data `CHAR` dapat lebih cepat

3. Karna **no_telp** tidak ada yang sama semua pasti berbeda dan nilainya unik maka menggunakan **constraints unique** artinya data dalam tabel **id_telpon** berbeda tidak ada yang sama.
 4. Nomor telpon dianggap opsional. nomor telepon hanya menjadi wajib saat pengguna melakukan langkah-langkah tertentu, Anda mungkin tidak ingin mengharuskan pengguna mengisinya pada tahap awal.
 5. **PRIMARY KEY** untuk membedakan data yang sama dan hanya boleh 1 dan tidak boleh tidak ada. Kalau **UNIQUE** sebuah kolom yang memiliki data yang berbeda atau tidak sama **unique** boleh 1,2,3 Dan seterusnya dan boleh tidak ada.
-

Insert, Select, Update, & Delete

- Setelah mempelajari cara untuk membuat, melihat struktur dan menampilkan daftar table sekarang kita akan mempelajari cara memasukan data pada table serta menampilkan hasil dari table tersebut, untuk melakukannya kita akan membaginya menjadi dua bagian yaitu Insert dan Select.

Insert

- Query satu ini memiliki fungsi untuk memasukkan sebuah nilai pada tabel yang telah kita buat dan kita akan mempelajari insert data 1 baris dan lebih satu baris

Insert 1 baris

- Untuk menginput data satu baris, kita menggunakan format seperti berikut;

```
INSERT INTO nama_tabel  
-> VALUES (data_1, data_2, data_3, data_4);
```

- **Result**

```
MariaDB [rental_rahmat]> INSERT INTO pelanggan  
-> VALUES (4, "Rehan", "Kz", "085233433121");  
Query OK, 1 row affected (0.037 sec)  
  
MariaDB [rental_rahmat]> select * from pelanggan;  
+-----+-----+-----+-----+  
| id_pelanggan | nama_depan | nama_belakang | no_telp |  
+-----+-----+-----+-----+  
| 1 | Rahman | Abdul | 081234567823 |  
| 2 | Dann | NT | 081248941011 |  
| 3 | Fachri | Ramadan | 083256734122 |  
| 4 | Rehan | Kz | 085233433121 |  
+-----+-----+-----+-----+  
4 rows in set (0.004 sec)
```

Insert lebih dari 1 baris

- Sedangkan untuk menginput nilai yang lebih dari satu baris kita menggunakan format sebagai berikut;


```
INSERT INTO [NAMA_TABLE]
-> VALUES (DATA_1, DATA_2, DATA_3, DATA_4),
-> (DATA_1, DATA_2, DATA_3, DATA_4),
-> (DATA_1, DATA_2, DATA_3, DATA_4);
```

- Result

```
MariaDB [rental_rahmat]> INSERT INTO pelanggan
-> VALUES (4, "fadhil", "Amir", "123456789123"),
-> (5, "Daud", "Reski", "987654321123"),
-> (6, "Ahmad", "Anugrah", "543219876321");
```

Query OK, 3 rows affected (0.255 sec)

Records: 3 Duplicates: 0 Warnings: 0

```
MariaDB [rental_rahmat]> SELECT * FROM pelanggan;
```

id_pelanggan	nama_depan	nama_belakang	no_telp
1	Rahman	Abdul	081234567823
2	Dann	NT	081248941011
3	Fachri	Ramadan	083256734122
4	fadhil	Amir	123456789123
5	Daud	Reski	987654321123
6	Ahmad	Anugrah	543219876321

6 rows in set (0.160 sec)

Select

- Selanjutnya query ini memiliki fungsi untuk menampilkan hasil dari tabel yang telah di inputkan (insert) data kedalam tabel tersebut, berbeda dengan `desc` yang hanya menampilkan struktur dari tabel query ini menampilkan hasil dari tabel

Select (Semua Tabel)

- Untuk menampilkan hasil dari seluruh tabel yang telah di buat atau menampilkan seluruh baris dan kolom kita menggunakan format sebagai berikut;

```
SELECT * FROM (NAMA_TABEL)
```

- Result

```
MariaDB [rental_rahmat]> SELECT * FROM pelanggan;
```

id_pelanggan	nama_depan	nama_belakang	no_telp
1	Rahman	Abdul	081234567823
2	Dann	NT	081248941011
3	Fachri	Ramadan	083256734122
4	fadhil	Amir	123456789123
5	Daud	Reski	987654321123
6	Ahmad	Anugrah	543219876321

```
6 rows in set (0.160 sec)
```

Select (hanya kolom)

- lalu untuk menampilkan beberapa kolom yang spesifik kita dapat menggunakan format yang sedikit berbeda dengan format all table, yaitu seperti dibawah ini :

```
SELECT NAMA_KOLOM_1, NAMA_KOLOM_2, NAMA_KOLOM_N FROM PELANGGAN;
```

- Result

```
MariaDB [rental_rahmat]> select id_pelanggan, no_telp from pelanggan;
```

id_pelanggan	no_telp
1	081234567823
2	081248941011
3	083256734122
4	123456789123
6	543219876321
5	987654321123

```
6 rows in set (0.014 sec)
```

Select kondisi "where"

- lalu kondisi yang saat satu ini berfungsi untuk mengambil data yang lebih spesifik dari sebuah field dengan simbol simbol aritmatika mulai dari "+", "-", "/", "%", ">", "<". Misalnya kita meminta untuk menampilkan field "Nama_Depan" pada "Id_Pelanggan" ke 2, kita dapat menggunakan simbol aritmatika seperti berikut :

```
SELECT Nama_Kolom FROM Nama_Table WHERE Id_Pelanggan=2;
```

- Result

```
MariaDB [rental_rahmat]> select nama_belakang from pelanggan where id_pelanggan=2;
+-----+
| nama_belakang |
+-----+
| NT             |
+-----+
1 row in set (1.567 sec)
```

Analisis

- Insert ialah query yang berfungsi untuk memasukkan data pada table yang telah kita buat.
- Select ialah query yang berfungsi untuk menampilkan hasil table dan select ini terbagi menjadi 3 bagian.
- 3 Jenis Select ialah Select All Table, Select Field Spesifik dan Select kondisi atau "Where".
- Where ini berisikan simbol simbol aritmatika mulai dari "+", "-", "/", "%", ">", "<".
- "*" simbol bintang ini memiliki makna "all" atau "semua"

Kesimpulan

- Kesimpulannya ialah *insert* bertugas untuk memasukkan nilai pada table yang telah dibuat dan *Select* berfungsi untuk menampilkan hasil dari table yang telah dibuat dan di input datanya dari Query sebelumnya, lalu *Select* ini dapat menampilkan semua sesuai dengan yang kita menggunakan misalnya jika ingin menampilkan seluruh table kita menggunakan simbol "*" atau All lalu jika ingin menampilkan beberapa field kita dapat menggunakan format hanya perlu memanggil nama fieldnya.
- Lalu yang terakhir ialah kondisi "Where" dimana kita dapat memanggil nama field dengan menggunakan simbol aritmatika, misalnya kita ingin memanggil field "Nama_Pelanggan" tapi hanya 'Id_Pelanggan' 2 kita dapat menggunakan format seperti ini `SELECT Nama_Kolom FROM Nama_Table WHERE Id_Pelanggan=2;`

Update

- Selanjutnya jika ingin mengganti nilai dari sebuah kolom tertentu kita bisa menggunakan Query *Update* lalu formatnya seperti dibawah ini :

```
Format :
UPDATE [Nama_Table] SET [Nama_Kolom]="Nilai_Pengganti" WHERE kondisi;
```

- Result

```
MariaDB [rental_rahmat]> UPDATE PELANGGAN SET nama_depan="Rahman" WHERE id_pelanggan=1;
Query OK, 1 row affected (0.079 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [rental_rahmat]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp      |
+-----+-----+-----+-----+
| 1           | Rahman    | Abdul         | 081234567823 |
| 2           | Dann      | NT            | 081248941011 |
| 3           | Fachri    | Ramadan       | 083256734122 |
+-----+-----+-----+-----+
3 rows in set (0.001 sec)

MariaDB [rental_rahmat]>
```

Delete

- Kita juga dapat menghapus baris pada table dengan Query *Delete*, untuk menghapus keseluruhan baris kita dapat menggunakan format seperti ini :

```
Format :
DELETE FROM [Nama_Table] WHERE [Nama_Kolom];
```

- Result

```
MariaDB [rental_rahmat]> delete from pelanggan where id_pelanggan=5;
Query OK, 1 row affected (0.231 sec)

MariaDB [rental_rahmat]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp      |
+-----+-----+-----+-----+
| 1           | Rahman    | Abdul         | 081234567823 |
| 2           | Dann      | NT            | 081248941011 |
| 3           | Fachri    | Ramadan       | 083256734122 |
| 4           | fadhil    | Amir         | 123456789123 |
| 6           | Ahmad     | Anugrah       | 543219876321 |
+-----+-----+-----+-----+
5 rows in set (0.079 sec)
```

Analisis

Update ialah Query untuk mengganti nilai yang telah ada pada sebuah table yang telah ada sebelumnya.

Delete ialah Query untuk menghapus baris pada sebuah tabel yang telah dibuat sebelumnya. Penggunaan Where masih sangat berperan penting dalam kondisi seperti ini.

Kesimpulan

dua Query yang akan dipelajari selanjutnya ialah untuk mengganti data dan menghapus baris data pada table. Query nya ialah *Update* untuk mengganti data yang telah ada pada table, dan Query *Delete* untuk menghapus nilai yang telah ada pada table yang telah kita buat. kedua Query ini memiliki format yang lumayan mirip, dimana memerlukan Where untuk menuliskan kondisinya.