**MODUL**

**SISTEM BASIS DATA**

****

**DISUSUN OLEH :**

**TIM LABORATORIUM**

**STIMIK SEPULUH NOPEMBER**

**JAYAPURA**

**2019**

**MODUL 1**

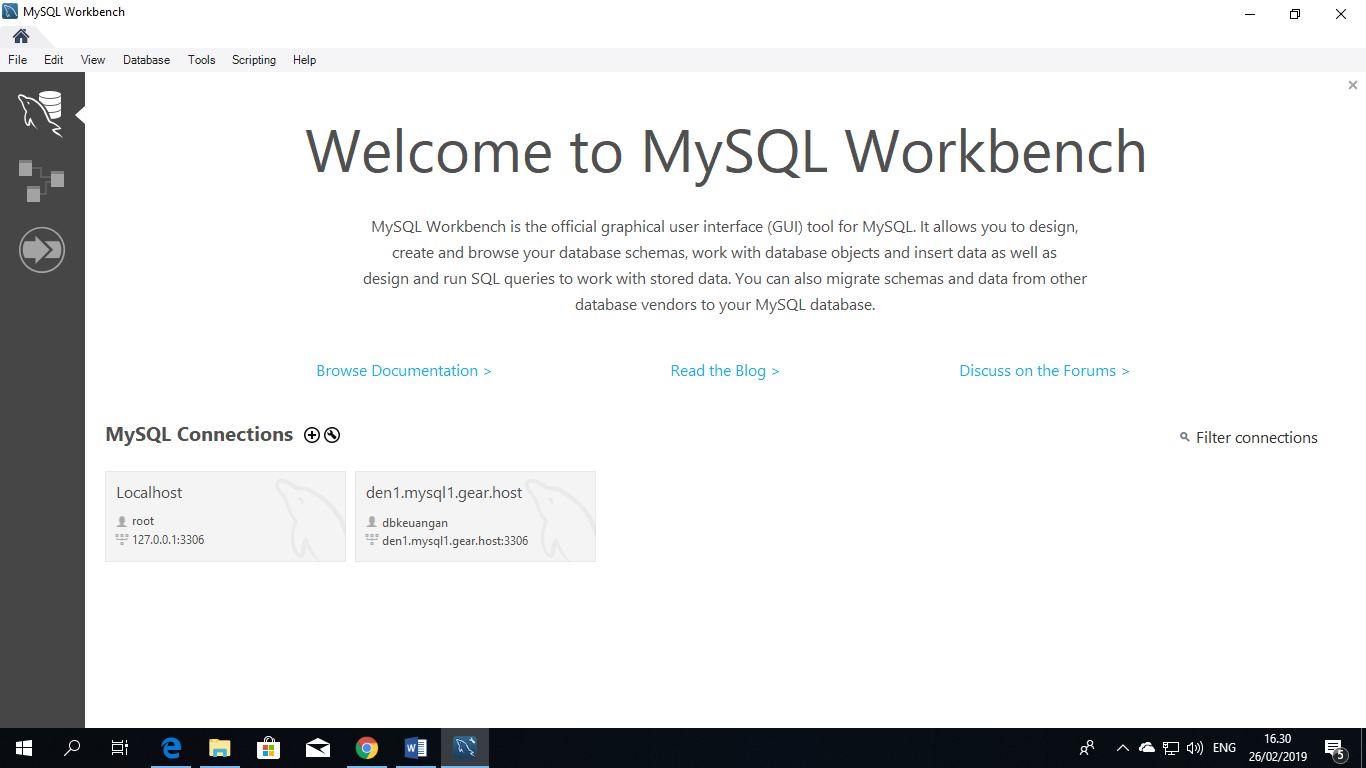
**PENGENALAN MYSQL WORKBENCH**

1. **Tujuan Praktikum**

Mahasiswa diharapkan dapat melakukan instalasi Aplikasi Workbench.

1. **Landasan Teori**

Pada saat MySQL dibuat sebagai versi open source dari SQL sebagai database RDBMS, MySQL tidak memiliki GUI sehingga perintah untuk mendisain database didapat dari command line (perintah baris) yang harus dimasukkan satu per satu. Namun seiring dengan penggunaanya yang semakin luas MySQL banyak didukung dengan GUI dari pihak ketiga. Salah satunya adalah phpMyAdmin, yang terkenal karena kemudahannya. Saat ini MySQL merupakan produk milik Oracle, dan saat ini MySQL memiliki MySQL Workbench sebagai mesin untuk mendesain database dengan interface visual.



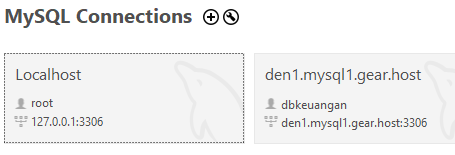
Gambar 1.1. Tampilan utama Mysql WorkBench

1. **Praktikum**

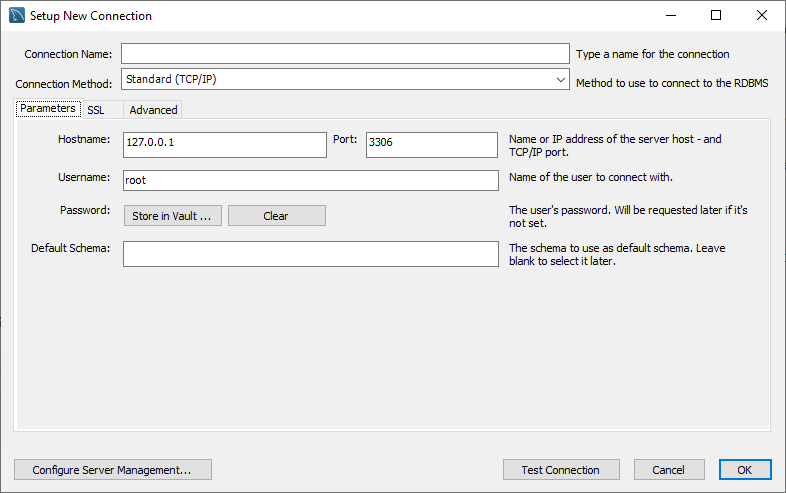
**Membuat Connection**

Ikuti Langkah-langkah pembuatan connection MySQL di bawah ini:

1. Buka Aplikasi MySQL Workbench kebudian pilih tombol tambah pada MySQL Connection



Sehingga muncul tampilan seperti berikut



Isikan semua isian dan tekan tombol test connection untuk mencoba koneksi databae, kemudian tekan tombol ok

**Membuat Model ERD dengan MySQL Workbench**

Buat Model ERD berdasarkan table berikut:

Buat Desain model Entity Relationship Diagram dengan menggunakan MySQL Workbench dari table berikut ini:

Tabel Mahasiswa

|  |  |  |
| --- | --- | --- |
| Field | Tipe Data | Keterangan |
| Id Mahasiswa | Int | Primery key |
| Nim | Varchar | - |
| Nama | Varchar | - |
| Alamat | Text | - |
| No\_telp | Text | - |
| Jk | Enum(L,P) | - |
| Tempat\_lahir | Varchar | - |
| Tanggal\_lahir | Date | - |
| Id\_Jurusan | Int | - |

Tabel Jurusan

|  |  |  |
| --- | --- | --- |
| Field | Tipe Data | Keterangan |
| Id\_Jurusan | Int | - |
| Nama\_Jurusan | Varchar | - |
| Id\_Jurusan | int | - |

Tabel Nilai

|  |  |  |
| --- | --- | --- |
| Field | Tipe Data | Keterangan |
| Id\_Nilai | Int | - |
| Nim | Varchar | - |
| Id\_mk | int | - |
| Huruf | Varchar | - |
| Angka | Varchar | - |
| Semester | Varchar | - |
| Th\_angkatan | Varchar | - |

Tabel KRS

|  |  |  |
| --- | --- | --- |
| Field | Tipe Data | Keterangan |
| Id\_krs | Int | - |
| Pembimbing | Varchar | - |
| Ips | Varchar | - |
| Ipk | Varchar | - |
| Sks | Varchar | - |
| Id\_krs | Int | - |
| Nim | VArchar | - |
| Th\_Akademik | Varchar | - |
| Semester | Varchar | - |

Tabel Mata Kuliah

|  |  |  |
| --- | --- | --- |
| Field | Tipe Data | Keterangan |
| Id\_mk | Int | - |
| Nama\_mk | Varchar | - |
| Sks | Varchar | - |
| No\_ruang | Varchar | - |
| Id\_Dosen | Int | - |
| Semester | Varchar | - |

Tabel Ruang

|  |  |  |
| --- | --- | --- |
| Field | Tipe Data | Keterangan |
| Id\_ruang | Int | - |
| Nama | Varchar | - |
| Kapasitas | Varchar | - |

Tabel Dosen

|  |  |  |
| --- | --- | --- |
| Field | Tipe Data | Keterangan |
| Id\_dosen | Int | - |
| Nama | Varchar | - |
| Tgl\_lahir | Date | - |
| Alamat | Varchar | - |
| No\_telp | Varchar | - |
| Jk | Enum(L,P) | - |

1. **Tugas Praktikum**

Buat Laporan dari hasil praktikum yang telah dilakukan.

**MODUL 2**

**LANJUTAN MODEL ERD**

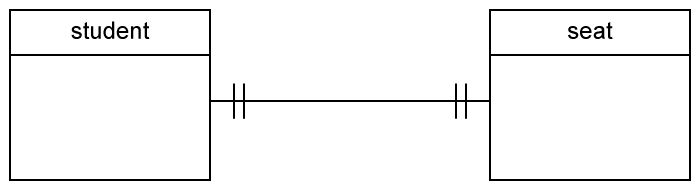
1. **Tujuan Praktikum**
2. Mahasiswa mampu untuk mendesain basis data menggunakan mysql workbench
3. Mahasiswa mengerti cara melakukan Synchronize database melalui model.
4. **Tugas Pendahuluan**

Pelajari materi Model ERD dan laporkan hasil pembelajaran dalam bentuk rangkuman materi.

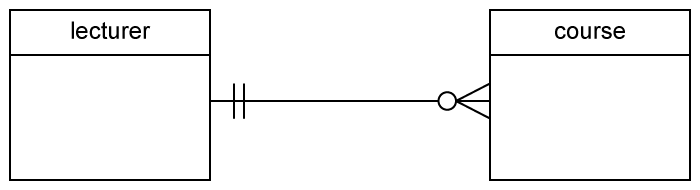
1. **Dasar Teori**

Hubungan relasi dengan

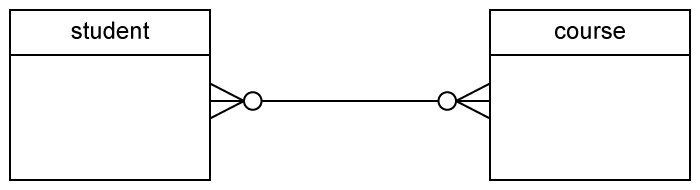
* One-to-one



* One-to-many

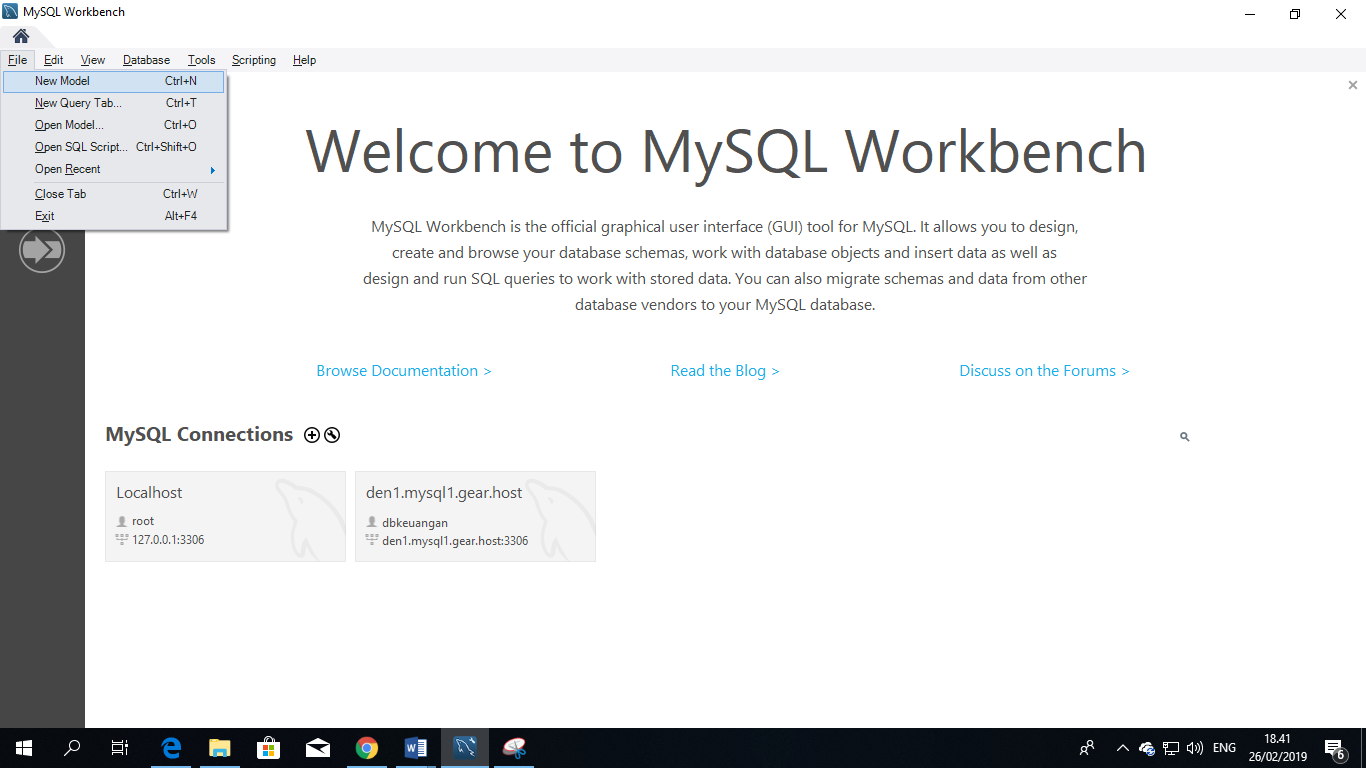


* Many-to-many

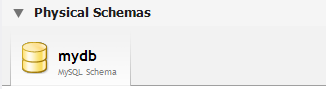


Pembuatan database dapat dilakukan dengan Melakukan Synchronize database melalui model ER, caranya sebagai berikut:

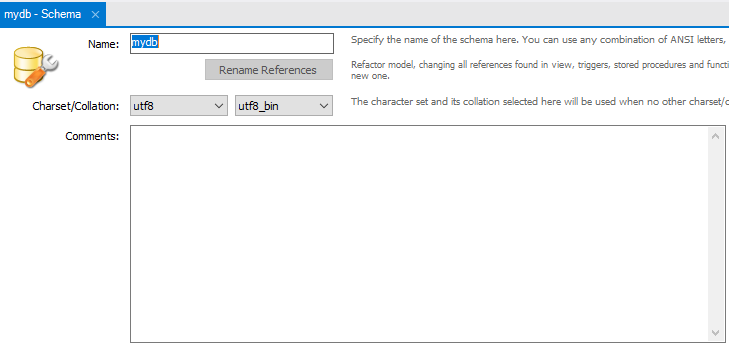
* 1. Buka Aplikasi MySQL Workbench,



* 1. Edit nama model (sesuaikan dengan nama database yang akan digunakan pada database) dengan cara click double pada nama model:

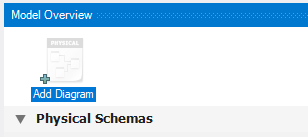


Contoh di atas click double pada “mydb” maka akan tapil seperti di bawah ini:

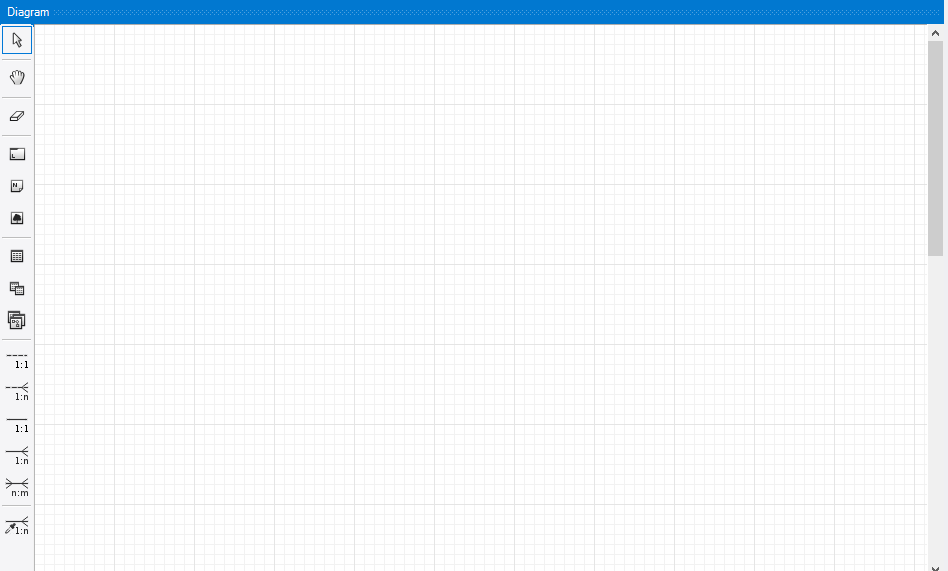


Set name sesuai dengan yang diinginkan.

* 1. Buat Diagram model dengan click double pada “Add Diagram”:



Maka atan terbuka form baru untuk diagram model seperti gambar di bawah ini



ERD model di gambarkan pada canvas yang telah tersedia,

* 1. Setelah selesai menggambar model ERD lakukan Synchronize dengan memilih menu Database 🡢 Synchronize Model

Selesaikan proses sampai selesai

1. **Kegiatan Praktikum**
   1. Buat Model ERD pada MySQL Workbench berdasarkan table hasil normalisasi dari form yang sudah dibuat
   2. Synchronize database MySQL sesuai dengan model ERD yang telah dibuat.
2. **Tugas Praktikum**
3. Buat Laporan hasil praktikum yang telah dilakukan sebelumnya.

**MODUL 3**

**MENGELOLAH OBYEK PRIMER DATABASE**

1. **Tujuan Praktikum**
2. Mahasiswa memahami cara pengelolaan obyek primer database
3. Mahasiswa mampu untuk mengelolah user akses pada database
4. Mahasiswa mampu memahami struktur table pada database,
5. Mahasiswa mengerti dan mampu untuk membuat view pada database
6. **Tugas Pendahuluan**

Siapkan ERD yang dibuat melalui proses normalisasi

1. **Dasar Teori**
2. User Database

User pada database adalah sebuah akun yang digunakan oleh pengguna basis data untuk dapat mengakses database untuk keperluan mengelolah database. konsep user yang ada di database MySQL yaitu user ada diluar database, berbeda dengan konsep user yang ada di oracle. Schema untuk user pada MySQL ditunjukkan pada gambar berikut:



Ilustrasi Schema user pada database MySQL

Untuk membuat user akses pada MySQL harus melalui superuser dalam hal ini user admin. Untuk membuat user akses pada database MySQL digunakan sintax:

“CREATE USER ‘nama\_user’@’nama\_hostuser’ IDENTIFIED BY ‘passworduser’;”

Secara default perintah sudah dapat menciptakan sebuah user tapi tidak dapat melakukan suatu pekerjaan, hal tersebut dikarenakan tidak adanya pengaturan akses yang dikenakan pada user tersebut. Untuk memberikan suatu akses terhadap suatu database kepada user maka digunakan perintah “GRANT”, sintax lengkapnya sebagai berikut:

“GRANT ALL ON \*.\* TO ‘[nama\_user]’@’[namahost]’;”

Perintah “GRANT ALL ON” digunakan untuk memberikan seluruh hak akses terhadap semua database (\*.\*).

Jika ingin memberikan printah lebih sepesifik bisa dilakukan dengan menyebutkan hak aksesnya saja misalnya “GRANT SELECT ON”, “GRANT UPDATE ON”, “GRANT INSERT ON”, “GRANT DELETE ON” dan lain sebagainya, tapi dengan menyebutkan nama database dan tablenya. Contoh:

“GRANT ALL ON Akademik TO ‘[nama\_user]’@’[namahost]’;”

Sintax di atas memberikan seluruh hak akses kepada user hanya terdapat database “Akademik” saja. Jika ingin lebih spesifik bisa dilakukan dengan perintah:

“GRANT SELECT ON nama\_database.nama\_table. TO ‘[nama\_user]’@’[namahost]’;”

Sintax di atas memberikan hak akses “SELECT” terhadap table pada database.

Setelah memberikan akses terhadap user perlu diikuti dengan perintah

“FLUSH PRIVILEGES”

Proses pembuatan user dan pemberian akses terhadap user sudah selesai, selanjutnya kita bisa melihat daftar user pada database dengan sintax sebagai berikut:

SELECT user, host FROM mysql.user;

Sedangkan untuk melihat akses yang berlaku pada user bisa dengan menggunakan sintax berikut

SELECT \* FROM information\_schema.user\_privileges;

Pertanyaan selanjutnya adalah “bagaimana jika kita ingin menghapus akses terdahap user?”. Proses penghapusan bisa dilakukan dengan printah “REVOKE” sintax lengkapnya seperti di bawah ini:

REVOKE ALL ON nama\_database.\* FROM ‘nama\_user’@’nama\_host’;

Perintah di atas jika dijalankan akan menghapus seluruh akses pada seluruh table pada sebuah database.

Untuk penghapusan user bisa dilakukan dengan sintax

DROP USER ‘nama\_user’@’nama\_host’;

1. Constraint

Constraint adalah batasan atau aturan yang ada pada table.

MySQL menyediakan beberapa tipe constraint berikut :

* NOT NULL

Suatu kolom yang didefinisikan dengan constraint NOT NULL tidak boleh berisi nilai NULL. Kolom yang befungsi sebagai kunci primer (primary key) otomatis tidak boleh NULL.

* UNIQUE

Mendefinisikan suatu kolom menjadi bersifat unik, artinya antara satu data dengan data lainnya namanya tidak boleh sama, misal alamat email.

* RIMARY KEY

Constraint PRIMARY KEY membentuk key yang unik untuk suatu table.

* FOREIGN KEY

FOREIGN KEY constraint didefinisikan pada suatu kolom yang ada pada suatu table, dimana kolom tersebut juga dimiliki oleh table yang lain sebagai suatu PRIMARY KEY, biasa dipakai untuk menghubungkan antara 2 tabel.

Contoh penggunaan constraint sebagai berikut:

create table Mahasiswa(

MahasiswaId INT NOT NULL AUTO\_INCREMENT,

NPM VARCHAR(9) NOT NULL

NamaMahasiswa VARCHAR(100) NOT NULL,

Kontak VARCHAR(40) NOT NULL,

TanggalLahir DATE,

PRIMARY KEY (MahasiswaId)

UNIQUE KEY MahasiswaId (MahasiswaId))

ENGINE=InnoDB;

1. View

**View** dapat didefenisikan sebagai **‘tabel virtual’**. Tabel ini bisa berasal dari tabel lain, atau gabungan dari beberapa tabel.

Tujuan dari pembuatan **VIEW** adalah untuk kenyamanan (mempermudah penulisan query), untuk keamanan (menyembunyikan beberapa kolom yang bersifat rahasia), atau dalam beberapa kasus bisa digunakan untuk mempercepat proses menampilkan data (terutama jika kita akan menjalankan query tersebut secara berulang). Contoh pembuatan view sebagai berikut:

CREATE VIEW TampilMahasiswa AS SELECT NPM, NamaMahasiswa, Kontak FROM Mahasiswa;

Cara pemanggilan dengan cara

SELECT \* FROM TampilMahasiswa;

1. Sequence

Sequence merupakan suatu object yang terdapat pada database yang berfungsi untuk melakukan penomoran otomatis, dan pada Mysql kita mengenal yang namanya dengan nama ***Auto Increment*** yang biasanya digunakan pada Primary Key yang terdapat pada sebuah table didalam database. Auto Increment tersebut bersifat otomatis yang nilainya selalu bertambah, nah pada tutorial kali ini kita akan membahas Cara Membuat Sequence pada Mysql.

1. **Kegiatan Praktikum**
2. Buka command prompt kemudian akses database MySQL seperti gambar di bawah ini



1. Buat sebuah database sesuai dengan ERD yang sudah disiapkan pada tugas pendahuluan
2. Ikuti prosedur untuk user dari membuat user sampai dengan penghapusan user, untuk pemberian akses gunakan database yang dibuat pada point 2
3. Ulangi point 3 sampai anda benar2 paham.
4. **Tugas Akhir Modul**

Buat laporan dari kegiatan praktikum.

**MODUL 4**

**ORDER BY, GROUP BY, UNION, AGGREGATE FUNCTION**

1. **Tujuan Praktikum**
2. Mahasiswa memahami penggunaan Order By pada query MySQL
3. Mahasiswa memahami penggunaan Group By pada query MySQL
4. Mahasiswa memahami penggunaan Anggregate Function pada query MySQL
5. **Dasar Teori**
6. Order By

Order By biasa digunakan dalam mengurutkan data yang dikembalikan oleh hasil queri

Sintax umumnya sebagai berikut:

Ascending

SELECT Field1, Field2, Field..n FROM NamaTabel ORDER BY field

Descending

SELECT Field1, Field2, Field..n FROM NamaTabel ORDER BY field DESC

1. Group By

Digunakan untuk menampilkan atau memilih sekumpulan data berdasarkan kelompok data tertentu.

* Pengelompokkannya biasanya diikuti oleh Aggregate Function
* Dalam implementasinya Aggregate function harus diikuti oleh Group By bila terdapat field lain yang dijadikan kriteria pengelompokkan.

1. Aggregate Function

Suatu fungsi yang digunakan untuk mendapatkan nilai tunggal berdasarkan hasil perhitungan nilai-nilai pada suatu kolom, contohnya adalah sebagai berikut :

MIN()

Digunakan jika ingin mengambil nilai terkecil pada sebuah data record di suatu tabel sintaxnya sebagai berikut:

SELECT Field-1,....Field-n, MIN (nama\_field) AS nama FROM tabel GROUP BY (Field-1,....Field-n)

MAX()

Max() merupakan kebalikan dari min(), yaitu untuk mengambil nilai tertinggi pada sebuah data record, sintax penggunaannya sebagi berikut:

SELECT Field-1,....Field-n, MAX (nama\_field) AS nama FROM tabel GROUP BY (Field-1,....Field-n)

AVG()

Fungsi ini digunakan untuk mencari nilai rata-rata pada data record di tabel tertentu.

Sintaxnya sebagai berikut:

SELECT AVG (nama\_field) AS nama FROM tabel GROUP BY (Field-1,....Field-n)

SUM()

Digunakan untuk menghitung total

COUNT()

Digunakan untuk memperoleh jumlah data pada sebuah tabel.

1. **Kegiatan Praktikum**
2. Buat table barang dan isi datanya seperti dibawah ini :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **kode\_brg** | **nama\_brg** | **harga\_modal** | **harga\_beli** | **stok** |
| B01 | Sabun | 2000 | 2500 | 15 |
| B02 | Pasta Gigi | 2500 | 3000 | 15 |
| B03 | Sikat Gigi | 3000 | 4000 | 10 |
| B04 | Rokok | 6000 | 7000 | 30 |
| B05 | Korek Api | 500 | 600 | 10 |

Buat SQL sehingga tampilannya sebagai berikut :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **kode\_brg** | **nama\_brg** | **harga\_modal** | **harga\_beli** | **stok** |
| B02 | Pasta Gigi | 2500 | 3000 | 15 |
| B03 | Sikat Gigi | 3000 | 4000 | 10 |
| B05 | Korek Api | 500 | 600 | 10 |

Buat SQL sehingga tampilannya sebagai berikut :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **kode\_brg** | **nama\_brg** | **harga\_modal** | **harga\_beli** | **stok** |
| B01 | Sabun | 2000 | 2500 | 15 |
| B03 | Sikat Gigi | 3000 | 4000 | 10 |

Buat SQL sehingga tampilannya sebagaia berikut :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **kode\_brg** | **nama\_brg** | **harga\_modal** | **harga\_beli** | **stok** |
| B01 | Sabun | 2000 | 2500 | 15 |
| B02 | Pasta Gigi | 2500 | 3000 | 15 |
| B03 | Sikat Gigi | 3000 | 4000 | 10 |
| B05 | Korek Api | 500 | 600 | 10 |

Buat SQL sehingga tampilannya sebagai berikut :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **kode\_brg** | **nama\_brg** | **harga\_modal** | **harga\_beli** | **stok** |
| B04 | Rokok | 6000 | 7000 | 30 |

Buat SQL sehingga tampilannya sebagai berikut :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **kode\_brg** | **nama\_brg** | **harga\_modal** | **harga\_beli** | **Stok** |
| B01 | Sabun | 2000 | 2500 | 15 |
| B04 | Rokok | 6000 | 7000 | 30 |

1. Implementasikan seluruh fungsi-fungsi yang ada pada Dasar Teori terhadap database yang sudah dibuat pada tugas sebelumnya.
2. **Tugas Ptaktikum**

Laporkan hasil praktikum dalam bentuk laporan.

**MODUL 5**

**PENGENDALIAN SERVER**

* + 1. **Tujuan Praktikum**

1. Mahasiswa memahami proses pengendalian server
2. Mahasiswa mampi mengimplementasikan proses pengendalian server
   * 1. **Dasar Teori**

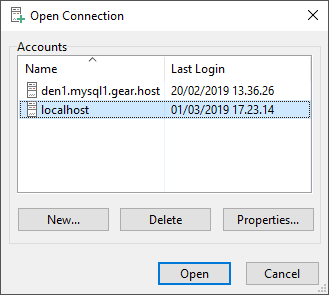
Bagian ini mempelajari tentang proses pengendalian server databases melalui computer client. Tahapan untuk pengendalian databases client-server sebagai berikut:

1. Pada sisi server, Daftarkan user beserta alamat ip client yang akan di gunakan untuk mengakses database, contohnya seperti sintax di bawah ini:

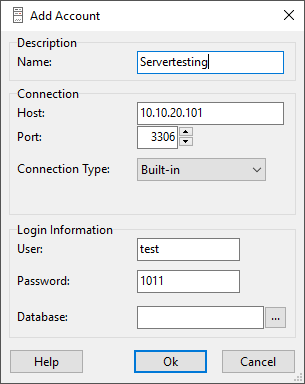
CREATE USER ‘nama\_user’@’ipclient’ IDENTIFIED BY ‘passworduser’;

Tentukan databases yang dapat diakses oleh client beserta jenis akses yang diberikan

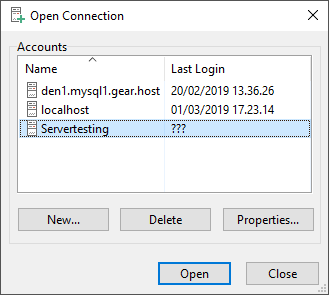
1. Pada sisi client buka aplikasi MySQL Front, buat koneksi ke server seperti contoh di bawah ini:



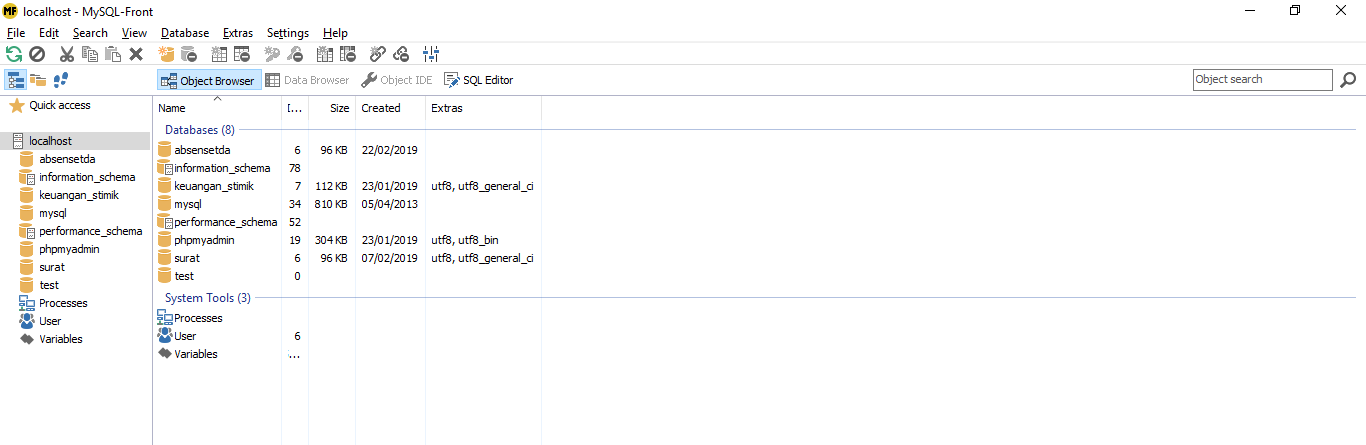
1. Pilih new kemudian tentukan nama koneksi (bisa apa saja), isikan host server (ip server), masukkan user beserta password sesuai dengan nama user yang sudah didaftarkan pada server



1. Kemudian click “ok” maka akan kembali ke form sebelumnya



1. Pilih koneksi yang telah dibuat kemudian click “open”, jika tidak ada kesalahan maka database akan di open



1. Coba lakukan queri sesuai dengan akses yang diberikan, lakukan juga queri untuk akses yang tidak diberikan.
   * 1. **Kegiatan Praktikum**
2. Buat kelompok (setiap kelompok terdiri dari 2 orang) yang akan bertindak sebagai server dan client
3. Lakukan tahapan pada dasar teori di atas sesuai dengan perannya masing-masing
   * 1. **Tugas Praktikum**

Buat laporan untuk masing-masing orang sesuai dengan hasil praktikum yang sudah dilakukan

**MODUL 6**

**FUNCTION DAN STORE PROCEDURE**

1. **Tujuan Praktikum**
2. Mahasiswa memahami dari function pada database
3. Mahasiswa mampu untuk mengimplementasikan function pada database MySQL
4. **Dasar Teori**

**Store procedure**

Store procedure dan function adalah rangkaian program yang disimpan dalam database dan dapat dipanggil oleh program lain atau melalui SQL Prompt. Store procedure ditulis dalam bentuk suatu script.

Untuk membuat *stored procedure/function* pada *database* digunakan pernyataan CREATE PROCEDURE atau CREATE FUNCTION.

### Keuntungan :

1. Cepat, kompilasi dilakukan didatabase (kadang disebut “pre compilation”) sehingga mengurangi traffic
2. Adanya pemisahan antara database dan access logic dengan application logic sehingga program aplikasi menjadi lebih sederhana dan lebih ringkas
3. Berupa obyek dalam database, sehingga menghilangkan ketergantungan terhadap bahasa program yang digunakan
4. Bersifat portable, jika bisa berjalan didatabase tersebut maka dipastikan jika database bisa terinstall dimanapun maka store procedure pasti bisa dijalankan

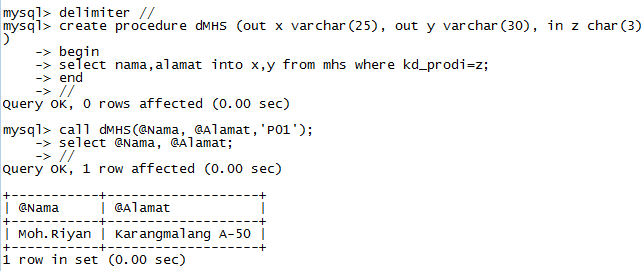
### Bentuk Umum Store Procedure

Agar lebih jelas, perhatikan contoh penggunaannya berikut ini :

### contoh 1:

Dari contoh diatas terlihat bahwa parameter “x” (sebagai OUT) digunakan untuk menampung hasil dari perintah *routine\_body*. Pernyataan “into x”, inilah yang mengakibatkan “x” menyimpan informasi **nama** (sebagai kolom yang ter-*select*). Untuk menjalankan *procedure* digunakan ststemen **call**. Pernyataan “call pMHS(@Nama)” menghasilkan informasi yang kemudian disimpan pada parameter “@Nama”. Kemudian untuk menampilkan informasi ke layar digunakan pernyataan “select @Nama”.

### Contoh 2 :



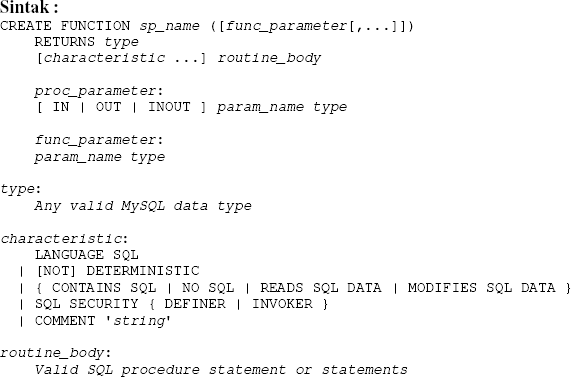
Dari contoh yang kedua ini terlihat bahwa parameter “z” (sebagai IN) digunakan sebagai jalur untuk masukan *routine* dan parameter “x” dan “y” digunakan untuk menampung hasil dari perintah *routine\_body*. Pernyataan “into x, y”, inilah yang mengakibatkan “x” dan “y” menyimpan informasi **nama** dan **alamat** (sebagai kolom yang ter-*select*).

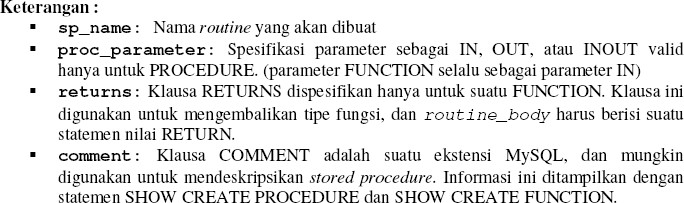
Pernyataan “call dMHS(@Nama, @Alamat)” menghasilkan informasi yang kemudian disimpan pada parameter **@Nama** dan **@Alamat**, sedangkan parameter “z” digunakan untuk menampung string „P01‟ yang kemudian digunakan untuk memproses *routine\_body* . Kemudian untuk menampilkan informasi ke layar digunakan pernyataan “select @Nama, @Alamat”.

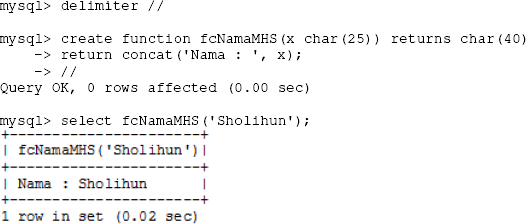
Jika diperhatikan pada **contoh1** dan **contoh2**, dalam membuat *routine* selalu menggunakan **delimiter.** Hal ini digunakan untuk mengubah pernyataan *delimiter* dari “;” ke “//” ketika suatu *procedure* sedang didefinisikan. Sehingga sebelum *delimiter* ditutup, meskipun sudah ditekan **enter** masih dianggap satu-kesatuan perintah. Jika menggunakan perintah **delimiter**, maka untuk menutupnya digunakan karakter *backslash* („\‟) karena karakter ini merupakan karakter **escape** untuk MySQL.

**Function**

Secara *default*, *routine* (*procedure/function*) diasosiasikan dengan *database* yang sedang aktif. Untuk dapat mengasosiasikan *routine* secara eksplisit dengan *database* yang lain, buat *routine* dengan format: **db\_name.sp\_name**.

MySQL mengijinkan beberapa *routine* berisi statemen DDL, seperti CREATE dan DROP. MySQL juga mengijinkan beberapa *stored procedure* (tetapi tidak *stored function*) berisi statemen SQL *transaction*, seperti COMMIT. *Stored function* juga berisi beberapa statemen baik yang secara eksplisit atau implisit **commit** atau **rollback.**

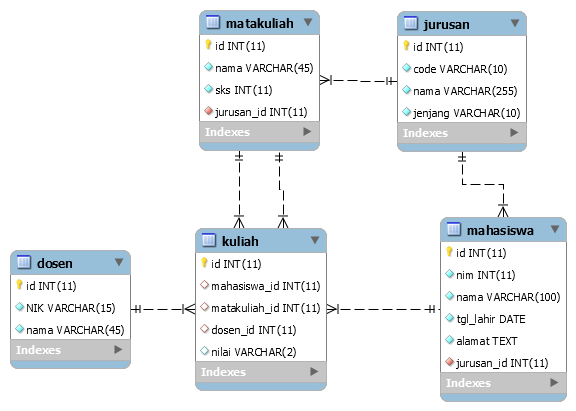


**Contoh**

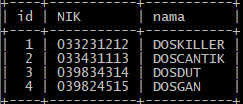
Dari contoh diatas terlihat bahwa parameter “x” diperlakukan sebagai IN karena sebagaimana dijelaskan sebelumnya bahwa fungsi hanya bisa dilewatkan dengan parameter IN. Kemudian untuk pengembalian nilainya, digunakan tipe data dengan kisaran nilai tertentu (dalam hal ini char(40)) dengan diawali pernyataan **returns**.

Pernyataam “concat('Nama : ', x)” merupakan *routine\_body* yang akan menghasilkan gabungan string “Nama :” dengan nilai dari parameter “x” yang didapat ketika fungsi ini dieksekusi. Perintah yang digunakan untuk mengeksekusi fungsi adalah “select fcNamaMHS('Sholihun')”.

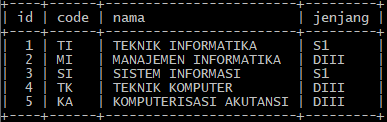
1. **Kegiatan Praktikum**
2. Buat database dengan nama akademik beserta tabel-tabelnya seperti gambar Entity Relationship Diagram di bawah ini:



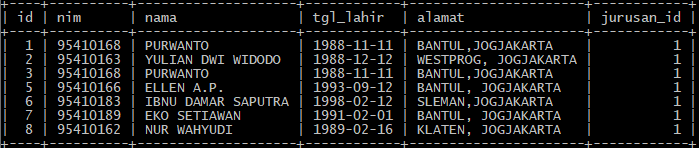
1. Masukkan data pada tabel-tabel tersebut.
2. Tabel Dosen



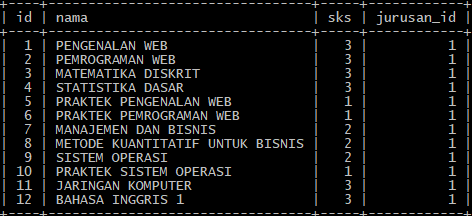
1. Tabel Jurusan



1. Tabel Mahasiswa



1. Tabel Matakuliah



1. Tabel Kuliah

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | mahasiswa\_id | matakuliah\_id | dosen\_id | nilai | id | mahasiswa\_id | matakuliah\_id | dosen\_id | nilai |
| 1 | 2 | 1 | 1 | A- | 49 | 6 | 1 | 1 | B |
| 2 | 2 | 2 | 1 | B+ | 50 | 6 | 2 | 1 | B |
| 3 | 2 | 3 | 3 | B | 51 | 6 | 3 | 3 | B |
| 4 | 2 | 4 | 4 | B | 52 | 6 | 4 | 4 | B |
| 5 | 2 | 5 | 1 | B | 53 | 6 | 5 | 1 | B |
| 6 | 2 | 6 | 1 | C- | 54 | 6 | 6 | 1 | C |
| 7 | 2 | 7 | 3 | B | 55 | 6 | 7 | 3 | B |
| 8 | 2 | 8 | 4 | B | 56 | 6 | 8 | 4 | A |
| 9 | 2 | 9 | 1 | B | 57 | 6 | 9 | 1 | A |
| 10 | 2 | 10 | 1 | A | 58 | 6 | 10 | 1 | A |
| 11 | 2 | 11 | 3 | A- | 59 | 6 | 11 | 3 | A |
| 12 | 2 | 12 | 4 | B+ | 60 | 6 | 12 | 4 | A |
| 13 | 8 | 1 | 1 | B | 61 | 1 | 1 | 1 | B |
| 14 | 8 | 2 | 1 | B+ | 62 | 1 | 2 | 1 | B |
| 15 | 8 | 3 | 3 | B | 63 | 1 | 3 | 3 | B |
| 16 | 8 | 4 | 4 | C | 64 | 1 | 4 | 4 | B |
| 17 | 8 | 5 | 1 | D+ | 65 | 1 | 5 | 1 | B |
| 18 | 8 | 6 | 1 | B | 66 | 1 | 6 | 1 | B |
| 19 | 8 | 7 | 3 | C- | 67 | 1 | 7 | 3 | B |
| 20 | 8 | 8 | 4 | A | 68 | 1 | 8 | 4 | B |
| 21 | 8 | 9 | 1 | A | 69 | 1 | 9 | 1 | B |
| 22 | 8 | 10 | 1 | B | 70 | 1 | 10 | 1 | A |
| 23 | 8 | 11 | 3 | B | 71 | 1 | 11 | 3 | B |
| 24 | 8 | 12 | 4 | B | 72 | 1 | 12 | 4 | A |
| 25 | 5 | 1 | 1 | B | 73 | 2 | 1 | 1 | B |
| 26 | 5 | 2 | 1 | A | 74 | 2 | 2 | 2 | A |
| 27 | 5 | 3 | 3 | C | 75 | 2 | 3 | 3 | B |
| 28 | 5 | 4 | 4 | B | 76 | 2 | 4 | 4 | B |
| 29 | 5 | 5 | 1 | A | 77 | 2 | 5 | 1 | C |
| 30 | 5 | 6 | 1 | A | 78 | 2 | 6 | 2 | A |
| 31 | 5 | 7 | 3 | B | 79 | 2 | 7 | 3 | A |
| 32 | 5 | 8 | 4 | B | 80 | 2 | 8 | 4 | A |
| 33 | 5 | 9 | 1 | B | 81 | 2 | 9 | 1 | A |
| 34 | 5 | 10 | 1 | A | 82 | 2 | 10 | 2 | A |
| 35 | 5 | 11 | 3 | C | 83 | 2 | 11 | 3 | A |
| 36 | 5 | 12 | 4 | A | 84 | 2 | 12 | 4 | C |
| 37 | 7 | 1 | 1 | A | 85 | 8 | 1 | 1 | B |
| 38 | 7 | 2 | 1 | A | 86 | 8 | 2 | 2 | C |
| 39 | 7 | 3 | 3 | A | 87 | 8 | 3 | 3 | B |
| 40 | 7 | 4 | 4 | B | 88 | 8 | 4 | 4 | C |
| 41 | 7 | 5 | 1 | B | 89 | 8 | 5 | 1 | B |
| 42 | 7 | 6 | 1 | B | 90 | 8 | 6 | 2 | B |
| 43 | 7 | 7 | 3 | C | 91 | 8 | 7 | 3 | B |
| 44 | 7 | 8 | 4 | B | 92 | 8 | 8 | 4 | B |
| 45 | 7 | 9 | 1 | C | 93 | 8 | 9 | 1 | B |
| 46 | 7 | 10 | 1 | B | 94 | 8 | 10 | 2 | A |
| 47 | 7 | 11 | 3 | A | 95 | 8 | 11 | 3 | A |
| 48 | 7 | 12 | 4 | A | 96 | 8 | 12 | 4 | A |

1. Buat stored procedure untuk menambah data dosen

DELIMITER //

CREATE PROCEDURE tambah\_dosen (

IN nikDosen VARCHAR(15),

IN namaDosen VARCHAR(45)

)

BEGIN

INSERT INTO dosen (‘id’,NIK,nama)

VALUES (‘’, nikDosen, namaDosen);

END //

DELIMITER ;

1. Buat stored procedure untuk menghapus data kuliah

DELIMITER //

CREATE PROCEDURE hapus\_kuliah (IN IdKuliah INT(11))

BEGIN

DELETE FROM kuliah WHERE id=IdKuliah;

END //

DELIMITER ;

1. Buat Store Procedure untuk menampilkan data penerbit

DELIMITER //

CREATE PROCEDURE tampil\_mahasiswa ()

BEGIN

SELECT \* FROM mahasiswa;

END //

DELIMITER ;

1. Menampilkan daftar store procedure yang telah dibuat dengan perintah

SELECT \* FROM mysql.proc where db=’perpustakaan’

1. Membuat Function untuk mengembalikan jumlah data pada tabel

DELIMITER $$  
CREATE FUNCTION sf\_tampil\_siswa\_kelas (p\_kelas int) RETURNS INT DETERMINISTIC  
BEGIN  
DECLARE jml INT;  
SELECT COUNT(\*) AS jml\_kelas INTO jml FROM tb\_siswa WHERE kelas = p\_kelas;  
RETURN jml;  
END$$  
DELIMITER ;

1. Function untuk cari nama penerbit berdasarkan kode penerbit :

DELIMITER $$

USE kampus$$

CREATE DEFINER=root@localhost FUNCTION TAMPILNAMA(NOMHS VARCHAR(30)) RETURNS varchar(30) CHARSET latin1

BEGIN

DECLARE NMAA VARCHAR(30);

SELECT nama FROM mahasiswa WHERE nim = NOMHS LIMIT 1 INTO NMAA;

RETURN NMAA;

END$$

DELIMITER ;

Cara pemanggilannya:

select TAMPILNAMA('95410163');

1. Function IP untuk mahasiswa :

DELIMITER $$

USE kampus$$

CREATE DEFINER=root@localhost FUNCTION HITUNgip(NOMHS CHAR(9)) RETURNS decimal(5,2)

BEGIN

DECLARE IP FLOAT;

SELECT

SUM(CASE

WHEN NILAI = 'A' THEN 4 \* b.sks

WHEN NILAI = 'A-' THEN 3.7 \* b.sks

WHEN NILAI = 'B+' THEN 3.4 \* b.sks

WHEN NILAI = 'B' THEN 3 \* b.sks

WHEN NILAI = 'B-' THEN 2.7 \* b.sks

WHEN NILAI = 'C+' THEN 2.4 \* b.sks

WHEN NILAI = 'C' THEN 2 \* b.sks

WHEN NILAI = 'C-' THEN 1.7 \* b.sks

WHEN NILAI = 'D+' THEN 1.4 \* b.sks

WHEN NILAI = 'D' THEN 1 \* b.sks

ELSE 0 \* b.sks

END) / SUM(b.sks) AS POINT

FROM

kuliah a,

matakuliah b,

mahasiswa c

WHERE

a.matakuliah\_id = b.id AND a.mahasiswa\_id = c.id AND c.nim = NOMHS

GROUP BY c.nim INTO IP;

RETURN IP;

END$$

DELIMITER;

untuk tes function 2 :

select hitungip('95410163');

1. Function 3 Hitung bobot :

DELIMITER $$

USE kampus$$

CREATE DEFINER=root@localhost FUNCTION HITUNGBOBOT(NIL CHAR(2), JSKS INT) RETURNS decimal(5,2)

BEGIN

IF NIL='A' THEN

RETURN 4\*JSKS;

END IF;

IF NIL='A-' THEN

RETURN 3.4\*JSKS;

END IF;

IF NIL='B+' THEN

RETURN 3.7\*JSKS;

END IF;

IF NIL='B' THEN

RETURN 3\*JSKS;

END IF;

IF NIL='B-' THEN

RETURN 2.7\*JSKS;

END IF;

IF NIL='C+' THEN

RETURN 2.4\*JSKS;

END IF;

IF NIL='C' THEN

RETURN 2\*JSKS;

END IF;

IF NIL='C-' THEN

RETURN 1.7\*JSKS;

END IF;

IF NIL='D+' THEN

RETURN 1.4\*JSKS;

END IF;

IF NIL='D' THEN

RETURN 1\*JSKS;

END IF;

IF NIL='E' THEN

RETURN 0\*JSKS;

END IF;

END$$

DELIMITER;

Nah untuk yang ke 3 sobat :

select HITUNGBOBOT('**A**',**2**);

1. Implementasikan contoh yang sudah ada pada database yang sudah sibuat dalam kelompok
2. **Tugas Praktikum**

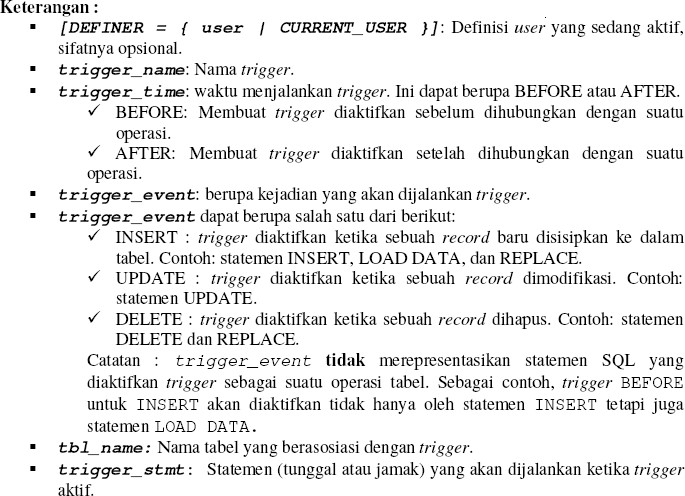
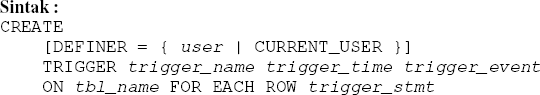
Buat laporan dari hasil praktikum yang sudah dilakukan

**MODUL 7**

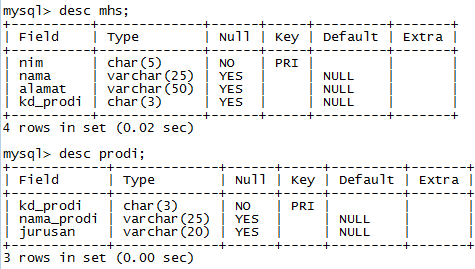
**TRIGGER**

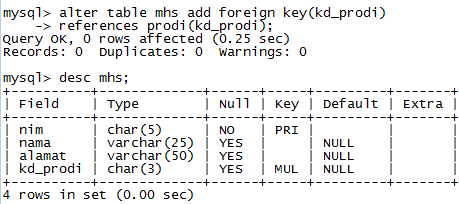
1. Dasar Teori

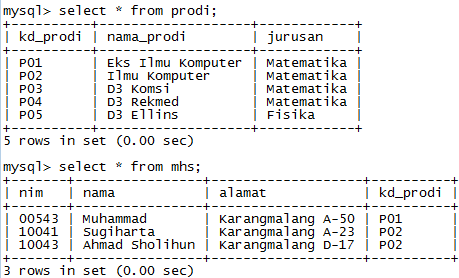
Pernyataan CREATE TRIGGER digunakan untuk membuat *trigger*, termasuk aksi apa yang dilakukan saat *trigger* diaktifkan. *Trigger* berisi program yang dihubungkan dengan suatu tabel atau *view* yang secara otomatis melakukan suatu aksi ketika suatu baris di dalam tabel atau *view* dikenai operasi INSERT, UPDATE atau DELETE.

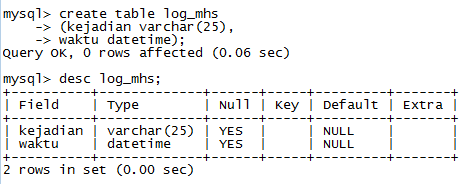


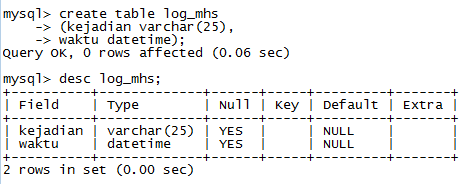
Sekarang kita masuk ke bahasan utama, yaitu implementasi. Untuk menerapkan TRIGGER, PROCEDURE, FUNCTION dan VIEW dibutuhkan suatu relasi, misalkan: *mahasiswa* dan *prodi*, sebagaimana yang diilustrasikan dengan perintah SQL di bawah ini.

1. Kegiatan Praktikum
2. Buatlah database dengan nama akademik.
3. Kemudian buatlah tabel mhs dan prodi dengan struktur sebagai berikut :
4. Agar kedua tabel diatas berelasi, buatlah kunci tamu pada tabel mhs yang mereferensi ke tabel prodi (kd\_prodi) sebagai berikut :

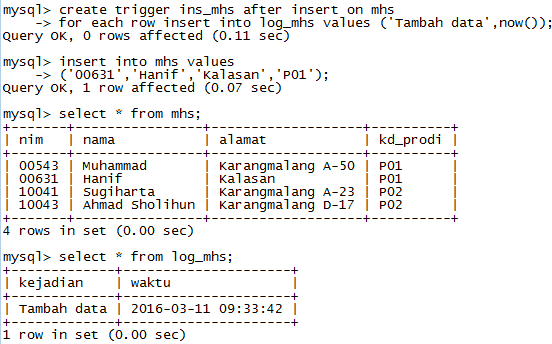


1. Isikan data ke tabel mhs dan prodi sebagai berikut :
2. Untuk pembuatan TRIGGER, contoh yang akan dibahas adalah mencatat kejadian- kejadian yang terjadi beserta waktunya pada tabel mhs, dan catatan-catatan tadi disimpan dalam tabel yang lain, misal **log\_mhs**. Misalkan struktur tabel **log\_mhs** adalah sebagai berikut:



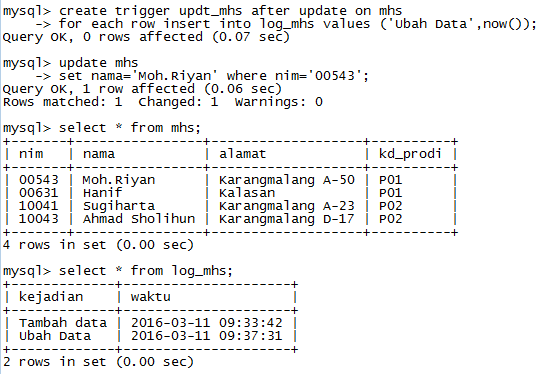


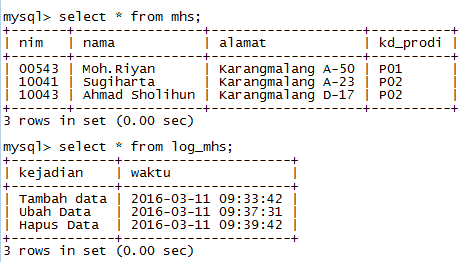
1. Buatlah TRIGGER untuk mencatat kejadian setelah dilakukan perintah INSERT pada tabel mhs dan disimpan ke dalam tabel log\_mhs sebagai berikut:



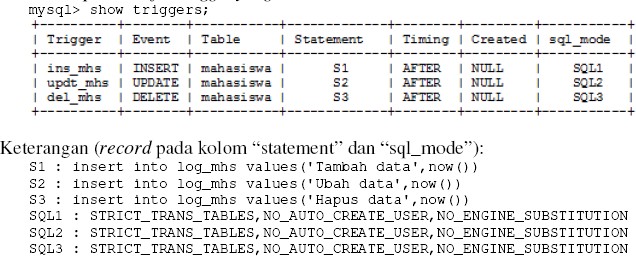
*Dari contoh diatas dapat dilihat bahwa ketika satu record pada tabel* ***mhs*** *disisipkan (insert), maka secara otomatis tabel* ***log\_mhs*** *akan disisipkan satu record, yaitu kejadian ‘Tambah data’ dan waktu saat record pada tabel* ***mhs*** *disisipkan*

1. Selanjutnya buatlah TRIGGER untuk mencatat perubahan data yang dilakukan setelah mendapatkan perintah UPDATE pada tabel mhs:



1. Kemudian TRIGGER untuk perintah DML DELETE pada tabel mhs sebagai berikut, dimana proses kejadiannya direkam pada tabel log\_mhs

*Dari contoh diatas dapat dilihat bahwa ketika satu record pada tabel* ***mhs*** *dihapus (delete), maka secara otomatis tabel* ***log\_mhs*** *akan disisipkan satu record, yaitu kejadian ‘Hapus data’ dan waktu saat record pada tabel* ***mhs*** *dihapus.*

1. Tampilkanlah semua TRIGGER yang telah dibuat

Karena hasil eksekusi perintah “show triggers” sangat panjang, tampilan di atas sengaja diedit dengan tujuan agar mudah dipahami

1. Tugas Praktikum

Implementasikan hasil praktikum di atas terhadap tabel kelompok, kemudian buat laporannya

**MODUL 8**

**BACKUP DAN RESTORE**

1. **Dasar Teori**

Sangat penting untuk mengenal perintah mysqldump,   Syntax yang digunakan adalah sebagai berikut.



Kode diatas adalah format penulisan mysqldump, lalu bagaimana contoh penulisannya, pada kesempatan kali ini jurnalweb.com akan mendemokan beberapa jenis cara backup mysql dengan berbagai tujuan.

Contoh cara backup

mysqldump -u root -p [password] nama\_database> jurnalwebcom.sql.

1. **Kegiatan Praktikum**

Lakukan backup database pada server kalian dengan CLI dan GUI