**MODUL PRAKTIKUM**

SISTEM BASIS DATA

**Modul 1**

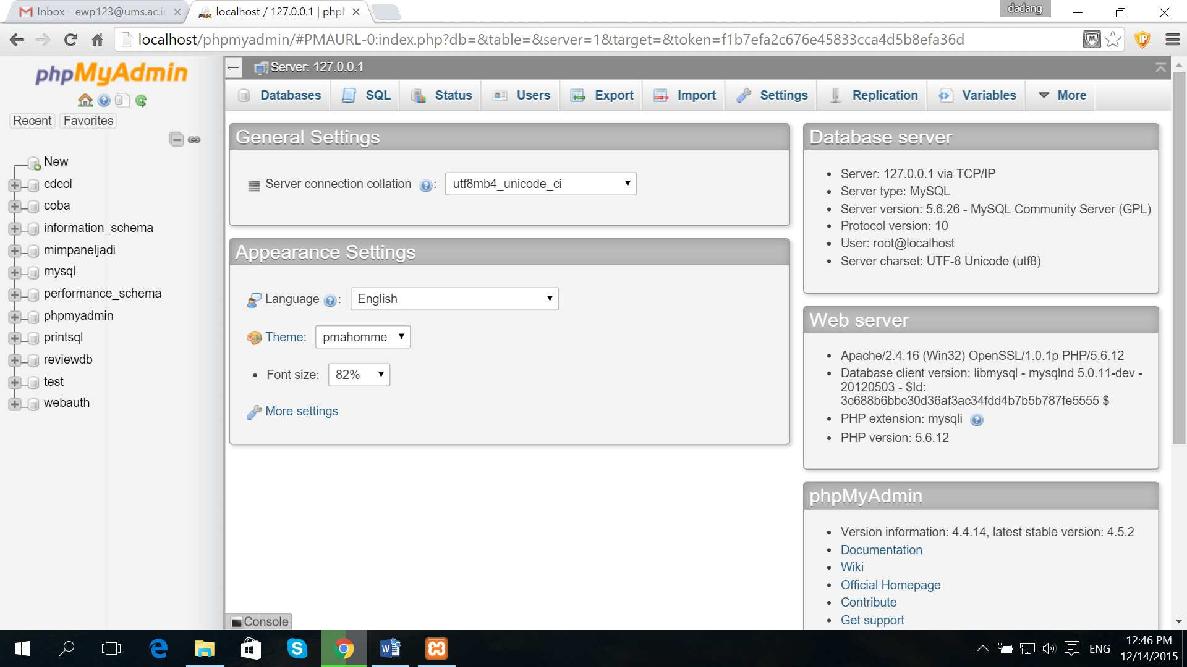
**Pengenalan dan Instalasi MySQL**

1. **Tujuan** 
   1. Mahasiswa memahami cara instalasi MySQL.
   2. Mahasiswa memahami cara konfigurasi MySQL
2. **Landasan Teori** 
   1. MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang multi-thread, multi-user, dengan sekitar 6 juta instalasi seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual di bawah lisensi komersial untuk kasus-kasus di mana penggunaannya tidak cocok dengan penggunaan GPL.

MySQL adalah sebuah implementasi dari sistem manajemen basis data relational (RDBMS) yang didistribusikan secara gratis. Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan bahwa perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basis data yang telah ada sebelumnya. SQL (Structured Query Language) merupakan sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Saat ini versi open source MySQL yang dapat didownload dan digunakan secara gratis adalah MySQL Community Server 5.529. Versi ini merupakan versi MySQL yang sering digunakan dalam pengembangan website. Biasanya dalam paket apache server XAMPP di Windows terdapat MySQL. Pada XAMPP, MySQL dikendalikan secara administratif menggunakan Bahasa PHP yang kemudian user interface nya dalam bentuk phpMyAdmin.



*Gambar 1.1 Tampilan Antarmuka PHPMyAdmin*

1. Instalasi MySQL pada XAMPP

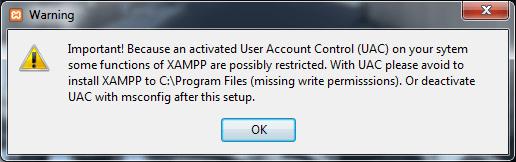
Ikuti langkah-langkah berikut ini untuk melakukan instalasi XAMPP. Untuk mendapatkan installer XAMPP anda dapat mengakses ke halaman https://www.apachefriends.org/download.html.

Langkah 1.

Non-aktifkan anti-virus karena dapat menyebabkan beberapa komponen XAMPP yang tidak bias di install dengan lancer.

Langkah 2.

Untuk pengguna Windows 7 atau Windows 8 anda akan melihat jendela pop up, peringatan tentang User Account Control (UAC) yang aktif pada sistem. Klik “OK” untuk melanjutkan installasi.



*Gambar 1.2. User Account Control Warning*

Langkah 3.

Mulai proses instalasi dengan klik dua kali pada installer XAMPP. Klik “Next” setelah splash screen.



*Gambar 1.3. Splash Screen XAMPP*

Langkah 4.

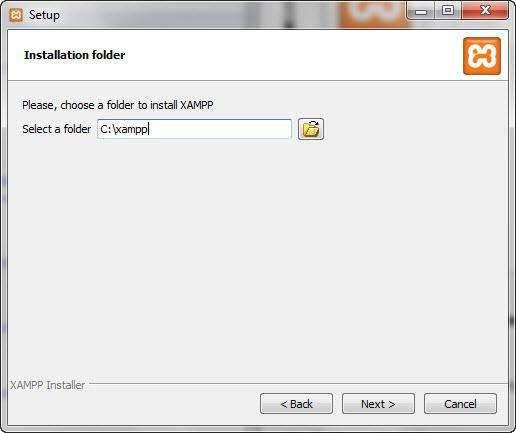
Di sini, kita dapat memilih komponen apa saja yang ingin kita install. Pilih pilihan default dan pilih “Next”.



*Gambar 1.4. Pilihan Komponen yang Akan diinstall*

Langkah 5.

Pilih folder sebagai tempat XAMPP akan diinstal, di folder ini kita akan menyimpan semua file aplikasi web kita, jadi pastikan untuk memilih drive yang masih memiliki banyak ruang (space).



*Gambar 1.5. Pemilihan Folder Instalasi*

Langkah 6.

Layar berikutnya adalah promo untuk Bitnami, sebuah took aplikasi untuk server perangkat lunak. Hapus centang pada kotak “Learn mo re about Bitnami for XAMPP”.



*Gambar 1.6 Promo Bitnami*

Langkah 7.

Sekarang Setup sudah siap untuk menginstall XAMPP. Klik Next dan tunggu instaler untuk membongkar paket-nya dan memasang komponen yang dipilih. Mungkin memakan waktu beberapa menit. Nanti mungkin kita akan diminta untuk menyetujui akses Firewall untuk komponen tertentu (seperti Apache) selama proses instalasi.

Langkah 8.

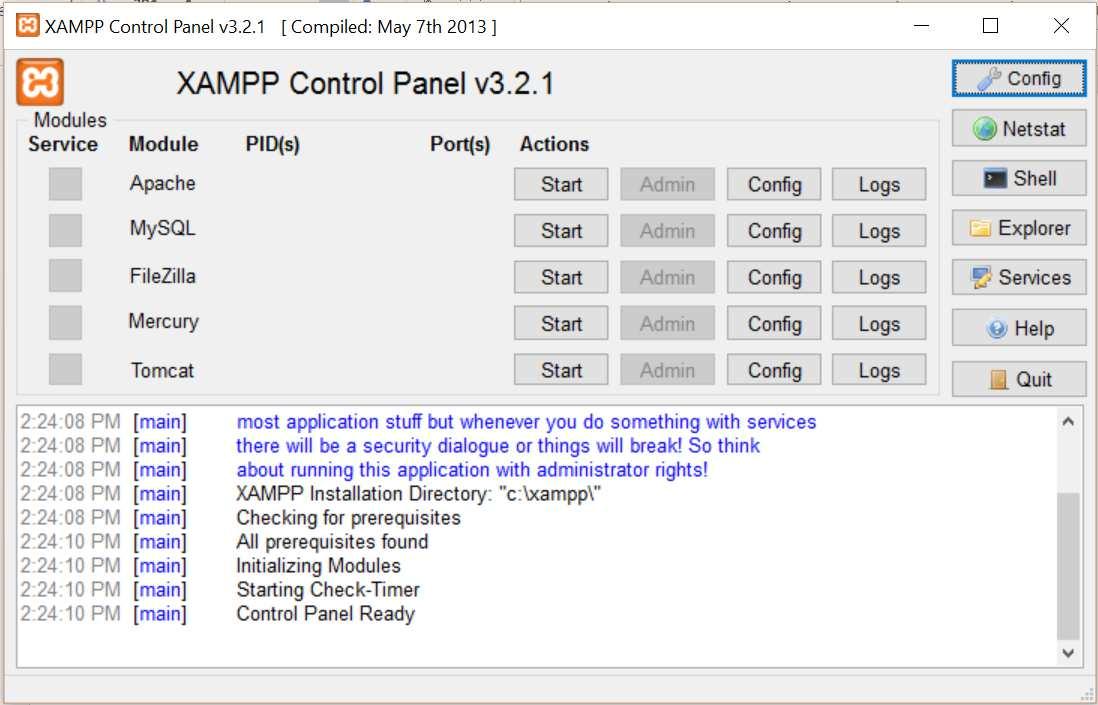
Proses Install sudah selesai! Pilih Kotak centang ‘Do you want to start the Control Panel now?’ untuk membuka panel kontrol XAMPP.



*Gambar 1.7 Finish Installation*

1. **Langkah-langkah Praktikum** 
   1. Mengakses PhpMyAdmin Langkah 1.

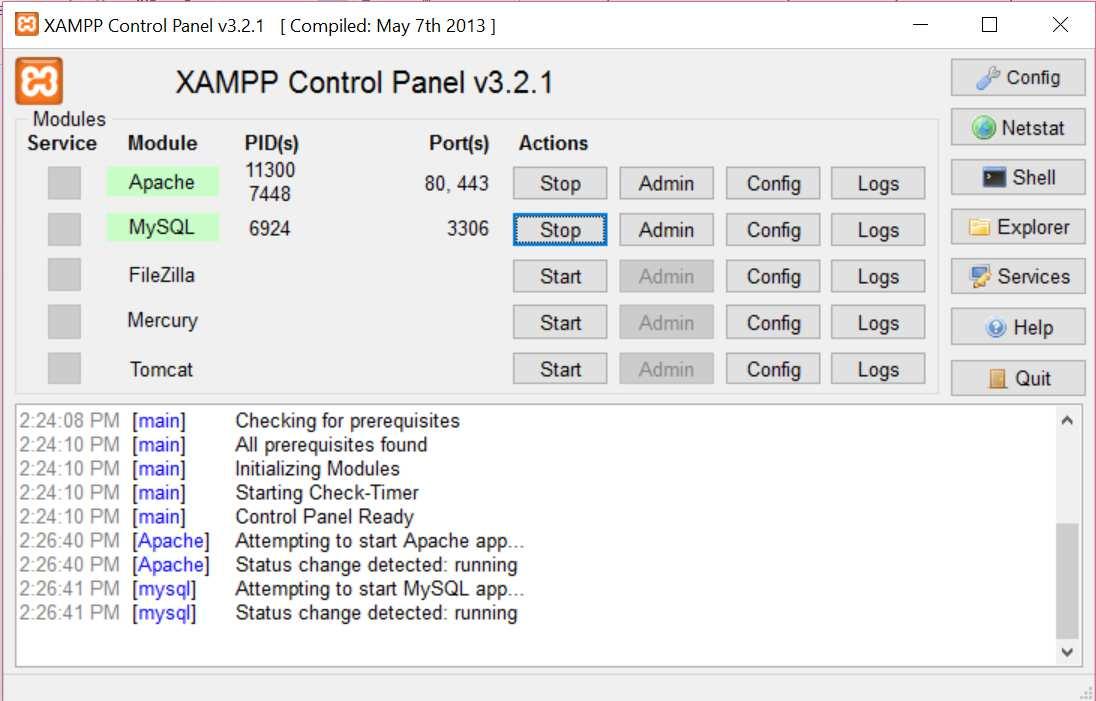
Buka XAMPP Control Panel.



*Gambar 1.8 XAMPP Control Panel*

Langkah 2.

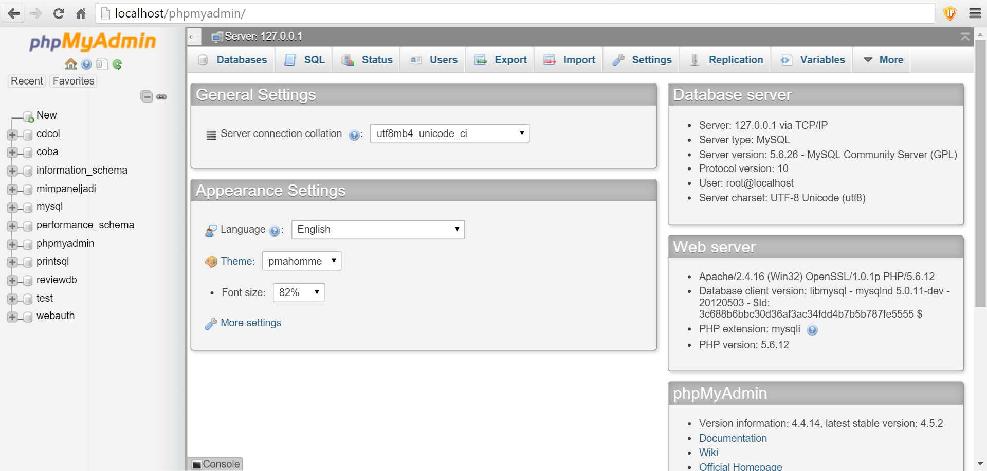
Jalankan Apache Server dan MySQL Server dengan menekan tombol “Start”. Tunggu hingga muncul warna hijau pada nama Module.



*Gambar 1.9. Service telah berjalan*

Langkah 3.

Buka web browser anda dan ketikan http://localhost/phpmyadmin/.



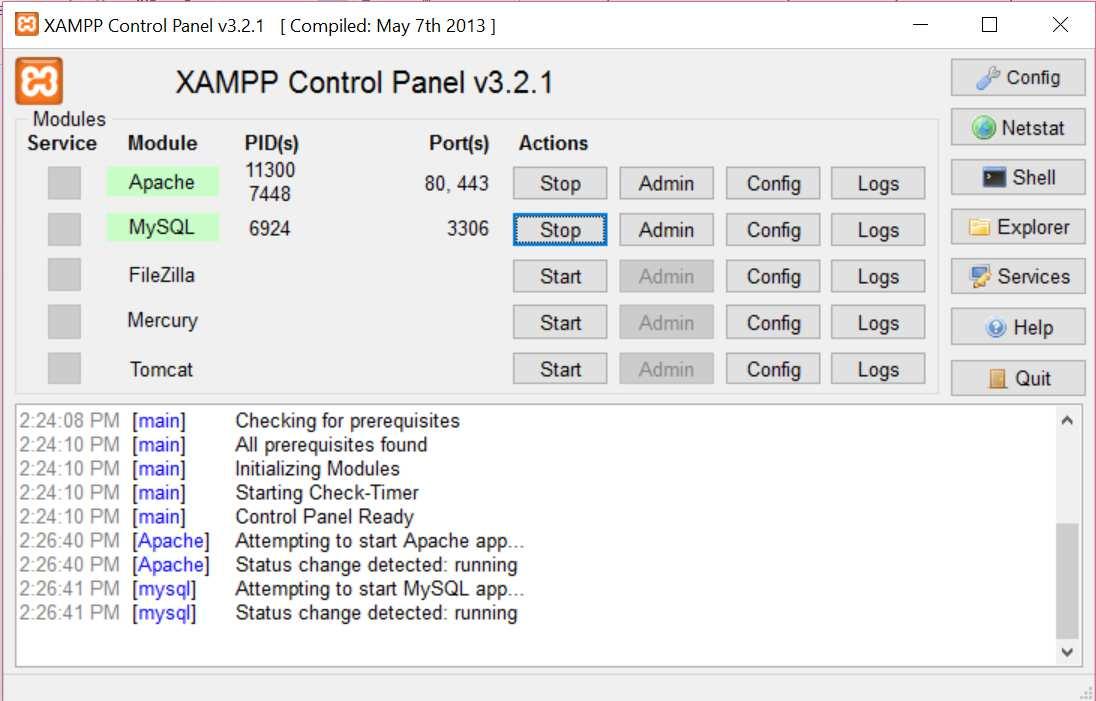
*Gambar 1.10. Tampilan halaman depan phpMyAdmin*

Langkah 4.

Pada halaman PhpMyAdmin ini kita dapat melakukan pembangunan basis data dan juga melakukan manipulasi isi basis data dengan MySQL. PhpMyAdmin merupakan halaman GUI administratif MySQL server yang saat ini paling banyak dipakai dalam pengembangan aplikasi berbasis web.

1. Mengakses MySQL lewat command prompt Langkah 1.

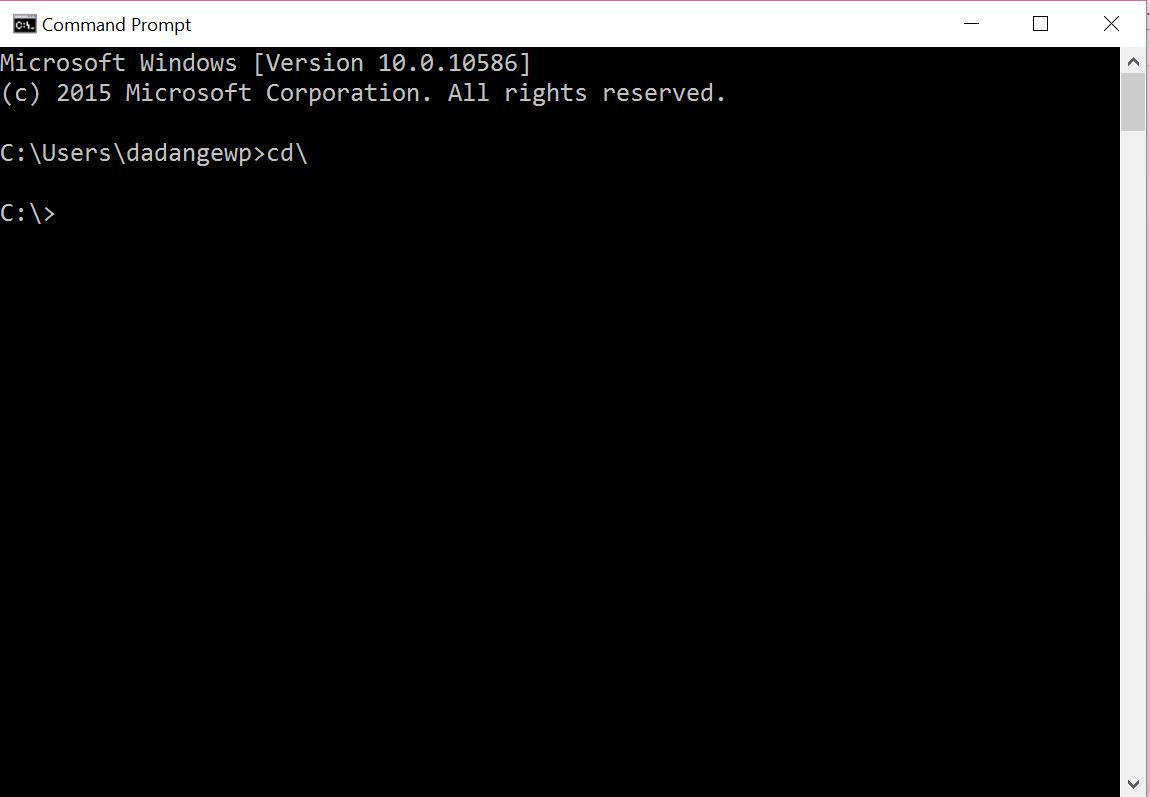
Pastikan bahwa server MySQL telah berjalan.



*Gambar 1.11. Service yang sedang berjalan*

Langkah 2.

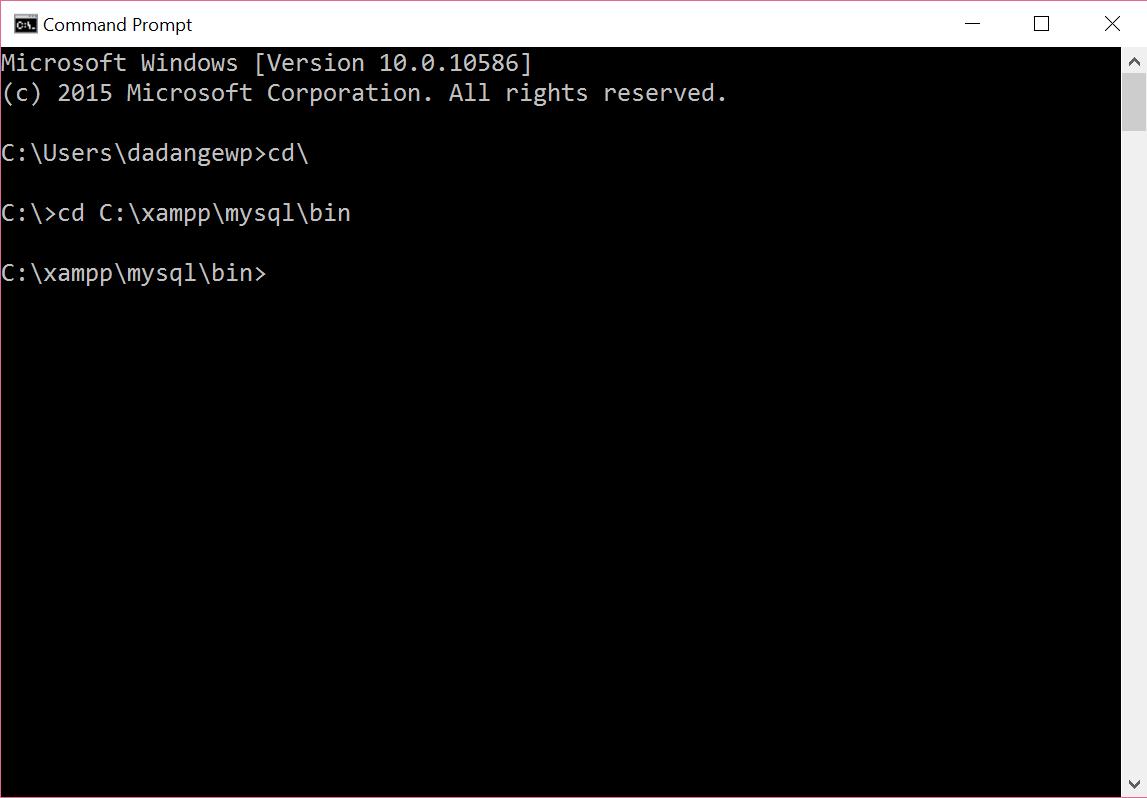
Buka command prompt dan ketik ‘cd\’ dan tekan “Ente r”. Sehingga anda akan berada di direktori (C:\).



*Gambar 1.12. Tampilan Window Command Prompt*

Langkah 3.

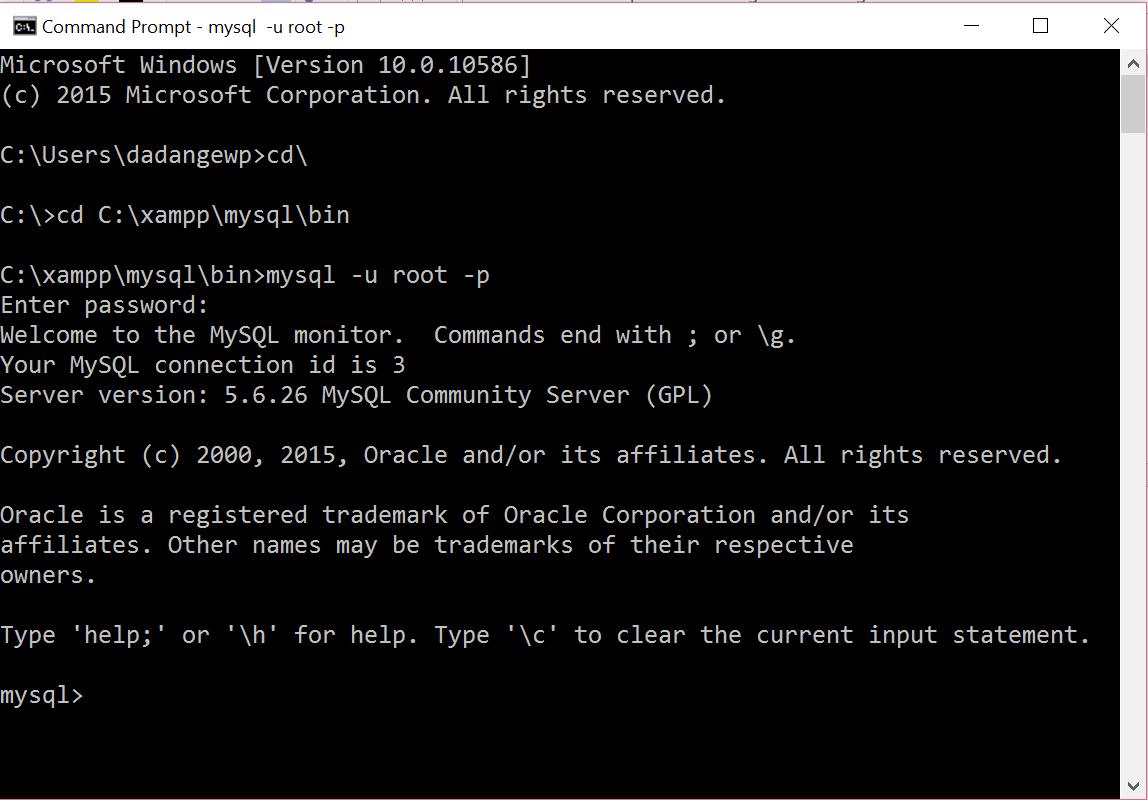
setelah itu arahkan ke folder C:\xampp\mysql\bin, caranya ketik ‘cd C:\xampp\mysql\bin’ (tanpa ‘ ‘) kemudian tekan tomb ol Enter.



*Gambar 1.13. Arahkan ke directory C:\xampp\mysql\bin*

Langkah 4.

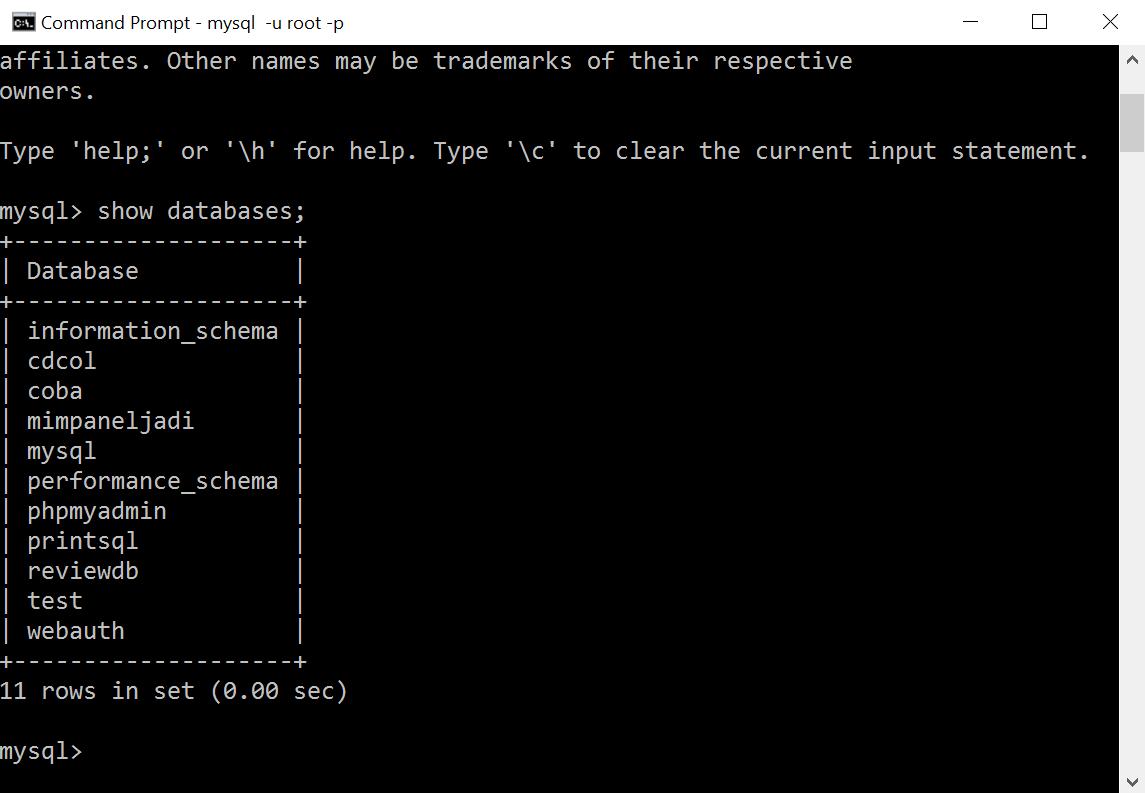
setelah berada di dalam folder C:\xampp\mysql\bin, baru anda dapat mengakses mysql. ketik : ‘mysql -u root -p’ (tanpa ‘ ‘) kemud ian tekan tombol Enter. Masukkan password (jika ada) kemudian klik tombol Enter lagi (seara default tidak ada password untuk root).



*Gambar 1.15. Login ke MySQL*

Langkah 5.

Setelah tampilan seperti di Gambar 1.15 berarti anda telah berhasil masuk ke MySQL sebagai root user. Untuk melihat database yang ada pada server anda dapat mengetikan ‘show databases;’. Jangan lupa unt uk selalu mengakhiri command dengan ‘;’.



*Gambar 1.16. Menampilkan Database yang ada di Server*

1. **Tugas** 
   1. Jelaskan mengapa dibutuhkan data!
   2. Jelaskan manfaat database dan contohnya!
   3. Untuk menentukan jenis database yang digunakan, apa yang menjadi acuan dalam pemilihan database tersebut?
   4. Jelaskan istilah atau terminology yang digunakan dalam Database (database,table, field, record)
   5. Bandingkan perbedaan pengolahan data secara manual dengan mengunakan system database.
   6. Mengapa dibutuhkan DBMS?
   7. Pada percobaan diatas ada beberapa field yang tipe data dan ukurannya berbeda. Jelaskan!

**Modul 2**

**Perancangan Basis Data**

1. **Tujuan** 
   1. Mahasiswa memahami makna entitas, atribut, dan relasi.
   2. Mahasiswa memahami tahap-tahap normalisasi.
   3. Mahasiswa mampu merancang basis data melalui tahap-tahap perancangannya.
2. **Landasan Teori** 
   1. Entitas

*Entity* adalah suatu obyek yang dapat dikenali dari obyek yang lain. Contoh : seseorangyang khusus, perusahaan, peristiwa, tanaman dan lain-lain. *Entity* mempunyai atribut, contoh : seseorang mempunyai nama dan alamat. Kumpulan entity adalah suatu kumpulan entity dengan tipe yang sama yang berbagi properti yang sama pula. Contoh : kumpulan orang, perusahaan, tanaman, tempat liburan dan lain-lain.

* 1. Atribut

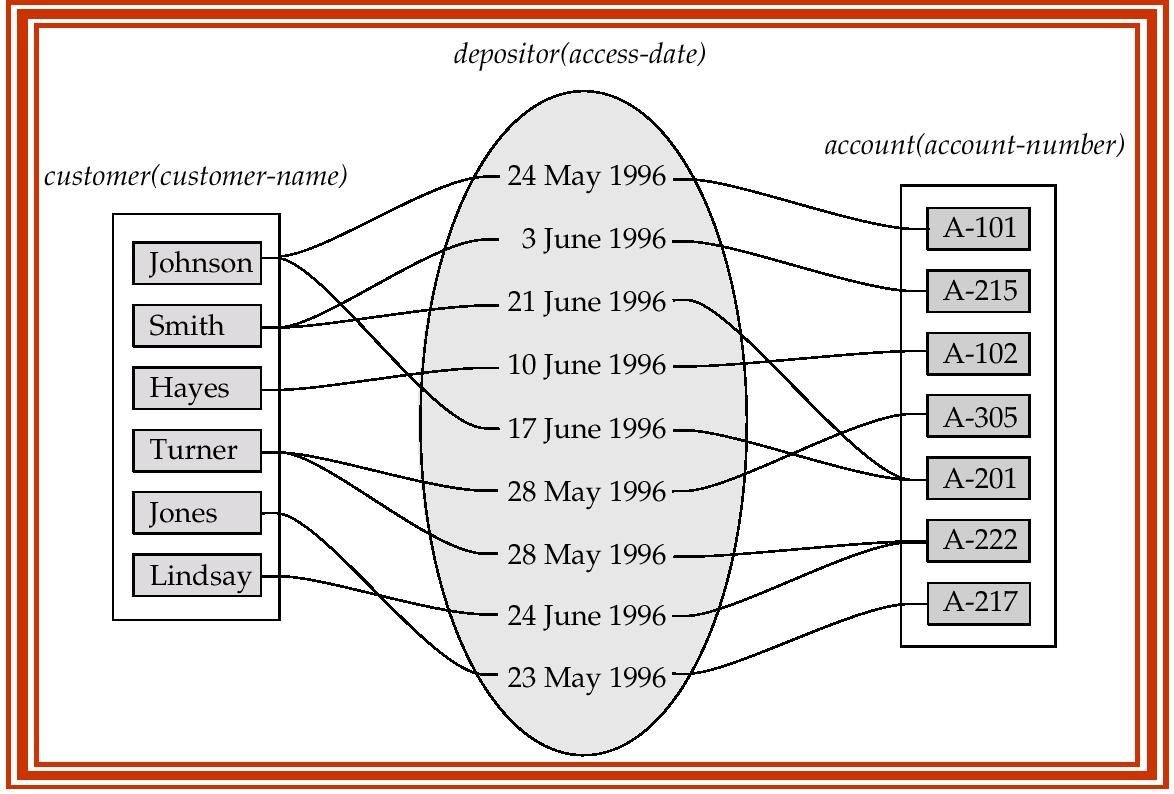
*Entity* ditampilkan oleh sekumpulan *attribute*, yang mana properti deskriptifnya dikuasaioleh seluruh anggota dalam kumpulan *entity*.

Tipe *attribute* :

* + 1. *Simple* (sederhana) dan *composite* (gabungan) *attributes*.
    2. *Single-valued* (satu-fungsi ) dan *multi-valued* (multi-fungsi ) *attributes*. Contoh *:* atribut multi-fungsi : nomor telepon
    3. *Derived* (turunan) *attributes* : dapat diperhitungkan dari atribut lain Contoh : umur,tanggal kelahiran.
  1. Relasi

*Relationship* adalah kesesuaian antar beberapa *entity*. *Relationship set* adalah hubungan

matematika antara *entity* n>2, tiap bagiannya diambil dari satuan *entity* {(*e*1, *e*2, … *en*) | *e*1 *E*1, *e*2 *E*2, …, *en*  *En*} dimana (*e*1, *e*2, …, *en*) adalah *relationship*. *Attribute* dapat merupakan properti dari *relationship set*.



Sebagai contoh: *depositor* merupakan *relationship set* antara *entity sets customer* dan *account* mungkin beratribut *access-date.* Mengacu pada jumlah *entity sets* yang terlibatdalam *relationship set*. *Relationship sets* yang melibatkan dua *entity sets* adalah *binary*

(atau tingkat dua). Umumnya, hampir semua *relationship sets* dalam sistem database adalah *binary*. *Relationship sets* mungkin melibatkan lebih dari dua *entity sets*.

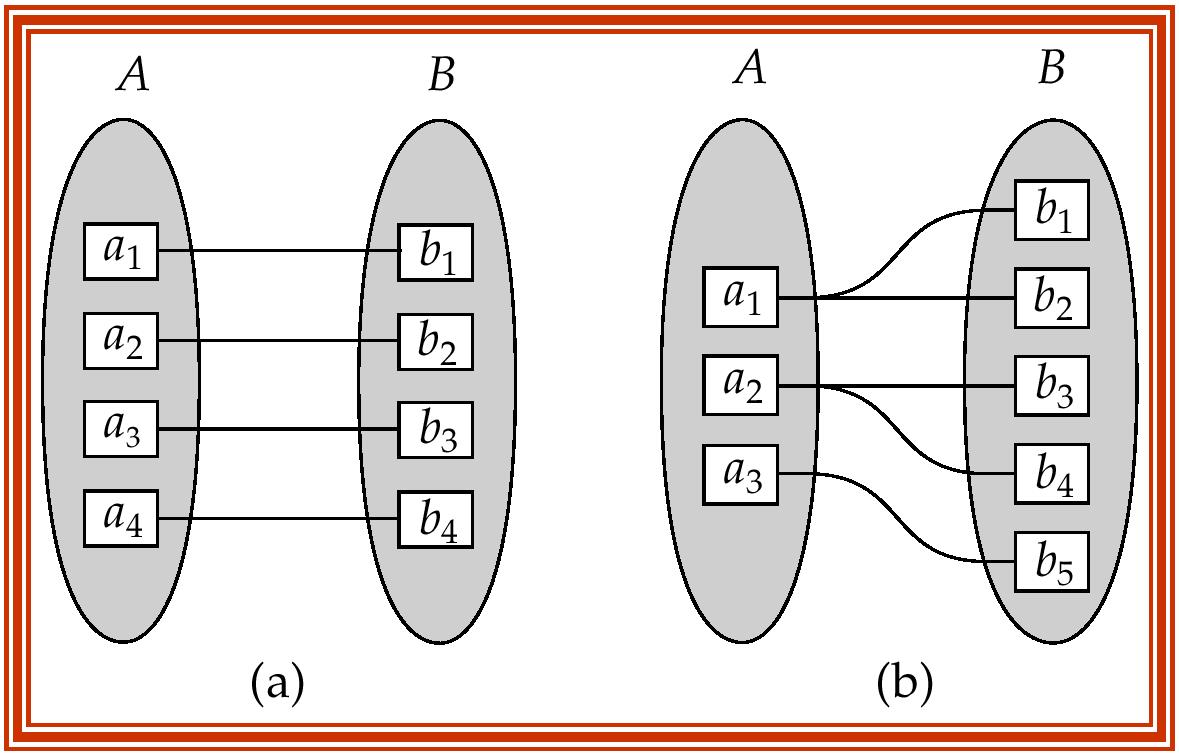
Contoh : misal seorang pegawai bank bekerja (bertanggung jawab) dalam beberapa cabang, dengan tugas yang berlainan dalam setiap cabang yang berlainan pula. Maka disini terdapat *relationship set ternary* antara entity sets pegawai (*employee)*, tugas (*job)* dan cabang(*branch)*. *Relationships* antara lebih dari dua *entity sets* sangat jarang.

1. Mapping Cardinalities

Mengungkap jumlah entitas ke entitas yang lain bisa dihubungkan melalui *relationship set*. Cardinalitas pemetaan paling banyak digunakan dalam menggambarkan *relationship sets* biner. Untuk *relationship set* biner cardinalitas pemetaan harus merupakan salah satu dari tipe berikut:

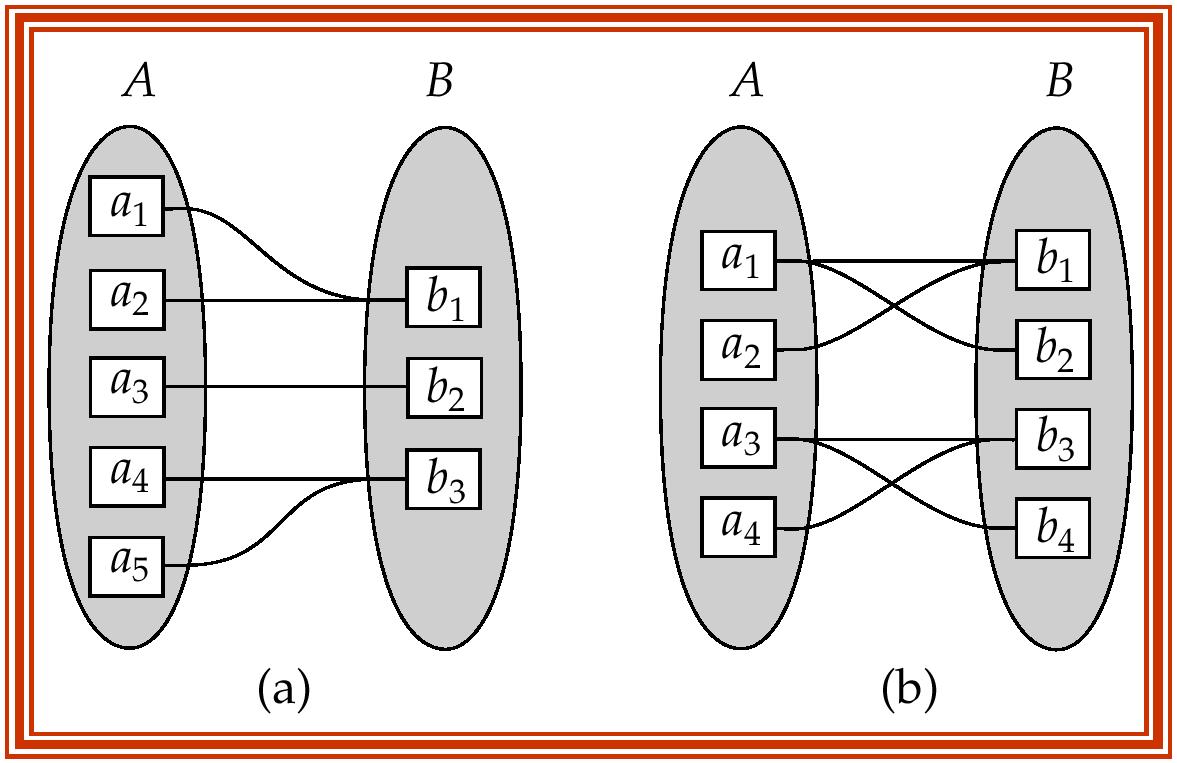
* 1. *One to one* (satu ke satu)
  2. *One to many* (satu ke banyak)
  3. *Many to one* (banyak ke satu)
  4. *Many to many* (banyak ke banyak)

14



*One to one* *One to many*

Catatan : Beberapa elemen di A dan B mungkin tidak dipetakan ke elemen manapun di himpunan lain.



*Many to one* *Many to many*

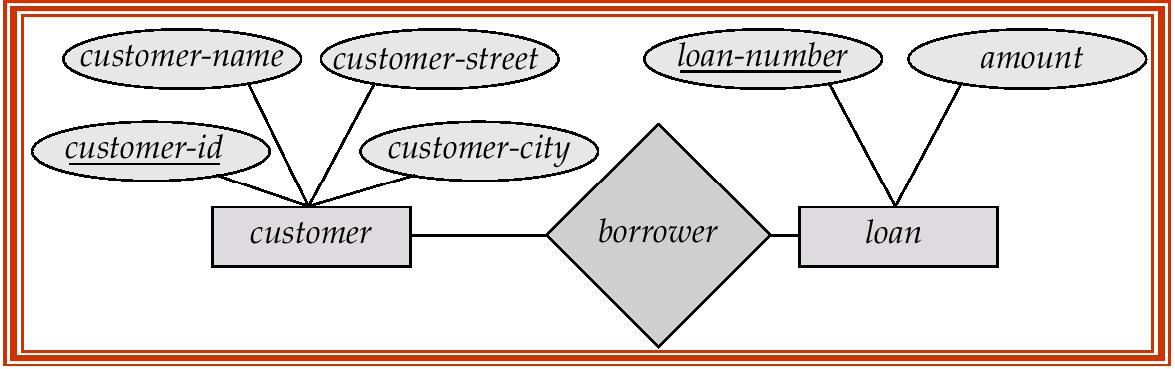
Catatan : Beberapa elemen di A dan B mungkin tidak dipetakan pada elemen mana pun di himpunan lain. *Mapping Cardinalities* mempengaruhi *ER Design*

1. Entity Relationship Diagram

Mencerminkan model database: hubungan antara entities (tabel-tabel) dan relationships (hubungan-hubungan) di antara entities tersebut.

15

Contoh ERD:



* + - ***Rectangles*** melambangkan set-set entitas .
    - ***Diamonds*** melambangkan set-set hubungan (*relationship*) .
    - ***Lines*** menghubungkan atribut dengan set-set entitas serta set-set entitas dengan set-set hubungan (*relationship*).
    - ***Ellipses*** mewakili*attributes*
      * ***Double ellipses*** mewakili atribut*multivalued*.
      * ***Dashed ellipses*** menunjukkan sifat yang didapat..
      * ***Underline*** menunjukkan sifat kunci pokok (*Primary key*)

1. Aturan untuk Rancangan Database
   1. Tiap baris harus berdiri sendiri (independent): tidak tergantung baris-baris yang lain, dan urutan baris tidak mempengaruhi model database.
   2. Tiap baris harus unik: tidak boleh ada 2 atau lebih baris yang sama persis.
   3. Kolom harus berdiri sendiri (independent): tidak tergantung kolom-kolom yang lain, dan urutan kolom tidak mempengaruhi model database.
   4. Nilai tiap kolom harus berupa satu kesatuan: tidak berupa sebuah daftar.
2. Tahap Pembuatan Database

a. Tahap 1 : Tentukan Entitas

Tentukan entities (object-object dasar) yang perlu ada di *database*.

Sifat-sifat *entity*:

* Signifikan: memang perlu disimpan di *database*
* Umum: tidak menunjuk pada sesuatu yang khusus
* Fundamental: dapat berdiri sendiri sebagai *entity* yang dasar dan independent
* Unitary: merupakan satu kesatuan yang tidak dapat dipecah lagi

16

1. Tahap 2 : Tentukan Atribut

Tentukan attributes (sifat-sifat) masing-masing *entity* sesuai kebutuhan database :

* + Tentukan sifat-sifat (*fields* atau kolom) yang dimiliki tiap *entity*, serta tipe datanya.
  + *Attribute* yang sesuai harus:
    1. Signifikan: memang penting dan perlu dicatat di dalam database
    2. Bersifat langsung (*direct*), bukan *derived*. Contoh *attribute direct*: tanggal\_lahir. Contoh *attribute derived*: umur
  + Tentukan *attribute* yang menjadi *Primary Key* untuk *entity* yang bersangkutan.
  + Jika satu *attribute* tidak cukup, gabungan beberapa *attribute* bisa menjadi

*Composite Primary Key*.

* + Jika *Composite Primary Key* banyak (lebih dari 3 *attribute*), sebaiknya menambahkan *attribute* buatan yang menjadi *Primary Key* yang tunggal.

1. Tahap 3 : Tentukan Relasi

Tentukan relationships (hubungan-hubungan) di antara entities tersebut :

* + Tentukan jenis hubungan di antara *entity* yang satu dengan entities yang lain.
  + Macam hubungan ada 3:
    1. *One-to-one* (1:1)
    2. *One-to-many* (1:n)
    3. *Many-to-many* (m:n)
  + Dalam membentuk hubungan di antara 2 entities, tentukan *attribute* mana yang digunakan untuk menghubungkan kedua entities tersebut.
  + Tentukan *entity* mana yang menjadi tabel utama, dan *entity* mana yang menjadi tabel kedua.
  + *Attribute* (dari tabel utama) yang menghubungkannya dengan tabel keduamenjadi *Foreign Key* di tabel kedua.

1. Tahap 4 : Pembuatan ERD

 Buat *Entity Relationship Diagram* (ERD) berdasarkan hasil dari Tahap

1- 3.

* + Ada berbagai macam notasi untuk pembuatan ERD.
  + Anda bisa menggunakan *software* khusus untuk menggambar ERD.

1. Tahap 5 : Normalisasi Basis Data
2. Tahap 6 : Implementasi Basis Data

17

1. **Alat dan Bahan** 
   1. Perangkat komputer dengan OS Windows 7.
   2. Modul Praktikum Sistem Basis Data
2. **Langkah-langkah Praktikum**

Rancanglah basis data untuk menyelesaikan permasalahan berikut ini.

Suatu perusahaan software diminta membuatkan basis data yang akan menangani data-data perbankan. Data-data yang akan ditanganinya adalah: data pribadi mengenai nasabah, data account deposit yang dimiliki oleh nasabah, cabang bank di mana nasabah membuka depositnya, dan data transaksi yang dilakukan nasabah. Nasabah boleh mempunyai lebih dari satu account deposit, dan satu account deposit boleh dimiliki oleh lebih dari satu nasabah sekaligus (*joint account*).

Langkah-langkah perancangan database perbankan :

* 1. Menentukan entities (object-object dasar) yang perlu ada di database.
     + **nasabah**: menyimpan semua data pribadi semua nasabah
     + **rekening**: menyimpan informasi semua rekening yang telah dibuka
     + **cabang\_bank**: menyimpan informasi tentang semua cabang bank
     + **transaksi**: menyimpan informasi tentang semua transaksi yang telah terjadi
  2. Menentukan attributes (sifat-sifat) masing-masing *entity* sesuai kebutuhan database
     + **nasabah**:

o id\_nasabah: nomor id untuk nasabah (integer) PK

o nama\_nasabah: nama lengkap nasabah (varchar(45))

o alamat\_nasabah: alamat lengkap nasabah (varchar(255))

* + - **rekening**:

o no\_rekening: nomor rekening (integer) PK

o pin : personal identification number (varchar(10)) o saldo: jumlah saldo rekening dalam Rp (integer)

* + - **cabang\_bank**:

o kode\_cabang: kode untuk cabang bank (varchar(10)) PK o nama\_cabang: nama lengkap cabang bank (varchar(20))

o alamat\_cabang: alamat lengkap cabang bank (varchar(255))

* + - **transaksi:**

o no\_transaksi: nomor transaksi (integer) PK

o jenis\_transaksi: kredit atau debit (varchar(10)) o tanggal: tanggal terjadinya transaksi (date)

o jumlah: besarnya transaksi dalam Rp (integer)

18

3. Menetukan relationship (hubungan) antar entitas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **nasabah** | **rekening** | **cabang\_bank** | **transaksi** |
|  |  |  |  |  |
| **nasabah** | - | m:n | - | 1:n |
|  |  |  |  |  |
| **rekening** |  | - | n:1 | 1:n |
|  |  |  |  |  |
| **cabang\_bank** |  |  | - | - |
|  |  |  |  |  |
| **transaksi** |  |  |  | - |
|  |  |  |  |  |

**Hubungan**

* **nasabah** memiliki **rekening**:

o Tabel utama: **nasabah, rekening**

o Tabel kedua: **nasabah\_has\_rekening** o Relationship: Many-to-many (m:n)

o Attribute penghubung: **id\_nasabah, no\_rekening** (FK **id\_nasabah,** **no\_rekening** di **nasabah\_has\_rekening**)

* **nasabah** melakukan **transaksi**:o Tabel utama: **nasabah**

o Tabel kedua: **transaksi**

o Relationship: One-to-many (1:n)

o Attribute penghubung: **id\_nasabah** (FK **id\_nasabah** di **transaksi**)

* **cabang\_bank** menangani **rekening**:o Tabel utama: **cabang\_bank**

o Tabel kedua: **rekening**

o Relationship: One-to-many (1:n)

o Attribute penghubung: **kode\_cabang** (FK **kode\_cabang** di **rekening**)

* **rekening** terlibat dalam **transaksi**:o Tabel utama: **rekening**

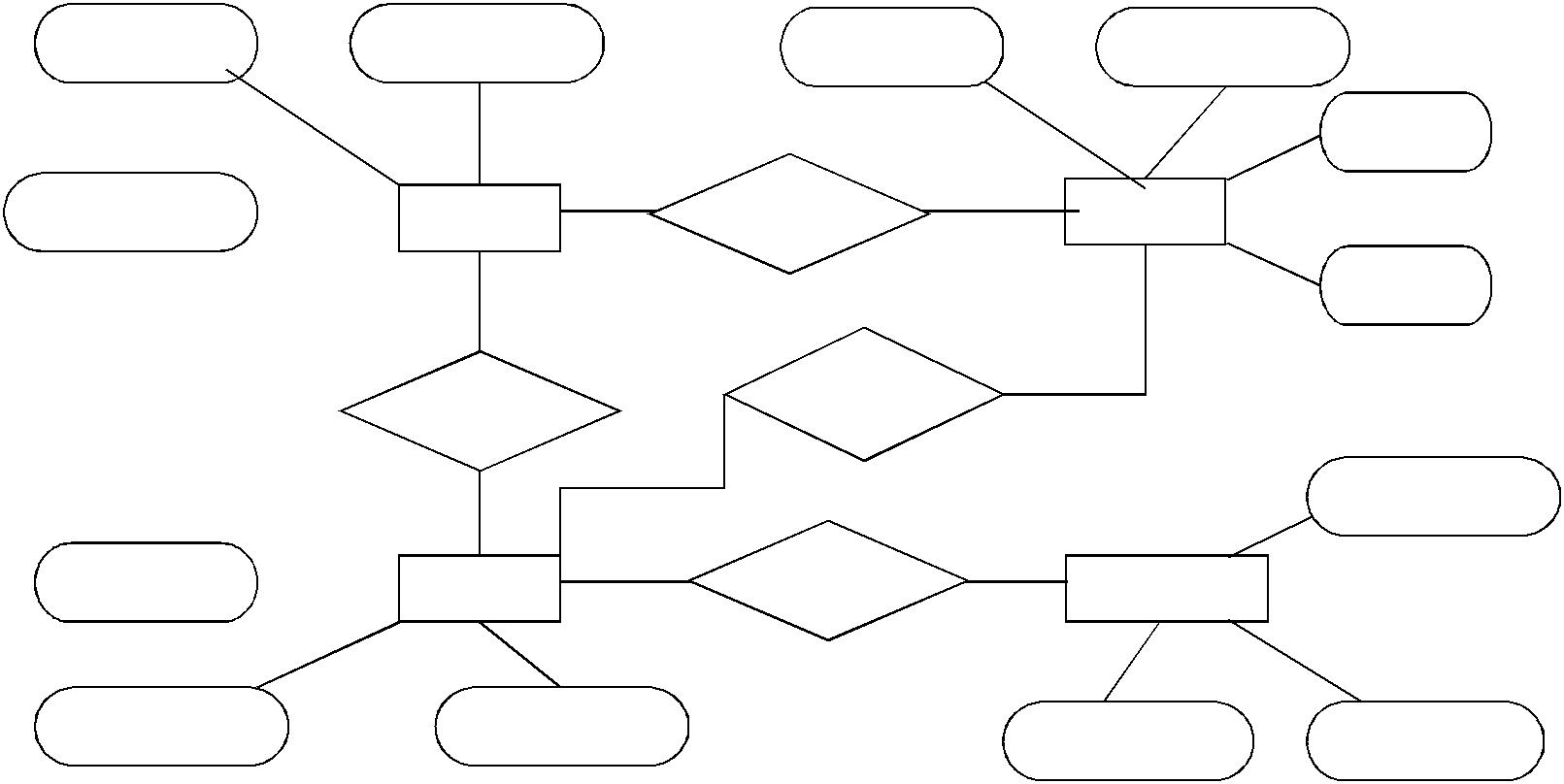
o Tabel kedua: **transaksi**

o Relationship: One-to-many (1:n)

o Attribute penghubung: **no\_rekening** (FK **no\_rekening** di **transaksi**)

19

4. Menggambar ERD Diagram



ID\_nasabah nama\_nasabah No\_transaksi jenis\_nasabah

tanggal

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| alamat\_nasabah |  |  | melakukan | Transaksi |  |
|  | Nasabah |  |
|  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

jumlah

melibatkan

memiliki

Kode\_cabang

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No\_rekening |  | Rekening | ditangani | Cabang\_bank |  |
|  |  |
|  |  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| pin | saldo | Alamat\_cabang | Nama\_cabang |  |
|  |  |  |

1. **Tugas** 
   1. Buatlah rancangan sebuah database untuk menangani data-data kuliah. Data-data yang akan ditanganinya adalah: data pribadi mengenai mahasiswa, data pribadi mengenai dosen, data mata kuliah dan data ruang kelas. Mahasiswa boleh mengambil lebih dari satu mata kuliah, dan satu mata kuliah boleh diambil oleh lebih dari satu mahasiswa sekaligus (*joint account*). Buatlah ER Diagram manual untuk kasus tersebut dari tahap 1 sampai tahap 4!
   2. Ambil contoh sembarang database (harus berbeda untuk setiap mahasiswa). Buatlah rancangan ER Diagram manual database tersebut dari tahap 1 sampai tahap 4, dengan ketentuan database minimal mengandung 4 buah entitas.

20

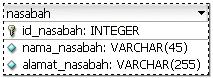
**Modul 3**

**Perancangan E-R Diagram dengan DBDesigner**

1. **Tujuan** 
   1. Mahasiswa mampu merancang basis data melalui tahap-tahap perancangannya.
   2. Mahasiswa mampu mewujudkan hasil perancangan basis data ke dalam diagram E-R menggunakan DBDesigner.
2. **Landasan Teori** 
   1. Landasan teori pada modul Bab 3.
3. **Alat dan Bahan** 
   1. Komputer dengan sistem operasi Windows XP
   2. Program aplikasi DBDesigner
   3. Modul Praktikum Sistem Berkas dan Basis Data
4. **Langkah-langkah Praktikum** 
   * 1. Menggambar ER Diagram dengan menggunakan DB Designer :
        1. Jalankan program aplikasi DB Designer
        2. Klik button *new table*   kemudian klik pada area kerja sehingga akan menghasilkan tabel baru.
        3. Double klik pada tabel baru untuk membuka tabel editor, ganti nama pada table name dengan nama *nasabah*, kemudian isikan atribut tabel dengan data sperti pada langkah nomor 2 yaitu sebagai berikut :

|  |  |
| --- | --- |
| **Coloumn Name** | **Data Type** |
|  |  |
| id\_nasabah | Integer |
|  |  |
| nama\_nasabah | Varchar(45) |
|  |  |
| alamat\_nasabah | Varchar(255) |
|  |  |

1. Klik  pada *coloumn name* id\_nasabah untuk mengeset id\_nasabah menjadi *primary key* sehingga berubah menjadi .
2. Klik  untuk menutup table editor sehingga tabel nasabah menjadi :



1. Lakukan langkah b sampai e untuk membuat tabel *rekening*, *cabang\_bank* dan *transaksi*.

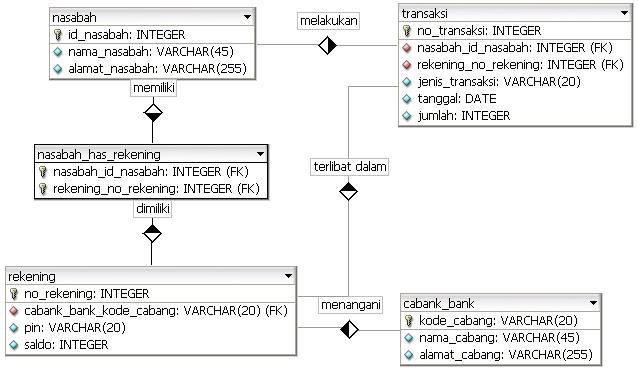
21

1. Setelah semua tabel dibuat, hubungkan setiap tabel dengan tabel lain dengan button sebagai berikut :

|  |  |
| --- | --- |
| **Button** | **Fungsi Relationship** |
|  |  |
|  | 1:n (*one to many*) |
|  |  |
|  | 1:1 (*one to one*) |
|  |  |
|  | n:m (*many to many*) |
|  |  |

Keterangan : klik salah satu button kemudian klik tabel yang akan dihubungkan.

1. Ubah nama relationship dengan membuka relationship editor, sehingga setelah selesai hasil akhir manjadi seperti berikut :



1. **Tugas**

Dari tugas pada modul 3 soal nomer 1 dan 2, buatlah rancangan basis data dari tahap 1 sampai tahap 4. Gunakan program DB Designer untuk membuat ER Diagram.

22

**Modul 4**

**Data Definition Language (DDL)**

1. **Tujuan** 
   1. Mahasiswa mampu membuat struktur tabel-tabel basis data berdasarkan perancangan di modul 3 dan mengimplementasikan tabel-tabel tersebut di basis data.
2. **Landasan Teori**

Basis data dapat diimplementasikan berdasarkan E-R diagram yang telah dibuat. Implementasi database bisa:

* + 1. Secara manual (dengan perintah SQL ‘CREATE TABLE’)
    2. Secara semi-manual dengan bantuan client berbasis GUI (MySQL Front, PgAccess, phpPgAdmin, dst.)
    3. Secara otomatis dengan CASE Tools (DBDesigner)

 Entity dan Relationships

Macam *Relationships* :

1. *One-to-One* : menggunakan notasi 1:1
   1. Satu *record* di *entity* pertama berhubungan dengan hanya satu *record* di *entity* kedua, dan demikian pula sebaliknya.
   2. Entity mana saja bisa menjadi tabel utama, sesuai dengan situasi/kebutuhan/analisis.
   3. *Primary Key* (PK) dari tabel utama menjadi *Foreign Key* (FK) di tabel kedua.
2. *One-to-Many* (atau *Many-to-One*) : menggunakan notasi 1:n
   1. Jika A terhadap B mempunyai *one-to-many relationship* (atau B terhadap A mempunyai *many-to-one relationship*): satu *record* di A bisa berhubungan dengan banyak *record* di B, tetapi satu *record* di B berhubungan dengan hanya satu *record* di A.
   2. *Entity* di sisi *One* dalam hubungan *One-to-Many* menjadi tabel utama, dan entity disisi *Many* menjadi tabel kedua.
   3. PK dari tabel utama menjadi FK di tabel kedua.
   4. Di contoh sebelumnya, A menjadi tabel utama dan B menjadi tabel kedua. Maka, ada FK di B yang merupakan PK di A.
3. *Many-to-Many* : menggunakan notasi m:n
   1. Jika A terhadap B mempunyai *many-to-many relationship*: satu *record* di A bisa berhubungan dengan banyak *record* di B, dan demikian pula sebaliknya.

23

* + - 1. Dalam implementasi database, harus ada sebuah tabel perantara di antara A dan B. A dan B menjadi tabel utama, dan tabel perantara menjadi tabel kedua.
      2. PK dari A menjadi FK di tabel perantara (tabel kedua), dan PK dari B juga menjadi FK di tabel kedua. Gabungan semua FK di tabel kedua menjadi *Composite* PK untuk tabel kedua.
      3. A terhadap tabel perantara mempunyai 1:n *relationship*.
      4. B terhadap tabel perantara juga mempunyai 1:n *relationship*.
  + Tahap Pembuatan Tabel
    1. Membuat semua tabel yang paling utama (yang tidak memiliki FK).
    2. Membuat semua tabel yang berhubungan langsung (atau memiliki relationship) dengan tabel yang dibuat di tahap sebelumnya. Mulailah secara urut dari tabel dengan jumlah FK yang paling sedikit ke yang paling banyak.
    3. Ulangi tahap 2 sampai semua tabel selesai dibuat.
* Implementasi Manual

Contoh untuk menentukan *Primary Key* (PK):

CREATE TABLE dosen (nip INTEGER PRIMARY KEY, nama\_dosen VARCHAR(45), alamat\_dosen VARCHAR(255));

CREATE TABLE ruang (kode\_ruang VARCHAR(20) PRIMARY KEY, lokasi\_ruang VARCHAR(255), kapasitas\_ruang INTEGER);

Contoh untuk menentukan *Foreign Key* (FK):

CREATE TABLE mahasiswa(nim INTEGER PRIMARY KEY, nip INTEGER REFERENCES dosen (nip), nama\_mhs VARCHAR(45), alamat\_mhs VARCHAR(255))

Contoh untuk banyak FK:

CREATE TABLE mata\_kuliah(kode\_mk INTEGER PRIMARY KEY, nip INTEGER REFERENCES dosen (nip), kode\_ruang VARCHAR(20) REFERENCES ruang (kode\_ruang), nama\_mk VARCHAR(45), deskripsi\_mk VARCHAR(255));

Contoh untuk menentukan *composite* PK:

CREATE TABLE mhs\_ambil\_mk(nim INTEGER REFERENCES mahasiswa (nim), kode\_mk INTEGER REFERENCES mata\_kuliah(kode\_mk), PRIMARY KEY(nim, kode\_mk));

24

* Referential Integrity
  1. Integritas database mengacu pada hubungan antar tabel melalui *Foreign Key* yang bersangkutan.
  2. Pada *insert*, *record* harus dimasukkan di tabel utama dahulu, kemudian baru di tabel kedua.
  3. Pada *delete*, record harus dihapus di tabel kedua dahulu, kemudian baru di tabel utama.
  4. Secara *default*, kebanyakan DBMS yang ada akan menolak *insert* atau *delete* yang melanggar integritas database.
* Advance Create Table Option
  1. Default

Untuk menentukan nilai default kolom jika tidak ada data yang di-insert untuk kolom itu:

CREATE TABLE mahasiswa(nim integer PRIMARY KEY, nama\_mhs varchar(45), fakultas varchar(5) DEFAULT ‘FKI’);

* 1. Not Null
     + Untuk membatasi agar nilai kolom tidak boleh NULL:

CREATE TABLE ruang(kode\_ruang varchar(20) PRIMARY KEY, lokasi\_ruang varchar(255) NOT NULL, kapasitas\_ruang integer NOT NULL);

* + - Jika kolom ditentukan NOT NULL, maka *insert* harus memasukkan nilai untuk kolom tersebut.
    - Bisa menggunakan DEFAULT sehingga nilai kolom ditambahkan secara otomatis.
  1. Unique
     + Untuk memastikan bahwa nilai kolom unik:

CREATE TABLE mata\_kuliah(kode\_mk integer PRIMARY KEY, nama\_mk varchar(45) UNIQUE);

* + - Untuk multikolom yang unik:

CREATE TABLE dosen(nip integer PRIMARY KEY, nama\_dosen varchar(45), alamat\_dosen varchar(255), UNIQUE (nama\_dosen, alamat\_dosen));

* 1. Check
     + Untuk membatasi nilai kolom, misalnya:

CREATE TABLE produk(kode\_produk integer PRIMARY KEY, nama\_produk varchar(45), harga integer, CHECK (harga <= 100000 AND kode\_produk > 100 ));

25

* + - * *Check* di atas membatasi bahwa harga harus maksimal Rp 100000, dankode\_produk harus di atas 100.
    1. Penentuan Referential Integrity
       - Contoh:

CREATE TABLE pemasok(kode\_pemasok integer PRIMARY KEY, nama\_pemasok varchar(45), kode\_produk integer REFERENCES produk ON DELETE CASCADE ON UPDATE CASCADE);

* + - * Untuk contoh di atas, jika ada *update* atau *delete* di tabel utama, maka tabel kedua secara otomatis disesuaikan.
    1. Action
       - NO ACTION atau RESTRICT: *update* atau *delete* tidak dilakukan. Ini merupakan pilihan *default*.
       - CASCADE: nilai kolom di tabel kedua disesuaikan dengan nilai kolom di tabel utama
       - SET NULL: nilai kolom di tabel kedua dijadikan NULL
       - SET DEFAULT: nilai kolom di tabel kedua dijadikan nilai DEFAULT (nilai DEFAULT harus ditentukan pada waktu pembuatan tabel).
    2. Autoincrement
       - Untuk fitur *autoincrement*, gunakan “serial”:

CREATE TABLE nasabah(id\_nasabah serial PRIMARY KEY, nama\_nasabah varchar(45));

* + - * Untuk contoh di atas, id\_nasabah tidak perlu di-*insert*, karena database secara otomatis akan menambahkannya secara urut.
      * “serial” hanya bisa dari 1 sampai 232.
      * Jika tidak cukup, gunakan “bigserial” yang bisa dar i 1 sampai 264.
      * Jika kolom menggunakan “serial” atau “bigserial”, j angan sekali-kali memasukkan nilai kolom tersebut secara manual! Biarkan database menambahkannya sendiri secara urut!
      * Penghapusan *record* tidak akan mempengaruhi urutan untuk serial dan *bigserial*. Nilai untuk kolom yang menggunakan *serial/bigserial* akan selalu bertambah 1, tidak akan pernah kembali mundur.

1. **Alat dan Bahan** 
   1. Komputer dengan sistem operasi Windows XP
   2. Program aplikasi XAMPP dengan PhpMyAdmin
   3. Modul Praktikum Sistem Berkas dan Basis Data

26

1. **Langkah-langkah Praktikum** 
   1. Jalankan XAMPP Control Panel.
   2. Jalankan server Apache dan MySQL.
   3. Buka Command Prompt dan login sebagai root ke MySQL seperti di langkah pada Modul 1.
   4. Buat database baru dengan perintah berikut ini. create database perbankan;
   5. Hubungkan ke dalam database yang telah dibuat dengan perintah berikut. Sehingga akan muncul pemberitahuan “database changed”.

use perbankan;

* 1. Membuat tabel nasabah dengan script berikut.

CREATE TABLE nasabah ( id\_nasabah INTEGER PRIMARY KEY,

nama\_nasabah VARCHAR(45) NOT NULL, alamat\_nasabah VARCHAR(255) NOT NULL

);

* 1. Membuat tabel cabang\_bank dengan script berikut.

CREATE TABLE cabang\_bank ( kode\_cabang VARCHAR(20) PRIMARY KEY,

nama\_cabang VARCHAR(45) UNIQUE NOT NULL, alamat\_cabang VARCHAR(255) NOT NULL

);

* 1. Membuat tabel rekening dengan script berikut.

CREATE TABLE rekening ( no\_rekening INTEGER PRIMARY KEY,

kode\_cabangFK VARCHAR(20) REFERENCES cabang\_bank(kode\_cabang) ON DELETE CASCADE ON UPDATE CASCADE,

pin VARCHAR(20) DEFAULT '1234' NOT NULL, saldo INTEGER DEFAULT 0 NOT NULL

);

* 1. Membuat tabel transaksi dengan script berikut ini.

CREATE TABLE transaksi ( no\_transaksi SERIAL PRIMARY KEY,

id\_nasabahFK INTEGER REFERENCES nasabah(id\_nasabah) ON DELETE SET NULL ON UPDATE CASCADE, no\_rekeningFK INTEGER REFERENCES rekening(no\_rekening) ON DELETE SET NULL ON UPDATE CASCADE,

27

jenis\_transaksi VARCHAR(20) DEFAULT 'debit' NOT NULL, tanggal DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP, jumlah INTEGER NOT NULL CHECK (jumlah>=20000)

);

1. Membuat tabel nasabah\_has\_rekening dengan script berikut ini.

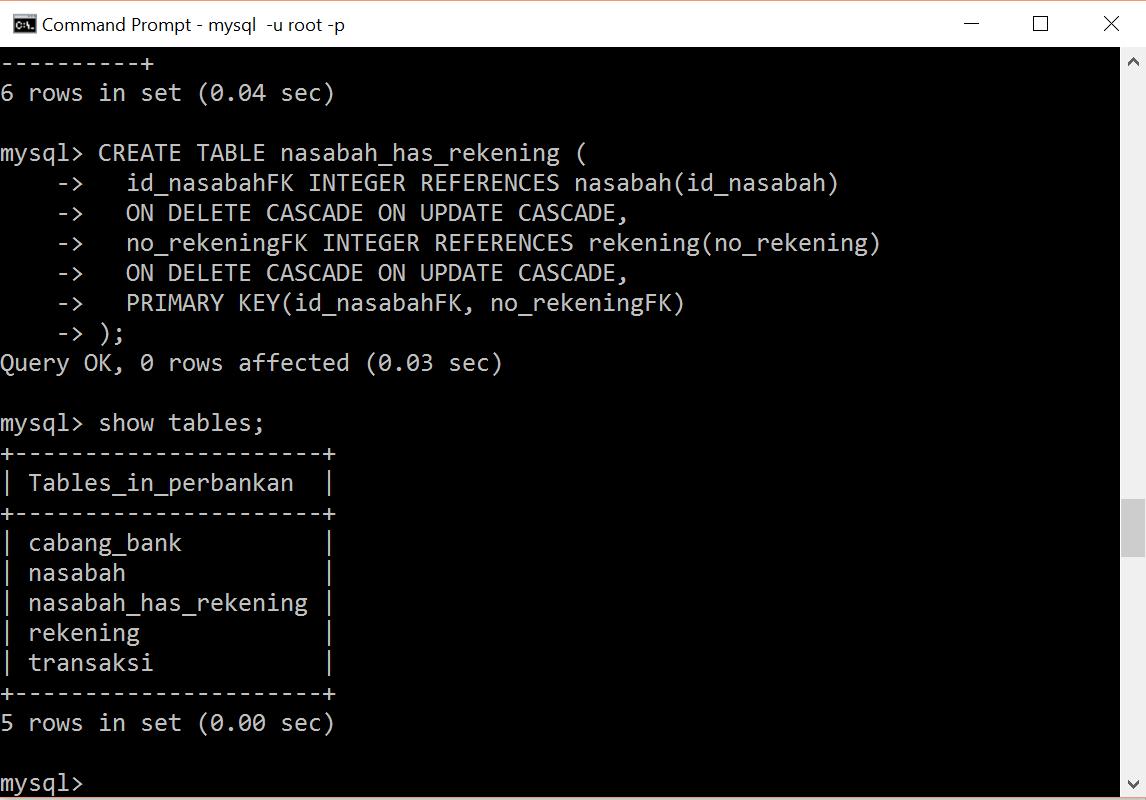
CREATE TABLE nasabah\_has\_rekening (

id\_nasabahFK INTEGER REFERENCES nasabah(id\_nasabah) ON DELETE CASCADE ON UPDATE CASCADE, no\_rekeningFK INTEGER REFERENCES rekening(no\_rekening) ON DELETE CASCADE ON UPDATE CASCADE,

PRIMARY KEY(id\_nasabahFK, no\_rekeningFK)

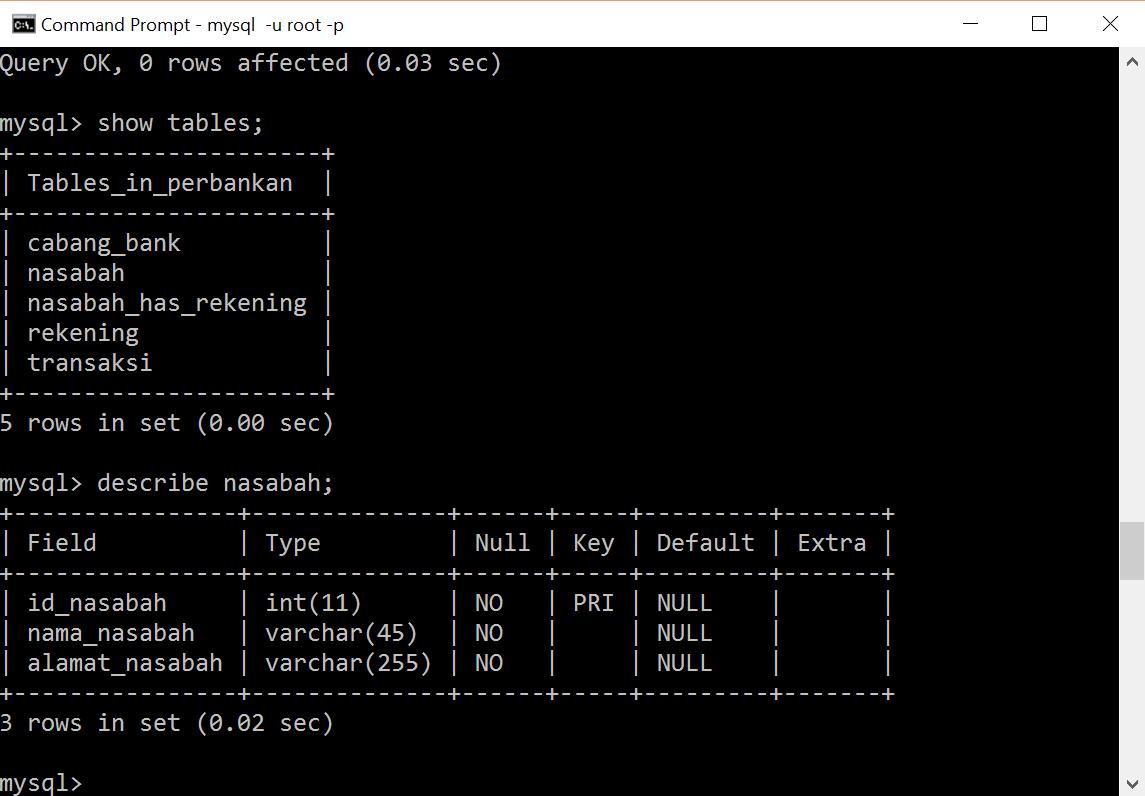
);

1. Untuk mengecek hasil pembuatan database gunakan perintah show tables;



28

1. Kemudian untuk melihat struktur tiap tabel dapat dilakukan dengan perintah describe. Misalkan untuk meihat struktur tabel nasabah dapat dilakukan dengan perintah describe nasabah;



1. **Tugas**

Implementasikan hasil rancangan database yang menangani data kuliah pada tugas modul 2 ke dalam program mysql.

29

**Modul 5**

**Data Manipulation Language (DML)**

1. **Tujuan** 
   1. Mahasiswa mampu memahami dan mengisi tabel dalam database.
   2. Mahasiswa mampu memahami dan memanipulasi data dalam database.
2. **Landasan Teori**

Data Manipulation Language (DML) adalah suatu statement yang dijalankan pada saat kita memerlukan :

* + - penambahan baris baru pada table.
    - memodifikasi baris yang ada pada table.
    - menghapus baris yang ada pada table.

DML Statement identik dengan operasi INSERT, MODIFY dan DELETE. Istilah Transaksi

mengandung pengertian kumpulan Statement DML yang membentuk suatu fungsi tertentu.

1. Menambahkan baris baru ke dalam tabel (INSERT)

Menambahkan baris baru ke dalam table menggunakan perintah INSERT. INSERT INTO table [(column [, column …] ) ]

VALUES (value [, value…] );

Contoh :

INSERT INTO nasabah(id\_nasabah, nama\_nasabah, alamat\_nasabah)

VALUES(1, ’Sutopo’,’ Jl. Jendral Sudirman 12’);

1. Perubahan data dalam tabel (UPDATE).

Untuk memodifikasi baris data yang ada pada table digunakan perintah UPDATE. Sintak dari perintah UPDATE :

UPDATE table

SET column = value [, column = value, …] [WHERE condition];

Contoh :

UPDATE nasabah

SET alamat\_nasabah = ‘Jalan Jenderal Sudirman No.12’ WHERE nama\_nasabah = ‘Sutopo’;

1. Menghapus data dalam tabel (DELETE).

Baris data yang ada pada table dapat dihapus dengan menggunakan perintah DELETE. DELETE FROM table

[WHERE condition]

30

Contoh :

DELETE FROM nasabah

WHERE nama\_nasabah = ‘Sutopo’;

1. Transaksi dalam database

Jika DML berkaitan dengan manipulasi data pada table, maka DDL (Data Definition Language) berkaitan dengan pendefinisian table, sedangkan DCL (Data Control Language) berkaitan kontrol transaksi. DDL dan DCL secara otomatis akan di-commit (dilakukan perubahan secara permanen) pada akhir dari transaksi. Ada 2 statement DCL yang penting yaitu COMMIT dan ROLLBACK, selain dari itu ada SAVEPOINT. Perintah COMMIT menandai perubahan secara permanen. pada data. Sedangkan ROLLBACK mengembalikan keadaan sesuai dengan titik (keadaan) yang ditandai dengan SAVEPOINT, atau jika ROLLBACK tidak diberi parameter maka keadaan akan dikembalikan pada titik perubahan yang terakhir. Transaksi akan diproses secara implicit atau dilakukan operasi COMMIT secara otomatis, untuk keadaan berikut :

* + Setelah Statement DDL diberikan.
  + Setelah Statement DCL diberikan.
  + Proses exit secara normal dari SQL\*PLUS.

Sedangkan perintah ROLLBACK secara otomatis akan dijalankan jika terjadi kondisi yang

abnormal atau terjadi system failure.

1. **Alat dan Bahan** 
   1. Komputer dengan sistem operasi Windows 7
   2. Program aplikasi XAMPP dengan PhpMyAdmin
   3. Modul Praktikum Sistem Berkas dan Basis Data
2. **Langkah-langkah praktikum** 
   1. Jalankan XAMPP Control Panel.
   2. Jalankan server Apache dan MySQL.
   3. Buka Command Prompt dan login sebagai root ke MySQL seperti di langkah pada Modul 1.
   4. Pilih database perbankan dengan perintah “use perba nkan”. Sehingga akan muncul pemberitahuan “database changed”
   5. Kemudian masukan data-data berikut ini ke dalam database perbankan.

31

**Tabel : nasabah**

|  |  |  |
| --- | --- | --- |
| **id\_nasabah** | **nama\_nasabah** | **alamat\_nasabah** |
|  |  |  |
| 1 | Sutopo | Jl. Jendral Sudirman 12 |
|  |  |  |
| 2 | Maryati | Jl. MT. Haryono 31 |
|  |  |  |
| 3 | Suparman | Jl. Hasanudin 81 |
|  |  |  |
| 4 | Kartika Padmasari | Jl. Manggis 15 |
|  |  |  |
| 5 | Budi Eko Prayogo | Jl. Kantil 30 |
|  |  |  |
| 6 | Satria Eka Jaya | Jl. Slamet Riyadi 45 |
|  |  |  |
| 7 | Indri Hapsari | Jl. Sutoyo 5 |
|  |  |  |
| 8 | Sari Murti | Jl. Pangandaran 11 |
|  |  |  |
| 9 | Canka Lokananta | Jl. Tidar 86 |
|  |  |  |
| 10 | Budi Murtono | Jl. Merak 22 |
|  |  |  |

**Tabel : cabang\_bank**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **kode\_cabang** | | **nama\_cabang** | |  |  | **alamat\_cabang** | |
|  |  |  |  |  |  | |  |
| BRUS | | Bank Rut Unit Surakarta | |  | Jl. Slamet Riyadi 18 | | |
|  |  |  |  |  |  | |  |
| BRUM | | Bank Rut Unit Magelang | |  | Jl. P. Tendean 63 | | |
|  |  |  |  |  |  | |  |
| BRUB | | Bank Rut Unit Boyolali | |  | Jl. Ahmad Yani 45 | | |
|  |  |  |  |  |  | |  |
| BRUK | | Bank Rut Unit Klaten | |  | Jl. Suparman 23 | | |
|  |  |  |  |  |  | |  |
| BRUY | | Bank Rut Unit Yogyakarta | |  | Jl. Anggrek 21 | | |
|  |  |  |  |  |  | |  |
| BRUW | | Bank Rut Unit Wonogiri | |  | Jl. Untung Suropati 12 | | |
|  |  |  |  |  |  |  |  |
| **Tabel : rekening** | |  |  |  |  |  |  |
|  |  |  |  |  | |  |  |
| **no\_rekening** |  | **kode\_cabang** |  | **pin** | |  | **saldo** |
|  |  |  |  |  | |  |  |
| 101 |  | BRUS |  | 1111 | |  | 500000 |
|  |  |  |  |  | |  |  |
| 102 |  | BRUS |  | 2222 | |  | 350000 |
|  |  |  |  |  | |  |  |
| 103 |  | BRUS |  | 3333 | |  | 750000 |
|  |  |  |  |  | |  |  |
| 104 |  | BRUM |  | 4444 | |  | 900000 |
|  |  |  |  |  | |  |  |
| 105 |  | BRUM |  | 5555 | |  | 2000000 |
|  |  |  |  |  | |  |  |
| 106 |  | BRUS |  | 6666 | |  | 3000000 |
|  |  |  |  |  |  |  |  |
|  |  | 32 | |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 107 | BRUS | 7777 | 1000000 |
|  |  |  |  |
| 108 | BRUB | 0000 | 5000000 |
|  |  |  |  |
| 109 | BRUB | 9999 | 0 |
|  |  |  |  |
| 110 | BRUY | 1234 | 550000 |
|  |  |  |  |
| 111 | BRUK | 4321 | 150000 |
|  |  |  |  |
| 112 | BRUK | 0123 | 300000 |
|  |  |  |  |
| 113 | BRUY | 8888 | 255000 |
|  |  |  |  |

**Tabel : nasabah\_has\_rekening**

|  |  |
| --- | --- |
| **id\_nasabah** | **no\_rekening** |
|  |  |
| 1 | 104 |
|  |  |
| 2 | 103 |
|  |  |
| 3 | 105 |
|  |  |
| 3 | 106 |
|  |  |
| 4 | 101 |
|  |  |
| 4 | 107 |
|  |  |
| 5 | 102 |
|  |  |
| 5 | 107 |
|  |  |
| 6 | 109 |
|  |  |
| 7 | 109 |
|  |  |
| 8 | 111 |
|  |  |
| 9 | 110 |
|  |  |
| 10 | 113 |
|  |  |
| 8 | 112 |
|  |  |
| 10 | 108 |
|  |  |

33

**Tabel : transaksi**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **no\_trans** | **no\_reke** | **id\_nasabah** |  | **jenis\_trans** | **tanggal** | **jumlah** |
| **aksi** | **ning** |  |  | **aksi** |  |  |
|  |  |  |  |  |  |  |
| 1 | 105 | 3 |  | debit | 2009-11-10 | 50000 |
|  |  |  |  |  |  |  |
| 2 | 103 | 2 |  | debit | 2009-11-10 | 40000 |
|  |  |  |  |  |  |  |
| 3 | 101 | 4 |  | kredit | 2009-11-12 | 20000 |
|  |  |  |  |  |  |  |
| 4 | 106 | 3 |  | debit | 2009-11-13 | 50000 |
|  |  |  |  |  |  |  |
| 5 | 107 | 5 |  | kredit | 2009-11-13 | 30000 |
|  |  |  |  |  |  |  |
| 6 | 104 | 1 |  | kredit | 2009-11-15 | 200000 |
|  |  |  |  |  |  |  |
| 7 | 110 | 9 |  | kredit | 2009-11-15 | 150000 |
|  |  |  |  |  |  |  |
| 8 | 102 | 5 |  | debit | 2009-11-16 | 20000 |
|  |  |  |  |  |  |  |
| 9 | 105 | 3 |  | kredit | 2009-11-18 | 50000 |
|  |  |  |  |  |  |  |
| 10 | 107 | 4 |  | debit | 2009-11-19 | 100000 |
|  |  |  |  |  |  |  |
| 11 | 103 | 2 |  | debit | 2009-11-19 | 100000 |
|  |  |  |  |  |  |  |
| 12 | 104 | 1 |  | debit | 2009-11-19 | 50000 |
|  |  |  |  |  |  |  |
| 13 | 107 | 4 |  | kredit | 2009-11-20 | 200000 |
|  |  |  |  |  |  |  |
| 14 | 105 | 3 |  | debit | 2009-11-21 | 40000 |
|  |  |  |  |  |  |  |
| 15 | 104 | 1 |  | kredit | 2009-11-22 | 100000 |
|  |  |  |  |  |  |  |
| 16 | 101 | 4 |  | kredit | 2009-11-22 | 20000 |
|  |  |  |  |  |  |  |
| 17 | 103 | 2 |  | debit | 2009-11-22 | 50000 |
|  |  |  |  |  |  |  |
| 18 | 102 | 5 |  | debit | 2009-11-25 | 50000 |
|  |  |  |  |  |  |  |
| 19 | 108 | 10 |  | debit | 2009-11-26 | 100000 |
|  |  |  |  |  |  |  |
| 20 | 106 | 3 |  | kredit | 2009-11-27 | 50000 |
|  |  |  |  |  |  |  |
| 21 | 103 | 2 |  | kredit | 2009-11-28 | 200000 |
|  |  |  |  |  |  |  |
| 22 | 105 | 3 |  | kredit | 2009-11-28 | 100000 |
|  |  |  |  |  |  |  |
| 23 | 102 | 5 |  | debit | 2009-11-30 | 20000 |
|  |  |  |  |  |  |  |
| 24 | 104 | 1 |  | debit | 2009-12-1 | 50000 |
|  |  |  |  |  |  |  |
| 25 | 103 | 2 |  | debit | 2009-12-2 | 40000 |
|  |  |  |  |  |  |  |
| 26 | 101 | 4 |  | debit | 2009-12-4 | 50000 |
|  |  |  |  |  |  |  |
| 27 | 103 | 2 |  | kredit | 2009-12-5 | 100000 |
|  |  |  |  |  |  |  |
| 28 | 102 | 5 |  | kredit | 2009-12-5 | 200000 |
|  |  |  |  |  |  |  |
|  |  |  | 34 | |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 29 | 109 | 7 | debit | 2009-12-5 | 100000 |
|  |  |  |  |  |  |
| 30 | 110 | 9 | debit | 2009-12-6 | 20000 |
|  |  |  |  |  |  |

* 1. Lakukan update untuk kasus-kasus berikut ini.
     1. Nasabah dengan nama “Indri Hapsari” pindah alamat k e “Jalan Slamet Riyadi No.34”.
     2. Cabang dengan kode “BRUW” pindah ke alamat “Jalan A . Yani No.23”.
  2. Lakukan penghapusan untuk kasus-kasus berikut.
     1. Nasabah dengan id “7” menutup rekeningnya.
     2. Cabang dengan nama\_cabang “Bank Rut Unit Magelang” menutup kantornya.

1. **Tugas**

Masukan beberapa *record* ke setiap tabel dalam database yang telah anda buat pada modul 4. Print out hasil implementasi rancangan tersebut dan analisa hasilnya. (Masing-masing 10 record untuk tabel master dan 20 record untuk tabel transaksi)

35

**Modul 6**

**Query Standar dan Variasinya**

1. **Tujuan** 
   1. Mahasiswa mampu menggunakan intruksi seleksi standard an beberapa variasi dasarnya.
2. **Landasan Teori**

*SELECT STATEMENTS* digunakan untuk menentukan atau memilih data yang akan ditampilkanketika melakukan *query* terhadap basis data. Struktur *select statements* di postgreSQL adalah sebagai berikut:

SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ] \* | expression [ AS output\_name ] [, ...]

[ FROM from\_item [, ...] ] [ WHERE condition ]

[ GROUP BY expression [, ...] ] [ HAVING condition [, ...] ]

[ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]

[ ORDER BY expression [ ASC | DESC | USING operator ] [, ...] ] [ FOR UPDATE [ OF tablename [, ...] ] ]

[ LIMIT { count | ALL } ] [ OFFSET start ]

* 1. Contoh *SELECT :*
     + Untuk melihat semua kolom dari suatu tabel:

SELECT \* FROM nasabah;

* + - Untuk melihat kolom(-kolom) tertentu:

SELECT nama\_nasabah FROM nasabah;

SELECT id\_nasabah, nama\_nasabah FROM nasabah;

* + - Secara umum:

SELECT <nama kolom, ...> FROM <nama tabel>;

36

1. Columns Alias (AS)

AS digunakan untuk mengganti nama kolom pada tampilan SELECT. Contoh:

SELECT nama\_nasabah AS “Nama Nasabah” FROM nasabah; SELECT nama\_nasabah AS “Nasabah”, alamat\_nasabah AS “Alamat Nasabah” FROM nasabah;

1. Where

Digunakan untuk membatasi hasil SELECT yang ditampilkan berdasarkan kondisi yang ditentukan.

Contoh:

SELECT nama\_nasabah FROM nasabah WHERE nama\_nasabah=‘Ali Topan’;

SELECT nama\_nasabah, alamat\_nasabah FROM nasabah WHERE id\_nasabah=2;

* + Bisa menggunakan >, <, <> (atau !=), >=, <= Gunakan AND atau OR untuk lebih dari satu kondisi:

SELECT\*FROM nasabah WHERE nama\_nasabah = ‘Rina Marsudi’ AND alamat\_nasabah=‘Jl. Kusumanegara 30’;

SELECT\*FROM nasabah WHERE nama\_nasabah=‘Ali Topan’ OR id\_nasabah = 2;

* + Pencarian NULL

Gunakan IS NULL untuk mencari NULL:

SELECT\*FROM rekening WHERE kode\_cabang IS NULL;

Gunakan IS NOT NULL untuk mencari yang tidak NULL:

SELECT\*FROM rekening WHERE kode\_cabang IS NOT NULL;

* + Pencarian String

Gunakan LIKE untuk mencari string tertentu:

SELECT\*FROM nasabah WHERE nama\_nasabah LIKE ‘Ali Topan’;

Bisa menggunakan %:

SELECT\*FROM nasabah WHERE alamat\_nasabah LIKE %negara%’;

Bisa menggunakan \_ untuk 1 huruf:

SELECT\*FROM nasabah WHERE nama\_nasabah LIKE ‘Ali T\_p\_n’;

Untuk pencarian yang *case insensitive* (tidak mempedulikan huruf besar atau kecil), gunakan ILIKE:

SELECT\*FROM nasabah WHERE nama\_nasabah ILIKE ‘% marsudi’;

37

1. Order by

Digunakan untuk mengurutkan hasil SELECT. Jenis-jenisnya adalah sebagai berikut ini : Untuk mengurutkan dari kecil ke besar:

SELECT\*FROM nasabah ORDER BY nama\_nasabah;

Untuk mengurutkan dari besar ke kecil:

SELECT\*FROM nasabah ORDER BY nama\_nasabah DESC;

Untuk melakukan pengurutan lebih dari satu kolom, pisahkan dengan tanda koma:

SELECT\*FROM nasabah\_has\_rekening ORDER BY no\_rekening, id\_nasabah;

Anda bisa menentukan DESC untuk kolom(-kolom) tertentu, misalnya:

SELECT\*FROM nasabah\_has\_rekening ORDER BY no\_rekening, id\_nasabah DESC;

SELECT\*FROM nasabah\_has\_rekening ORDER BY no\_rekening DESC, id\_nasabah;

1. Limit dan Offset
   * Digunakan untuk membatasi jumlah baris yang ditampilkan dalam SELECT. Contoh:

**Hanya menampilkan 3 baris pertama:**

SELECT\*FROM nasabah ORDER BY id\_nasabah LIMIT 3;

**Menampilkan 2 baris setelah melewati 2 baris pertama:**

SELECT\*FROM nasabah ORDER BY id\_nasabah LIMIT 2 OFFSET 2;

* + Perhatian: penggunaan LIMIT sebaiknya selalu digunakan bersama dengan ORDER BY, sehingga urutan yang ditampilkan akan selalu konsisten.
  + LIMIT dan OFFSET sangat berguna dalam tampilan yang berbasis web (melalui web browser dengan menggunakan PHP atau JSP) agar tampilan data tidak terlalu besar dan bisa lebih rapi. Tampilan data yang banyak bisa diatur dan dibagi menjadi beberapa halaman (pages).

1. Tabel Join
   * Cross Join

Menggabungkan semua record dari tabel pertama dengan semua record di tabel kedua.

Banyaknya record dari cross join = jumlah record tabel pertama X jumlah record tabel kedua

Contoh:

SELECT\*FROM rekening CROSS JOIN cabang\_bank;

* + Inner Join

38

Menghubungkan 2 (atau lebih) tabel berdasarkan attribute penghubung. Metode 1:

SELECT\*FROM rekening INNER JOIN cabang\_bank USING(kode\_cabang);

Metode 2:

SELECT\*FROM rekening INNER JOIN cabang\_bank ON rekening.kode\_cabang = cabang\_bank.kode\_cabang;

Metode 3:

SELECT \* FROM rekening NATURAL INNER JOIN cabang\_bank;

Metode 4:

SELECT \* FROM rekening, cabang\_bank WHERE rekening.kode\_cabang= cabang\_bank.kode\_cabang;

Perhatian: Untuk INNER JOIN, Anda dapat menghilangkan kata ‘INNER’. Jadi, cukup dengan kata ‘JOIN’ saja.

Dengan metode 4, jika kolom yang ingin ditampilkan ada di lebih dari 2 tabel, maka Anda harus menentukan tabel mana yang diinginkan.

Contoh:

SELECT nasabah.id\_nasabah, nama\_nasabah, no\_rekening FROM nasabah, nasabah\_has\_rekening WHERE nasabah.id\_nasabah= nasabah\_has\_rekening.id\_nasabah;

* Tabel Alias

Untuk kemudahan penulisan SQL, kita bisa membuat table alias. Contoh:

SELECT\*FROM nasabah A, nasabah\_has\_rekening B WHERE A.id\_nasabah=B.id\_nasabah;

SELECT A.id\_nasabah, nama\_nasabah, no\_rekening FROM nasabah A, nasabah\_has\_rekening B WHERE A.id\_nasabah=B.id\_nasabah;

* Distinct

Dalam table join, kadang-kadang ada informasi yang berulang. Untuk menghilangkan pengulangan tersebut, gunakan DISTINCT.

Contoh:

SELECT DISTINCT nama\_nasabah, alamat\_nasabah FROM nasabah NATURAL JOIN nasabah\_has\_rekening;

39

Perhatikan perbedaan dengan berikut:

SELECT nama\_nasabah, alamat\_nasabah FROM nasabah NATURAL JOIN nasabah\_has\_rekening;

* Right Outer Join

Menampilkan hasil join tabel pertama (sisi kiri) dengan tabel kedua (sisi kanan), serta semua *record* di tabel kedua (sisi kanan/*right*):

SELECT\*FROM rekening NATURAL RIGHT OUTER JOIN cabang\_bank;

Ketiga metode pertama yang telah disebutkan untuk INNER JOIN juga berlaku untuk RIGHT OUTER JOIN, yaitu dengan menggunakan USING, ON, atau NATURAL.

* Left Outer Join

Menampilkan hasil join tabel pertama (sisi kiri) dengan tabel kedua (sisi kanan), serta semua *record* di tabel pertama (sisi kiri/*left*):

SELECT\*FROM rekening NATURAL LEFT OUTER JOIN cabang\_bank;

Ketiga metode yang telah disebutkan untuk RIGHT OUTER JOIN juga berlaku untuk LEFT OUTER JOIN, yaitu dengan menggunakan USING, ON, atau NATURAL.

* Full Outer Join

Menampilkan hasil join tabel pertama dengan tabel kedua, serta semua record di kedua tabel tersebut:

SELECT\*FROM rekening NATURAL FULL OUTER JOIN cabang\_bank;

Ketiga metode yang telah disebutkan untuk LEFT/RIGHT OUTER JOIN juga berlaku untuk FULL OUTER JOIN, yaitu dengan menggunakan USING, ON, atau NATURAL.

* Inner vs Outer Join

Dalam *Inner Join*: yang ditampilkan hanyalah hasil dari *table join* yang berhasil, yaitu semua *record* yang berhubungan di kedua tabel yang digabungkan.

Dalam *Outer Join*: selain menampilkan hasil dari *Inner Join, Outer Join* juga menampilkan semua *record* yang tidak berhubungan di kedua tabel yang digabungkan.

* Multiple Join

Untuk lebih dari 2 tabel, tinggal diteruskan saja JOINnya. Misalnya:

SELECT\*FROM nasabah NATURAL JOIN nasabah\_has\_rekening NATURAL JOIN rekening;

Cara lain:

40

SELECT\*FROM nasabah A, nasabah\_has\_rekening B, rekening C

WHERE A.id\_nasabah=B.id\_nasabah AND

B.no\_rekening=C.no\_rekening;

Jika melakukan *multiple join* (lebih dari 2 tabel), Anda harus memperhatikan urutan *join*. Urutan *table join* perlu mengikuti alur *relationship* yang tertera di ER Diagram.

Oleh karena itu, sebaiknya Anda menggunakan ER Diagram agar bisa menghasilkan *table join* yang benar.

1. **Alat dan Bahan** 
   1. Komputer dengan sistem operasi Windows 7
   2. Program aplikasi XAMPP dengan PhpMyAdmin
   3. Modul Praktikum Sistem Berkas dan Basis Data
2. **Langkah-langkah Praktikum** 
   1. Jalankan XAMPP Control Panel.
   2. Jalankan server Apache dan MySQL.
   3. Buka Command Prompt dan login sebagai root ke MySQL seperti di langkah pada Modul 1.
   4. Pilih database perbankan dengan perintah “use perba nkan”. Sehingga akan muncul pemberitahuan “database changed”
   5. Tampilkan nama bank dan alamat bank untuk semua cabang bank dan diurutkan berdasarkan nama bank dengan kode berikut :

SELECT nama\_cabang, alamat\_cabang FROM cabang\_bank ORDER BY nama\_cabang;

* 1. Tampilkan nomor rekening, pin, dan jumlah saldo untuk semua rekening dan diurutkan berdasarkan jumlah saldo dari yang paling besar ke yang paling kecil dengan kode berikut

SELECT no\_rekening, pin, saldo FROM rekening ORDER BY saldo DESC;

* 1. Tampilkan nomor rekening, nama nasabah, dan alamat nasabah dari semua nasabah yang memiliki rekening dan diurutkan berdasarkan nama nasabah dengan kode berikut.

SELECT rekening.no\_rekening, nasabah.nama\_nasabah, nasabah.alamat\_nasabah FROM rekening, nasabah, nasabah\_has\_rekening WHERE nasabah.id\_nasabah= nasabah\_has\_rekening.id\_nasabahFK AND rekening.no\_rekening=nasabah\_has\_rekening.no\_rekeningFK ORDER BY nasabah.nama\_nasabah;

* 1. Tampilkan nomor rekening, nama nasabah, dan jumlah saldo untuk semua rekening yang dimiliki oleh nasabah dan diurutkan berdasarkan nama nasabah dengan kode berikut :

SELECT rekening.no\_rekening, nasabah.nama\_nasabah, rekening.saldo

FROM rekening, nasabah, nasabah\_has\_rekening WHERE

41

nasabah.id\_nasabah= nasabah\_has\_rekening.id\_nasabahFK AND rekening.no\_rekening= nasabah\_has\_rekening.no\_rekeningFK ORDER BY nasabah.nama\_nasabah;

1. **Tugas** 
   1. Tampilkan nama nasabah, alamat nasabah, jenis transaksi dan jumlah transaksi dimana jenis transaksinya adalah kredit dan diurutkan berdasarkan nama nasabah!
   2. Tampilkan nomor rekening, nama nasabah, jenis transaksi dan jumlah transaksi yang melakukan transaksi pada tanggal 21 November 2009 dan diurutkan berdasarkan nama nasabah!
   3. Tampilkan nomor rekening, nama nasabah, jenis transaksi dan jumlah transaksi dimana jumlah transaksi = Rp 20.000!
   4. Tampilkan nomor rekening, nama nasabah dan alamat nasabah dimana nama nasabah diawali dengan kata ‘Su’!
   5. Tampilkan nomor rekening dengan alias ‘Nomor Rekening’, nama nasabah dengan alias ‘Nama Nasabah’, jumlah transaksi dengan alias ‘Juml ah Transaksi’ dimana jenis transaksinya adalah debit! Urutkan berdasarkan nama nasabah!

42

**Modul 7**

**Aggregasi**

1. **Tujuan** 
   1. Mahasiswa mampu menggunakan beberapa fungsi agregasi dalam melakukan seleksi data.
2. **Landasan Teori** 
   1. IN
      * Contoh:

SELECT\*FROM rekening WHERE kode\_cabang IN (‘BRUM’, ‘BRUL’);

* + - Perintah SQL di atas sama dengan:

SELECT\*FROM rekening WHERE kode\_cabang = ‘BRUM’ OR kode\_cabang = ‘BRUL’;

* + - Tidak ada batas banyaknya nilai yang bisa ada di dalam IN (...).
  1. NOT IN
     + Contoh:

SELECT\*FROM rekening WHERE kode\_cabang NOT IN (‘BRUS’, ‘BRUM’);

* + - Perintah SQL di atas sama dengan:

SELECT\*FROM rekening WHERE kode\_cabang <> ‘BRUS’ AND kode\_cabang <> ‘BRUM’;

* + - Nilai NULL tidak akan tampil dalam IN dan NOT IN.
    - Perhatikan perbedaan penggunaan OR dan AND dalam IN dan NOT IN.
  1. BETWEEN
     + Contoh:

SELECT\*FROM rekening WHERE saldo BETWEEN 500000 AND 1000000;

* + - Perintah SQL di atas sama dengan:

SELECT\*FROM rekening WHERE saldo>=500000 AND saldo<= 1000000;

* + - Nilai yang pertama dalam BETWEEN harus lebih kecil dari nilai yang kedua.
    - Bisa untuk *string*.
  1. NOT BETWEEN
     + Contoh:

SELECT\*FROM rekening WHERE saldo NOT BETWEEN 500000 AND 1000000;

* + - Perintah SQL di atas sama dengan:

SELECT\*FROM rekening WHERE saldo<500000 OR saldo>1000000;

43

* + Perhatikan perbedaan penggunaan AND dan OR dalam BETWEEN dan NOT BETWEEN.

1. AGGREGATE FUNCTIONS Fungsi-fungsi untuk *aggregate*:
   * ***MIN()*** 
     + Digunakan untuk mencari nilai terkecil dari sekumpulan *record*. Contoh:

SELECT MIN(saldo) FROM rekening;

* + - Bisa dibatasi dengan WHERE clause sehingga hanya *record*(-*record*) tertentu yang ditelusuri:

SELECT MIN(saldo) FROM rekening WHERE kode\_cabang=‘BRUS’;

* + ***MAX()*** 
    - Digunakan untuk mencari nilai terbesar dari sekumpulan *record*. Contoh:

SELECT MAX(saldo) FROM rekening;

* + - Juga bisa dibatasi dengan WHERE *clause*:

SELECT MAX(saldo) FROM rekening WHERE kode\_cabang=‘BRUS’;

* + ***COUNT()*** 
    - Digunakan untuk menghitung banyaknya *record*. Contoh:

SELECT COUNT(\*) FROM nasabah;

SELECT COUNT(nama\_nasabah) FROM nasabah; SELECT COUNT(alamat\_nasabah) FROM nasabah;

* + - Juga bisa dibatasi dengan WHERE clause.
    - Jika kita ingin menghitung banyaknya *record* yang unik (tidak ada pengulangan), gunakan DISTINCT:

SELECT COUNT(DISTINCT alamat\_nasabah) FROM nasabah;

* + ***SUM()*** 
    - Digunakan untuk menjumlahkan nilai-nilai dari sekumpulan *record*. Contoh:

SELECT SUM(saldo) FROM rekening;

* + - Bisa dibatasi dengan WHERE *clause*:

SELECT SUM(saldo) FROM rekening WHERE kode\_cabang=‘BRUS’;

* + ***AVG()*** 
    - Digunakan untuk menghitung rata-rata nilai dari sekumpulan *record*. Contoh:

44

SELECT AVG(saldo) FROM rekening;

* + Bisa dibatasi dengan WHERE *clause*:

SELECT AVG(saldo) FROM rekening WHERE kode\_cabang=‘BRUS’;

* + Beberapa *aggregate functions* bisa digabungkan dalam satu perintah SQL:

SELECT MIN(saldo), MAX(saldo), AVG(saldo) FROM rekening;

* + Bisa menambahkan ekspresi aritmatika:

SELECT SUM(saldo + 1000) FROM rekening; SELECT SUM(saldo) + 1000 FROM rekening; SELECT MAX(saldo) - MIN(saldo) FROM rekening;

* + Bisa menggunakan *Column Alias (AS)* untuk membuat tampilan lebih profesional.
* ***GROUP BY***

– Digunakan untuk mengelompokkan sekumpulan *record* berdasarkan (kolom-kolom) tertentu.

Contoh:

SELECT jenis\_transaksi FROM transaksi GROUP BY jenis\_transaksi; SELECT jenis\_transaksi, tanggal FROM transaksi GROUP BY jenis\_transaksi, tanggal;

– Hasil yang sama bisa didapatkan dengan menggunakan DISTINCT:

SELECT DISTINCT jenis\_transaksi, tanggal FROM transaksi;

– Jika menggunakan GROUP BY, semua *field* yang ingin ditampilkan dalam SELECT harus tercantum di GROUP BY.

Contoh yang salah:

SELECT jenis\_transaksi, tanggal FROM transaksi GROUP BY jenis\_transaksi;

SELECT jenis\_transaksi, tanggal FROM transaksi GROUP BY tanggal;

Contoh yang benar:

SELECT jenis\_transaksi, tanggal FROM transaksi GROUP BY jenis\_transaksi, tanggal;

* ***HAVING***

– Merupakan pasangan dari GROUP BY, digunakan untuk membatasi kelompok yang ditampilkan:

SELECT jenis\_transaksi, tanggal FROM transaksi GROUP BY jenis\_transaksi, tanggal HAVING jenis\_transaksi=‘kredit’;

– Hasil yang sama bisa didapatkan dengan:

45

SELECT jenis\_transaksi, tanggal FROM transaksi WHERE

jenis\_transaksi=‘kredit’ GROUP BY jenis\_transaksi, tanggal;

– Jika menggunakan HAVING, maka pembatasan dilakukan setelah hasil dikelompokkan dalam GROUP BY.

– Jika menggunakan WHERE, maka pembatasan dilakukan sebelum hasil dikelompokkan dalam GROUP BY.

– *Field* (-*field*) yang disebut di HAVING harus ada di GROUP BY, atau berupa *aggregate functions*.

Contoh yang salah:

SELECT jenis\_transaksi, tanggal FROM transaksi GROUP BY jenis\_transaksi, tanggal HAVING jumlah=50000;

Contoh yang benar:

SELECT jenis\_transaksi, tanggal FROM transaksi WHERE jumlah=50000 GROUP BY jenis\_transaksi, tanggal;

* ***GROUP BY* dan *AGGREGATE***

– GROUP BY sangat cocok untuk *aggregate functions*. Dengan menggunakan GROUP BY, kita bisa mengelompokkan record-record dan menghitung *min, max,* *count, sum,* dan *avg* untuk masing-masing kelompok.

Contoh:

SELECT kode\_cabang, MIN(saldo), MAX(saldo), COUNT(\*), SUM(saldo), AVG(saldo) FROM rekening GROUP BY kode\_cabang;

– Bisa digabungkan dengan tabel join dan ORDER BY:

SELECT nama\_cabang, SUM(saldo) FROM rekening NATURAL JOIN cabang\_bank GROUP BY nama\_cabang ORDER BY nama\_cabang;

– Hasil di atas menampilkan total saldo untuk masing-masing cabang\_bank.

– Perintah SQL di bawah menampilkan banyaknya nasabah yang dilayani oleh masing-masing cabang bank:

SELECT nama\_cabang, COUNT(DISTINCT id\_nasabah) FROM cabang\_bank NATURAL JOIN rekening NATURAL JOIN nasabah\_has\_rekening GROUP BY nama\_cabang ORDER BY nama\_cabang;

– Contoh dengan HAVING:

SELECT kode\_cabang, SUM(saldo), COUNT(\*) FROM rekening GROUP BY kode\_cabang HAVING SUM(saldo) >= 5000000 ORDER BY kode\_cabang;

46

Karena SUM(saldo) hanya bisa dihitung setelah hasil dikelompokkan dengan GROUP BY, maka kita harus menggunakan HAVING untuk membatasi hasil berdasarkan SUM(saldo) >= 5000000. Kita tidak bisa menggunakan WHERE.

1. **Alat dan Bahan** 
   1. Komputer dengan sistem operasi Windows 7
   2. Program aplikasi XAMPP dengan PhpMyAdmin
   3. Modul Praktikum Sistem Berkas dan Basis Data
2. **Langkah-langkah Praktikum** 
   1. Jalankan XAMPP Control Panel.
   2. Jalankan server Apache dan MySQL.
   3. Buka Command Prompt dan login sebagai root ke MySQL seperti di langkah pada Modul 1.
   4. Pilih database perbankan dengan perintah “use perba nkan”. Sehingga akan muncul pemberitahuan “database changed”
   5. Tampilkan tanggal transaksi, jenis transaksi, dan jumlah transaksi untuk semua transaksi yang dilakukan oleh Sutopo dan Canka Lokananta dan diurutkan berdasarkan tanggal transaksi dengan kode berikut :

SELECT transaksi.tanggal, transaksi.jenis\_transaksi, transaksi.jumlah FROM nasabah, transaksi WHERE nasabah.id\_nasabah=transaksi.id\_nasabahFK AND nasabah.nama\_nasabah IN ('Sutopo','Canka Lokananta') ORDER BY transaksi.tanggal;

* 1. Tampilkan tanggal transaksi, nama nasabah, jenis transaksi, dan jumlah transaksi untuk semua transaksi yang terjadi dari 15 November sampai 20 November 2009 dan diurutkan berdasarkan tanggal transaksi dan nama nasabah dengan kode berikut :

|  |  |  |
| --- | --- | --- |
| SELECT | transaksi.tanggal, | nasabah.nama\_nasabah, |
| transaksi.jenis\_transaksi, transaksi.jumlah | | FROM nasabah, transaksi |

WHERE transaksi.tanggal BETWEEN '2009-11-15' AND '2009-11-20' AND

nasabah.id\_nasabah=transaksi.id\_nasabahFK ORDER BY

transaksi.tanggal, nasabah.nama\_nasabah;

1. Tampilkan jenis transaksi dan total jumlah transaksi (dalam rupiah) untuk tiap jenis transaksi dan diurutkan berdasarkan jenis transaksi dengan kode berikut :

SELECT transaksi.jenis\_transaksi AS "Jenis Transaksi", SUM(jumlah) AS "Jumlah (Rp)" FROM transaksi GROUP BY transaksi.jenis\_transaksi ORDER BY transaksi.jenis\_transaksi;

1. Tampilkan jenis transaksi, jumlah transaksi yang terbesar serta yang terkecil untuk tiap jenis transaksi dan diurutkan berdasarkan jenis transaksi dengan kode berikut :

47

SELECT jenis\_transaksi AS "Jenis Transaksi", MAX(jumlah) AS "Transaksi Terbesar", MIN(jumlah) AS "Transaksi Terkecil" FROM transaksi GROUP BY transaksi.jenis\_transaksi ORDER BY transaksi.jenis\_transaksi;

* 1. Tampilkan jenis transaksi, total jumlah transaksi (dalam rupiah), dan banyaknya transaksi yang tercatat untuk tiap jenis transaksi yang terjadi sebelum bulan Desember 2009 dan diurutkan berdasarkan jenis transaksi dengan kode berikut :

SELECT jenis\_transaksi AS "Jenis Transaksi", SUM(jumlah) AS "Jumlah (Rp)", COUNT(jumlah) AS "Jumlah Transaksi" FROM transaksi WHERE tanggal BETWEEN '2009-11-1' AND '2009-11-30' GROUP BY transaksi.jenis\_transaksi ORDER BY transaksi.jenis\_transaksi;

1. **Tugas** 
   * 1. Tampilkan jenis transaksi, jumlah transaksi dalam Rp dan total transaksi untuk nasabah yang bernama akhiran ‘Kartika Padmasari’ untuk masi ng-masing jenis transaksi!
     2. Berapa jumlah total saldo yang dimiliki oleh Maryati?
     3. Tampilkan jumlah transaksi yang ditangani oleh masing-masing cabang bank!
     4. Tampilkan nama nasabah dan jumlah saldo yang memiliki saldo antara Rp 500.000 sampai Rp 2.000.000!
     5. Tampilkan nama nasabah, tanggal transaksi dan jumlah transaksi dalam Rp dimana jumlah transaksi di atas Rp 100.000 dan urutkan berdasarkan jumlah transaksi dari yang besar ke yang kecil!

48

**Modul 8**

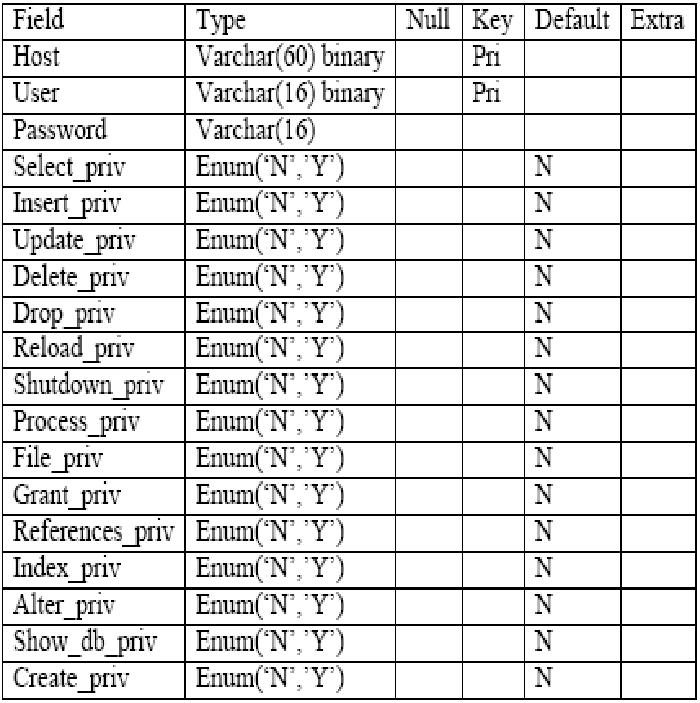
**Data Control Language**

1. **Tujuan** 
   1. Mahasiswa memahami perintah-perintah Data Control Language.
   2. Mahasiswa mampu mengimplementasikan perintah GRANT dan REVOKE.
2. **Landasan Teori**

Data Control Language (DCL) merupakan perintah-perintah yang dapat digunakan untuk menjaga keamanan basis data. Perintah tersebut dapat dipakai untuk menentukan akses basis data hanya dapat dilakukan oleh orang-orang tertentu dan dengan macam akses yang dibatasi pula. Adapun Objek-Objek DCL dalam Mysql diantaranya :

* 1. Tabel User dari Database MySQL

Tabel user adalah tabel yang ada dalam database MySQL. Tabel user hanya diperuntukkan bagi seorang Administrator (root). Tabel user bersifat global, apapun perubahan yang terjadi pada tabel ini akan mempengaruhi jalannya keseluruhan system MySQL. Tabel ini yang nantinya digunakan untuk mendefinisikan user dari basis data. Adapun isi dari tabel tersebut adalah sebagai berikut.



49

Sintaks Umum :

SELECT, UPDATE, DELETE, INSERT, FLUSH

Contoh Penggunaan :

SELECT host, user, password FROM user; DELETE FROM user WHERE user=’’;

UPDATE user SET password=password(‘xxxxxxxxxx’) WHERE user=’root’; insert into user (User,Password) values(‘Winda’,’w1nd’);

FLUSH PRIVILEGES;

Perintah FLUSH PRIVILEGES adalah suatu perintah untuk mengaktifkan perubahan-perubahan yang terjadi pada user, seperti hak akses, penggantian password pada user, dsb. Perintah FLUSH PRIVILEGES ini hukumnya wajib dilaksanakan setelah Anda melakukan perubahan (apapun juga) secara langsung ke dalam tabel user atau ke dalam database mysql.

1. Tabel Table\_Priv dari database MySQL

Tabel\_priv berfungsi mengatur tabel apa saja yang dapat diakses oleh seorang user, berikut jenis izin aksesnya. Tingkat akses hanya untuk tabel. Pada prinsipnya hanya bekerja seperti db table, kalau tidak digunakan untuk tabel sebagai ganti database.

Sintaks Umum :

SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT, REFERENCES, INDEX, ALTER, CREATE VIEW, SHOW VIEW

Contoh :

Insert into tables\_priv (User, table\_name, table\_priv) values (’Febe,

Elfrida’, ’Point\_Of\_Sales.jenis’, ’select’);

1. GRANT

Grant berfungsi untuk membuat user baru dan memberikan hak istimewa. Grant adalah salah satu privilege untuk tabel. Grant digunakan untuk memberikan privilege kepada tabel yang didefinisikan kepada pemakai lain. Privilege untuk pemakai dalam perintah grant didefinisikan dengan menggunakan nama-nama privilege. Nama privilege memudahkan administrator untuk dapat memberikan priivilege tanpa harus tahu apa nama field dan tabel yang harus diisi. Perintah grant secara otomatis akan menambah data pemakai apabila data nama pemakai yang disertakan pada perintah tersebut belum ada dalam tabel user. Perintah grant memudahkan administrator untuk tidak perlu melakukan perintah pendefinisian privilege dengan menggunakan sql. Karena dengan menggunakan sql, kita harus hafal nama tabel yang harus diisi, field apa saja yang harus diisi, jumlah field yang harus diisi. Kesalahan mudah dilakukan dengan menggunakan perintah sql karena 38 ketidaktelitian atau ketidakhafalan nama tabel dan nama field yang harus diisi.

50

Sintaks umum :

GRANT hak\_akses ON nama\_tabel TO pemakai;

Contoh :

GRANT SELECT ON Point\_Of\_Sales.jenis TO Febe;

GRANT ALL PRIVILEGES ON Point\_Of\_Sales.User TO Admin; SHOW GRANTS FOR Admin;

1. REVOKE

Untuk menghapus batasan hak akses yang telah diatur dengan menggunakan perintah GRANT, digunakan perintah REVOKE.

Sintaks Umum :

REVOKE hak\_akses ON nama\_tabel FROM namaAccount@namaHost; REVOKE hak\_akses ON nama\_database.nama\_tabel FROM user; \ Contoh Penggunaan :

REVOKE SELECT ON Point\_Of\_Sales.jenis FROM Febe;

REVOKE ALL PRIVILEGES ON Point\_Of\_Sales.user FROM Admin;

1. Show Processlist

Digunakan untuk menampilkan kegiatan apa saja yang terjadi pada MySQL server atau menampilkan informasi mengenai thread yang dieksekusi di server. Bila terdapat kegiatan yang membahayakan kita sebagai admin dapat menghentikan dengan perintah KILL atau MySQLAdmin.

Sintaks Umum :

SHOW PROCESSLIST;

1. Kill

Kill berfungsi menghentikan thread server / untuk membunuh proses yang sedang berjalan Sintaks Umum :

KILL nomor\_Id; Contoh Penggunaan :

KILL 2;

1. Optimize Table

Tabel yang sering mengalami proses penghapusan dan penambahan akan menyebabkan struktur yang tidak teratur secara fisik atau telah terjadi fragmentasi. Penghapusan data dalam jumlah besar mempunyai peluang terjadinya fragmentasi. Terutama untuk data bertipe VARCHAR, TEXT atau BLOB. Tidak semua DBMS dapat melakukan fragmentasi, kita dapat melihat dukungan setiap DBMS Untuk mengatasi masalah fragmentasi solusinya adalah melakukan OPTIMIZE TABLE Untuk MySQL versi 3.23 ke atas mendukung fasilitas OPTIMIZE TABLE. Perlu diketahui pada saat OPTIMIZE TABLE dikerjakan, semua tabel

51

akan di-lock (terkunci) Proses fragmentasi sebaiknya dilakukan secara berkala, misalnya setiap minggu atau setiap bulan.

Sintaks Umum :

OPTIMIZE TABLE tabel\_1, tabel\_2, tabel\_3, tabel\_n; Contoh Penggunaan :

OPTIMIZE TABLE Point\_Of\_Sales.User, Point\_Of\_Sales.item,

Point\_Of\_Sales.jualGlobal;

1. **Alat dan Bahan** 
   * 1. Komputer dengan sistem operasi Windows 7
     2. Program aplikasi XAMPP dengan PhpMyAdmin
     3. Modul Praktikum Sistem Berkas dan Basis Data
   1. **Langkah-langkah Praktikum** 
      1. Jalankan XAMPP Control Panel.
      2. Jalankan server Apache dan MySQL.
      3. Buka Command Prompt dan login sebagai root ke MySQL seperti di langkah pada Modul 1.
      4. Pilih database perbankan dengan perintah “use mysql ”. Sehingga akan muncul pemberitahuan “database changed”
      5. Masukan user baru ke tabel user dengan sintaks berikut ini.

insert into user (user,password) values (‘myuser’,’myuser’);

* + 1. Berikan hak akses kepada user “myuser” dengan sinta ks berikut ini.

GRANT SELECT on perbankan.nasabah to myuser; FLUSH PRIVILEGES;

GRANT SELECT on perbankan.cabang\_bank to myuser; FLUSH PRIVILEGES;

* + 1. Keluar sebagai user root dengan sintax “exit;”.
    2. Login lagi sebagai “myuser” dengan sintaks sebagai berikut. mysql –u myuser
    3. Ubah database yang digunakan ke database perbankan dengan sintaks “use perbankan”. Sehingga akan muncul konfirmasi “database changed”.
    4. Coba lakukan select pada tabel nasabah dengan sintaks berikut. select \* from nasabah;

Selanjutnya coba lakukan insert pada tabel nasabah dengan sintaks berikut. insert into nasabah values(‘31’,’Susilo’,’Sukoharjo’);

* + 1. Kemudian ketikan perintah “exit”. Kemudian login la gi sebagai root.
    2. Kemudian hilangkan salah satu privilege untuk user myuser dengan sintaks berikut ini.

REVOKE SELECT on perbankan.nasabah from myuser;

52

FLUSH PRIVILEGE;

* 1. Kemudian exit dan kembali login sebagai user “myuse r”.
  2. Gunakan database perbankan dengan sintaks “use perb ankan”.
  3. Selanjutnya silahkan dicoba untuk menggunakan perintah SELECT pada tabel nasabah.

**E. Tugas**

* 1. Buat user baru sesuai dengan nama anda masing-masing.
  2. Beri Privilege untuk user tersebut dengan ketentuan sebagai berikut ini.
     1. Dapat melakukan INSERT, UPDATE, dan DELETE pada tabel nasabah.
     2. Hanya dapat melakukan SELECT pada tabel cabang\_bank.
  3. Coba lakukan kasus berikut ini dengan login sebagai user yang dibuat dan screenshot hasilnya.
     1. Lakukan perintah SELECT pada tabel nasabah.
     2. Lakukan perintah INSERT pada tabel nasabah.
     3. Lakukan perintah INSERT pada tabel cabang\_bank.
  4. Analisa hasil percobaan kalian tersebut.

53

**Modul 9**

**SubQuery**

1. **Tujuan** 
   1. Mahasiswa mampu mengimplementasikan SubQuery untuk menyelesaikan kasus pengambilan data dari suatu basis data.
2. **Landasan Teori**

Dalam aplikasi yang menggunakan database tidak akan pernah terlepas dalam perolehan kembali data. Untuk memunculkan data yang dikemas lebih baik agar memudahkan pengguna data untuk mengerti dan memahami data tentunya tidak akan terlepas dari query-query sql yang lebih komplek apalagi data antara table satu dengan table lainnya saling berhubungan.

Sangat panjang jika kita menulis subquery dalam satu baris(baris tunggal) dan multiple-row subqueris(banyak baris) dimana hanya satu kolom saja yang ingin ditampilkan dengan perintah select dan ini digunakan untuk mengevaluasi dari expression dalam induk perintah select. Jika kita ingin membandingkan dua atau lebih kolom, kita harus menulis sebuah gabungan klausa WHERE menggunakan operator logika. Menggunakan multiple-subqueries, kita dapat mengkombinasikan duplikat kondisi where ke dalam satu klausa WHERE.

Sintaks Umum.

select kolom, kolom,…

from table

where (kolom, kolom,…) IN

select kolom, kolom,.. From table

Where kondisi);

1. SubQuery Perbandingan Kolom

Multiple-kolom membandingkan ke dalam subquery dan dijadikan campuran perbandingan atau pasangan perbandingan. Jika kita mempertimbangkan contoh “menampilan secara detail dari pekerja yang bekerja di beberapa department dan memiliki beberapa menager seperti ‘Daniel’”, kita dapat melihat hasil nya dari statement berikut:

Select first\_name, last\_name, manager\_id, department\_id from employees

where manager\_id IN (select manager\_id from employees where firt\_name=’Daniel)

AND department\_id IN (select department\_id From employees where firt\_name=’Daniel’);

54

2. SubQuery Pasangan Perbandingan Subquery

Menampilkan secara detail dari pekerja yang di manage oleh beberapa manager and bekerja di beberapa department sebagai pekerja yang first\_name nya adalah “John”.

select employee\_id, manager\_id, department\_id from employees where

(manager\_id, department\_id) IN

( select manager\_id, department\_id from employees where first\_name='John')

AND first\_name <> 'John'

Pada contoh diatas adalah contoh tampilan hasil dari query kombinasi nilai dari kolom manager\_id dan kolom department\_id untuk masung-asing baris pada table employees dengan nilai pada kolom manager\_id dan kolom department\_id untuk pekertja dengan first\_name adalah “John”, subquery untuk menndapatk an kembali nilai dari manager\_id dan department\_id untuk employees dengan first\_name “Jo hn” dieksekusi.

Nilai perbandingan dengan kolom manager\_id dan department\_id untuk masing-masing baris dalam table employees. Jika kombinasi sesuai, baris akan ditampikan. Dalam output, record dari employees dengan first\_name “Jo hn” tidak akan ditampilkan.

3. SubQuery NonPasangan Perbandingan Subquery

Menampilkan secara detail dari pekerja yang dimanage beberapa manager pekerja dengan firt\_name “John” dan bekerja di beberap depa rtment dengan first\_name John

select employee\_id, manager\_id, department\_id from employees

where manager\_id IN (select manager\_id from employees

where first\_name='John') AND

department\_id IN (select department\_id from employees

where first\_name='John') AND first\_name<>'John';

55

4. Ekspresi-ekspresi Scalar Subquery

Scalar subquery adalah subquery yang mengembalikan tepat satu nilai kolom dari satu baris. Scalar subquery dapat digunakan di:

Bagian kondisi dan expresi dari DECODE and CASE Semua klausa dari SELECT kecuali GROUP BY Contoh scalar subquery dapa CASE expression :

select employee\_id, last\_name,( CASE

when department\_id=

(select department\_id from departments where location\_id=1800)

THEN 'Canada' ELSE 'USA' END) location from employees

Contoh subquery dalam klausa order by :

select employee\_id, last\_name from employees e

order by ( select department\_name from departments d where e.department\_id=d.department\_id);

5. Korelasi Subquery

Korelasi subquery digunakan untuk proses baris-perbaris.Masng-masing subquery di eksekusi setelah setiap baris setelah query paling luar. Oracle server menampilkan sebuah korelasi query mengacu sebuah kolom dari table yang menunjuk dapa parent statement. Korelasi subquery dievalusi ketika setiap baris diproses oleh statement induk. Induk statement dapat berupa select, update atau delete.

Sintaks umum :

Select kolom1, kolom2,… From table1 outer Where kolom1 operator (select kolom1, kolom2,.. From table2

Where expresi=outer.expresi2);

Menampilkan semua pekerja yang mendapatkan lebih dari rata-rata gaji dari department tertentu

select last\_name, salary, department\_id from

employees outer

56

where salary >

(select AVG(salary)

from employees

where department\_id=outer.department\_id);

1. Menggunakan Operator Exists

Operator EXISTS mengecek keberadaan dari baris dalam hasil subquery. Jika Nilai baris dari subquery ditemukan:

o Pencarian tidak akan dilanjutkan dalam query terdalam o Kondisi tetap benar

Jika Nilai baris dari subquery tidak ditemukan: o Kondisi akan salah

o Pencarian akan diteruskan samapi pada query paling dalam. Menemukan pekerja yang paling sedikit memberikan laporan : select employee\_id, last\_name, job\_id, department\_id from employees outer

where EXISTS ( select 'X' from employees where manager\_id=outer.employee\_id);

1. Korelasi Update

Menggunakan korelasi query untuk update baris berdasar pada baris dari table lain Sintaks Umum.

Update table1 alias

Set kolom=(select expression

From table2 alias

Where alias1.kolom=alias2.kolom);

Denormalisasi table EMP16 dengan menambahkan sebuah kolom untuk memasukan pada

department name.

Create table emp16 as select \* from employees;

Alter table emp16 add(department\_name varchar2(25));

Select department\_name from emp16;

update emp16 e set department\_name=

(select department\_name from departments d where e.department\_id=d.department\_id);

Select department\_name from emp16;

57

* + 1. Korelasi Delete

Menggunakan korelasi subquery untuk mendelete row berdasarkan pada baris dari table lain Sintaks umum :

DELETE from table1 alias WHERE kolom operator

(select expression from table alias2 Where alias1.kolom=alias2.kolom);

DELETE from emp16 E where employee\_id=(

select employee\_id from employees where employee\_id=E.employee\_id);

select \* from emp16;

1. **Alat dan Bahan** 
   * 1. Komputer dengan sistem operasi Windows 7
     2. Program aplikasi XAMPP dengan PhpMyAdmin
     3. Modul Praktikum Sistem Berkas dan Basis Data
   1. **Langkah Praktikum** 
      1. Jalankan XAMPP Control Panel.
      2. Jalankan server Apache dan MySQL.
      3. Buka Command Prompt dan login sebagai root ke MySQL seperti di langkah pada Modul 1.
      4. Pilih database perbankan dengan perintah “use perba nkan”. Sehingga akan muncul pemberitahuan “database changed”
      5. Tampilkan detail data nasabah yang pernah melakukan transaksi kredit dengan menggunakan syntax subquery dibawah ini:

SELECT \* FROM nasabah where nasabah.id\_nasabah IN (select DISTINCT transaksi.id\_nasabahFK from transaksi where jenis\_transaksi='kredit');

* + 1. Tampilkan detail data nasabah yang tidak pernah melakukan segala jenis transaksi dengan menggunakan syntax sub query dibawah ini:

SELECT \* FROM nasabah where nasabah.id\_nasabah NOT IN (select DISTINCT transaksi.id\_nasabahFK from transaksi);

* + 1. Tampilkan cabang bank yang memiliki nilai rata-rata saldo paling besar untuk seluruh rekening dalam cabang tersebut menggunakan syntax sub query dibawah ini:

SELECT cabang\_bank.kode\_cabang, cabang\_bank.nama\_cabang, AVG(rekening.saldo) as rata2 from cabang\_bank, rekening where cabang\_bank.kode\_cabang=rekening.kode\_cabangFK group by cabang\_bank.kode\_cabang having rata2 >= ALL (select AVG (saldo) FROM `rekening` group by kode\_cabangFK);

58

* 1. Tampilkan data transaksi yang melibatkan jumlah uang lebih besar dari rata-rata nilai keseluruhan transaksi yang pernah dilakukan menggunakan syntax sub query dibawah ini:

Select \* from transaksi where jumlah > (SELECT AVG(jumlah) FROM `transaksi`);

1. **Tugas**

Gunakan sub query untuk mendapatkan data-data berikut dari databse data kuliah yang telah anda bangun pada modul 4 dan 5.

* 1. Ambil salah satu mata kuliah. Tampilkan daftar mahasiswa yang tidak mengambil mata kuliah tersebut.
  2. Satu dosen dapat mengampu lebih dair satu mata kuliah dan satu mahasiswa dapat mengambil lebih dari satu mata kuliah. Tampilkan daftar mahasiswa yang mengambil semua mata kuliah yang diampu oleh salah satu dosen!
  3. Karena salah satu mata kuliah (A) dihilangkan, seluruh mahasiswa yang mengambil matakuliah tersebut dipindahkan untuk mengambil mata kuliah lain (B). Lakukan update data menggunakan sub query!

59

**Modul 10**

**MySQL dan Python**

1. **Tujuan** 
   1. Mahasiswa memahami cara koneksi MySQL pada Python
   2. Mahasiswa memahami penggunaan MySQL pada Python
2. **Landasan Teori** 
   1. MySQL Connector

MySQL connector adalah framework yang dibuat oleh MySQL untuk dapat mendukung berbagai macam bahasa pemrograman. Dengan memanfaatkan MySQL connector, pengembang perangkat lunak dapat mengolah database yang dibangun menggunakan bahasa pemrograman lain. Berikut adalah beberapa connector yang disediakan oleh MySQL :

* + - ADO.NET Driver (Connector/NET)
    - ODBC Driver (Connector/ODBC)
    - JDBC Driver (Connector/J)
    - Python Driver (Connector/Python)
    - C++ Driver (Connector/C++)
    - C Driver (Connector/C)
    - C API (mysqlclient)
  1. Bahasa Pemrograman Python

Python adalah bahasa pemrograman yang diciptakan oleh Guido van Rossum pada tahun 1991. Python adalah salah satu bahasa pemrograman tingkat tinggi yang bersifat intrepreter, interaktif, object oriented, dan multi-platform. Sebagai bahasa tingkat tinggi python termasuk salah satu bahasa pemrogramman yang mudah di pelajari karena syntaks yang jelas dan elegan, di kombinasikan dengan penggunaan module-module siap pakai dan struktur data tingkat tinggi yang efisien.

Bahasa pemrograman ini mudah dibaca dan terstruktur karena menggunakan indentasi. Berbeda dengan java atau C yang memanfaatkan kurung kurawal untuk menandai blok program, pada bahasa pemrograman ini setiap blok kode dipisahkan dengan susunan indentasi. Untuk memasukkan sub program dalam sebuah blok program, sub program tersebut harus diletakkan satu atau lebih spasi dari kolom suatu blok.

60

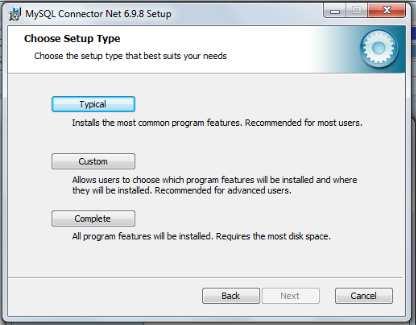
1. Koneksi MySQL pada Python

Sebelum dapat melakukan installasi MySQL Connector, terlebih dahulu lakukan installasi Python. Python telah secara otomatis terinstall pada sistem operasi UNIX dan sistem operasi sejenis UNIX seperti Linux, OS X, dan FreeBSD. Pada sistem operasi Windows, installer python dapat diakses pada http://python.org/download/.

* + Unduh konektor untuk python pada halaman berikut http://dev.mysql.com/downloads/connector/python/. Pastikan untuk mengunduh versi konektor yang sesuai dengan versi python yang terinstall pada perangkat anda.
  + Lakukan instalasi package tersebut.

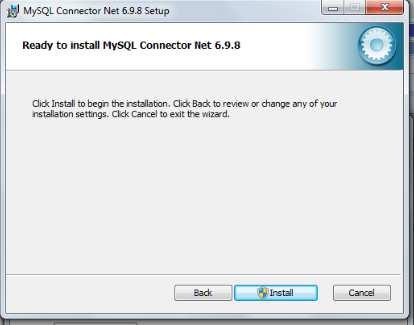


* Pilih mode installasi yang diinginkan



61

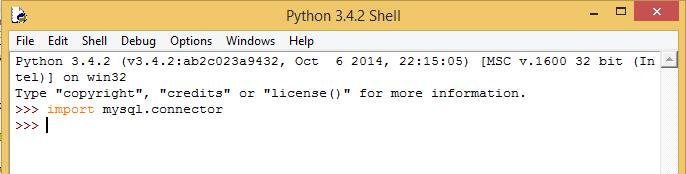
* Pilih tombol install untuk memulai installasi



* Tunggu beberapa saat sampai proses installasi selesai dilakukan. Kemudian tekan finish.

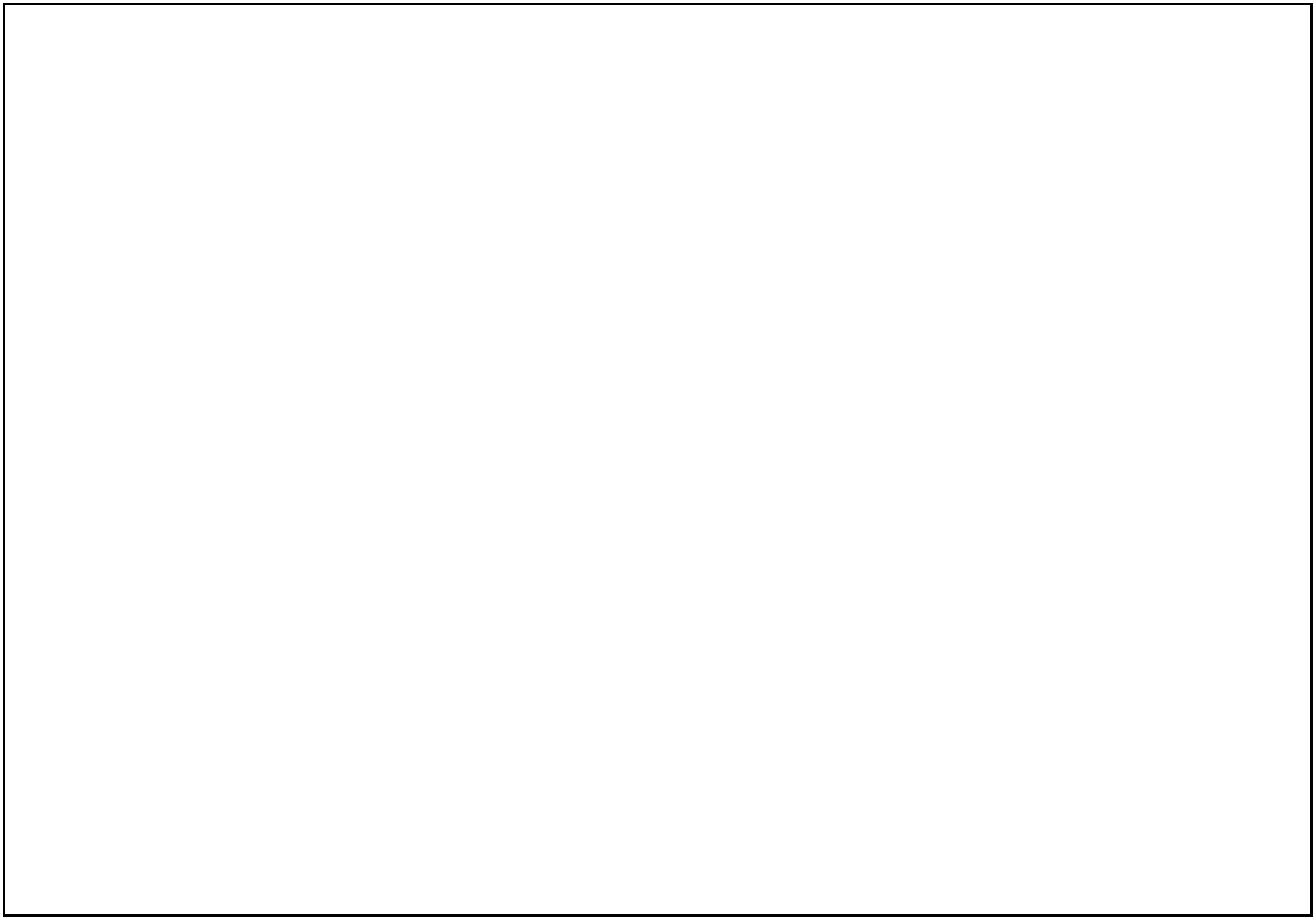


* Untuk melakukan pengecekan, masukkan syntax “impor t mysql.connector” pada python command line. Apabila tidak terdapat pemberitahuan kesalahan, maka installasi konektor berhasil dilakukan.



62

1. **Alat dan Bahan** 
   1. Python yang sudah terinstall.
   2. MySQL connector yang sudah terinstall.
2. **Langkah-Langkah Praktikum** 
   1. Jalankan XAMPP Control Panel.
   2. Jalankan server Apache dan MySQL.
   3. Buat sebuah file python dengan ekstensi .py
   4. Masukkan kode berikut ini untuk memasukkan data transaksi baru pada database perbankan.



from datetime import date, datetime, timedelta import mysql.connector

cnx = mysql.connector.connect(user='root', database='perbankan') cursor = cnx.cursor()

tanggal = datetime.now().date() tambah\_transaksi = ("INSERT INTO transaksi "

"(`id\_nasabahFK`, `no\_rekeningFK`, `jenis\_transaksi`, `tanggal`, `jumlah`) " "VALUES (%s, %s, %s, %s, %s)")

data\_transaksi = ('9', '110','kredit', tanggal, '500') cursor.execute(tambah\_transaksi, data\_transaksi)

cnx.commit()

cursor.close()

cnx.close()

63

1. Lakukan query untuk mengetahui transaksi yang dilakukan hari ini dengan menggunakan k



from datetime import date, datetime, timedelta import mysql.connector

cnx = mysql.connector.connect(user='root', database='perbankan') cursor = cnx.cursor()

query = ("SELECT `id\_nasabahFK`,`jenis\_transaksi`, `tanggal`, `jumlah`" "FROM transaksi "

"WHERE YEAR(tanggal)='2016'") cursor.execute(query)

for (id\_nasabahFK,jenis\_transaksi, tanggal,jumlah) in cursor:

print("nasabah dengan ID {} melakukan transaksi {} pada {:%d %b %Y} sejumlah {}".format( id\_nasabahFK,jenis\_transaksi, tanggal,jumlah))

cursor.close()

cnx.close()

1. **Tugas** 
   1. Buat kode program python untuk melakukan perintah INSERT, UPDATE, dan DELETE pada data transaksi.
   2. Buatlah kode program python untuk mendapatkan :
      1. Data nasabah.
      2. Data nasabah yang melakukan transaksi antara bulan oktober sampai desember.

64