*Adityo Permana W*

# MODUL

## Sistem Basis Data

*Lanjut*


MySQL

Microsoft SQL Server


**Nama :**

**NPM :**

**Program Studi Sarjana Informatika**
**Fakultas Teknologi Informasi dan Elektro**
**Universitas Teknologi Yogyakarta**

**2019**

# SISTEM BASIS DATA LANJUT

**Oleh:**

**Adityo Permana Wibowo**

**Universitas Teknologi Yogyakarta**
**2019**

Diterbitkan oleh:
Universitas Teknologi Yogyakarta
Jl. Siliwangi, Jombor, Sleman, Yogyakarta
Email : publikasi@uty.ac.id
Website : uty.ac.id

# Sistem Basis Data Lanjut

ISBN : 978-623-92626-6-2
-

Oleh : Adityo Permana Wibowo, S.Kom., M.Cs.

-
Edisi ke-1
Cetakan Pertama, 2019

# KATA PENGANTAR

Potensi penggunaan komputer saat ini benar-benar dapat dirasakan, terutama dalam pengolahan data dimana dapat menghasilkan informasi yang berkualitas. Modul Sistem Basis Data Lanjut ini dirancang untuk melanjutkan materi dari modul Sistem Basis Data pada semester sebelumnya. Materi pada modul ini dibuat menyesuaikan kebutuhan dunia kerja saat ini dan dimaksudkan untuk menunjang kemampuan (*soft skill*) mahasiswa dalam mengimplementasikan sistem basis data. Modul ini diharapkan memberikan bekal kepada mahasiswa agar siap terjun dalam dunia kerja dan mengetahui bagaimana proses untuk melakukan implementasi sebuah basis data dengan pemrograman aplikasi yang berbasis visual.

Modul ini terdiri atas 14 bab, yang isinya dimulai dari materi review modul Sistem Basis Data terdiri dari 3 bab, kemudian dilanjutkan materi manajemen user database, view, trigger, stored procedure, function, event, dan terakhir materi tentang pemrograman client dan server menggunakan database MySQL dan aplikasi berbasis GUI yaitu Netbeans IDE 8.0. Masing-masing bab juga disertakan latihan studi kasus tentang implementasi basis data sehingga diharapkan mahasiswa bisa lebih mengerti untuk pembuatan basis data menggunakan DBMS tersebut.

Akhir kata, kami tim penyusun yang terdiri dari mengucapkan terimakasih kepada semua pihak yang telah membantu menyelesaikan modul ini. Kritik dan saran sebagai penyempurnaan modul ini sangat kami harapkan bisa dikirimkan melalui email adityopw@staff.uty.ac.id.

**Yogyakarta, Desember 2019**

**Penyusun,**

**Adityo Permana Wibowo, S.Kom., M.Cs.**

# DAFTAR ISI

# PENDAHULUAN

## I. Deskripsi Materi

Materi Sistem Basis Data Lanjut di modul ini mengambil refrensi dari buku yang ditulis oleh Abraham Sylberschat, dengan judul buku Database System Concept. Buku tersebut diterbitkan tahun 2011.

Modul ini merupakan lanjutan dari materi Sistem Basis Data sebelumnya. Materi sebelumnya membahas tentang basis data mulai dari perancangan basis data sampai dengan implementasi menggunakan perangkat lunak DBMS MySQL. Selanjutnya, materi yang dibahas pada modul ini mulai dari pembahasan SubQuery, manajemen user dan privileges, pembuatan aplikasi basis data menggunakan pemrograman berbasis GUI yaitu Netbeans IDE 8.0, sampai dengan pembahasan program aplikasi basis data yang berbasis *client* dan *server*. Masing-masing materi disertakan berbagai macam studi kasus sehingga diharapkan materi bisa lebih dimengerti oleh mahasiswa. Pemilihan perangkat lunak DBMS MySQL tersebut dikarenakan banyak digunakan saat dunia kerja, sehingga mahasiswa disiapkan untuk pemahaman terkait perangkat lunak DBMS MySQL tersebut.

Setelah mempelajari modul ini diharapkan mahasiswa dapat mempunyai landasan atau struktur mengenai perancangan basis data dan pengetahuan tentang Bahasa Query yang kuat, sampai dengan pembuatan program aplikasi basis data berbasis jaringan client dan server sehingga nantinya saat menghadapi pemecahan suatu permasalahan dalam membangun suatu basis data dapat mengatasinya dengan mudah.

## II. Prasyarat

Sebelum menggunakan modul ini diharapkan mahasiswa sudah memenuhi beberapa prasyarat, antara lain:

a) Mahasiswa mampu menggunakan command prompt meskipun berbasis windows.
b) Mahasiswa mempunyai pemahaman logika yang baik (sudah menempuh matakuliah prasyarat yaitu Logika Informatika).
c) Mahasiswa sudah lulus matakuliah Sistem Basis Data.
d) Mahasiswa sudah lulus matakuliah Jaringan Komputer.
e) Mahasiswa sudah lulus matakuliah pemrograman.

## III. Petunjuk Penggunaan Modul

Modul ini dapat digunakan mahasiswa dengan pertimbangan sebagai berikut:

a) Mahasiswa telah memiliki modul dan telah membaca modul sebelum mata praktikum dimulai.
b) Mahasiswa mempelajari serta mengidentifikasi isi modul yang diuraikan lebih rinci oleh asisten pengampu.
c) Mahasiswa dan asisten mendiskusikan materi untuk mencari penyelesaian terhadap kasus tertentu.
d) Mahasiswa menyimpulkan isi materi yang didiskusikan.
e) Mahasiswa menjawab soal latihan yang diberikan.
f) Pemberian pengayaan materi bagi mahasiswa yang telah memahami dan menyelesaikan soal latihan.
g) Memberikan tinjauan ulang terhadap materi sekaligus mengidentifikasi kesulitan-kesulitan mahasiswa dalam memahami materi.

## IV. Standar Kompetensi

a) Mahasiswa mampu menjelaskan konsep sistem basis data.
b) Mahasiswa mampu memecahkan permasalahan basis data menggunakan perintah query.
c) Mahasiswa mampu memecahkan permasalahan basis data terkait pengelolaan user dan hak akses masing-masing user.
d) Mahasiswa mampu menerapkan konsep basis data pada sebuah pemrograman aplikasi berbasis GUI yang berbasis jaringan *client* dan *server*.

# BAB 1. STRUCTURE QUERY LANGUAGE (SQL) dalam 1 TABEL

## 1.1. KOMPETENSI DASAR

Setelah mempelajari Bab ini, mahasiswa:

1) Memahami pengertian dan fungsi query DDL
2) Memahami pengertian dan fungsi query DML
3) Memahami pengertian dan fungsi query Fungsi dan Aritmatika

## 1.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa:

1) Mampu menjelaskan dan mengimplementasikan query DDL.
2) Mampu menjelaskan dan mengimplementasikan query DML.
3) Mampu menjelaskan dan mengimplementasikan query Fungsi dan Aritmatika

## 1.3. URAIAN MATERI

### 1.3.1. Query Data Definition Language

DDL (*Data Definition Language*) merupakan perintah dasar dalam SQL yang digunakan untuk membuat basis data dan mendefinisikan struktur tabel dalam basis data tersebut. Struktur tabel yang dimaksud terdiri dari nama kolom, lebar kolom, penentuan *key*, dan relasi antar tabel. Statement query DDL adalah kumpulan perintah terdiri dari Create, Alter, Drop untuk mendefinisikan tipe data dari objek-objek basis data. Objek basis data (pada MySQL) tersebut terdiri dari *database*, *table*, *view*, *index*, *stored procedure*, *function*, *trigger*. Perintah-perintah DDL seperti terlihat pada Tabel 1.1.

**Tabel 1.1** Perintah DDL

| Create (pembuatan) | Alter/Rename (Perubahan) | Drop (Penghapusan) |
|---|---|---|
| CREATE DATABASE | ALTER DATABASE | DROP DATABASE |
| CREATE FUNCTION | ALTER FUNCTION | DROP FUNCTION |
| CREATE INDEX | ALTER PROCEDURE | DROP INDEX |
| CREATE PROCEDURE | ALTER TABLE | DROP PROCEDURE |
| CREATE TABLE | ALTER VIEW | DROP TABLE |
| CREATE TRIGGER | RENAME TABLE | DROP TRIGGER |
| CREATE VIEW | | DROP VIEW |

Contoh penggunaan perintah query DDL untuk *create table*:

```
CREATE TABLE ms_karyawan (
      kd_cbg varchar(10) not null,
      kd_kary varchar(10) not null,
      nm_dpan varchar(8) default null,
      nm_blkg varchar(9) default null,
      jns_klamin varchar(8) default null,
primary key (kode_karyawan))
```

### 1.3.2. Query Data Manipulation Language

DML (*Data Manipulation Language*) merupakan perintah dalam SQL yang memungkinkan pengguna untuk memanipulasi data dalam tabel. Yang dimaksud memanipulasi data adalah:

a) Menambah data baru pada suatu tabel.
b) Merubah data yang sudah ada pada suatu tabel.
c) Menghapus data pada suatu tabel.
d) Menampilkan informasi yang disimpan dalam tabel pada sebuah basis data

DML juga bertujuan untuk memudahkan pemakai untuk mengakses data sebagaimana direpresentasikan oleh model data. Perintah DML pada MySQL terdiri dari *Call, Delete, Do, Handler, Insert, Load Data Infile, Replace, Select, Truncate, Update*. Hanya saja pada umumnya perintah DML yang biasa digunakan adalah *Insert, Update, Delete, Select*. Jenis DML terdiri dari:

a) **Prosedural**: mensyaratkan pemakai untuk menentukan data apa yang diinginkan serta bagaimana cara mendapatkannya.
b) **NonProsedural**: membuat pemakai dapat menentukan data apa yang diinginkan tanpa menyebutkan bagaimana cara mendapatkannya.

Contoh penggunaan perintah query DML untuk *insert* data ke dalam tabel ms_karyawan, seperti pada gambar di bawah ini:

```
INSERT INTO ms_karyawan (kd_cbg, kd_kary, nm_dpan, nm_blkg, jns_klamin)
VALUES ('cbg03','kary08','Abdul','Query','laki-laki');
```

Contoh penggunaan perintah query DML untuk *update* data pada tabel ms_karyawan, seperti pada gambar di bawah ini:

```
UPDATE ms_karyawan SET nm_blkg='ALEX' WHERE kd_kary='KA08';
```

Contoh penggunaan perintah query DML untuk *delete* data pada tabel ms_karyawan, seperti pada gambar di bawah ini:

```
DELETE FROM ms_karyawan WHERE kd_kary='KA08';
```

### 1.3.3. Query Fungsi

Terdapat beberapa perintah operator atau fungsi yang biasa digunakan di SQL, antara lain SUM, MIN, MAX, COUNT, AVG, dan Group By. Sebelum membahas tentang operator tersebut, ada baiknya perlu diingat-ingat kembali konsep penggunaan operator dan fungsi yang sudah pernah dibahas pada modul semester sebelumnya. Mulai dari Syntak Dasar SQL tentang SUM. SUM adalah suatu fungsi pada SQL yang digunakan untuk menjumlahkan nilai dari sekumpulan record. Selanjutnya fungsi MIN, MIN adalah suatu fungsi pada SQL yang digunakan untuk mendapatkan nilai terkecil dari sekumpulan *record*. Kemudian fungsi MAX, MAX adalah kebalikan dari MIN, yaitu fungsi yang digunakan untuk mendapatkan nilai tertinggi dari sekumpulan record. Kemudian penggunaan COUNT. Count adalah suatu fungsi pada SQL yang digunakan untuk mendapatkan jumlah baris atau record dari suatu tabel. Kemudian penggunaan AVG (AVERAGE). AVG adalah suatu fungsi pada SQL yang digunakan untuk mendapatkan nilai rata-rata dari sekumpulan nilai *record* suatu tabel. Terakhir penggunaan Group BY. Fungsi Group By digunakan untuk menampilkan berdasarkan pengelompokkan sebuah kolom.

### 1.3.4. AGREGATE

Fungsi Agregate digunakan untuk melakukan proses perhitungan secara cepat. Yang termasuk dalam fungsi aggregate antara lain SUM, MIN, MAX, AVG (Average), dan COUNT.

Fungsi-fungsi yang bisa digunakan dalam SQL adalah:

- **MAX** → untuk mencari nilai maksimal dari suatu kolom

```
SELECT MAX(umur) FROM ms_karyawan;
```

- **MIN** → untuk mencari nilai minimal dari suatu kolom

```
SELECT MIN(umur) FROM ms_karyawan;
```

- **AVG** → untuk mencari nilai rata-rata

```
SELECT AVG(umur) FROM ms_karyawan;
```

- **SUM** → untuk mencari nilai jumlah

```
SELECT SUM(umur) FROM ms_karyawan;
```

- **COUNT** → untuk mencari nilai cacah

```
SELECT COUNT(kd_karyawan) FROM ms_karyawan;
```

- **LIKE** → untuk mencari karakter yang memenuhi syarat *LIKE*

```
SELECT nama_kary FROM ms_karyawan WHERE nama_kary
                LIKE %Ahmad%;
```

- **GROUP BY** → untuk mengelompokkan *record*

```
SELECT nama_kary, alamat_kary FROM ms_karyawan
                GROUP BY alamat_kary;
```

- **DISTINCT** → untuk meniadakan duplikasi hasil *record*

### 1.3.5. ARITMATIKA

Fungsi aritmatika digunakan untuk menghasilkan perhitungan yang diambil dari record atau perhitungan tersendiri. Yang termasuk dalam fungsi aritmatika antara lain penjumlahan (+), pengurangan (-), perkalian (*), pembagian (/), pembagian sisa (%).

Contoh:

```
SELECT 18+2 AS JUMLAH, 18-2 AS SELISIH, 18*2 AS
PERKALIAN, 18/2 AS PEMBAGIAN, 18%2 AS MODULUS;
```

## 1.4. SOAL LATIHAN

**Tabel Buku**

| KD_BUKU | JUDUL_BUKU | KD_KARANG | KD_TERBIT | JUMLAH |
|---|---|---|---|---|
| 21 | Kalkulus | 10 | 1 | 10 |
| 22 | Metode Numerik | 11 | 2 | 20 |
| 23 | Sistem Basis Data | 12 | 3 | 40 |
| 24 | Pengantar Teknologi Informasi | 12 | 3 | 41 |
| | | | row(s) 1 - 4 of 4 | |

Berdasarkan tabel di atas, jawab pertanyaan berikut menggunakan perintah query:

1) Dengan menggunakan perintah query, inputkan 1 data pada tabel buku. Data tersebut yaitu: Kode Buku = 27; Judul Buku = Informatika Sosial; Kode Pengarang = 12; Kode Penerbit = 3; Jumlah stok buku = 35;
2) Tampilkan judul buku yang jumlah bukunya lebih dari 30 buah.
3) Tampilkan jumlah buku yang diterbitkan oleh penerbit yang mempunyai kode 3.
4) Tampilkan judul buku yang jumlah bukunya terbanyak.

# BAB 2. SQL MULTI TABEL

## 2.1. KOMPETENSI DASAR

Setelah mempelajari bab ini mahasiswa:

1) Memahami penggunaan perintah query multi tabel menggunakan konsep WHERE-AND
2) Memahami penggunaan perintah query multi tabel menggunakan konsep JOIN

## 2.2. INDIKATOR

Setelah mempelajari bab ini mahasiswa:

1) Mampu menjelaskan dan mengimplementasikan perintah query WHERE-AND
2) Mampu menjelaskan dan mengimplementasikan perintah query JOIN

## 2.3. URAIAN MATERI

## 2.4. SQL menggunakan WHERE-AND

Didalam basis data relasional dimungkinkan untuk mengakses satu atau lebih tabel dalam suatu database dalam waktu bersamaan. Perintah query relasi bisa dilakukan dengan 2 macam perintah, yaitu WHERE-AND dan JOIN. Perintah JOIN sendiri mempunyai beberapa fungsi lagi antara lain INNER JOIN, OUTER JOIN, LEFT JOIN, RIGHT JOIN.

Contoh sintak dasar perintah query WHERE-AND:

```
SELECT nama_tabel1.nama_field1, nama_tabel2.nama_field2
FROM nama_tabel1, nama_tabel2, nama_tabel3
WHERE nama_tabel1.key1=nama_tabel2.key2 AND
nama_tabel3.key3=nama_tabel2.key2;
```

## 2.5. SQL menggunakan JOIN

Perintah query JOIN umumnya digunakan untuk menampilkan data dari beberapa tabel (relasi tabel). terdapat berbagai macam tipe JOIN antara lain Inner Join, Left Join, Right Join. Sebagai bahan pembelajaran, perhatikan relasi tabel dibawah ini.

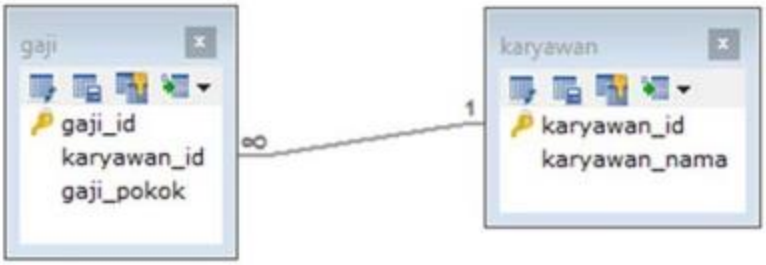



Table Karyawan:

| karyawan_id | karyawan_nama |
|---|---|
| 001 | Santi |
| 002 | Dewi |
| 003 | Joko |
| 004 | Putri |
| 005 | Nadya |
| 006 | Febri |

Table Gaji:

| gaji_id | karyawan_id | gaji_pokok |
|---|---|---|
| GJ001 | 001 | 2000000 |
| GJ002 | 003 | 2500000 |
| GJ003 | 002 | 2200000 |
| GJ004 | 004 | 1900000 |
| GJ005 | 005 | 2000000 |

### 2.5.1. Inner Join

Selain menggunakan perintah WHERE-AND, INNER JOIN merupakan jenis join yang paling umum digunakan untuk menampilkan record dari beberapa tabel. WHERE AND dan INNER JOIN menghasilkan record relasi yang sama. Perbedaan perintah tersebut terletak pada relasinya yaitu ON (untuk JOIN) dan AND (untuk WHERE-AND). Tipe join ini akan mengambil semua row dari table asal dan table tujuan dengan kondisi nilai key yang terkait saja, dan jika tidak maka row tersebut tidak akan muncul. Kalau tidak terdapat kondisi key terkait antar table, maka semua row dari kedua table dikombinasikan.

Contoh:

```
SELECT * FROM karyawan INNER JOIN gaji ON
karyawan.karyawan_id=gaji.karyawan_id;
```

Hasil eksekusi query diatas adalah

| gaji_id | karyawan_id | gaji_pokok | karyawan_id | karyawan_nama |
|---|---|---|---|---|
| GJ001 | 001 | 2000000 | 001 | Santi |
| GJ002 | 003 | 2500000 | 003 | Joko |
| GJ003 | 002 | 2200000 | 002 | Dewi |
| GJ004 | 004 | 1900000 | 004 | Putri |
| GJ005 | 005 | 2000000 | 005 | Nadya |

Baris data yang ditampilkan sebanyak 5 baris data, karena INNER JOIN hanya memperhitungkan kondisi key yang terkait antara tabel karyawan dengan tabel gaji. Sedangkan

karyawan dengan karyawan_id=’006’ tidak ditampilkan karena tidak terkait dengan tabel gaji. Jika dibuat diagram venn-nya akan terlihat seperti gambar berikut:


### 2.5.2. Left Join

LEFT JOIN atau biasa juga dikenal dengan LEFT OUTER JOIN merupakan perintah join untuk menampilkan semua data sebelah kiri dari table yang di joinkan dan menampilkan data sebelah kanan yang cocok dengan kondisi join. Jika tidak ditemukan kecocokan, maka akan di set NULL secara otomatis.

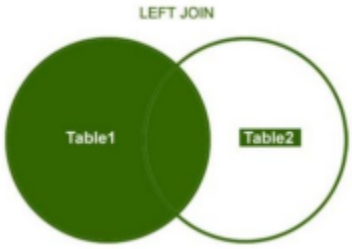
Contoh:

```
SELECT * FROM karyawan LEFT JOIN gaji ON
karyawan.karyawan_id=gaji.karyawan_id;
```

Perintah query di atas akan menghasilkan output sebagai berikut:

| karyawan_id | karyawan_nama | gaji_id | karyawan_id | gaji_pokok |
|---|---|---|---|---|
| 001 | Santi | GJ001 | 001 | 2000000 |
| 003 | Joko | GJ002 | 003 | 2500000 |
| 002 | Dewi | GJ003 | 002 | 2200000 |
| 004 | Putri | GJ004 | 004 | 1900000 |
| 005 | Nadya | GJ005 | 005 | 2000000 |
| 006 | Febri | *(NULL)* | *(NULL)* | *(NULL)* |

Banyaknya record yang ditampilkan sebanyak 6 record. Karena LEFT JOIN akan menampilkan semua table sebelah kiri dari kondisi join yaitu table karyawan. Semua data pada table karyawan akan ditampilkan, meskipun tidak ada kecocokan key pada table gaji. Jika dibuat diagram venn-nya akan terlihat seperti gambar berikut:



Selain kondisi diatas, LEFT JOIN juga bisa menampilkan data yang hanya kondisi key pada table tamu (*foreign key*) kosong (*NULL*).

Contoh:

```
SELECT * FROM karyawan LEFT JOIN gaji ON
karyawan.karyawan_id=gaji.karyawan_id
WHERE gaji.karyawan IS NULL;
```

Perintah query di atas akan menghasilkan output sebagai berikut:

| karyawan_id | karyawan_nama | gaji_id | karyawan_id | gaji_pokok |
|---|---|---|---|---|
| 006 | Febri | *(NULL)* | *(NULL)* | *(NULL)* |

Data yang ditampilkan hanya 1 record. Hal ini dikarenakan, hanya ada satu data yang belum memiliki kecocokan key pada table tamu. Untuk mempermudah anda memahami perbedaan antara kedua LEFT JOIN ini coba perhatikan diagram venn berikut:


Dengan melihat perbedaan dari diagram venn tersebut, maka Anda akan mudah memahami bagaimana left join ini bekerja. Ingat LEFT JOIN ini sangat penting untuk anda pahami, karena disaat Anda mulai mengerjakan project yang cukup kompleks, maka anda akan banyak berkutat dengan left join ini.

Contoh pada kasus diatas, hanya dengan memanfaatkan left join kita bisa menampilkan semua data karyawan yang sudah ada gajinya dan siapa yang belum ada gajinya. Selain itu anda juga dapat menampilkan semua data karyawan yang belum ada gajinya dengan fungsi LEFT JOIN WHERE NULL.

### 2.5.3. Right Join

Kebalikan dari LEFT JOIN adalah RIGHT JOIN, atau biasa juga dikenal dengan RIGHT OUTER JOIN. RIGHT JOIN akan menampilkan semua data yang ada di table sebelah kanan dan

mencari kecocokan key pada table sebelah kiri. Jika tidak ditemukan kecocokan, maka akan di set NULL secara otomatis pada table sebelah kiri.

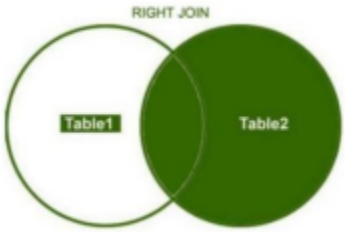
Contoh:

```
SELECT * FROM karyawan RIGHT JOIN gaji ON
karyawan.karyawan_id=gaji.karyawan_id;
```

Query di atas akan menampilkan output seperti gambar berikut:

| gaji_id | karyawan_id | gaji_pokok | karyawan_id | karyawan_nama |
|---|---|---|---|---|
| GJ001 | 001 | 2000000 | 001 | Santi |
| GJ002 | 003 | 2500000 | 003 | Joko |
| GJ003 | 002 | 2200000 | 002 | Dewi |
| GJ004 | 004 | 1900000 | 004 | Putri |
| GJ005 | 005 | 2000000 | 005 | Nadya |
| *(NULL)* | *(NULL)* | *(NULL)* | 006 | Febri |

Pada output diatas, anda dapat melihat bahwa terdapat NULL pada table sebelah kiri. Hal ini dikarenakan tidak ditemukan kecocokan key diantara kedua table. Untuk lebih mudah memahaminya, perhatikan diagram venn berikut:




Selain kondisi diatas, RIGHT JOIN juga bisa menampilkan data yang hanya kondisi key pada table tamu (*foreign key*) kosong (*NULL*).

Contoh:

```
SELECT * FROM karyawan RIGHT JOIN gaji
ON karyawan.karyawan_id=gaji.karyawan_id
WHERE gaji.karyawan IS NULL;
```

Query di atas akan menghasilkan output seperti gambar berikut:

| gaji_id | karyawan_id | gaji_pokok | karyawan_id | karyawan_nama |
|---|---|---|---|---|
| *(NULL)* | *(NULL)* | *(NULL)* | 006 | Febri |

Data yang ditampilkan hanya 1 record. Hal ini dikarenakan, hanya ada satu data yang belum memiliki kecocokan key pada table tamu. Untuk mempermudah anda memahami

perbedaan antara kedua RIGHT JOIN ini coba perhatikan diagram venn berikut dan bandingkan dengan diagram venn sebelumnya:


## 2.6. SOAL LATIHAN

Berdasarkan soal latihan 1.4, tambahkan kembali 2 tabel untuk membentuk relasi tabel. Tabel tersebut antara lain:

**Tabel Pengarang**

| KD_KARANG | PENGARANG | ALAMAT |
|---|---|---|
| 12 | Adityo PW | Sleman |
| 10 | Suparman | Sleman |
| 11 | Irwanto | JOgja |
| | row(s) 1 - 3 of 3 | |

**Tabel Penerbit**

| KD_TERBIT | PENERBIT | ALAMAT |
|---|---|---|
| 1 | GaYa Media | Jl. Aspal |
| 2 | Ditanam Press | Jl. Aspal Apik |
| 3 | Ty0 Publisher | Jl. Brongkalan Boto |
| | row(s) 1 - 3 of 3 | |

Berdasarkan 3 tabel yang terbentuk (Tabel Buku, Tabel Pengarang, Tabel Penerbit), jawab pertanyaan berikut menggunakan perintah query.

1) Tampilkan judul buku, nama pengarang dan nama penerbit dari masing-masing buku menggunakan query WHERE-AND, INNER JOIN, LEFT JOIN, dan RIGHT JOIN. Kemudian perhatikan perbedaan dari masing-masing hasilnya selanjutnya jelaskan perbedaan hasil tersebut.
2) Tampilkan nama pengarang dan judul buku yang jumlah bukunya lebih dari 30 buah.
3) Tampilkan judul buku yang ditulis oleh pengarang bernama AdityoPW
4) Tampilkan jumlah buku yang di terbitkan oleh penerbit Ty0 Publisher.

# BAB 3. SUBQUERY

## 3.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami konsep Subquery pada MySQL
2) Memahami penggunaan Subquery pada MySQL

## 3.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa:

1) Mampu menjelaskan konsep Subquery pada MySQL
2) Mampu menjelaskan dan mengimplementasikan Subquery pada MySQL

## 3.3. URAIAN MATERI

Subquery adalah query didalam query. Artinya seleksi data berdasarkan dari hasil seleksi data yang telah ada. Sintak SQL nya sama dengan sintak SQL pada umumnya, hanya saja kondisi setelah WHERE atau from diikuti dengan query baru atau sub query. Sintak dasar Subquery SQL yaitu:

```
SELECT      field-1, field-2,……, field-n
FROM        nama_tabel
WHERE       kriteria(SELECT field-2,……,
            field-n FROM nama_tabel
            WHERE kriteria);
```

Di bawah ini adalah contoh penggunaan subquery:

```
SELECT * FROM t1 WHERE column1 = (SELECT
column1 FROM t2);
```

Dari contoh tersebut, **SELECT * FROM t1 ...** merupakan query yang paling luar (*outer query*), dan **SELECT column1 FROM t2)** merupakan *subquery*. Bisa dikatakan bahwa subquery adalah sekumpulan query yang ada di *outer query*, sehingga ada kemungkinan subquery terdapat dalam subquery lagi dan kemudian setumpukan subquery tersebut ada di sebuah query. Subquery bisa diibaratkan anaknya, dan *outer query* adalah induknya.

**Keuntungan utama dari subquery adalah**:

Subquery membuat query-query menjadi tersusun, sehingga ada kemungkinan untuk memisahkan / memberi batasan area untuk setiap bagian statement.

- Subquery menyediakan cara alternatif untuk menjalankan operasi-operasi yang dalam keadaan lain akan membutuhkan JOIN dan UNION yang kompleks.
- Subquery lebih mudah dibaca dibanding JOIN atau UNION yang kompleks.

Sebuah subquery bisa mengembalikan scalar (single value/satu nilai), single row (satu baris), single column (satu kolom), atau table (satu/lebih baris dari satu/lebih kolom). Ada beberapa batasan untuk tipe statement yang subquery bisa digunakan di dalamnya. Sebuah subqeury dengan **SELECT** yang biasa dapat berisikan: **DISTINCT**, **GROUP BY**, **ORDER BY**, **LIMIT**, joins, indexhints, **UNION**, comments, dan fungsi-fungsi lainnya. Sedangkan, outer statement (statement yang paling luar) yang memiliki subquery tidak hanya bisa menggunakan **SELECT** saja, tapi dia juga bisa menggunakan **INSERT**, **UPDATE**, **DELETE**, **SET**, atau **DO.**

### 3.3.1. Penggunaan Perbandingan untuk Subquery

Penggunaan operator perbandingan di query sudah umum digunakan sesuai dengan kebutuhannya. Operator perbandingan yang umum digunakan antara lain: =, >, <, >=, <=, <>, !=, ⇔. Perbandingan perintah Subquery juga biasa menggunakan operator LIKE.

### 3.3.2. Penggunaan Subquery SELECT setelah FROM

Subquery dengan statement SELECT pada umumnya digunakan setelah operator perbandingan. Akan tetapi sebenarnya statement SELECT Subquery juga bisa digunakan setelah FROM. Penggunaannya seperti di bawah ini:

```
SELECT . . . FROM (subquery) [AS] name . . .;
```

Tulisan [AS] name adalah wajib dicantumkan, karena setiap label setelah kata FROM harus memiliki nama. Setiap nama kolom di Subquery SELECT harus memiliki nama yang unik. Contoh penggunaan subquery setelah FROM seperti di bawah ini:

```
SELECT sb1, sb2, sb3
FROM (SELECT s1 AS sb1, s2 AS sb2, s3*2 AS sb3 FROM t1) AS sb
WHERE sb1 > 1;
```

Contoh lainnya penggunaan subquery setelah FROM yang ditambahkan dengan fungsi aritmatika akan menjadi seperti di bawah ini:

```
SELECT AVG(sum_column1)
FROM (SELECT SUM(column1) AS sum_column1 FROM t1 GROUP BY column1) AS t1;
```

Sedangkan, subquery (sum_column1) bisa dipanggil oleh outer query yang membutuhkan, contohnya untuk dihitung AVERAGE-nya seperti contoh di atas.

### 3.3.3. Subquery 1 Tabel

Pada prinsipnya Subquery bisa digunakan untuk menampilkan data dari 1 tabel maupun multi tabel. Sebagai contoh, perhatikan tabel buku berikut:

| KD_BUKU | JUDUL_BUKU | KD_KARANG | KD_TERBIT | JUMLAH |
|---|---|---|---|---|
| 21 | Kalkulus | 10 | 1 | 10 |
| 22 | Metode Numerik | 11 | 2 | 20 |
| 23 | Sistem Basis Data | 12 | 3 | 40 |
| 24 | Pengantar Teknologi Informasi | 12 | 3 | 41 |
| | | | row(s) 1 - 4 of 4 | |

Berdasarkan tabel buku di atas, terdapat pertanyaan yaitu Tampilkan judul buku yang jumlahnya paling banyak. Dari pertanyaan tersebut dapat dipahami bahwa diminta untuk menampilkan judul buku yang jumlahnya paling banyak, akan tetapi di pertanyaan tersebut tidak dituliskan kriteria berapa jumlah buku yang paling banyak, maka dengan demikian konsep Subquery yang akan digunakan adalah perlu dicari dulu berapa jumlah terbesar dari jumlah buku yang terdaftar, kemudian baru ditampilkan judul bukunya. Maka dengan demikian perintah Subquerynya adalah:

```
SELECT judul_buku FROM buku
WHERE jumlah = (SELECT MAX(jumlah) FROM buku);
```

### 3.3.4. Subquery 2 tabel

Perhatikan 2 tabel di bawah ini:

**Tabel Buku**

| KD_BUKU | JUDUL_BUKU | KD_KARANG | KD_TERBIT | JUMLAH |
|---|---|---|---|---|
| 21 | Kalkulus | 10 | 1 | 10 |
| 22 | Metode Numerik | 11 | 2 | 20 |
| 23 | Sistem Basis Data | 12 | 3 | 40 |
| 24 | Pengantar Teknologi Informasi | 12 | 3 | 41 |
| | | | row(s) 1 - 4 of 4 | |

**Tabel Pengarang**

| KD_KARANG | PENGARANG | ALAMAT |
|---|---|---|
| 12 | Adityo PW | Sleman |
| 10 | Suparman | Sleman |
| 11 | Irwanto | JOgja |
| | row(s) 1 - 3 of 3 | |

Berdasarkan 2 tabel di atas, terdapat pertanyaan, siapa nama pengarang yang jumlah bukunya paling banyak. Sama hal nya dengan pertanyaan sebelumnya, maka perlu dicari dulu jumlah terbanyaknya, kemudian baru direlasikan dengan tabel pengarang untuk menampilkan nama pengarangnya. Sehingga perintah Subquerynya adalah:

```
SELECT pengarang FROM buku, pengarang
WHERE buku.kd_karang=pengarang.kd_karang AND buku.jumlah =
(SELECT MAX(buku.jumlah) FROM buku);
```

## 3.4. SOAL LATIHAN

Buatlah tabel dengan komposisi seperti dibawah ini:

**Buku**(kd_buku, judul_buku, pengarang, penerbit, jml_buku)

**Petugas(**kd_petugas, nama_petugas, alamat_petugas**)**

**Pelanggan(**kd_pelanggan, nama_pelanggan, alamat_pelanggan**)**

**Pembelian(**kd_beli, kd_petugas, kd_pelanggan, tanggaljual, total_jual, total_item**)**

**DetailPembelian(**kd_detail, kd_beli, kd_buku, jml_beli**)**

Berdasarkan komposisi tabel diatas, jawablah pertanyaan berikut menggunakan perintah query.

1. Tampilkan nama pelanggan yang melakukan pembelian dengan total item terbanyak.
2. Tampilkan nama pelanggan yang mempunyai alamat yang sama.
3. Tampilkan judul buku yang dibeli paling banyak
4. Tampilkan judul buku yang jumlahnya diatas rata-rata

# BAB 4. MANAJEMEN USER DAN PRIVILEGES

## 4.1. KOMPETENSI DASAR

Setelah mempelajari Bab ini, mahasiswa:

1) Memahami pengertian user pada sebuah *database*
2) Memahami pengertian privileges pada sebuah *database*
3) Memahami cara pengaturan manajemen user, privileges, grant, dan revoke

## 4.2. INDIKATOR

Setelah mempelajari Bab ini, mahasiswa:

1) Mampu menjelaskan tentang pengertian user pada sebuah *database*
2) Mampu menjelaskan tentang privileges pada sebuah *database*
3) Mampu menjelaskan tentang cara pengaturan manajemen user, *privileges, grant, dan revoke* pada sebuah *database*

## 4.3. URAIAN MATERI

Pengguna basis data atau *database user* merupakan hal yang sangat penting dalam sebuah pengelolaan basis data. pada umumnya *database user* terdiri dari Database Administrator (DBA) dan *end user*. DBA merupakan pengguna yang paling utama dimana mempunyai hak akses paling banyak bahkan hampir semua hak akses pada database dimiliki oleh DBA. Hak akses yang dimaksud adalah manipulasi *database* dan data didalam *database* tersebut, contohnya antara lain *create database, create table, alter table, drop table, insert* data dalam *table*, *update* data dalam *table*, *delete* data dalam *table* dan lain sebagainya. Pengguna lain selain DBA adalah *end user*. *End User* hanya bisa mengakses dan mengelola *database* berdasarkan akses yang diberikan oleh DBA. Pemberian hak akses pada databse menggunakan perintah *grant*, sedangkan untuk membatalkan hak akses yang diberikan menggunakan perintah *revoke*. Dalam istilah query, perintah untuk mengatur pengguna basis data dinamakan *Data Control Language* (DCL).

### 4.3.1. User

Setiap DBMS menyediakan satu akun default yang nantinya bisa digunakan sebagai DBA. Pada DBMS MySQL, akun default yang biasa digunakan untuk DBA *login* yaitu menggunakan *username root* dengan *password* kosong (tanpa *password*). Jika salah satu sudah ada yang

bertindak sebagai DBA, maka untuk alasan keamanan sebaiknya *password user root* segera diganti dan dibuatkan pengguna baru dibawah *root*.

Dalam aplikasi database, khususnya yang berhubungan dengan aplikasi komputer, hak akses terkait operasi query jumlahnya sedikit dan tidak dilakukan secara bersama-sama. Oleh karena itu setiap perintah basis data, tidak perlu semuanya menggunakan *user* dengan level *root*.

Perintah dasar membuat *user* pada basis data khususnya MySQL, seperti terlihat pada sintak dibawah ini.

```
CREATE USER nama_user@nama_mesin IDENTIFIED by password
```

**Keterangan sintak:**

- Nama_user : nama user yang akan dibuat
- Nama_mesin : asal mesin atau host/IP Address user mengakses.
  Jika diakses menggunakan host/komputer itu sendiri (lokal), maka diisi "localhost". Namun jika diakses dari mesin lain bisa menggunakan "%".
- Password : password yang digunakan

**Contoh 1 penggunaan Sintak (localhost):**

```
CREATE USER admin@localhost IDENTIFIED by Adm1n321
```

Berdasarkan perintah diatas, artinya adalah membuat user dengan nama "Admin" dari mesin "Localhost" dan dengan password "Adm1n321".

**Contoh 2 penggunaan Sintak (mesin lain):**

```
CREATE USER admin@% IDENTIFIED by Adm1n321
```

Berdasarkan perintah diatas, artinya adalah membuat user dengan nama "Admin" dari mesin atau komputer manapun dan dengan password "Adm1n321". Tanda '%' mirip seperti LIKE Statement pada perintah SQL.

Jika ingin menampilkan perintah user yang sudah dibuat dengan perintah diatas, bisa menggunakan perintah di bawah ini:

```
SELECT User, Password, Host FROM user
```

User yang sudah dibuat juga bisa dihapus, dengan menggunakan perintah di bawah ini:

```
DROP user nama_user
```

### 4.3.2. Privilege

Privilege adalah hak akses atas sesuatu, sedangkan *System Privilege* adalah hak akses terhadap *database*. Hak akses untuk memanipulasi isi database objek adalah *object privilege*.

Perintah yang digunakan untuk memberiakn hak akses database terdiri dari:

a) Select_priv : memberikan hak user untuk menampilkan data.
b) Insert_priv : memberikan hak user untuk memasukan data.
c) Update_priv : memberikan hak user untuk merubah/memperbarui data
d) Delete_priv : memberikan hak user untuk menghapus data
e) Drop_priv : memberikan hak user untuk menghapus database, tabel, kolom.
f) Reload_priv : memberikan hak user untuk melakukan refresh server.
g) Shutdown_prive : memberikan hak akses user untuk mematikan daemon server MySQL.
h) Process_priv : memberikan hak user untuk melihat proses server MySQL.
i) File_priv : memberikan hak user untuk mengkonversi data dari database menjadi sebuah file.
j) Grant_priv : memberikan hak user untuk memberikan hak akses ke user lain.
k) References_priv :
l) Index_priv : memberikan hak user untuk membuat dan menghapus index pada database.
m) Alter_priv : memberikan hak user untuk merubah objek database.
n) Show_db_priv : memberikan hak user untuk melihat semua database.
o) Super_priv : memberikan hak user untuk melakukan pengaturan pada database.
p) Create_tmp_table_priv : memberikan hak user untuk membuat temporary tabel.
q) Lock_tables_priv : memberikan hak user untuk memberikan kunci (password) untuk mengakses suatu tabel.
r) Repl_slave_priv : memberikan hak user untuk membaca logs pada master replikasi.

### 4.3.3. Grant

Grant merupakan salah satu perintah SQL yang termasuk ke dalam kelompok DCL (Data Contor Language). Perintah grant digunakan untuk memberikan/mengijinkan user untuk mengakses tabel di dalam sebuah database. Terdapat beberapa perintah grant yang biasa digunakan sesuai dengan kebutuhannya. Perintah Grant hanya bisa dilakukan oleh administrator pada database tersebut.

Beberapa perintah dasar grant yaitu:

a) CREATE : memperbolehkan user membuat basis data/tabel
b) SELECT : memperbolehkan user melakukan pencarian data/menerima data
c) INSERT : memperbolehkan user menambah data baru pada sebuah tabel
d) UPDATE : memperbolehkan user merubah data baru pada sebuah tabel
e) DELETE : memperbolehkan user menghapus data baru pada sebuah tabel
f) DROP : memperbolehkan user menghapus seluruh basis data/tabel

Pemakaian perintah diatas menggunakan perintah SQL. Sintak SQL perintah grant sepeti terlihat di bawah ini.

a) Sintak Umum

```
GRANT hak_akses ON nama_database.nama_tabel TO
user@nama_host
```

b) Sintak pengaturan untuk hak akses penuh kepada suatu user (‘tyo’) dan pada host ‘localhost’

```
GRANT ALL PRIVILEGES ON *.* TO ‘tyo’@’localhost’
```

c) Sintak pengaturan untuk hak akses tertentu (select, insert) pada user ‘tyo’ dan pada server ‘localhost’

```
GRANT select,insert ON *.* TO ‘tyo’@’localhost’
```

### 4.3.4. Revoke

Sama halnya dengan Grant, Revoke juga merupakan salah satu perintah SQL yang termasuk ke dalam kelompok DCL (Data Contor Language). Kebalikan dengan perintah grant, revoke digunakan untuk membatalkan ijin user yang sudah diberikan dari perintah grant untuk mengakses tabel di dalam sebuah database. Penggunaan perintah revoke yang biasa digunakan seperti dibawah ini:

a) Sintak umum

```
REVOKE hak_akses ON nama_database.nama_tabel FROM
user@nama_host
```

b) Sintak menghapus hak akses penuh yang sudah diberikan untuk user ‘tyo’

```
REVOKE ALL PRIVILEGES ON *.* FROM ‘tyo’@’localhost’
```

## 4.4. STUDI KASUS

### 4.4.1. Membuat User Baru

Pembuatan user baru di database MySQL pada dasarnya bisa menggunakan perintah query *insert*, dimana akan dimasukan data user baru ke dalam tabel user. Tabel user merupakan tabel default dari MySQL yang menyimpan semua data-data user yang akan mengakses database MySQL. Perintah untuk membuat *user* baru ke dalam tabel user seperti dibawah ini:

```
CREATE USER tyo@localhost IDENTIFIED by ty0321
```

Perintah diatas menunjukkan bahwa membuat user baru dengan nama ‘tyo’ yang mengakses database secara ‘*localhost’*, password loginnya adalah ‘ty0321’.

Setelah perintah menambahkan user sudah berhasil dieksekusi, selanjutnya gunakan perintah ‘***Flush Privileges’***. Perintah ini digunakan untuk me-reload, memerintahkan server untuk membaca ulang hak akses.

### 4.4.2. Memberikan Hak Akses User

Jika user sudah berhasil dibuat, selanjutnya diberikan wewenang/hak akses kepada user baru tersebut. Pemberian hak akses menggunakan perintah query update. Contoh sintaknya seperti dibawah ini:

```
UPDATE user SET select_priv=’y’,
        insert_priv=’y’,
        update_priv=’y’,
        delete_priv=’y’,
        create_priv=’y’,
         drop_priv=’y’,
         alter_priv=’y’
       WHERE user=’tyo’;
```

Berdasarkan perintah query diatas, menunjukkan bahwa diberikan hak akses yang terdiri dari pencarian data (*select_priv*), menambahkan data (*insert_priv*), merubah data (*update_priv*), hapus data (*delete_priv*), membuat tabel (*create_priv*), hapus tabel (*drop_priv*), dan rubah struktur tabel (*alter_priv*).

## 4.5. SOAL LATIHAN

1) Buatlah beberapa user dengan hak akses berbeda-beda sesuai dengan jenis-jenis hak akses dalam basis data. Ketentuan hak akses terdiri dari:
   a) User hanya bisa melihat data
   b) User bisa melihat dan menambah data
   c) User hanya bisa memperbaiki data
   d) User bisa memperbaiki dan menghapus data
   e) Use rmemiliki semua hak akses
2) Uji masing-masing user dengan koneksi ke MySQL menggunakan SQLYog. Pastikan user dengan privileges dapat melihat data, tidak bisa melakukan penambahan, perbaikan dan pengapusan data.

# BAB 5. VIEW

## 5.1. KOMPETENSI DASAR

Setelah mempelajari Bab ini, mahasiswa:

1) Memahami pengertian dan fungsi view pada sebuah DBMS
2) Memahami pengertian dan fungsi trigger pada sebuah DBMS

## 5.2. INDIKATOR

Setelah mempelajari Bab ini, mahasiswa:

1) Mampu menjelaskan tentang pengertian view pada sebuah DBMS
2) Mampu memecahkan permasalahan database menggunakan view
3) Mampu menjelaskan tentang pengertian trigger pada sebuah DBMS
4) Mampu memecahkan permasalahan database menggunakan trigger

## 5.3. URAIAN MATERI

### 5.3.1. View

*View* merupakan salah satu objek basis data yang merepresentasikan sub himpunan dari data yang berasal dari satu atau lebih tabel. *View* adalah perintah *query* yang disimpan pada basis data dengan suatu nama tertentu sehingga bisa digunakan setiap saat untuk melihat data tanpa menuliskan ulang *query* tersebut. *View* bisa disebut juga dengan tabel virtual. Pembuatan *view* digunakan untuk mempermudah penulisan *query*, menyembunyikan beberapa kolom yang bersifat rahasia serta untuk mempercepat menampilkan data.

Pada pembahasan ini perintah *view* terdiri dari *create*, *replace*, *drop*, dan *using*. Jika perintah *View* digunakan untuk menampilkan data dari beberapa tabel, lain halnya jika ingin menyimpan data ke beberapa tabel, yaitu menggunakan *Stored Procedure*. *Stored Procedure* merupakan perintah *query* yang digunakan untuk menyimpan, merubah, dan menghapus data yang disimpan di dalam basis data dengan suatu nama tertentu sehingga bisa dipanggil setiap saat.

#### 5.3.1.1. Create View

Perintah untuk membuat view diawali dengan "*Create View*" kemudian dilanjutkan nama view, kemudian nama kolom, nama tabel dan disertakan kondisi jika diperlukan. Perintah pembuatan view seperti dibawah ini:

```
CREATE VIEW nama_view AS
SELECT kolom_1, kolom_2, kolom_n
FROM nama_tabel
WHERE kondisi;
```

#### 5.3.1.2. Replace View

View yang sudah dibuat, bisa dilakukan perubahan berdasarkan kebutuhan. Misalkan menambahkan kolom, menambah tabel, dan menambahkan filter pada *where clause*. Untuk merubah view menggunakan sintak seperti dibawah ini:

```
CREATE OR REPLACE VIEW nama_view AS
SELECT kolom_1, kolom_2, kolom_n
FROM nama_tabel
WHERE kondisi;
```

#### 5.3.1.3. Drop View

Selain bisa dirubah, view yang sudah dibuat bisa dihapus jika sudah tidak dibutuhkan. Untuk menghapus view menggunakan sintak seperti dibawah ini:

```
DROP VIEW nama_view;
```

Penggunaan perintah *drop view* seperti di bawah ini:

```
DROP VIEW v_buku;
```

## 5.4. STUDI KASUS

### 5.4.1. Create View

Pada dasarnya *view* merupakan tabel virtual yang terbentuk dari satu atau beberapa tabel yang ada dalam basis data. Pada studi kasus ini akan dijelaskan pembuatan *view* untuk 1 tabel dan 2 tabel. Contoh penggunaan sintak *create view* untuk 1 tabel yaitu:

```
CREATE VIEW v_buku AS
SELECT kd_buku, judul_buku, jml_buku
FROM buku
WHERE jml<=3;
```

Sintak create *view* di atas menunjukkan pembuatan tabel virtual dari tabel buku yang berisi data buku yang jumlahnya kurang dari 3 buah. Dengan demikian jika ingin mengetahui jumlah buku yang kurang dari 3 cukup memanggil “v_buku” saja, tidak perlu membuat perintah *query select* lagi.

Selanjutnya, contoh penggunaan sintak *create view* dua tabel, yaitu:

```
CREATE VIEW v_kategoribuku AS
SELECT kd_buku, judul_buku, namakategori
FROM buku, kategori
WHERE kategori.kd_kategori=buku.kd_kategori;
```

Sintak create *view* di atas menunjukkan pembuatan tabel virtual dari tabel buku dan tabel kategori yang berisi data buku berdasarkan kategorinya. Dengan demikian jika ingin mengetahui data buku beserta kategorinya, cukup memanggil "v_kategoribuku" saja, tidak perlu membuat perintah *query select* lagi.

### 5.4.2. Replace View

Berdasarkan view yang sudah dibuat di atas (v_kategoribuku), misalkan akan ditambahkan kondisi yaitu jumlah buku yang kurang dari 3 buah. Sehingga perintah viewnya adalah:

```
CREATE OR REPLACE VIEW v_kategoribuku AS
SELECT kd_buku, judul_buku, namakategori
FROM buku, kategori
WHERE  kategori.kd_kategori=buku.kd_kategori,
buku.jml_buku<=3;
```

## 5.5. SOAL LATIHAN

Buatlah basis data penjualan buku yang terdiri dari tabel:

**Buku**(kd_buku, judul_buku, kd_pengarang, kd_penerbit, stok)

**Pengarang**(kd_pengarang, nama_pengarang, alamat_pengarang)

**Penerbit**(kd_penerbit, nama_penerbit, alamat_penerbit)

**Petugas(**kd_petugas, nama_petugas, alamat_petugas**)**

**Pelanggan(**kd_pelanggan, nama_pelanggan, alamat_pelanggan**)**

**Penjualan(**kd_jual, kd_petugas, kd_pelanggan, tanggaljual, total_jual, total_item**)**

**DetailPenjualan(**kd_detail, kd_jual, kd_buku, jml_beli**)**

Berdasarkan tabel yang sudah dibuat, selanjutnya buatlah view untuk menampilkan data dengan ketentuan:

5. Data stok buku berdasarkan penerbit
6. Data jumlah buku berdasarkan pengarang
7. Data penjualan buku yang dilayani oleh petugas tertentu
8. Data buku yang dibeli oleh oleh pelanggan tertentu

# BAB 6. TRIGGER

## 6.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami pengertian dan fungsi trigger pada MySQL
2) Memahami penggunaan trigger pada MySQL

## 6.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa:

1) Mampu menjelaskan tentang pengertian dan fungsi trigger pada MySQL.
2) Mampu mengimplementasikan trigger pada MySQL.

## 6.3. URAIAN MATERI

### 6.3.1. Pengertian Trigger

*Trigger* adalah object basis data yang berhubungan langsung dengan tabel dan akan aktif apabila suatu kejadian (*event*) terjadi pada tabel tersebut. *Event* adalah suatu proses yang terdiri dari INSERT, UPDATE, dan DELETE pada sebuah tabel. Sebagai contoh apabila ada suatu proses *query Insert* ke tabel 1 maka sebelum (*BEFORE*) atau sesudah (*AFTER*) suatu proses *query* INSERT dijalankan, *trigger* akan beraksi jika dihubungkan dengan tabel 1 tersebut. Beberapa *event* yang bisa digunakan untuk mengeksekusi trigger yaitu:

a) **Before Insert**: dijalankan ketika data dimasukkan dalam tabel
b) **After Insert**: dijalankan ketika data masuk ke dalam tabel
c) **Before Update**: dijalankan sebelum proses update data
d) **After Update**: dijalankan setelah proses update data
e) **Before Delete**: dijalankan sebelum proses delete data
f) **After Delete**: dijalankan setelah proses delete data

Sintak dasar penulisan trigger seperti terlihat dibawah ini:

```
CREATE TRIGGER nama_trigger [BEFORE|AFTER] [INSERT|UPDATE|DELETE]
ON nama_tabel FOR EACH ROW (statement)
```

Keterangan:

a) **Nama_trigger**: berisi nama trigger yang akan digunakan
b) **Trigger time**: kapan eksekusi trigger (Before / After)
c) **Trigger event**: proses yang terjadi (Insert / Update / Delete)

d) **Nama_tabel**: berisi nama tabel yang akan di *trigger*

e) **Statement**: berisi perintah SQL yang akan diproses. Jika perintah lebih dari satu maka gunakan dalam blok statement BEGIN….END.

### 6.3.2. Insert Trigger

*Insert trigger* merupakan salah satu perintah input data yang terjadi pada suatu tabel dan memicu proses input data pada tabel lain. Perintah trigger ini nantinya juga bisa digunakan sebagai history penambahan data pada sebuah basis data produk. Contoh perintah trigger yaitu:

```
DELIMITER
CREATE TRIGGER hist_barang
AFTER INSERT ON barang FOR EACH ROW
BEGIN
 INSERT INTO loghist_barang SET
 id_barang=loghist_idbarang,
 nama=loghist_nama,
 harga=loghist_harga,
 diskon=loghist_diskon,
 waktu_input=NOW();
END;
DELIMITER;
```

Perintah di atas, menunjukkan bahwa akan terjadi proses input data barang ke dalam tabel loghist_barang setelah terjadi proses input data barang ke dalam tabel barang. Proses tersebut terlihat dari perintah AFTER INSERT pada tabel barang dan selanjutnya terjadi proses INSERT INTO juga pada tabel loghist_barang.

### 6.3.3. Update Trigger

Sama halnya dengan *insert*, *update* trigger merupakan salah satu perintah ubah data yang terjadi pada suatu tabel dan memicu proses input data pada tabel lain. Perintah trigger ini nantinya juga bisa digunakan sebagai history perubahan data pada sebuah basis data produk. Contoh perintah trigger yaitu:

```
DELIMITER
CREATE TRIGGER histupdate_barang
BEFORE UPDATE ON barang FOR EACH ROW
BEGIN
 INSERT INTO loghist_ubahbarang SET
 id_barang=loghist_idbarang,
 nama=loghist_nama,
 harga=loghist_harga,
 diskon=loghist_diskon,
 waktu_update=NOW();
END;
DELIMITER;
```

Perintah di atas, menunjukkan bahwa akan terjadi proses input data barang ke dalam tabel loghist_ubahbarang setelah terjadi proses input data barang ke dalam tabel barang. Proses tersebut terlihat dari perintah BEFORE UPDATE pada tabel barang dan selanjutnya terjadi proses INSERT INTO juga pada tabel loghist_ubahbarang.

### 6.3.4. Delete Trigger

Delete trigger merupakan salah satu perintah input data yang terjadi pada suatu tabel dan memicu proses input data pada tabel lain. Perintah trigger ini nantinya juga bisa digunakan sebagai history penambahan data pada sebuah basis data produk. Contoh perintah trigger yaitu:

```
DELIMITER
CREATE TRIGGER hist_barang
BEFORE DELETE ON barang FOR EACH ROW
BEGIN
 INSERT INTO loghist_hapusbarang SET
 id_barang=loghist_idbarang,
 nama=loghist_nama,
 harga=loghist_harga,
 diskon=loghist_diskon,
 waktu_input=NOW();
END;
DELIMITER;
```

Perintah di atas, menunjukkan bahwa akan terjadi proses input data barang ke dalam tabel loghist_hapusbarang setelah terjadi proses hapus data barang ke dalam tabel barang. Proses tersebut terlihat dari perintah BEFORE DELETE pada tabel barang dan selanjutnya terjadi proses INSERT INTO juga pada tabel loghist_hapusbarang.

### 6.3.5. Menghapus Trigger

Untuk menghapus trigger yang sudah dibuat, bias menggunakan perintah:

```
DROP TRIGGER nama_trigger;
```

## 6.4. SOAL LATIHAN

Dengan menggunakan perintah trigger buatlah tambahan field untuk menyimpan waktu (tanggal) pada proses:

1. pendaftaran anggota pada tabel anggota
2. pendataan buku pada tabel buku

# BAB 7. STORED PROCEDURE

## 7.1. KOMPETENSI DASAR

Setelah mempelajari bab ini mahasiswa:

1) Memahami pengertian Stored Procedure pada sebuah DMBS.
2) Memahami fungsi dan penggunaan Stored Procedure pada sebuah DBMS.

## 7.2. INDIKATOR

Setelah mempelajari bab ini mahasiswa:

1) Mampu menjelaskan pengertian Stored Procedure pada sebuah DMBS.
2) Mampu mengimplementasikan fungsi dan penggunaan Stored Procedure pada sebuah DBMS.

## 7.3. URAIAN MATERI

*Stored Procedure* merupakan serangkaian *sub routine* (baris program) yang dibuat dan disimpan dalam basis data. *Stored Procedure* memberikan banyak keuntungan bagi developer terutama dari sisi sekuritas basis data dan pengembangan *software* yang *multiplatform* serta membutuhkan banyak *teamwork.*

*Stored Procedure* berisi kumpulan sintak-sintak SQL yang menghasilkan *output* tertentu. *Stored Procedure* bisa mengurangi *traffic network* dan *overhead*. Pengelolaan data tidak dilakukan disisi aplikasi, melainkan disisi *database server*. Penggunaan *Stored Procedure* antara lain untuk proses manipulasi data (*insert*, *update*, dan *delete*), selain itu juga bisa membatasi akses langsung ke tabel dari basis data.

Perintah *stored procedure* pada umumnya digunakan untuk proses *insert*, *update* dan *delete* pada sebuah DBMS. *Stored Procedure* memiliki variable parameter yang dibedakan menjadi:

1) **IN**: Variabel parameter hanya dapat digunakan untuk menerima input saja. IN menjadi nilai default dari variabel parameter
2) **OUT**: Variabel parameter hanya dapat digunakan untuk menyimpan hasil output saja.
3) **INOUT**: Variabel parameter yang digunakan untuk menerima input dan menyimpan hasil output.

## 7.4. Perintah Dasar Stored Procedure

Setiap perintah *stored procedure* selalu diawali dan diakhiri dengan fungsi delimiter. Delimiter adalah sebuah tanda batas akhir dari suatu perintah bagi SQL untuk mengeksekusi sebuah perintah query. Secara default delimiter pada sebuah perintah SQL adalah tanda titik koma (;), namun ada juga yang menggunakan garis pipa (|) sebagai batas perintah. Penggunaannya sangat sederhana, sebelum mendefinisikan objek tersebut kita gunakan statement "DELIMITER"

```
DELIMITER |
CREATE PROCEDURE storedprocedure_name ([proc_parameter[....]])
[characteristic ……] routine_body
```

diikuti tanda pemisah baru. Setelah di akhir pendefinisian kita kembalikan delimiter lagi kepada tanda titik koma. Sintak dasar perintah stored procedure yaitu:

Keterangan:

- **Proc_parameter** : parameter stored procedure yang terdiri dari IN, OUT, INOUT.
- **Routine_body** : kumpulan perintah stored procedure yang terdiri dari peritnah SQL.

## 7.5. Studi Kasus Pembuatan Stored Procedure

### 7.5.1. Perintah Stored Procedure untuk Insert

Sebagaimana yang sudah dijelaskan diatas, bahwa *stored procedure* merupakan kumpulan perintah SQL yang dibuat untuk mempermudah proses yang digunakan di beberapa tempat, sehingga dalam penggunaannya hanya memanggil nama *stored procedure*-nya saja. Sintak dasar *stored procedure insert*, yaitu:

```
DELIMITER |
CREATE PROCEDURE tambahbarang
 (IN kd_brg char(10), IN nm_brg varchar(40), IN hrg_sat int(5),
IN sat varchar(5), IN stok int(3))

Begin
 INSERT INTO barang (kode_barang, nama_barang, harga_satuan,
 satuan, jml_barang) values (kd_brg, nm_brg, hrg_sat, sat,
 stok)
End |
DELIMITER;
```

### 7.5.2. Perintah Stored Procedure untuk update

Sama halnya *stored procedure insert*, perintah *stored procedure* untuk *update* yaitu:

```
DELIMITER |
CREATE PROCEDURE ubahbarang
 (IN kd_brg char(10), IN nm_brg varchar(40), IN hrg_sat int(5),
IN sat varchar(5), IN stok int(3))

Begin
 UPDATE barang SET nm_brg=@nama_barang, hrg_sat=@harga_satuan,
 sat=@satuan, stok=@jml_barang WHERE kd_brg=@kode_barang;
End |
DELIMITER;
```

### 7.5.3. Perintah Stored Procedure untuk Delete

Sama halnya *stored procedure update*, perintah *stored procedure* untuk *delete* yaitu:

```
DELIMITER |
CREATE PROCEDURE hapusbarang
 (IN kd_brg char(10), IN nm_brg varchar(40), IN hrg_sat int(5),
IN sat varchar(5), IN stok int(3))

Begin
 Delete FROM barang WHERE kd_brg=@kode_barang;
End |
DELIMITER;
```

## 7.6. SOAL LATIHAN

Perhatikan tabel berikut:

| KD_BUKU | JUDUL_BUKU | KD_KARANG | KD_TERBIT | JUMLAH |
|---|---|---|---|---|
| 21 | Kalkulus | 10 | 1 | 10 |
| 22 | Metode Numerik | 11 | 2 | 20 |
| 23 | Sistem Basis Data | 12 | 3 | 40 |
| 24 | Pengantar Teknologi Informasi | 12 | 3 | 41 |
| | | | row(s) 1 - 4 of 4 | |

Buatlah perintah Stored Procedure untuk menambah data, merubah data, dan menghapus data.

# BAB 8. FUNCTION

## 8.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami pengertian function menggunakan database MySQL.
2) Memahami konsep function menggunakan database MySQL

## 8.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa:

1) Mampu menjelaskan tentang pengertian function menggunakan database MySQL.
2) Mampu menjelaskan dan mengimplementasikan function menggunakan database MySQL.

## 8.3. URAIAN MATERI

Stored Procedure dan Store Function merupakan routine yang diasosiasikan secara default ke dalam database aktif. Untuk dapat mengasosiasikan routine secara eksplisit dengan database lain, perlu dibuat routine dengan format: db_name.sp_name.

Sintak dasar Function dalam database MySQL, yaitu:

```
DELIMITER |
CREATE FUNCTION sp_name ([func_parameter[. . . . . .]])
      RETURNS type
      [characteristic . . .] routine_body
      Proc_parameter:
      [IN | OUT | INOUT] param_name type
      Func_parameter:
      Param_name type
Type:
      Any valid MySQL data type
Characteristic:
Language SQL
| [NOT] DETERMNISTIC
| (CONTAINS SQL|NOSQL|READS SQL DATA|MODIFIES SQL DATA)
| SQL SECURITY {DEFINER|INVOKER}
| COMMENT 'string'
Routine body:
      Valid SQL procedure statement or statements
```

Keterangan:

- **sp_name**: Nama routine yang akan dibuat
- **proc_parameter**: spesifikasi parameter sebagai IN, OUT, atau INOUT valid hanya untuk PROCEDURE. (parameter FUNCTION selalu sebagai parameter IN)
- **returns**: Klausa RETURNS dispesifikan hanya untuk suatu FUNCTION. Klausa ini digunakan untuk mengembalikan tipe fungsi dan routine_body harus berisi suatu statemen nilai RETURNS.
- **Comment**: Klausa COMMENT adalah suatu ekstensi MySQL, dan mungkin digunakan untuk mendeskripsikan stored Procedure. Informasi ini ditampilkan dengan statemen SHOW CREATE PROCEDURE dan SHOW CREATE FUNCTION.

Contoh penggunaan Function untuk proses diskon.

```
DELIMITER |
CREATE FUNCTION diskon(jumlah INT) RETURNS int(11)

Begin
 DECLARE potdiskon INT; CASE
  WHEN (jumlah >= 100) THEN SET diskon = 10;
  WHEN (jumlah >= 50 AND jumlah < 100) THEN SET diskon = 5;
  WHEN (jumlah >= 20 AND jumlah < 50) THEN SET diskon = 3;
 ELSE SET diskon = 0; END CASE;
RETURN potdiskon;
END//
```

Pemanggilan function diskon:

```
SELECT diskon('70');
```

Contoh lain function untuk menghitung volume segitiga

```
DELIMITER |
CREATE FUNCTION volumesegitiga(Panjang INT, lebar INT, tinggi
INT) RETURNS int(11)
DETERMINISTIC

Begin
 DECLARE volum INT;
 Set volum = Panjang*lebar*tinggi;
RETURN volum;
END//
```

Pemanggilan function volumesegitiga:

```
SELECT volumesegitiga(12,13,2);
```

## 8.4. SOAL LATIHAN

Buatlah perhitungan denda pengembalian buku menggunakan perintah function. Jumlah besaran denda adalah Rp. 1000,- perhari dihitung dari tanggal harus dikembalikan.

# BAB 9. EVENT

## 9.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami pengertian event pada database MySQL
2) Memahami penggunaan event pada database MySQL

## 9.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa:

1) Mampu menjelaskan tentang pengertian event pada database MySQL
2) Mampu menjelaskan dan mengimplementasikan event pada database MySQL.

## 9.3. URAIAN MATERI

Event pada database MySQL biasa digunakan sebagai scheduler yang bertugas mengelola penjadwalan dan menjalankan perintah tertentu secara terjadwal dan otomatis. Scheduler merupakan suatu jadwal yang sudah ditentukan dimana pada jadwal tersebut akan terjadi sebuah eksekusi suatu proses pada database yang sudah ditentukan. Cara kerjanya adalah seperti “Cron Job” pada system Linux dan Windows Task Scheduler pada system Windows.

Untuk memulai membuat event, sebaiknya dilihat dulu apakah feature schedulenya hidup atau tidak dengan cara mengetikkan pada jendela *command prompt* perintah seperti di bawah ini:

```
SHOW VARIABLES LIKE ‘event_scheduler’;
```

Hasil eksekusi perintah diatas kurang lebihnya adalah:

```
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| event_scheduler | ON    |
+-----------------+-------+
1 row in set (0.02 sec)
```

Jika muncul hasil seperti di atas, maka event scheduler MySQL dalam keadaan hidup atau ON.

Untuk menghidupkan atau mematikan event scheduler MySQL dengan menggunakan inisialisasi parameter angka 0 dan 1. Angka 0 artinya mematikan, angka 1 artinya menghidupkan. Misalkan untuk menghidupkan event scheduler perintahnya seperti di bawah ini:

```
SET GLOBAL event_scheduler=1;
```

### 9.3.1. Pembuatan Event Scheduler berdasarkan tanggal dan waktu

Perintah event berdasarkan tanggal dan waktu yang sudah ditentukan. Sintak dasarnya seperti terlihat di bawah ini.

```
CREATE EVENT event_name
ON SCHEDULE
AT {DATE AND TIME}
DO
{SQL COMMAND};
```

Contoh implementasi event berdasarkan tanggal dan waktu yang sudah ditentukan, misalkan membuat insert data ke tabel buku pada tanggal 1 Februari 2019 jam 07:20:27.

```
CREATE EVENT event_buku
ON SCHEDULE AT '2019-02-01 07:20:27'
DO
Insert into buku values (null,'sch_1',now());
```

Hasil eksekusi perintah di atas, kurang lebihnya akan seperti:

```
select * from tbl_sch;
+----+-------+---------------------+
| no | des   | time                |
+----+-------+---------------------+
|  1 | sch 1 | 2009-06-01 18:06:49 |
+----+-------+---------------------+
```

### 9.3.2. Pembuatan Event Scheduler berdasarkan pengulangan waktu

Selain perintah berdasarkan tanggal dan waktu, event scheduler bisa juga berdasarkan pengulangan waktu. Sintak dasarnya seperti terlihat pada gambar dibawah ini.

```
CREATE EVENT event_name
ON SCHEDULE
EVERY {x}
{SECOND|MINUTE|HOUR|DAY|MONTH|YEAR|WEEK}
DO
{SQL COMMAND};
```

Contoh implementasi event berdasarkan pengulangan waktu yang sudah ditentukan. Misalkan terjadi proses insert data ke tabel buku tiap menit. Sintak implementasinya seperti terlihat pada gambar dibawah ini.

```
CREATE EVENT buku_2
ON SCHEDULE EVERY 1 MINUTE
DO
Insert into buku values(null, 'sch2', now());
```

Hasil eksekusi perintah di atas, kurang lebihnya akan seperti:

```
+----+-------+---------------------+
| no | des   | time                |
+----+-------+---------------------+
|  2 | sch 2 | 2009-06-01 18:16:27 |
|  3 | sch 2 | 2009-06-01 18:17:27 |
|  4 | sch 2 | 2009-06-01 18:18:27 |
|  5 | sch 2 | 2009-06-01 18:19:27 |
|  6 | sch 2 | 2009-06-01 18:20:27 |
|  7 | sch 2 | 2009-06-01 18:21:27 |
|  8 | sch 2 | 2009-06-01 18:22:27 |
|  9 | sch 2 | 2009-06-01 18:23:27 |
| 10 | sch 2 | 2009-06-01 18:24:27 |
| 11 | sch 2 | 2009-06-01 18:25:27 |
+----+-------+---------------------+
10 rows in set (0.01 sec)
```

## 9.4. Ubah dan Hapus Event Scheduler

Selain perintah create, event scheduler juga bisa dilakukan operasi ubah dan hapus event. Ubah event yang dimaksud adalah mengubah nama event dan waktu eksekusi. Sintak dasar ubah event schedular seperti yang terlihat di bawah ini.

```
ALTER EVENT event_name
ON SCHEDULE schedule
RENAME to event_name2
DO
{SQL COMMAND};
```

Contoh penggunaan perintah alter event seperti terlihat seperti di bawah ini.

```
ALTER EVENT e_sch_2
ON SCHEDULE every 2 minute
RENAME to e_sch_3
DO
Insert into tbl_sch values (null, 'sch 3', now());
```

Untuk menghapus event scheduler, menggunakan perintah seperti di bawah ini.

```
DROP EVENT event_name;
```

## 9.5. SOAL LATIHAN

Buatlah event scheduler untuk mendata:

1. Tiap hari
2. Tiap minggu
3. Tiap bulan

# BAB 10. KONEKSI DATABASE

## 10.1. KOMPETENSI DASAR

Setelah mempelajari bab ini mahasiswa:

1) Memahami aplikasi berbasis GUI, Netbeans IDE.
2) Memahami pembuatan form dan komponen-komponen pada aplikasi Netbeans IDE.
3) Memahami cara menghubungkan basis data dengan aplikasi berbasis GUI.
4) Memahami cara memanggil data dari database dan menampilkan data di aplikasi Netbeans IDE.

## 10.2. INDIKATOR

Setelah mempelajari bab ini mahasiswa:

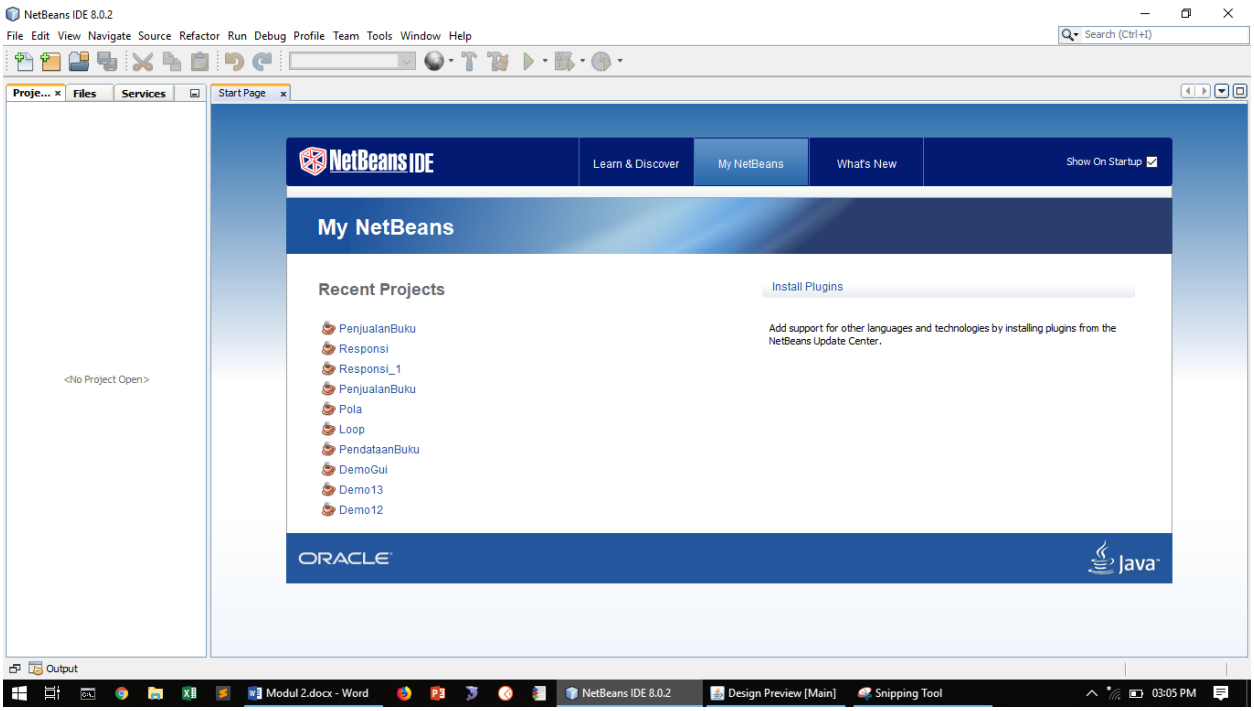
1) Mampu menjelaskan tentang aplikasi Netbeans IDE.
2) Mampu menjelaskan tentang pembuatan form dan komponen-komponen pada aplikasi Netbeans IDE.
3) Mampu menjelaskan cara menghubungkan basis data dengan aplikasi berbasis GUI.
4) Mampu menjelaskan tentang cara memanggil data dari database dan menampilkan data di aplikasi Netbeans IDE.

## 10.3. URAIAN MATERI

### 10.3.1. Netbeans IDE

Materi koneksi database pada modul ini, membahas aplikasi pemrograman basbasis GUI menggunakan Netbeans IDE 8.0. Netbeans IDE 8.0 merupakan salah satu aplikasi pengembang terintegrasi berbasis GUI yang bersifat open source. Sehingga gratis dan bisa dilakukan pengembangan untuk aplikasi desktop, seluler, dan web. Dengan adanya IDE (Integrated Development Environment) maka mendukung pengembangan aplikasi dalam berbagai Bahasa antara lain Java, HTML5, PHP, dan C++. IDE juga bisa berjalan pada beberapa system operasi antara lain Windows, Linux, Mac OS, dan beberapa sistem berbasis Unix lainnya.

IDE menyediakan dukungan teknologi JDK 7 dan perangkat tambahan Java terbaru. Ini adalah IDE pertama yang menyediakan dukungan untuk JDK 7, Java EE 7, dan JavaFX 2. IDE mendukung penuh Java EE menggunakan standar terbaru untuk Java, XML, layanan Web, dan SQL dan sepenuhnya mendukung GlassFish Server, implementasi refrensi Java EE. Tampilan halaman Netbeans IDE 8.0 terlihat pada seperti gambar dibawah ini.

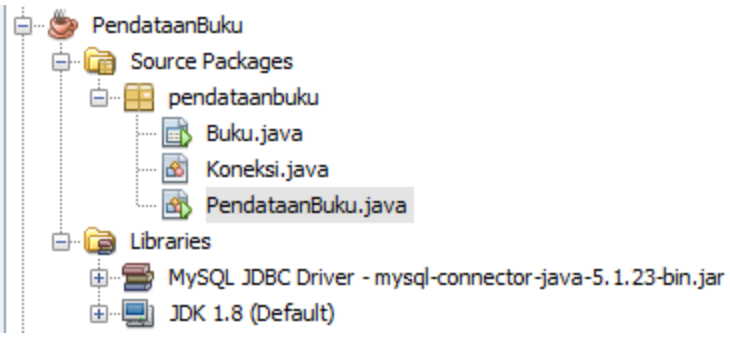



### 10.3.2. Membuat Form di Netbeans IDE

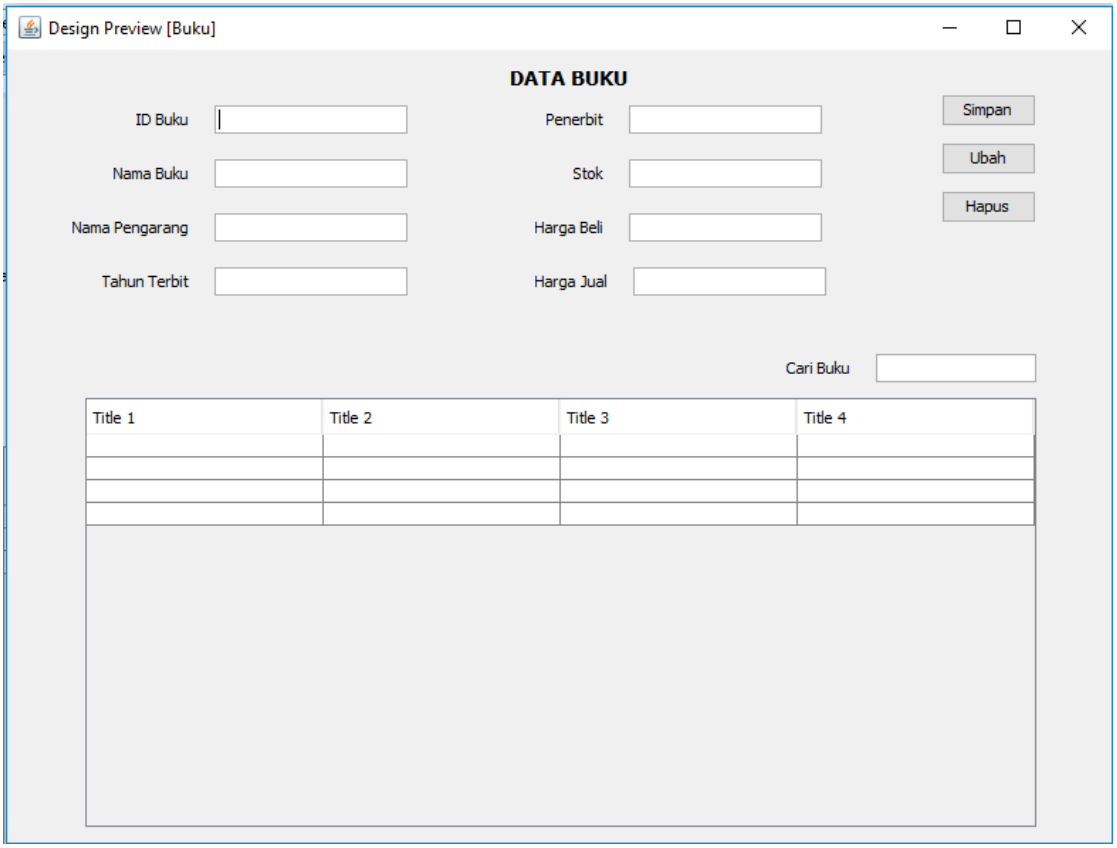
Sebelum memulai pembuatan form di Netbeans IDE, pastikan table yang akan dijadikan untuk database pemanggilan data sudah tersedia dan data sudah terisikan. Dalam hal ini database yang akan digunakan adalah MySQL dan menggunakan tabel buku, struktur tabelnya seperti terlihat pada gambar dibawah ini.

| Field | Type | | Null | Key | Default | | Extra |
|---|---|---|---|---|---|---|---|
| id_buku | varchar(20) | 11B | NO | PRI | (NULL) | 0K | |
| nama_buku | varchar(50) | 11B | NO | | (NULL) | 0K | |
| nama_pengarang | varchar(50) | 11B | NO | | (NULL) | 0K | |
| tahun_terbit | int(11) | 7B | NO | | (NULL) | 0K | |
| penerbit | varchar(30) | 11B | NO | | (NULL) | 0K | |
| stok | int(11) | 7B | NO | | (NULL) | 0K | |
| harga_beli | int(11) | 7B | NO | | (NULL) | 0K | |
| harga_jual | int(11) | 7B | NO | | (NULL) | 0K | |

Untuk membuat form pada Netbeans, sebelumnya buatlah nama project pada Netbeans IDE, misalkan dalam hal ini dibuat project dengan nama PendataanBuku. Kemudian buatlah struktur project PendataanBuku. Akan terbentuk Class PendataanBuku. Class Koneksi, dan jFrame Buku. Selanjutnya tambahkan Library MySQL JDBC Driver pada project. Sehingga terlihat seperti pada gambar dibawah ini.




Selanjutnya buatlah tampilan desain jFrame Buku. Pada tampilan jFrame Buku, tambahkan komponen jLabel, jTextField, jButton dan jTable. Sehingga menjadi seperti pada gambar dibawah ini.




jLabel digunakan untuk menuliskan label/nama kolom yang akan diisikan. jTextField digunakan untuk inputan data yang nantinya akan disimpan dalam database. jButton digunakan untuk tombol eksekusi proses, dalam hal ini proses yang terjadi adalah proses simpan data, proses ubah data, dan proses hapus data. jTable digunakan sebagai tempat untuk menampilkan data yang dipanggil dari database.

### 10.3.3. Menghubungkan basis data dengan form Netbeans IDE

Untuk menghubungkan basis data dengan jFrame Buku adalah dengan mengisikan source code pada Class Koneksi.java yang akan digunakan untuk menghubungkan Netbeans IDE dengan database MySQL. Isikan source code seperti pada gambar dibawah ini.

```
import java.sql.*;
/**
 *
 * @author Horizon
 */
public class Koneksi {
    public Connection koneksi;
    public Statement statement;
    public Koneksi() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (Exception e) {
            System.err.println("JDBC Driver tidak ditemukan : "+e.getMessage());
        }
        koneksi = null;
        try {
            koneksi = DriverManager.getConnection("jdbc:mysql://localhost/penjualan_buku","root","");
            statement = koneksi.createStatement();
        } catch (Exception e) {
            System.err.println("Koneksi DB Error : "+e.getMessage());
        }
        if (koneksi!=null) {
            System.out.println("Koneksi DB berhasil");
        }
    }
}
```

### 10.3.4. Menampilkan data pada aplikasi Netbeans IDE

Setelah diisikan source code untuk menghubungkan form dengan database, selanjutnya lakukan import package pada jFrame Buku dengan mengetikan source code seperti dibawah ini.

```
import java.sql.*;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

Kemudian buat objek koneksi dari Class Koneksi, dengan source code:

```
private Koneksi koneksi = new Koneksi();
```

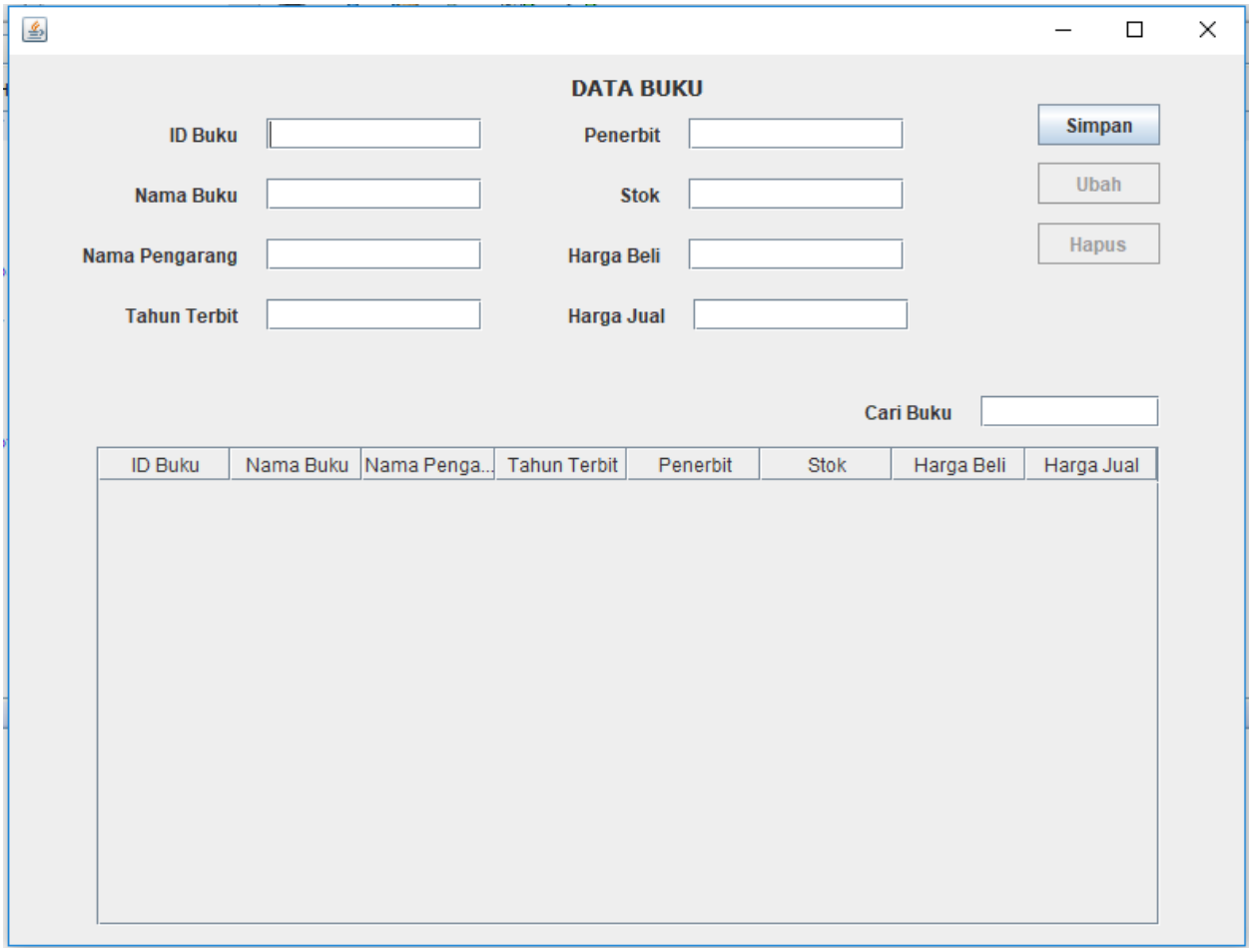
Jika package dan objek koneksi sudah dibuat, selanjutnya adalah membuat method initDataBuku untuk memanggil data buku dari database. Source code method seperti terlihat dibawah ini.

```
private void initDataBuku(String sql){
    DefaultTableModel tableModel = new DefaultTableModel();
    tableModel.addColumn("ID Buku");
    tableModel.addColumn("Nama Buku");
    tableModel.addColumn("Nama Pengarang");
    tableModel.addColumn("Tahun Terbit");
    tableModel.addColumn("Penerbit");
    tableModel.addColumn("Stok");
    tableModel.addColumn("Harga Beli");
    tableModel.addColumn("Harga Jual");

    try {
        ResultSet resultSet = koneksi.statement.executeQuery(sql);
        while (resultSet.next()) {
            tableModel.addRow(new Object[]{
                resultSet.getString("id_buku"),
                resultSet.getString("nama_buku"),
                resultSet.getString("nama_pengarang"),
                resultSet.getString("tahun_terbit"),
                resultSet.getString("penerbit"),
                resultSet.getString("stok"),
                resultSet.getString("harga_beli"),
                resultSet.getString("harga_jual"),

            });
        }
        tableBuku.setModel(tableModel);
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(rootPane, e.getMessage());
    }
}
```

Jika semua class dan project sudah dibuat, selanjutnya dilakukan test koneksi. Jika koneksi sudah berhasil, akan tampil nama-nama kolom pada jTable yang terdapat pada jFrame Buku. Nama kolom tersebut sama dengan kolom yang ada pada database. Seperti pada gambar form dibawah ini.




## 10.4. SOAL LATIHAN

Dengan mengikuti langkah-langkah di atas, buatlah program aplikasi sederhana menggunakan Netbeans IDE yang bisa menampilkan data dari database MySQL. Catat semua error yang terjadi.

# BAB 11. OPERASI DML DAN PENCARIAN

## 11.1. KOMPETENSI DASAR

Setelah mempelajari bab ini mahasiswa:

1) memahami operasi input, ubah, dan hapus data ke dalam database menggunakan Netbeans IDE.
2) Memahami operasi pencarian menggunakan data di database melalui Netbeans IDE

## 11.2. INDIKATOR

Setelah mempelajari bab ini mahasiswa:

1) Mampu membuat aplikasi sederhana yang berisi proses input, ubah, dan hapus data menggunakan Netbeans IDE.
2) Mampu membuat aplikasi sederhana yang bisa digunakan untuk operasi pencarian data melalui Netbeans IDE.

## 11.3. URAIAN MATERI

Sebelum memulai percobaan operasi input data, ubah data, hapus data dan pencarian data menggunakan Netbeans IDE, pastikan materi sebelumnya (Bab 10) sudah di pahami. Jika masih ada yang kurang paham dengan materi Bab 10 silakan untuk bertanya kepada dosen atau asisten dosen yang bersangkutan.

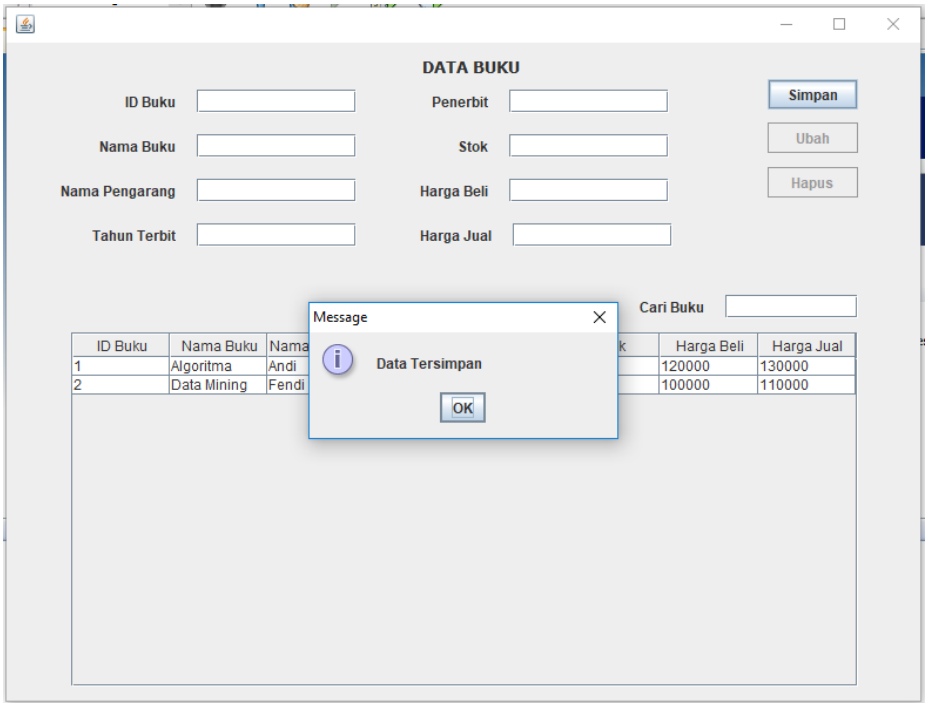
Materi pada bab ini merupakan lanjutan dari Bab 10. Sehingga masih menggunakan tabel dan database yang sama dengan materi Bab 10, yaitu tabel buku dan menggunakan settingan yang sudah dibuat sebelumnya pada Bab 10.

### 11.3.1. Operasi Input Data

Sebelum mengisikan source code untuk input data, pastikan sudah melakukan langkah-langkah sebelumnya, yaitu mulai dari pembuatan struktur *project*, penambahan *library*, pembuatan *Class* dan *Object*, *import package*. Selanjutnya mengisikan perintah-perintah input data pada unit editornya. Penulisan coding seperti terlihat pada gambar dibawah ini.

```
//    Menyimpan data buku
    private void simpanDataBuku(){
        String sql = "INSERT INTO buku(id_buku, nama_buku, nama_pengarang, "
                + "tahun_terbit, penerbit, stok, harga_beli, harga_jual) "
                + "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
        try {
            PreparedStatement ps = koneksi.koneksi.prepareStatement(sql);
            ps.setString(1, fieldIdBuku.getText());
            ps.setString(2, fieldNamaBuku.getText());
            ps.setString(3, fieldNamaPengarang.getText());
            ps.setString(4, fieldtahunTerbit.getText());
            ps.setString(5, fieldPenerbit.getText());
            ps.setString(6, fieldStok.getText());
            ps.setString(7, fieldHargaBeli.getText());
            ps.setString(8, fieldHargaJual.getText());
            ps.executeUpdate();
            reset();
            JOptionPane.showMessageDialog(rootPane, "Data Tersimpan");
            initDataBuku("SELECT buku.* FROM buku");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(rootPane, "Data Gagal Disimpan "
                    + "\nError: "+e.getMessage());
        }
    }
```

Selanjutnya, lakukan percobaan untuk menginputkan satu data pada form yang sudah terhubung dengan database. Jika proses penyimpanan berhasil, akan mengeluarkan notifikasi bahwa data sudah berhasil disimpan. Seperti pada gambar di bawah ini.



### 11.3.2. Operasi Ubah Data

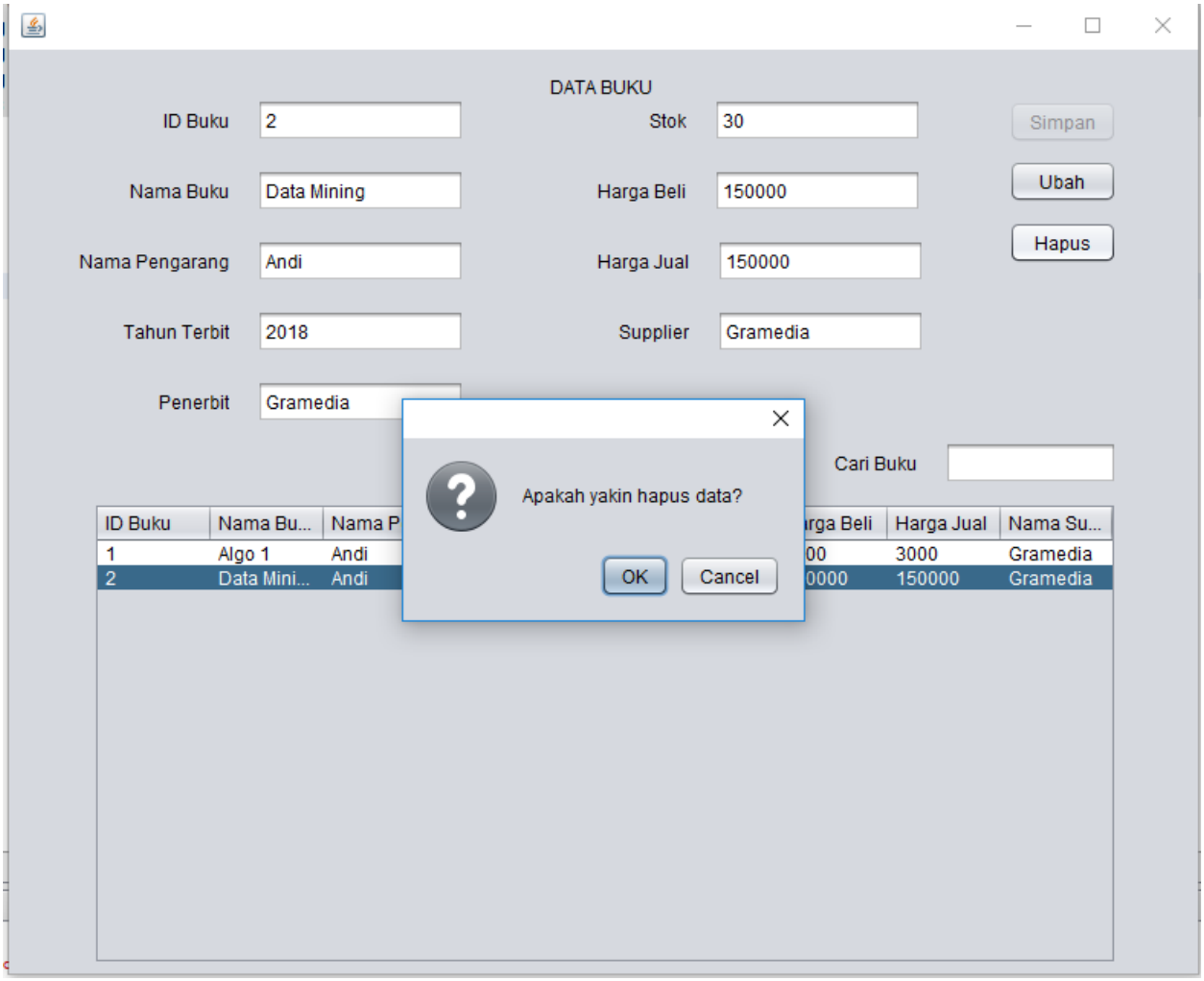
Untuk operasi ubah data, dengan mengisikan source code seperti dibawah ini.

```
//   Mengubah data buku
    private void ubahDataBuku(){
        String sql = "UPDATE buku SET nama_buku=?, nama_pengarang=?, "
                + "tahun_terbit=?, penerbit=?, stok=?, harga_beli=?, "
                + "harga_jual=? WHERE id_buku = ?";
        try {
            PreparedStatement ps = koneksi.koneksi.prepareStatement(sql);
            ps.setString(8, fieldIdBuku.getText());
            ps.setString(1, fieldNamaBuku.getText());
            ps.setString(2, fieldNamaPengarang.getText());
            ps.setString(3, fieldtahunTerbit.getText());
            ps.setString(4, fieldPenerbit.getText());
            ps.setString(5, fieldStok.getText());
            ps.setString(6, fieldHargaBeli.getText());
            ps.setString(7, fieldHargaJual.getText());
            System.out.println(ps.toString());
            ps.executeUpdate();
            reset();
            JOptionPane.showMessageDialog(rootPane, "Data Tersimpan");
            initDataBuku("SELECT buku.* FROM buku");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(rootPane, "Data Gagal Disimpan "
                    + "\nError: "+e.getMessage());
        }
    }
```

### 11.3.3. Operasi Hapus Data

Untuk operasi hapus data seperti terlihat pada gambar dibawah ini.

```
//    Menghapus data buku
    private void hapusDataBuku(){
        String sql = "DELETE FROM buku WHERE id_buku = ?";
        try {
            PreparedStatement ps = koneksi.koneksi.prepareStatement(sql);
            ps.setString(1, fieldIdBuku.getText());
            ps.executeUpdate();
            initDataBuku("SELECT buku.* FROM buku");
            reset();
            JOptionPane.showMessageDialog(rootPane, "Data Terhapus");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(rootPane, "Data Gagal Dihapus "
                    + "\nError: "+e.getMessage());
        }
    }
```




### 11.3.4. Pencarian Data

Operasi pencarian data dengan mengisikan source code seperti dibawah ini.

```
//    Melakukan pencarian data buku berdasarkan nama
    private void cariDataBuku(String params){
        initDataBuku("SELECT buku.* FROM buku WHERE nama_buku LIKE '%"+params+"%'");
    }
```

## 11.4. SOAL LATIHAN

Dari aplikasi yang sudah dibuat di atas, tambahkan satu form lagi yaitu Form Anggota yang berisi data anggota perpustakaan. Terlebih dahulu buatlah tabel anggota pada MySQL, kemudian dihubungkan dengan Form Anggota yang dibuat pada Netbeans IDE. Isikan data anggota melalui Form Anggota Netbeans IDE.

Catat semua error yang terjadi, dan sertakan pula solusi yang dilakukan.

# BAB 12. PEMROGRAMAN CLIENT DAN SERVER

## 12.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami konsep pemrograman Client dan Server menggunakan Netbeans IDE.
2) Memahami langkah-langkah pembuatan koneksi Client dan Server menggunakan Netbeans IDE

## 12.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa:

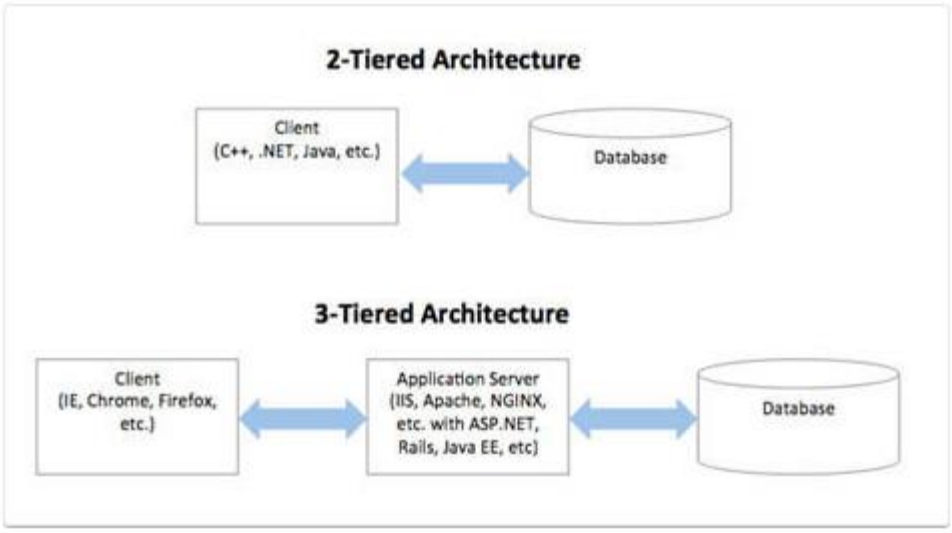
1) Mampu menjelaskan tentang konsep pemrograman client dan server menggunakan Netbeans IDE.
2) Mampu menjelaskan dan mengimplementasikan koneksi client dan server menggunakan Netbeans IDE.

## 12.3. URAIAN MATERI

### 12.3.1. Pemrograman Client dan Server

Pemrograman Client dan Server merupakan salah satu pemrograman yang menggunakan konsep mendistribusikan aplikasi dan database pada beberapa computer/host. Berdasarkan arti kata, Client merupakan host yang nantinya akan berisi aplikasi yang bertugas melakukan request ke Server. Sementara server merupakan host yang nantinya akan berisi aplikasi dan database yang bertugas melayani permintaan dari client. Secara spesifikasi hardware, tentunya server harus lebih besar dari pada client. Pemrograman Client dan Server bisa digunakan untuk dua jenis pemrograman, yaitu pemrograman basbasis web dan pemrograman berbasis desktop. Pemrograman berbasis web sudah pasti menggunakan konsep client dan server karena pengguna yang mengakses website adalah sebagai client.

Arsitektur Pemrograman Client dan Server dibagi menjadi 2, yaitu two tier dan dan three tier. Perbedaan arsitektur client dan server terlihat pada gambar di bawah ini.




### 12.3.2. Koneksi Client dan Server menggunakan Netbeans IDE

Sebelum memulai pembuatan aplikasi client dan server menggunakan Netbeans IDE, buatlah 3 tabel yang nantinya akan digunakan sebagai praktikum pembuatan aplikasi client dan server. Tabel tersebut antara lain tabel buku, tabel penjualan dan tabel detail penjualan. Komposisinya adalah sebagai berikut:

**Tabel Buku**

| Field | Type | | Null | Key | Default | | Extra |
|---|---|---|---|---|---|---|---|
| id_buku | varchar(20) | 11B | NO | PRI | *(NULL)* | 0K | |
| nama_buku | varchar(50) | 11B | NO | | *(NULL)* | 0K | |
| nama_pengarang | varchar(50) | 11B | NO | | *(NULL)* | 0K | |
| tahun_terbit | int(11) | 7B | NO | | *(NULL)* | 0K | |
| penerbit | varchar(30) | 11B | NO | | *(NULL)* | 0K | |
| stok | int(11) | 7B | NO | | *(NULL)* | 0K | |
| harga_beli | int(11) | 7B | NO | | *(NULL)* | 0K | |
| harga_jual | int(11) | 7B | NO | | *(NULL)* | 0K | |

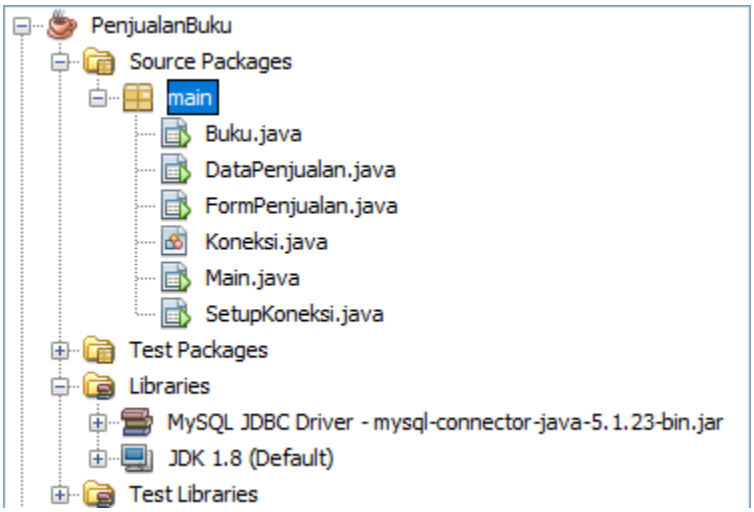
**Tabel Penjualan**

| Field | Type | | Null | Key | Default | | Extra |
|---|---|---|---|---|---|---|---|
| no_penjualan | varchar(30) | 11B | NO | PRI | *(NULL)* | 0K | |
| tanggal | date | 4B | NO | | *(NULL)* | 0K | |
| nama_pembeli | varchar(30) | 11B | NO | | *(NULL)* | 0K | |
| total | int(11) | 7B | NO | | *(NULL)* | 0K | |
| bayar | int(11) | 7B | YES | | *(NULL)* | 0K | |
| kembali | int(11) | 7B | YES | | *(NULL)* | 0K | |

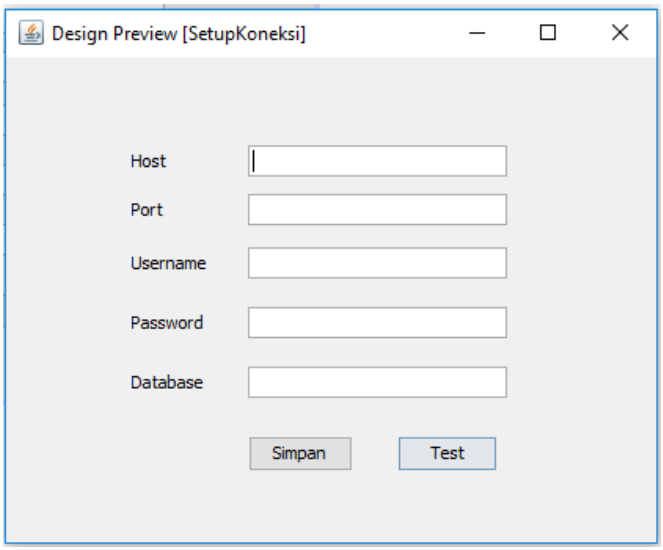
**Tabel Detail Penjualan**

| Field | Type | | Null | Key | Default | | Extra |
|---|---|---|---|---|---|---|---|
| id_detail | int(11) | 7B | NO | PRI | *(NULL)* | 0K | auto_increment |
| no_penjualan | varchar(30) | 11B | YES | | *(NULL)* | 0K | |
| id_buku | varchar(20) | 11B | YES | | *(NULL)* | 0K | |
| jumlah | int(11) | 7B | YES | | *(NULL)* | 0K | |

Selanjutnya buatlah project pada Netbeans Ide dengan nama PenjualanBuku. Tambahkan struktur project yang terdiri dari Class Koneksi, jFrame Buku, jFrame DataPenjualan, jFrame FormPenjualan, jFrame Main, dan jFrame SetupKoneksi, serta tambahkan pula jFrame SetupKoneksi, sehingga seperti dibawah ini.




Untuk file jFrame SetupKoneksi.java, buatlah tampilan desain seperti di bawah ini.



Untuk melakukan koneksi ke dalam database MySQL melalui form SetupKoneksi, diperlukan file Class Koneksi.java. File ini berisi source code memanggil JDBC Driver yang nantinya akan menghubungkan Netbeans IDE dengan database MySQL. Source code tersebut seperti pada gambar di bawah ini.

```
public class Koneksi {
    public Connection koneksi;
    public Statement statement;
    public static String host;
    public static String port;
    public static String db;
    public static String user;
    public static String password;

    public Koneksi() {
    }
    public Connection getConnection(){
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (Exception e) {
            System.err.println("JDBC Driver tidak ditemukan : "+e.getMessage());
        }
        koneksi = null;
        try {
            koneksi = DriverManager.getConnection("jdbc:mysql://"+
                    Koneksi.host+":"+Koneksi.port+"/"+Koneksi.db,
                    Koneksi.user,Koneksi.password);
            statement = koneksi.createStatement();
        } catch (Exception e) {
            System.err.println("Koneksi DB Error : "+e.getMessage());
        }
        return koneksi;
    }
}
```

Kemudian, untuk mengisikan setting awal form koneksi, buatlah method setValue() seperti di bawah ini.

```
public void setValue(){
    Koneksi koneksi = new Koneksi();
    if (koneksi.getConnection() == null) {
        fieldHost.setText("localhost");
        fieldPort.setText("3306");
        fieldUsername.setText("root");
        fieldPassword.setText("");
        fieldDataBase.setText("");
    }
    else{
        fieldHost.setText(Koneksi.host);
        fieldPort.setText(Koneksi.port);
        fieldUsername.setText(Koneksi.user);
        fieldPassword.setText(Koneksi.password);
        fieldDataBase.setText(Koneksi.db);
    }

}
```

Selanjutnya, untuk melakukan percobaan koneksi (test connection), buatlah method test() seperti di bawah ini.

```
public void test(){
    Koneksi.host = fieldHost.getText();
    Koneksi.port = fieldPort.getText();
    Koneksi.user = fieldUsername.getText();
    Koneksi.password = String.copyValueOf(fieldPassword.getPassword());
    Koneksi.db = fieldDataBase.getText();
    Koneksi koneksi = new Koneksi();

    if (koneksi.getConnection() != null) {
        JOptionPane.showMessageDialog(rootPane, "Koneksi Berhasil");
    }
    else{
        JOptionPane.showMessageDialog(rootPane, "Koneksi Gagal");
    }
}
```

Untuk menyimpan konfigurasi koneksi database, buatlah method simpan() seperti dibawah ini.

```
public void simpan(){
    Koneksi.host = fieldHost.getText();
    Koneksi.port = fieldPort.getText();
    Koneksi.user = fieldUsername.getText();
    Koneksi.password = String.copyValueOf(fieldPassword.getPassword());
    Koneksi.db = fieldDataBase.getText();

    this.dispose();
}
```

Sebelum settingan konfirugrasi koneksi disimpan dalam database, ada baiknya dilakukan validasi terlebih dahulu. Validasi dilakukan untuk memastikan bahwa data konfigurasi koneksi yang akan disimpan sudah sesuai dengan yang akan digunakan. Untuk melakukan validasi perlu dibuatkan sebuah method validateForm(), seperti di bawah ini.

```
public Boolean validateForm(){
    if (
            fieldHost.getText().equals("") ||
            fieldPort.getText().equals("") ||
            fieldUsername.getText().equals("") ||
            fieldDataBase.getText().equals("")
            ) {
        return false;
    }else{
        return true;
    }
}
```

Untuk memanggil method test, isikan source code di bawah ini pada button test.

```
private void buttonTestActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (validateForm()) {
        test();
    }else{
        JOptionPane.showMessageDialog(rootPane, "Lengkapi Semua Data Terlebih Dahulu");
    }
}
```

Setelah koneksi dinyatakan berhasil, selanjutnya disimpan melalui button simpan. Untuk memanggil method simpan, isikan source code dibawah ini.

```
private void buttonSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (validateForm()) {
        simpan();
    } else {
        JOptionPane.showMessageDialog(rootPane, "Lengkapi Semua Data Terlebih Dahulu");
    }

}
```

## 12.4. SOAL LATIHAN

Buatlah koneksi client dan server menggunakan aplikasi Netbeans IDE dengan database MySQL. Pastikan koneksi sudah berhasil. Catat semua error yang terjadi, dan sertakan pula solusi yang dilakukan.

# BAB 13. PEMROGRAMAN CLIENT DAN SERVER 2

## 13.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami Pemrograman Client dan Server menggunakan Netbeans IDE dan MySQL.
2) Memahami proses tambah, ubah, dan hapus data menggunakan konsep client dan server menggunakan Netbeans IDE dan MySQL.

## 13.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa:

1) Mampu menjelaskan tentang pemrograman Client dan Server menggunakan Netbeans IDE dan MySQL.
2) Mampu menjelaskan dan mengimplementasikan proses tambah, ubah, hapus data menggunakan konsep client dan server menggunakan Netbeans IDE dan MySQL.

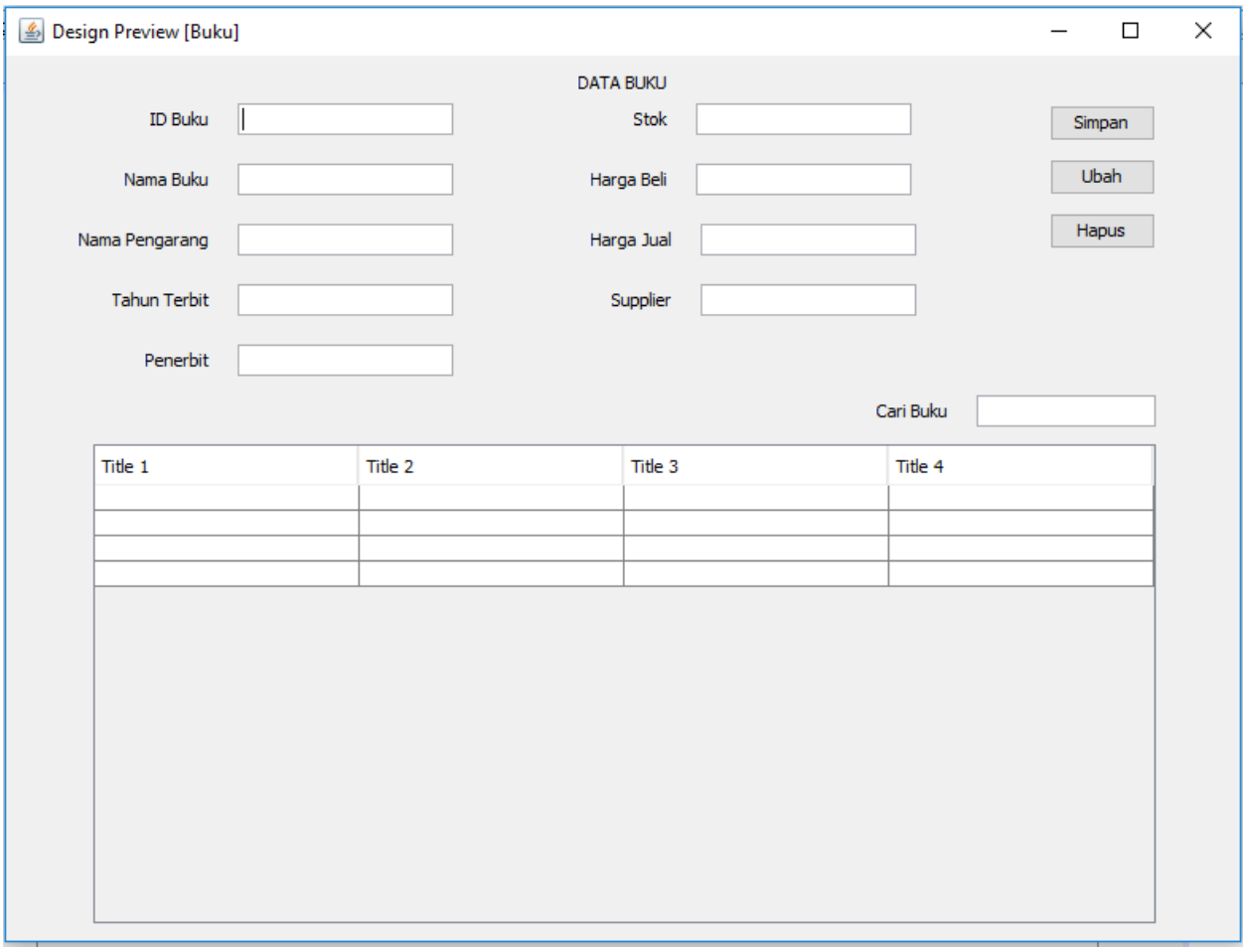
## 13.3. URAIAN MATERI

Materi pembahasan pada bab ini merupakan lanjutan dari materi pada bab sebelumnya (Bab 12). Bab sebelumnya membahas tentang bagaimana cara melakukan settingan koneksi jaringan client dan server dari Netbeans IDE ke MySQL.

Materi bab ini membahas tentang pembuatan aplikasi Netbeans IDE sebagai client yang melakukan operasi input data, ubah data, dan hapus data ke server. Operasi input data, ubah data, dan hapus data berdasarkan 3 tabel yang sudah dibuat pada Bab 12.

### 13.3.1. Tambah Data

Pembahasan pertama adalah tampilan jFrame Buku.java. Buatlah tampilan desain jFrame Buku yang terdiri dari komponen jLabel, jTextField, jButton, dan jTabel seperti dibawah ini.




Selanjutnya adalah Import Package pada jFrame Buku dengan mengisikan source code dibawah ini.

```
import java.sql.*;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

Kemudian, buatlah objek koneksi dari Class koneksi

```
    private Koneksi koneksi = new Koneksi();
```

Setelah objek koneksi dari Class Koneksi sudah dibuat, selanjutnya adalah buat method initDataBuku() yang digunakan untuk menampilkan data buku dari tabel buku. Data buku tersebut nantinya akan ditampilkan pada komponen jTable yang terdapat pada jFrame Buku. Source Code initDataBuku() seperti terlihat dibawah ini.

```
private void initDataBuku(String sql){
    DefaultTableModel tableModel = new DefaultTableModel();
    tableModel.addColumn("ID Buku");
    tableModel.addColumn("Nama Buku");
    tableModel.addColumn("Nama Pengarang");
    tableModel.addColumn("Tahun Terbit");
    tableModel.addColumn("Penerbit");
    tableModel.addColumn("Stok");
    tableModel.addColumn("Harga Beli");
    tableModel.addColumn("Harga Jual");
    tableModel.addColumn("Nama Supplier");
    try {
        ResultSet resultSet = koneksi.statement.executeQuery(sql);
        while (resultSet.next()) {
            tableModel.addRow(new Object[]{
                resultSet.getString("id_buku"),
                resultSet.getString("nama_buku"),
                resultSet.getString("nama_pengarang"),
                resultSet.getString("tahun_terbit"),
                resultSet.getString("penerbit"),
                resultSet.getString("stok"),
                resultSet.getString("harga_beli"),
                resultSet.getString("harga_jual"),
                resultSet.getString("nama_supplier")
            });
        }
        tableBuku.setModel(tableModel);
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(rootPane, e.getMessage());
    }
}
```

Jika koneksi method untuk sudah dibuat, kemudian mulai membuat method untuk manipulasi data. Dalam hal ini method yang pertama untuk manipulasi adalah method simpanDataBuku(). Method ini berisi perintah coding untuk melakukan proses tambah/input data baru ke dalam database melalui form Netbeans IDE. Method simpanDataBuku() seperti terlihat pada gambar di bawah ini.

```
private void simpanDataBuku(){
    String sql = "INSERT INTO buku(id_buku, nama_buku, nama_pengarang, "
            + "tahun_terbit, penerbit, stok, harga_beli, harga_jual, "
            + "nama_supplier) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
    try {
        PreparedStatement ps = koneksi.koneksi.prepareStatement(sql);
        ps.setString(1, fieldIdBuku.getText());
        ps.setString(2, fieldNamaBuku.getText());
        ps.setString(3, fieldNamaPengarang.getText());
        ps.setString(4, fieldtahunTerbit.getText());
        ps.setString(5, fieldPenerbit.getText());
        ps.setString(6, fieldStok.getText());
        ps.setString(7, fieldHargaBeli.getText());
        ps.setString(8, fieldHargaJual.getText());
        ps.setString(9, fieldNamaSupplier.getText());
        ps.executeUpdate();
        reset();
        JOptionPane.showMessageDialog(rootPane, "Data Tersimpan");
        initDataBuku("SELECT buku.* FROM buku Order By nama_buku asc");
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(rootPane, "Data Gagal Disimpan \nError: "+e.getMessage());
    }
}
```

Biasanya juga disediakan souce code untuk mengosongkan form input data (reset). Pengosongan ini biasanya digunakan setelah proses manipulasi selesai dijalankan (sebagai tanda bahwa proses manipulasi berhasil dijalankan). Untuk method reset() seperti di bawah ini.

```
public void reset(){
    fieldCariBuku.setText(null);
    fieldHargaBeli.setText(null);
    fieldHargaJual.setText(null);
    fieldIdBuku.setText(null);
    fieldNamaSupplier.setText(null);
    fieldNamaBuku.setText(null);
    fieldNamaPengarang.setText(null);
    fieldPenerbit.setText(null);
    fieldStok.setText(null);
    fieldtahunTerbit.setText(null);
    fieldIdBuku.setEditable(true);
    buttonSimpanBuku.setEnabled(true);
    buttonHapusBuku.setEnabled(false);
    buttonUbahBuku.setEnabled(false);
}
```

Sebelum data disimpan (tambah dan ubah data), direkomendasikan untuk dilakukan validasi data untuk memastikan kebenaran data yang akan disimpan dalam database. Untuk melakukan validasi data diperlukan method validateData() yang berisi source code seperti gambar di bawah ini.

```
public Boolean validateData(){
    if (
            fieldIdBuku.getText().equals("") ||
            fieldNamaBuku.getText().equals("") ||
            fieldNamaPengarang.getText().equals("") ||
            fieldtahunTerbit.getText().equals("") ||
            fieldPenerbit.getText().equals("") ||
            fieldStok.getText().equals("") ||
            fieldHargaBeli.getText().equals("") ||
            fieldHargaJual.getText().equals("") ||
            fieldNamaSupplier.getText().equals("")
        ) {
        JOptionPane.showMessageDialog(rootPane, "Lengkapi semua data");
        return false;
    }
    else {
        return true;
    }
}
```

Selain validasi data, juga ada fasilitas untuk memanggil data melalui klik mouse yang ada di jTable dan menampilkannya di jTextField yang terdapat pada jFrame Buku. Pemanggilan data biasanya digunakan sebelum proses ubah dan hapus data untuk memastikan data yg dipilih sebelum diubah atau dihapus. Pemanggilan data tersebut menggunakan event mouse click, seperti pada gambar di bawah ini.

```
private void tableBukuMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    buttonHapusBuku.setEnabled(true);
    buttonUbahBuku.setEnabled(true);
    buttonSimpanBuku.setEnabled(false);
    fieldIdBuku.setEditable(false);
    int row = tableBuku.getSelectedRow();
    fieldIdBuku.setText(tableBuku.getValueAt(row, 0).toString());
    fieldNamaBuku.setText(tableBuku.getValueAt(row, 1).toString());
    fieldNamaPengarang.setText(tableBuku.getValueAt(row, 2).toString());
    fieldtahunTerbit.setText(tableBuku.getValueAt(row, 3).toString());
    fieldPenerbit.setText(tableBuku.getValueAt(row, 4).toString());
    fieldStok.setText(tableBuku.getValueAt(row, 5).toString());
    fieldHargaBeli.setText(tableBuku.getValueAt(row, 6).toString());
    fieldHargaJual.setText(tableBuku.getValueAt(row, 7).toString());
    fieldNamaSupplier.setText(tableBuku.getValueAt(row, 8).toString());

}
```

Untuk memanggil method simpanDataBuku(), pada button simpan perlu diisikan source code simpanBuku. Source code tersebut pada prinsipnya adalah memanggil method, sehingga

ketika button lain yang memerlukan proses yang sama, cukup memanggil method saya. Source code button simpanBuku seperti terlihat pada gambar di bawah ini.

```
private void buttonSimpanBukuActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (validateData()) {
        simpanDataBuku();
    }

}
```

Untuk menghindari kesalahan pengguna (human error) pada saat menggunakan aplikasi, bisa ditambahkan constructor yang nantinya bisa mengkondisikan form pada saat dibuka. Dalam hal ini constuctor yang akan ditambahkan adalah menampilkan data buku secara urut berdasarkan nama buku, dan me-nonaktif-kan button ubah dan hapus buku. Sehingga dengan demikian ketika form buku ditampilkan pertama kali button yang aktif adalah button simpan. Source code constructor yang dimaksud seperti terlihat pada gambar di bawah ini.

```
public Buku() {
    initComponents();
    initDataBuku("SELECT buku.* FROM buku Order By nama_buku asc");
    buttonHapusBuku.setEnabled(false);
    buttonUbahBuku.setEnabled(false);

}
```

### 13.3.2. Ubah Data Buku

Selanjutnya, buatlah method ubahDataBuku() yang berisi perintah-perintah berupa source code untuk proses ubah data buku. Seperti terlihat pada gambar di bawah ini.

```
private void ubahDataBuku(){
    String sql = "UPDATE buku SET nama_buku=?, nama_pengarang=?, "
            + "tahun_terbit=?, penerbit=?, stok=?, harga_beli=?, harga_jual=?, "
            + "nama_supplier=? WHERE id_buku = ?";
    try {
        PreparedStatement ps = koneksi.koneksi.prepareStatement(sql);
        ps.setString(9, fieldIdBuku.getText());
        ps.setString(1, fieldNamaBuku.getText());
        ps.setString(2, fieldNamaPengarang.getText());
        ps.setString(3, fieldtahunTerbit.getText());
        ps.setString(4, fieldPenerbit.getText());
        ps.setString(5, fieldStok.getText());
        ps.setString(6, fieldHargaBeli.getText());
        ps.setString(7, fieldHargaJual.getText());
        ps.setString(8, fieldNamaSupplier.getText());
        System.out.println(ps.toString());
        ps.executeUpdate();
        reset();
        JOptionPane.showMessageDialog(rootPane, "Data Tersimpan");
        initDataBuku("SELECT buku.* FROM buku Order By nama_buku asc");

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(rootPane, "Data Gagal Disimpan \nError: "+e.getMessage());
    }
}
```

Setelah source code simpanBuku yang memanggil method simpanDataBuku(), selanjutnya source code ubahbuku yang memanggil method ubahDataBuku() yang memangil method ubahDataBuku(), seperti terlihat pada gambar di bawah ini.

```
private void buttonUbahBukuActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (validateData()) {
        ubahDataBuku();
    }
}
```

### 13.3.3. Hapus Data Buku

Selanjutnya adalah method hapusDataBuku() yang berisi Source Code untuk menghapus data buku. Seperti terlihat pada gambar di bawah ini.

```
private void hapusDataBuku(){
    String sql = "DELETE FROM buku WHERE id_buku = ?";
    try {
        PreparedStatement ps = koneksi.koneksi.prepareStatement(sql);
        ps.setString(1, fieldIdBuku.getText());
        ps.executeUpdate();
        reset();
        JOptionPane.showMessageDialog(rootPane, "Data Terhapus");
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(rootPane, "Data Gagal Dihapus \nError: "+e.getMessage());
    }
}
```

Begitu juga source code hapusbuku yang digunakan untuk memanggil method hapusDataBuku(). Seperti terlihat pada gambar dibawah ini.

```
private void buttonHapusBukuActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int state = JOptionPane.showConfirmDialog(
            rootPane,
            "Apakah yakin hapus data?", null,
            JOptionPane.WARNING_MESSAGE);
    if (state == 0) {
        hapusDataBuku();
    }

}
```

### 13.3.4. Pencarian Data Buku

Selain operasi manipulasi, juga ditambahkan operasi pencarian data pada jFrame Buku. Untuk pencarian data digunakan method cariDataBuku() seperti terlihat pada gambar di bawah ini.

```
private void cariDataBuku(String params){
    initDataBuku("SELECT buku.* FROM buku "
            + "WHERE nama_buku LIKE '%"+params+"%'");
}
```

Isikan source code event keyReleased pada fieldCariBuku yang digunakan untuk melakukan pencarian atau memanggil method cariBuku. Source code ini akan melakukan pencarian ketika keyword diketikkan pada field cari. Source codenya seperti terlihat di bawah ini.

```
private void fieldCariBukuKeyReleased(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    String nama_buku = fieldCariBuku.getText();
    cariDataBuku(nama_buku);
}
```

## 13.4. SOAL LATIHAN

Berdasarkan pembahasan di atas, ulangi pembahasan diatas untuk Form Penjualan dan Form Detail Penjualan. Catat semua error yang terjadi, dan sertakan pula solusi yang dilakukan.

# BAB 14. STUDI KASUS

## 14.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa memahami konsep pemrograman Client dan Server menggunakan aplikasi Netbeans IDE dengan database MySQL.

## 14.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mmapu membangun sebuah aplikasi berbasis Client dan Server menggunakan aplikasi Netbeans IDE dengan database MySQL.

## 14.3. SOAL

Seorang pengusaha perusahaan penjualan barang spare part motor mengalami permasalahan terkait pendataan barang jualannya. Pengusaha tersebut membutuhkan aplikasi yang bisa mendata stok barang dan keuntungan penjualan. Pengguna aplikasi tersebut adalah petugas Gudang, petugas kasir dan admin (pemilik). Stok barang bertambah dengan adanya transaksi pembelian barang dari supplier yang didata dan dilayani oleh petugas Gudang. Sementara stok berkurang dari adanya transaksi pembelian barang oleh konsumen (member) yang dilayani oleh kasir. Admin bisa memantau stok barang dan keuntungan yang didapat selama periode tertentu.

Anda sebagai seorang programmer, buatkan sebuah aplikasi berbasis client dan server sederhana menggunakan Bahasa pemrograman Netbeans IDE 8.0 untuk membantu pengusaha tersebut. Aplikasi tersebut terdiri dari 2 client (Kasir dan petugas gudang) dan 1 server (admin/pemilik)

**diterbitkan oleh :**
**Universitas Teknologi Yogyakarta**
**Jl. Siliwangi, Jombor, Sleman, Yogyakarta**
**Email : publikasi@uty.ac.id**
**Website : uty.ac.id**

ISBN 978-623-92626-6-2