BAB V
MANIPULASI DATA

V.1 MEMASUKKAN DATA DALAM DATABASE MYSQL

Sebelum Anda dapat melakukan hal apapun dengan data dalam database, data harus sudah ada. Untuk alasan inilah, pernyataan SQL yang perlu Anda pelajari adalah memasukkan data dalam database. Ketika Anda menambahkan data dalam database, sebenarnya Anda menambahkannya ke tabel-tabel secara individu dalam database tersebut.

V.1.1 MENGGUNAKAN PERNYATAAN INSERT UNTUK MENAMBAHKAN DATA

Pernyataan INSERT merupakan metode yang paling umum digunakan untuk memasukkan data secara langsung ke dalam sebuah tabel.
Berikut ini adalah sintaks dalam pernyataan INSERT :

```
<pernyataan insert>::=
INSERT [LOW_PRIORITY | DELAYED] [IGNORE] [INTO]
{<opsi nilai> | <opsi set> | <opsi select>}

< opsi nilai >::=
<nama tabel> [(<nama kolom> [{, < nama kolom >}...])]
VALUES ({<ekspresi> | DEFAULT} [{, {< ekspresi > | DEFAULT}}...])
[{, ({<ekspresi > | DEFAULT} [{, {< ekspresi > | DEFAULT}}...])}...]

< opsi set >::=
< nama tabel >
SET < nama kolom >={< ekspresi > | DEFAULT}
[{, < nama kolom >={< ekspresi > | DEFAULT}}...]

< opsi select >::=
< nama tabel > [(<nama kolom > [{, < nama kolom >}...])]
```

```
<pernyataan select>
```

Anda dapat menggunakan pernyataan INSERT untuk menambahkan data ke tabel manapun dalam database MySQL. Ketika Anda menambahkan data Anda harus melakukannya dari baris ke baris, Anda harus memasukkannya dengan tepat satunilai per kolom. Jika Anda menspesifikasikan nilai yang lebih sedikit dari jumlah kolom yang ada, nilai default atau null akan dimasukkan ke kolom yang nilainya tidak dispesifkasikan.
Sekarang perhatikan baris pertama sintaks pada pernyataan INSERT:

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE] [INTO]
```

Sebagaimana Anda lihat, pada bars pertama memerlukan kata kunci INSERT dan sejumlah opsi-opsi. Opsi pertama adalah LOW_PRIORITY dan DELAYED. Anda dapat menyertakan salah satu dari opsi ini, tidak bisa keduanya. Jika Anda menspesifikasikan LOW_PRIORITY, pernyataan tidak akan dieksekusi sampai tidak ada koneksi klien yang sedang mengakses tabel yang sama dimana pernyataan INSERT juga sedang mengakses. Hal ini dapat menyebabkan penundaan yang lama sementara Anda menunggu pernyataan tersebut dieksekusi. Pada saat itu, Anda tidak dapat melakukan aksi apapun. Jika Anda menspesifikasikan DELAYED, eksekusi juga akan ditunda, namun Anda dapat melanjutkan untuk melakukan aksi lain sementara pernyataan INSERT berada dalam antrian. Opsi LOW_PRIORITY dan DELAYED dapat digunakan hanya untuk memasukkan data terhadap tabel MyISAM dan ISAM.

Opsi berikut yang dapat Anda tentukan adalah klausa IGNORE. Opsi ini diterapkan terutama pada pernyataan INSERT yang menambahkan banyak baris ke suatu tabel. Jka Anda menspesifikasikan IGNORE, baris-baris yang ditambahkan akan diacuhkan jika mereka berisi sebuah nilai yang terduplikasi terhadap nilai dari *primary key* atau *index* unik. Pernyataan INSERT melanjutkan untuk menambahkan baris-baris yang tersisa. Jika Anda tidak menspesifikasikan IGNORE, nilai-nilai yang terduplikasi akan membatalkan proses penambahan data.

Menggunakan alternatif <opsi nilai> pada pernyataan INSERT

Alternatif <opsi nilai> pada pernyataan INSERT mengijinkan Anda untuk menambahkan satu atau lebih baris ke suatu tabel. Sintaksnya adalah sebagai berikut:

```
<opsi nilai>::=
```

```
<nama tabel> [(<nama kolom> [{, < nama kolom >}...])]
VALUES ({<ekspresi> | DEFAULT} [{, {<ekspresi> | DEFAULT}}...])
        [{, ({<ekspresi> | DEFAULT} [{, {<ekspresi> | DEFAULT}}...])}...]
```

Sebagaimana Anda lihat dalam sintaks, Anda harus menyediakan nama tabel dan sebuah klausa VALUES. Anda juga mempunyai pilihan untuk menspesifikasikan satu atau lebih kolom setelah nama tabel. Jika Anda menspesifikasikan nama kolom, mereka harus ditutup dalam tanda kurung dan dipisahkan oleh koma.

Sekali Anda menspesifikasikan nama tabel dan kolom opsional, Anda harus menspesifikasikan klausa VALUES. Klausa harus menyertakan minimal satu nilai, dimana direpresentasikan oleh <ekspresi> atau kata kunci DEFAULT. Jika Anda menyertakan nama kolom setelah nama kolom, klausa VALUES harus menyertakan sebuah nilai untuk setiap kolom, sesuai dengan daftar kolom yang ada. Jika Anda tidak menentukan nama kolom, Anda harus menyediakan sebuah nilai untuk setiap kolom dalam tabel, sesuai dengan daftar kolom yang terdefinisi dalam tabel.

Kita akan membuat pernyataan pemasukan data berdasarkan definisi tabel berikut:

```
CREATE TABLE CD
(
IDCD SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
NamaCD VARCHAR(50) NOT NULL,
Copyright YEAR,
JumlahDisk TINYINT UNSIGNED NOT NULL DEFAULT 1,
JumlahStok TINYINT UNSIGNED,
JumlahCadangan TINYINT UNSIGNED NOT NULL,
JumlahTersedia TINYINT UNSIGNED NOT NULL,
TipeCD VARCHAR(20),
WaktuPenambahan TIMESTAMP
);
```

Jika Anda lupa dengan penjelasan elemen-elemen dalam definisi tabel silakan dibaca modul pada pertemuan sebelumnya.

Tabel dalam definisi ini, dimana bernama CD, menyimpan informasi mengenai *compact disks*. Anggap saja Anda ingin menggunakan pernyataan INSERT untuk menambah informasi mengenai CD Anda yang bernama *Surgamu – Koleksi Ungu*. Anda dapat menset pernyataan Anda dalam berbagai cara. Pertama adalah menentukan nilai dari setiap kolom, tanpa menspesifikasikan nama dari kolom, contoh:

```
INSERT INTO CD
VALUES (NULL, 'Surgamu – Koleksi Ungu',2006, 2, 10, 3,
JumlahStok - JumlahCadangan, 'Slow', NULL);
```

Dalam pernyataan ini, baris pertama berisi kata kunci perintah INSERT, opsional kata kunci INTO, dan nama tabel (CD). Klausa VALUES meliputi sebuah nilai untuk setiap kolom, dimasukkan secara berurut dimana sesuai dengan urutan penampakan kolom dalam definisi tabel. Nilai-nilai ditutup dalam tanda kurung dan dipisahkan oleh koma.

Kolom pertama diisi dengan NULL. Nilai ini digunakan pada kolom IDCD, dimana dikonfigurasi dengan opsi AUTO_INCREMENT dan merupakan *primary key*. Dengan menspesifikasikan NULL, nilai inkrementasi berikutnya secara otomatis ditambahkan pada kolom tersebut ketika Anda menambahkan baris ini ke tabel. Karena ini adalah baris pertama yang ditambahkan ke tabel, nilai 1 dimasukkan ke kolom IDCD.

Nilai berikut dalam klausa VALUES adalah menunjuk pada kolom NamaCD. Karena nilainya string, dia ditutup dengan tanda kutip satu pada kedua sisinya.
Catatan, untuk menambahkan karakter tertentu dalam string , Anda harus memakai tanda *backslash*. Misal:

- String = 'Do'a Ibu'
  Anda harus menuliskan 'Do\'a Ibu'
- Begitu juga untuk tanda kutip ganda (''), *backslash* (\), persen (%), dan *underscore* (_).

Empat nilai berikut yang ditentukan dalam klausa VALUES adalah data tanggal dan numerik yang menunjuk kolom-kolom dalam definisi tabel (Copyright = 2006, JumlahDisk =2, JumlahStok = 10, dan JumlahCadangan = 3).

Nilai yang diberikan untuk kolom JumlahTersedia (JumlahStok-JumlahCadangan) adalah merupakan ekspresi yang menggunakan dua nama kolom dan tanda operator aritmatika minus (-) untuk mengurangi nilai sebanyak pada kolom JumlahCadangan dari nilai pada kolom JumlahStok untuk memperoleh jumlah total CD yang tersedia untuk dijual. Dalam hal ini totalnya adalah 7.

Nilai berikutnya adalah Slow, dimana dimasukkan pada kolom TipeCD. Nilai terakhir adalah NULL, dimana digunakan untuk kolom WaktuPenambahan. Kolom dikonfigurasi sebagai kolom TIMESTAMP, yang berarti bahwa waktu dan tanggal sekarang ditambahkan secara otomatis pada kolom tersebut.

Cara lain untuk menambahkan tanggal sekarang adalah dengan menggunakan fungsi NOW(), daripada NULL. Jika hanya tanggal saja, Anda dapat mengugnakan fungsi CURDATE() atau CURTIME() untuk waktu.

Berikut ini juga contoh untuk menambahkan data ke tabel CD:

```
INSERT LOW_PRIORITY INTO CD (NamaCD, Copyright, JumlahDisk, JumlahStok,
JumlahCadangan, JumlahTersedia, TipeCD)
VALUES ('Opick – Semesta Bertasbih', 2006, DEFAULT, 13, 2,
JumlahStok - JumlahCadangan, 'Nasyid');
```

Dalam pernyataan ini, kolom IDCD dan WaktuPenambahan tidak dispesifikasikan. Karena IDCD merupakan kolom AUTO_INCREMENT, sebuah nilai inkrementasi secara otomatis ditambahkan ke kolom tersebut. Dan karena kolom WaktuPenambahan merupakan kolom TIMESTAMP, waktu dan tanggal sekarang dimasukkan ke kolomnya.

Untuk kolom JumlahDisk, DEFAULT digunakan. Ini menandakan bahwa nilai default harus dimasukkan ke kolom. Dalam hal ini, nilai 1. Sebaliknya, nilai yang tertera sama dengan nilai yang Anda lihat sebelumnya. Dalam contoh terakhir ini terdapat opsi LOW_PRIORITY dalam klausa INSERT. Sehingga, pernyataan ini tidak diproses dan klien akan terus menunggu sampai semua koneksi klien telah selesai mengakses tabel yang dituju.

Sekarang kita akan menampilkan sebagian data dari tabel CD, misal IDCD, NamaCD, dan JumlahDisk, hasilnya :

```
+----------+----------------------------------------+------------------          +
| IDCD | NamaCD                                 | JumlahDisk     |
+----------+----------------------------------------+------------------          +
|    1    | Surgamu – Koleksi Ungu          |       2          |
|    2    | Opick – Semesta Bertasbih       |       1          |
+----------+----------------------------------------+------------------          +
2 rows in set (0.00 sec)
```

Contoh lain yang lebih sederhana:

```
INSERT INTO CDs (NamaCD)
VALUES ('GIGI ');
```

Sedangkan berikut ini contoh untuk menambahkan banyak baris sekaligus:

```
INSERT INTO CDs (NamaCD, Copyright, JumlahDisk,
JumlahStok, JumlahCadangan, JumlahTersedia,
VALUES ('Slank 99', 1999, 1, 9, 0,
JumlahStok-JumlahCadangan, 'Blues'),
('Metallica The Unforgiven', 1990, 1, 14, 2,
JumlahStok-JumlahCadangan, 'Metal'),
('Boomerang – Bawalah Aku', 1988, 1, 6, 1,
JumlahStok-JumlahCadangan, 'Rock');
```

V.1.2 MENGGUNAKAN PERNYATAAN REPLACE UNTUK MENAMBAHKAN DATA

Selain pernyataan INSERT untuk menambahkan data ke dalam tabel, Anda juga dapat menggunakan pernyataan REPLACE. Perbedaan dengan INSERT adalah bagaimana nilai dalam kolom *primary key* dan indeks *unique* diperlakukan. Dalam pernyataan INSERT, jika Anda mencoba memasukkan baris yang berisi nilai indeks *unique* atau *primary key* yang sudah terdapat dalam tabel, Anda tidak bisa

menambahkannya dalam baris tersebut. Pernyatan REPLACE akan menghapus baris lama dan menambahkan baris yang baru.

Sintaksnya adalah:

Sedangkan berikut ini contoh untuk menambahkan banyak baris sekaligus:

```
<pernyataan replace>::=
REPLACE [LOW_PRIORITY | DELAYED] [INTO]
{< opsi nilai > | < opsi set > | < opsi select >

<opsi nilai>::=
< nama tabel > [(<nama kolom > [{, < nama kolom >}...])]
VALUES ({<ekspresi > | DEFAULT} [{, {< ekspresi > | DEFAULT}}...])
[{, ({<ekspresi > | DEFAULT} [{, {< ekspresi > | DEFAULT}}...])}...]

<opsi set>::=
<nama tabel>
SET < nama kolom >={<ekspresi> | DEFAULT}
[{, < nama kolom >={<ekspresi> | DEFAULT}}...]

<opsi select>::=
<nama tabel> [(<nama kolom> [{, < nama kolom >}...])]
<pernyataan select>
```

Contoh.

Terdapat definisi tabel sebagai berikut:

```
CREATE TABLE Inventori
(
IDProduk SMALLINT UNSIGNED NOT NULL PRIMARY KEY,
JumlahStok SMALLINT UNSIGNED NOT NULL,
JumlahPemesanan SMALLINT UNSIGNED NOT NULL,
TanggalUpdate DATE
```

Karakteristik utama dalam tabel Inventori adalah kolom IDProduk dikonfigurasi sebagai *primary key*. Dia tidak menggunakan opsi AUTO_INCREMENT, sehingga Anda harus menyediakan sebuah nilai untuk kolom ini. Berikut ini contoh pernyataan REPLACE untuk menambahkan data pada setiap kolom dalam tabel Inventori:

```
REPLACE LOW_PRIORITY INTO Inventori
VALUES (101, 20, 25, '2006-10-14');
```

Anda dapat menampilkan nilai-nilai yang telah Anda tambahkan dalam tabel Inventori dengan mengeksekusi perintah SELECT berikut:

```
SELECT * FROM Inventori
```

Misalkan Anda ingin menambahkan data baru sebagai berikut:

```
REPLACE LOW_PRIORITY INTO Inventori
VALUES (101, 10, 25, '2006-10-16');
```

Dalam pernyataan tersebut juga disertakan nilai 101. Hasilnya, baris yang asli dengan nilai IDProduk tersebut akan dihapus dan baris baru akan ditambahkan.

V.2 MENGUPDATE DATA DALAM DATABASE MYSQL

Pernyataan utama yang digunakan untuk memodifikasi data dalam database MySQL adalah pernyataan UPDATE. Sintaksnya adalah sebagai berikut :

```
<pernyataan update>::=
UPDATE [LOW_PRIORITY] [IGNORE]
<single table update> | <joined table update>

<single table update>::=
<nama tabel>
```

```
SET <nama kolom>=<ekspresi> [{, <nama kolom>=<ekspresi>}...]
[WHERE <definisi where>]
[ORDER BY <nama kolom> [ASC | DESC] [{, <nama kolom> [ASC | DESC]}...]]
[LIMIT <jumlah baris>]

<joined table update>::=
<nama tabel> [{, <nama tabel>}...]
SET <nama kolom>=<ekspresi> [{, <nama kolom>=<ekspresi>}...]
[WHERE <definisi where>]
```

Baris pertama dari sintaks berisi kata kunci perintah UPDATE bersamaan dengan opsi LOW_PRIORITY dan IGNORE, keduanya seperti yang Anda lihat pada pernyataan INSERT. Anda sebaiknya menggunakan opsi LOW_PRIORITY ketika Anda ingin menunda eksekusi pernyataan UPDATE sampai tidak ada lagi koneksi klien yang sedang mengakses tabel target. Anda sebaiknya menggunakan opsi IGNORE jika Anda ingin update dilanjutkan meskipun ditemukan nilai duplikat pada *primary key* dan indeks *unique*. (Baris dengan nilai duplikat tidak diupdate).

MENGUPDATE TABEL TUNGGAL (*SINGLE TABLE*)

Untuk mengupdate tabel tunggal dalam database MySQL, dimana tidak ada kondisi join yang diambil ke dalam akun untuk melakukan peng-update-an, Anda sebaiknya membuat pernyataan UPDATE yang menggunakan alternatif <*single table update*>, dimana ditunjukkan sintaks berikut ini :

```
<single table update>::=
<nama tabel>
SET <nama kolom>=<ekspresi> [{, <nama kolom>=<ekspresi>}...]
[WHERE <definisi where>]
[ORDER BY <nama kolom> [ASC | DESC] [{, <nama kolom> [ASC | DESC]}...]]
[LIMIT <jumlah baris>]
```

Sebagaimana terlihat pada sintaks, Anda harus menspesifikasikan nama tabel dan klausa SET. Klausa SET meliputi, minimal, sebuah nama kolom dan ekspresi yang berhubungan, dihubungkan dengan tanda sama dengan (=). Informasi mengeset sebuah nilai untuk satu kolom tertentu. Jika Anda ingin

menyertakan lebih dari satu kolom, Anda harus memisahkan pasangan kolom/ekspresi dengan koma.

Contoh.

Membuat tabel Buku.

```
CREATE TABLE Buku
(
IDBuku SMALLINT NOT NULL PRIMARY KEY,
NamaBuku VARCHAR(40) NOT NULL,
Stok SMALLINT NOT NULL
)
ENGINE=INNODB;
```

Memasukkan data ke dalam tabel Buku.

```
INSERT INTO Buku
VALUES (101, 'Noncomformity: Writing on Writing', 12),
(102, 'The Shipping News', 17),
(103, 'Hell\'s Angels', 23),
(104, 'Letters to a Young Poet', 32),
(105, 'A Confederacy of Dunces', 6),
(106, 'One Hundred Years of Solitude', 28);
```

Tabel berikut adalah tabel Pembelian, yang menyertakan *foreign key* yang mereferensi ke tabel Buku.

```
CREATE TABLE Pembelian
(
IDPembelian SMALLINT NOT NULL PRIMARY KEY,
IDBuku SMALLINT NOT NULL,
Kuantitas TINYINT (40) NOT NULL DEFAULT 1,
TanggalBeli TIMESTAMP,
FOREIGN KEY (IDBuku) REFERENCES Buku (IDBuku)
)
```

```
ENGINE=INNODB;
```

Memasukkan data ke dalam tabel Pembelian.

```
INSERT INTO Pembelian
VALUES (1001, 103, 1, '2006-10-12 12:30:00'),
(1002, 101, 1, '2006-10-12 12:31:00'),
(1003, 103, 2, '2006-10-12 12:34:00'),
(1004, 104, 3, '2006-10-12 12:36:00'),
(1005, 102, 1, '2006-10-12 12:41:00'),
(1006, 103, 2, '2006-10-12 12:59:00'),
(1007, 101, 1, '2006-10-12 13:01:00'),
(1008, 103, 1, '2006-10-12 13:02:00'),
(1009, 102, 4, '2006-10-12 13:22:00'),
(1010, 101, 2, '2006-10-12 13:30:00'),
(1011, 103, 1, '2006-10-12 13:32:00'),
(1012, 105, 1, '2006-10-12 13:40:00'),
(1013, 106, 2, '2006-10-12 13:44:00'),
(1014, 103, 1, '2006-10-12 14:01:00'),
(1015, 106, 1, '2006-10-12 14:05:00'),
(1016, 104, 2, '2006-10-12 14:28:00'),
(1017, 105, 1, '2006-10-12 14:31:00'),
(1018, 102, 1, '2006-10-12 14:32:00'),
(1019, 106, 3, '2006-10-12 14:49:00'),
(1020, 103, 1, '2006-10-12 14:51:00');
```

Perhatikan bahwa nilai-nilai yang ditambahkan ke kolom IDBuku meliputi hanya nilai-nilai yang terdapat dalam kolom IDBuku pada tabel Buku. Kolom IDBuku di Pembelian merupakan kolom yang mereferensi/mengkait, dan kolom IDBuku di tabel Buku merupakan kolom yang direferensi/dikait.

Setelah membuat tabel dan menambahkan data pada tabel-tabel tersebut, Anda dapat memodifikasi datanya. Contoh:

```
UPDATE Buku
SET Stok=Stok+10;
```

Dalam pernyataan ini karena tidak kondisi khusus yang harus dipenuhi, berarti akan mengubah seluruh nilai dari kolom Stok ditambahkan dengan 10.
Sedangkan untuk memodifikasi data dengan memnuhi kondisi tertentu, Anda bisa menambahkannya dengan klausa WHERE.

```
UPDATE Pembelian
SET Kuantitas=2
WHERE IDPembelian=1001;
```

Dalam pernyataan di atas berarti akan mengupdate kolom Kuantitas menjadi 2 dimana IDPembelian = 1001.

```
UPDATE Pembelian
SET Kuantitas=Kuantitas+1
WHERE IDPembelian=1001;
```

Sedangkan dalam pernyataan di atas, baris dengan IDPembelian = 1001, nilai dari kuantitasnya yang lama akan ditambahkan dengan 1.

```
UPDATE LOW_PRIORITY Buku
SET Stok=Stok+10
WHERE Stok<30;
```

Setiap rekod/baris pada tabel Buku yang mempunyai Stok lebih kecil dari 30 akan ditambah dengan 10.

```
UPDATE Pembelian
SET Kuantitas=Kuantitas+1
WHERE IDBuku=103
ORDER BY TanggalPembelian DESC
LIMIT 5;
```

Mengupdate tabel Pembelian yang mempunyai IDBuku = 103, dimana nilai Kuantitas ditambah 1. Diurutkan berdasarkan TanggalPembelian secara DESCENDING pada 5 baris pertama.

MENGUPDATE TABEL JOIN (*JOINED TABLE*)

Dalam contoh pernyataan UPDATE sebelumnya Anda telah melihat bahwa update hanya dilakukan pada tabel-tabel secara individual tanpa menjoinkannya dengan tabel lain. Meskipun tabel-tabel berisi *foreign key* yang direferensi/dikait oeh tabel lain, Anda tidak menspesifikasikan kondisi *no join* dalam pernyataan UPDATE.
Sintaksnya adalah sebagai berikut:

```
<joined table update>::=
<nama tabel> [{, <nama tabel>}...]
SET <nama kolom>=<ekspresi> [{, <nama kolom>=<ekspresi>}...]
[WHERE <definisi where>]
```

Contoh.

```
UPDATE Buku, Pembelian
SET Buku.Stok = Buku.Stok - Pembelian.Kuantitas
WHERE Buku.IDBuku= Pembelian.IDBuku
        AND Pembelian.IDPembelian=1002;
```

Dalam pernyataan di atas berarti, mengupdate tabel Buku, dimana kolom Stok yang baru merupakan hasil pengurangan kolom Stok yang lama dengan kolom Kuantitas pada tabel Pembelian dengan syarat IDBuku di tabel Buku sama dengan IDBuku di tabel Pembelian dan IDPembelian di tabel Pembelian = 1002.

Lihatlah pada pernyataan di atas. Dalam klausa UPDATE disertakan nama dari kedua tabel yaitu Buku dan Pembelian. Meskipun Anda hanya melakukan peng-update-an pada tabel Buku, Anda harus menspesifikasikan kedua tabel karena keduanya Anda sertakan dalam tabel yang saling join (*joined tables*). Perhatikan bahwa nama-nama tabel dipisahkan oleh koma.

Klausa SET pada pernyataan ini menggunakan nama kolom *qualified* untuk memberikan ekspresi pada kolom Stok. Sebuah nama kolom *qualified* adalah didahului oleh nama tabel dan sebuah tanda titik. Hal ini mengijinkan MySQL (dan Anda) untuk membedakan antara kolom-kolom pada tabel yang berbeda-beda yang mempunyai nama yang sama. Misal, kolom pertama (Buku.Stok) menunjuk pada kolom Stok pada tabel Buku.

Berikut ini contoh mengupdate tabel join dengan banyak nilai sekaligus:

```
UPDATE Buku, Pembelian
SET Pembelian.Kuantitas = Pembelian.Kuantitas + 2,
      Buku.Stok= Buku.Stok - 2
WHERE Buku.IDBuku= Pembelian. IDBuku
      AND Pembelian.IDPembelian = 1002;
```

V.3 MENGHAPUS DATA DALAM DATABASE MYSQL

Sintaksnya adalah sebagai berikut:

```
<pernyataan delete>::=
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
{<single table delete> | <from join delete> | <using join delete>

<single table delete>::=
FROM <nama tabel>
[WHERE <definisi where>]
[ORDER BY <nama kolom> [ASC | DESC] [{, <nama kolom> [ASC | DESC]}...]]
[LIMIT <jumlah baris>]

<from join delete>::=
<nama tabel>[.*] [{, <nama tabel>[.*]}...]
FROM <nama tabel> [{, <nama tabel>}...]
[WHERE <definisi where>]
```

```
<using join delete>::=
FROM <nama tabel>[.*] [{, <nama tabel>[.*]}...]
USING <nama tabel> [{, <nama tabel>}...]
[WHERE <definisi where>]
```

Contoh menghapus pada tabel tunggal.

```
DELETE FROM Pembelian
```

```
DELETE FROM Pembelian
WHERE IDPembelian=1020;
```

```
DELETE LOW_PRIORITY FROM Pembelian
WHERE IDBuku=103
ORDER BY TanggalPembelian DESC
LIMIT 1;
```

Menghapus pada *Joined Tables*.

```
DELETE Pembelian.*
FROM Buku, Pembelian
WHERE Buku.IDBuku = Pembelian.IDBuku
      AND Buku.NamaBuku='Where I\'m Calling From'
```