MODUL PRAKTIKUM

BASIS DATA 2

D3 Manajemen Informatika
Fakultas Teknik
Universitas Trunojoyo

MODUL I

ENTITY-RELATIONSHIP

Tujuan :
Mampu memahami konsep data model dari basis data dan merancang skema basis data dalam ER model dan dipresentasikan dalam bentuk gambar (ER Diagram).

Tugas Pendahuluan:

1. Apakah yang anda ketahui tentang ER-Model?
2. ER-Model digunakan dalam fase desain basis data konseptual, sebutkan dan jelaskan 6 tahap proses desain basis data tersebut?
3. Sebutkan dan jelaskan dari elemen-elemen dasar pada ER-Model?
4. Sebutkan dan jelaskan macam-macam batasan integritas (integrity constraints) dalam ER-Diagram?

1. Dasar Teori

Tinjauan Desain Basis Data

Pada ER model, gambaran dunia nyata diistilahkan dalam objek dan relasinya. ER model biasa digunakan untuk mengembangkan inisial dari desain basis data. ER model menyediakan suatu konsep yang bermanfaat yang dapat mengubah deskripsi informal dari apa yang diinginkan oleh user menjadi hal yang lebih detail, presisi dan deskripsi detail tersebut dapat diimplementasikan ke dalam DBMS.

Pada konteks yang lebih luas, ER-model digunakan dalam fase desain basis data konseptual. Berikut ini proses desain basis data yang dapat dibagi dalam 6 tahap ER model biasanya digunakan pada tiga tahap pertama dari proses desain dibawah ini:

1. Analisa Kebutuhan
   Pertama kali yang harus dipahami adalah kebutuhan user terhadap basis data. Proses pada tahap ini biasanya berlangsung secara informal meliputi diskusi dengan kelompok user, melakukan studi terhadap lingkungan operasi dan kemungkinan perubahan yang dilakukan, analisa terhadap dokumentasi dari aplikasi yang sudah ada yang dimaksudkan untuk diganti atau disempurnakan oleh basis data. Beberapa metodologi diusulkan dan alat bantu (tools) dikembangkan untuk mendukung proses ini.

2. Desain Konseptual Basis data
   Informasi dikumpulkan pada bagian analisis kebutuhan dan digunakan untuk mengembangkan deskripsi tingkat tinggi dari data yang disimpan dalam basis data, dengan cosntraints yang digunakan untuk menangani data-data ini. Pada tahap ini digunakan ER model.

3. Desain logika Basis data
   Pada tahap ini mengubah desain konseptual basis data ke dalam skema basis data dalam model data yang dipilih oleh DBMS. Dalam hal ini yang

dipilih adalah DBMS Relasional. Hasilnya berupa skema konseptual yang disebut juga dengan skema logika.

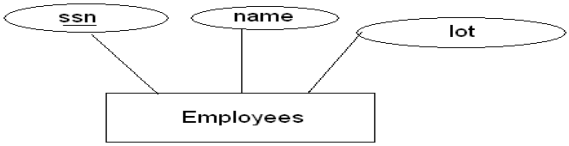
4. Skema Perbaikan
   Pada tahap ini, himpunan relasi dalam skema basis data relasional dianalisa untuk mengidentifikasi persoalan yang akan muncul, kemudian memperbaikinya. Dalam hal ini dilakukan normalisasi, dengan restrukturisasi ulang untuk memastikan beberapa properti yang dikehendaki.

5. Desain Fisik Basis data
   Pada tahap ini, ditentukan workload (masukan) yang harus disupport, memperbaiki desain berbasis data untuk memastikan kriteria performance yang diinginkan sudah tercapai. Tahap ini melibatkan indeks pada beberapa tabel dan pengelompokan (clustering) beberapa tabel, atau melibatkan pula desain ulang bagian skema basis data yang didapatkan dari tahap desain awal..

6. Desain Keamanan
   Pada tahap ini, didentifikasi kumpulan user yang berbeda dengan peranannya masing-masing (misalnya tim pengembangan produk, bagian customer, manajer produk, dll). Untuk tiap peran dan sekelompok user harus diidentifikasi bagian mana dari database yang dapat mereka akses dan bagian mana yang tidak boleh diakses.

Entiti, Atribut dan Himpunan Entiti

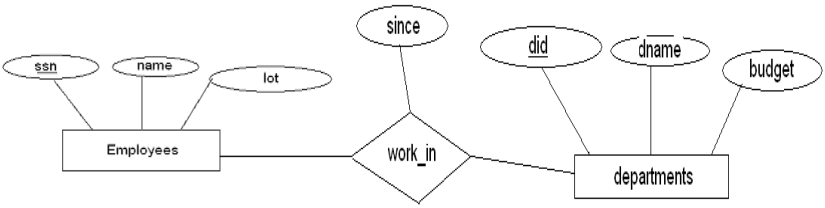
Entity adalah obyek dunia nyata yang dapat dibedakan dari obyek yang lain. Entity digambarkan (dalam basis data) dengan menggunakan himpunan atribut. Himpunan entiti yang sejenis disimpan dalam himpunan entiti. Himpunan entiti adalah kumpulan entity yang sejenis.



Gambar 1.2 Entiti Pegawai (Employee)

Relasi dan Himpunan Relasi

Relasi adalah asosiasi diantara dua atau lebih entity. Misalnya ani bekerja di departemen farmasi. Sedangkan himpunan relasi adalah himpunan relasi yang sejenis. Himpunan relasi n-ary R berelasi dengan sejumlah himpunan entity n E1 .... En sehingga himpunan entity yang sama dapar berpartisipasi dalam himpunan relasi yang berbeda. Atau mempunyai peran yang berbeda dalam suatu himpunan yang sama.



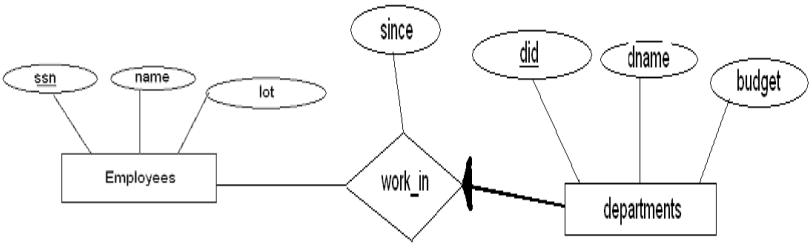
Gambar 1.3. Relasi dan Entity

Fitur Tambahan untuk ER Model

1.4.1 Batasan Kunci (Key Constraints)

Pada model database relasional, kaitannya atau asosiasi antara dua buah table disebut hubungan (relationship). Hubungan dapat berupa:

- 1 to 1, yakni satu data pada suatu tabel berpasangan dengan hanya satu data pada tabel lain. Contoh : seorang mahasiswa hanya dimungkinkan mempunyai sebuah no_reg dan satu no_reg hanya dapat ditugaskan pada satu orang mahasiswa saja.
- 1 to many, yakni satu data pada suatu tabel berpasangan dengan banyak data pada tabel lain. Contoh : Seorang mahasiswa dapat mengambil lebih dari satu matakuliah.
- Many to 1, yakni banyak data pada suatu tabel berpasangan dengan suatu data pada tabel lain.
- Many to Many, yakni banyak data pada suatu tabel berpasangan dengan banyak data pada tabel lain. Contoh : seorang dosen dapat mengajar banyak mahasiswa dan sebaliknya seorang mahasiswa juga dapat diajar lebih dari satu dosen.

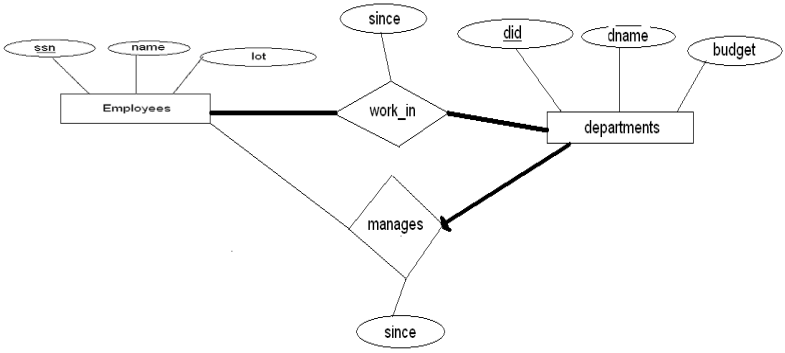


Gambar 1.4 Contoh Key Constraint antar Entity

1.4.2 Batasan Partisipasi (Participation Constraints)

Apakah setiap departemen mempunyai seorang manajer?

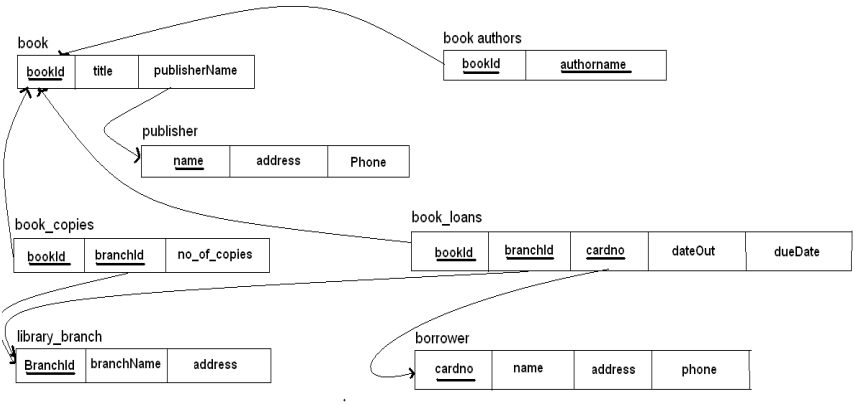
Jika semua departemen pasti mempunyai manager maka partisipasi *Departements* dalam manages dapat dikatakan total. Sebaliknya jika tidak semua departemen memiliki manager maka partisipasinya adalah partial.



Gambar 1.5 Contoh Participation Constraint

2. Kegiatan Praktikum

1. Buatkan rancangan entity dan relasi tiap tabel dari sistem perpustakaan berikut ini:

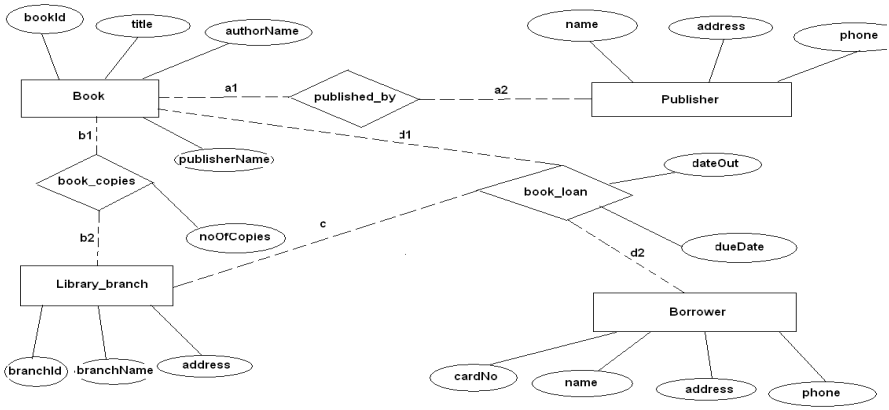


Gambar 1.6 skema relasi database untuk sistem perpustakaan

2. Tugas tambahan (ditentukan oleh asisten praktikum)!

3. Tugas Akhir

Dari kasus I pada kegiatan praktikum maka bagaimana desain batasan kunci nya?



I. Relasi published_by :
   a1 : -------- to a2 : ---------

II. Relasi book_copies :
   b1 : ------- to b2 : ----------

III. Relasi book_loan :
   d1 : ------- to c : ----------
   c : -------- to d2 : ---------

MODUL II
DATA DEFINITION LANGUAGE (DDL)

Tujuan :

Mampu memahami perintah-perintah untuk menjelaskan objek dari database dan mendefinisikan atribut-atribut database, tabel dan batasan-batasan terhadap suatu atribut serta hubungan antar tabel.

Tugas Pendahuluan :

1. Apakah yang anda ketahui tentang DDL?
2. Sebutkan statemen DDL untuk mendefinisikan kerangka database?
3. Apa yang anda ketahui tentang batasan kunci baik primer (primary key constraints) maupun foreign keys?
4. Bagaimana index dibuat dan kapan index perlu dibuat dan tidak perlu dibuat?
5. Apa yang anda ketahui view dan sebutkan dua macam tipe dari view?

1. Dasar Teori

1.1 Statement Data Definition Languange

DDL adalah perintah-perintah yang digunakan untuk menjelaskan objek dari database. Dengan kata lain DDL merupakan kelompok perintah yang berfungsi untuk mendefinisikan atribut-atribut database, tabel, atribut(kolom), batasan-batasan terhadap suatu atribut serta hubungan antar tabel. Yang termasuk kelompok DDL ini adalah:

- CREATE : untuk membuat database, tabel, view dan index
- ALTER : untuk mengubah struktur tabel
- DROP : untuk menghapus database, tabel, view dan index

1.2 Database, Tabel, Index dan View

1.2.1 Database

Database adalah sekumpulan data/informasi yang diorganisasikan dengan beberapa cara logika, saling berhubungan dan digunakan untuk keperluan tertentu. Sintaks untuk membuat database adalah:

```
CREATE DATABASE nama_db;
```

sedangkan sintaks untuk menghapus database adalah:

```
DROP DATABASE nama_db;
```

1.2.2 Tabel

Tabel adalah ekuivalensi dari sebuah entitas dalam sebuah ER model. Sebuah tabel terdiri dari beberapa kolom yang disebut dengan field dan sebuah field terdiri dari beberapa baris (record). Sedangkan field adalah ekuivalensi dari atribut sebuah entitas dalam sebuah ER model. Dapat juga dianggap sebagai bentuk pengelompokan data pada sebuah tabel. Record adalah satuan data atomik (terkecil) yang ada dalam sebuah tabel. Sebuah record terbentuk dari beberapa informasi/data dari beberapa field. Contoh : ”05621045, Anamisa, Jl. Pinang No.75” adalah sebuah record yang tersimpan dalam field tabel mahasiswa.

Untuk membuat sebuah relationship diperlukan field-field yang saling berpadanan. Field-field yang dimaksud adalah memiliki batasan integritas. Batasan integritas adalah suatu kondisi yang harus bernilai benar untuk suatu

instance dalam basis data, diantaranya adalah: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY. Sintaks untuk penulisan constraint:

```
CREATE TABLE nama_tabel (
        nama_column datatype [DEFAULT expr] [column_constraint],
        ….
        CONSTRAINT [table_constraint][,….]);
```

- Constraint NOT NULL

Suatu kolom yang didefinisikan dengan constraint NOT NULL tidak boleh berisi nilai NULL. Kolom yang berfungsi sebagai kunci primer (primary key) otomatis tidak boleh NULL. Syntaks untuk constraint NOT NULL adalah:

```
CREATE TABEL nama_tabel (
    nama_kolom      tipe_data,
    .......         …..
    CONTRAINT nama_kolom_constraint NOT NULL);
```

- Constraint UNIQUE

Mendefinisikan suatu kolom menjadi bersifat unik. Sintaks untuk constraint UNIQUE adalah:

```
CREATE TABEL nama_tabel (
    nama_kolom      tipe_data,
    .......         …..
    CONTRAINT table_constraint UNIQUE (nama_kolom_constraint)
);
```

- Constraint PRIMARY KEY

Membentuk key yang unik untuk suatu tabel. Kolom yang didefinisikan sebagai primary key akan mengidentifikasi suatu baris data menjadi unik.

```
CREATE TABLE nama_tabel (
    nama_kolom tipedata,
    …….
    CONSTRAINT table_constraint PRIMARY KEY
    ( nama_kolom_constraint )
```

```
);
```

Contoh penulisan constraint:

```
CREATE TABLE employees(
employee_id varchar( 6 ) ,
last_name VARCHAR( 20 ) ,
email VARCHAR( 20 ) ,
CONSTRAINT employee_id_pk PRIMARY KEY ( employee_id )
);
```

- Constraint FOREIGN KEY

Mendefinisikan pada suatu kolom yang ada pada suatu table, dimana kolom tersebut juga dimiliki oleh table yang lain sebagai suatu PRIMARY KEY.

Sintaks untuk constraint FOREIGN KEY adalah:

```
CREATE TABLE nama_tabel (
    nama_kolom tipedata,
    …….
    CONSTRAINT table_constraint FOREIGN KEY
    ( nama_kolom_constraint ) REFERENCES
    table_contraints_kunci_primer (nama_kolom_kunci_primer)
);
```

Sedangkan untuk menghapus tabel, dapat dilakukan dengan sintaksnya sebagai berikut:

```
DROP TABLE nama_tabel;
```

Sedangkan untuk mengubah tabel, dapat dilakukan dengan menggunakan sintaks sebagai berikut:

```
ALTER TABLE nama_tabel
    ADD (nama_kolom_baru type_kolom [BEFORE nama_kolom])
    MODIFY (nama_kolom_lama type_kolom)
     DROP (nama_kolom_lama type_kolom);
```

1.2.3 Index

Index adalah skema object yang digunakan untuk meningkatkan kecepatan dalam mendapatkan baris data yang diinginkan dengan menggunakan pointer. Dapat mereduksi disk I/O dengan menggunakan metode pengaksesan untuk melokasikan data secara cepat. Serta indepedent dari tabel yang diindeks.

Cara membuat index adalah index yang unik dibuat secara otomatis pada saat mendefinisikan constraint PRIMARY KEY atau UNIQUE dalam definisi tabel. Sedangkan secara manual user dapat membuat index non-unik pada kolom yang ada untuk meningkatkan kecepatan akses. Syntax untuk membuat index pada satu atau lebih kolom:

```
CREATE INDEX  nama_index
ON tabel(nama_kolom);
```

Contoh :

```
CREATE INDEX emp_last_name_idx ON employees(
last_name);
```

Sedangkan sintaks untuk menghapus index dari data dictionary digunakan DROP INDEX adalah:

```
DROP INDEX nama_index;
```

1.2.4. View

salah satu objek database, yang secara logika merepresentasikan sub himpunan dari data yang berasal dari satu atau lebih tabel. Kegunaan view adalah untuk membatasi akses database, membuat query kompleks secara mudah, mengijinkan independensi data dan untuk menampilkan view (pandangan) data yang berbeda dari data yang sama. Ada dua macam tipe view adalah simple view dan complex view. Berikut ini perbandingan antara simple view dan complex view adalah:

Tabel 2.1 Perbandingan Simple View dan Complex View

Fitur	Simple view	Complex View
Jumlah Table	Satu	Satu atau lebih
Berisi Fungsi	Tidak	Ya
Berisi Group Data	Tidak	Ya
DML melalui View	Ya	Tidak selalu

View dapat dibuat dengan perintah CREATE VIEW. Subquery dapat dicantumkan dalam CREATE VIEW, tapi subquery yang digunakan tidak boleh berisi klausa ORDER BY. Syntax penulisan VIEW adalah:

```
CREATE VIEW nama_view
 AS subquery ;
```

Contoh:

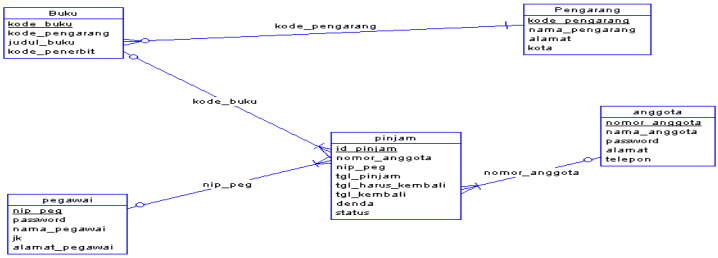
```
CREATE VIEW empvu80
AS select employee_id, last_name
    from employees
    where department_id = 80;
```

Sedangkan untuk memanggil data dari view, digunakan perintah yang sama seperti memanggil data dari tabel. Dan untuk menghapus view dengan sintaks sebagai berikut:

```
DROP VIEW nama_view;
```

2. Kegiatan Praktikum

1. Membuat system informasi perpustakaan dimana ada koleksi-koleksi buku yang disediakan, peminjaman oleh anggota perpustaan maka dalam system tersebut akses yang dapat dilakukan oleh anggota adalah dapat melihat data-data buku yang pernah dipinjam, denda yang harus dibayar juga melakukan pencarian buku-buku diperpustakaan tersebut. Dengan desain database sebagai berikut:


Buku
kode_buku
kode_pengarang
judul_buku
kode_penerbit
kode_pengarang
Pengarang
kode_pengarang
nama_pengarang
alamat
kota
kode_buku
pinjam
id_pinjam
nomor_anggota
nip_peg
tgl_pinjam
tgl_harus_kembali
tgl_kembali
denda
status
anggota
nomor_anggota
nama_anggota
password
alamat
telepon
nomor_anggota
nip_peg
pegawai
nip_peg
password
nama_pegawai
jk
alamat_pegawai

```
varchar(50) default NULL,
rchar(50) default NULL,
archar(20) default NULL,
nomor_anggota_pk PRIMARY KEY  (`nomor_anggota`)

`buku` (
 varchar(15) NOT NULL default '',
rang` varchar(10) NOT NULL default '',
` varchar(50) NOT NULL default '',
bit` varchar(10) default NULL,
allint(6) default '0',
kode_buku_pk PRIMARY KEY  (`kode_buku`),
kode_pengarang_fk FOREIGN KEY ( kode_pengarang ) REFERENCES pengarang


`pegawai` (
archar(20) NOT NULL default '',
varchar(50) default NULL,
ai` varchar(30) default NULL,
nt(6) default NULL,
` varchar(50) default NULL,
nip_peg_pk PRIMARY KEY  (`nip_peg`)

`pengarang` (
rang` varchar(10) NOT NULL default '',
rang` varchar(30) NOT NULL default '',
rchar(50) default NULL,
har(20) default NULL,
kode_pengarang_pk PRIMARY KEY  (`kode_pengarang`)

`pinjam` (
 int(11) NOT NULL auto_increment,
 varchar(30) NOT NULL default '',
ota` varchar(10) NOT NULL default '',
archar(20) NOT NULL default '',
` date NOT NULL default '0000-00-00',
kembali` date NOT NULL default '0000-00-00',
i` date default NULL,
(11) default NULL,
allint(6) default '0',
id_pinjam_pk PRIMARY KEY  (`id_pinjam`),
kode_buku_fk FOREIGN KEY ( kode_buku ) REFERENCES buku (kode_buku),
```

2. Amatilah apa yang terjadi dari kegiatan 1!
3. Tugas tambahan (ditentukan oleh asisten praktikum)!

3. **Tugas Akhir**

1. Buatlah desain database untuk suatu perusahaan :
    a. Tabel department dengan memiliki primary key pada field department_id. Dan tabel department terdiri atas field-field: department_id, department_name, manager_id dan location_id.
    b. Tabel employee dengan memiliki constraint foreign key pada department_id. Dan tabel employess terdiri dari: employee_id, last_name NOT NULL, email, salary, commission_pct, hire_date NOT NULL.
    c. Tabel-Tabel lain yang saling berelasi (seperti tabel persediaan barang, tabel pemasok barang, dll).
2. Buat view empvu80 yang berisi id_number, name, sal, department_id dari pegawai yang bekerja di department = 80. kemudian tampilkan struktur dari view empvu80.
3. Buat non-unique index pada kolom FOREIGN KEY yang ada pada tabel EMPLOYEE.

MODUL III
DML (DATA MANIPULATION LANGUAGE)

Tujuan :

- Praktikan dapat memahami dan mengisi tabel dalam database
- Praktikan dapat memahami dan memanipulasi data dalam database

Tugas Pendahuluan

- Apa yang anda ketahui tentang DML?
- Apa yang anda ketahui tentang perintah INSERT, UPDATE dan DELETE?
- Sebutkan macam-macam klausa maupun operator yang terdapat dalam perintah SELECT!

Dasar Teori

DML (Data Manipulation Language) adalah bahasa yang memungkinkan pengguna mengakses atau memanipulasi data seperti yang diatur oleh model data. Manipulasi data adalah :

- Pengambilan informasi yang disimpan dalam basisdata
- Penempatan informasi baru dalam basisdata
- Penghapusan informasi dari basisdata
- Modifikasi informasi yang disimpan dalam basisdata

DML (Data Manipulation Language) merupakan bahasa yang bertujuan memudahkan pemakai untuk mengakses data sebagaimana direpresentasikan oleh model data. Ada 2 jenis DML, yaitu :

- Prosedural, yang mensyaratkan agar pemakai menentukan, data apa yang diinginkan serta bagaimana cara mendapatkannya.
- Nonprosedural, yang membuat pemakai dapat menentukan data apa yang diinginkan tanpa menyebutkan bagaimana cara mendapatkannya.

Query adalah pernyataan yang meminta pengguna mengambil informasi. Bagian DML yang terlibat dalam pengambilan informasi disebut bahasa query. Istilah bahasa query sering disamakan dengan istilah bahasa manipulasi data. Sedangkan SQL adalah sebuah sintaks untuk mengeksekusi query.

Praktikum

Coba semua contoh operasi DML di bawah ini :

1. **Pernyataan INSERT INTO**

Pernyataan INSERT INTO digunakan untuk memasukkan data baru pada tabel.

Sintaks :

```
INSERT INTO nama_tabel
VALUES (nilai1, nilai2, ...)
```

Urutan nilai yang diletakkan dalam tanda kurung disesuaikan dengan urutan kolom dalam tabel. Akan tetapi kita bisa menentukan kolom-kolom yang akan diisi dengan data baru, yaitu :

```
INSERT INTO nama_tabel (kolom1, kolom2, ...)
VALUES (nilai1, nilai2, ...)
```

Kolom-kolom yang tidak disebutkan pada Insert secara otomatis akan diisi dengan Null dan kolom yang tidak disebutkan dalam Insert haruslah yang tidak Not Null.
Contoh :

Insert Into Mahasiswa
Values ('01012','Irwan','Jl.Beo 23','Bogor')

2. **Pernyataan UPDATE**

Pernyataan UPDATE digunakan untuk modifikasi data dalam tabel.
Sintaks :

UPDATE nama_tabel
SET nama_kolom = nilai_baru
WHERE nama_kolom = nilai

Pada pernyataan diatas :

- SET untuk menentukan kolom yang akan diubah dan nilai penggantinya.
- WHERE menentukan kondisi dari baris-baris yang akan diganti.

Contoh :

Update Mahasiswa Set Nama ='Riri'
Where NPM='01010'

3. **Pernyataan DELETE**

Pernyataan DELETE digunakan untuk menghapus baris pada tabel.
Sintaks :

DELETE FROM nama_tabel
WHERE nama_kolom = nilai

Contoh :
Untuk menghapus baris pada tabel Mahasiswa yang nilai NPMnya adalah 01013, anda bisa memberikan pernyataan seperti berikut :

Delete From Mahasiswa
Where NPM ='01013'

Dalam perintah DELETE jika kita ingin menghapus semua data pada tabel tanpa menghapus tabel maka Where tidak perlu disebutkan.

DELETE FROM nama_tabel
Atau
DELETE * FROM nama_tabel

4. **Pernyataan SELECT**

Secara umum perintah SELECT hanya difungsikan untuk menampilkan data yang ada di dalam suatu tabel. Tetapi dalam pengembangannya, perintah ini akan menjadi sebuah perintah yang sangat penting dan berpengaruh hingga saat pemrograman di stored procedures dan triggers. Dalam basis data 2 ini yang kita bahas adalah semua pernyataan SELECT yang digunakan untuk memilih data dari tabel, yang mana hasilnya disimpan dalam tabel hasil yang disebut Result Set.

Sintaks :

```
SELECT nama_kolom
FROM nama_tabel
```

Untuk memilih beberapa kolom gunakan pernyataan SELECT sebagai berikut :

```
SELECT nama_kolom1, nama_kolom2, ...
FROM nama_tabel
```

Contoh :

Pernyataan untuk menampilkan kolom NPM dan Nama yang terdapat pada tabel Mahasiswa.

```
Select NPM, Nama
From Mahasiswa
```

Untuk memilih semua kolom dari tabel, dapat menuliskan tanda asterisk (*) sesudah kata Select. Simbol * berarti semua kolom, seperti berikut :

```
SELECT *
FROM nama_tabel
```

Contoh :

```
Select *
From Mahasiswa
```

Dalam perintah SELECT banyak sekali perintah/klausa/operator yang bisa digunakan untuk memanipulasi data dengan lebih rinci, diantaranya adalah distinct, klausa where, like, alias, order by, group by, having, operator AND, OR, between ... AND, fungsi aggregate, dan masing banyak lagi yang lainnya.

Tugas :

Dari topik pembuatan aplikasi basis data yang telah anda buat, buatlah masing-masing perintah DML berikut :

1. Sisipkan min 10 baris data dalam masing-masing tabel
2. Tambahkan min 2 operasi update dan delete
3. Buatlah perintah SQL untuk menampilkan data yang berasal dari dua tabel dan tiga tabel.
4. Buatlah perintah SQL yang menggunakan klausa Order By, Group By, dan Having.
5. Buatlah perintah SQL yang menggunakan operator AND dan OR.
6. Buatlah perintah SQL yang menggunakan fungsi Agregate (masing-masing 1).
7. Buatlah perintah SQL yang menggunakan operator IN dan BETWEEN... AND.
8. Buatlah perintah SQL untuk menampilkan data yang berasal dari dua tabel dan tiga tabel dengan menggunakan perintah JOIN.
9. Buatlah perintah SQL yang menggunakan UNION, INTERSECT, dan EXCEPT.
10. Buatlah perintah SQL yang menggunakan operator comparison ANY dan ALL.

MODUL IV
STORED PROCEDURE DAN TRANSAKSI

Tujuan :

1. Mampu memahami dan membuat procedure atau function dalam basis data
2. Mampu menggunakan perintah-perintah dalam stored procedure

Tugas Pendahuluan :

1. Apa yang anda ketahui tentang stored procedure !
2. Bagaimana cara membuat procedure atau function dalam basis data !
3. Apa kegunaan perintah delimiter dalam stored procedured ! jelaskan dengan contoh

1.

1. Teori

1.1. Stored Procedure

Stored procedure dan stored function merupakan fasilitas baru dari MySQL versi 5.0. Stored procedure merupakan sekumpulan SQL yang disimpan ke dalam server MySQL. Keuntungan menggunakan store procedure, klien MySQL tidak perlu menuliskan perintah SQL ke server namun hanya perlu memanggil procedure yang sudah disimpan di server (jika tersedia). Perbedaan antara procedure dan function pada MySQL hampir mirip dengan procedure dan function pada bahasa pemrograman. Function mengembalikan suatu nilai skalar dan dapat dipanggil di dalam statement procedure atau function lain. Procedure dipanggil melalui perintah CALL dan dapat mengembalikan nilai melalui variabel output.

Stored Procedure adalah prosedur (spt subprogram dalam bhs pemrograman) yang disimpan di dalam database.
Mysql mendukung dua jenis “rutin” (subprogram):

- Stored procedure yang dapat dipanggil,

- fungsi yang menghasilkan nilai yang dapat dipakai dalam statemen SQL lain.

Statemen yang menciptakan stored procedure

```
CREATE procedure procedure1              /* nama */
(IN parameter1 INTEGER)                  /* parameter */
BEGIN                                    /* awal blok */
  DECLARE variable1 CHAR(10);            /* variabel */
  IF parameter1 = 17 THEN                /* awal IF */
   SET variable1 = 'burung';             /* assignment */
  ELSE
   SET variable1 = 'kelelawar';          /* assignment */
  END IF;                                /* akhir IF */
  INSERT INTO table1 VALUES (variable1); /* statement */
END  /*akhir blok  */
```

Dengan stored procedure eksekusi menjadi cepat. Tidak ada kompilasi.
Peningkatan kecepatan datang dari reduksi lalu-lintas jaringan. Jika ada pekerjaan pengecekan berulang, looping, multiple statement, dikerjakan dengan pemanggilan tunggal ke prosedur yang telah disimpan ke server.

Stored procedure adalah komponen. Andaikan aplikasi kemudian ditulis dalam bahasa berbeda, tidak ada masalah, karena logika berada didalam database bukan dalam aplikasi.

Stored Procedure adalah portable. Stored procedure ditulis dalam SQL, Anda bisa jalankan pada setiap platform dimana Mysql dijalankan disitu.

Create Procedure dan Create Function

Sebelum membuat procedure atau function terlebih dahulu pastikan bahwa Anda sudah masuk ke dalam suatu database (*use nama_database*). Bentuk umum dari perintah create procedure dan function adalah

```
Create procedure
<nama_procedure>
(parameter)
<karakteristik procedure>
<badan program>
```

Create function

<nama_procedure>
(parameter)
<karakteristik function>
<tipe data return>
<badan program>

Di mana :

Parameter = terdiri dari jenis parameter [IN, OUT, atau INOUT], nama parameter dan tipe data parameter.
Karakteristik = terdiri dari bahasa SQL, komentar, dsb.
Tipe data = tipe data yang dapat di*return* adalah semua tipe data yang valid di MySQL.
Program = semua syntax procedure SQL yang valid.

Jenis parameter ada 3 yaitu : IN berarti variabel parameter hanya berfungsi sebagai masukan, OUT berarti variabel parameter berfungsi sebagai tempat untuk menyimpan nilai keluaran dari procedure, dan INOUT berarti variabel parameter berfungsi sebagai masukan dan penyimpan nilai keluaran procedure.

2. Kegiatan Praktikum

1. Buat database dengan nama dbparkir, kemudian aktifkan

```
mysql> use dbparkir;
Database changed
```

2. Buat tabel kendaraan dalam database dbparkir
3. Tampilkan msql> select * from dbparkir;

```
mysql> select * from kendaraan;
+--------------+-----------------+
| no_kendaraan | jenis_kendaraan |
+--------------+-----------------+
| AG3561VA     | sepeda motor    |
| AG351JP      | sepeda motor    |
| AG4444ZA     | mobil           |
| AG4531CA     | sepeda motor    |
| 9887         | pesawat         |
+--------------+-----------------+
5 rows in set (0.06 sec)
```

4. Memilih delimiter

```
mysql> delimiter //
```

Kemudian sebelum create procedure dijalankan terdapat perintah delimiter. Yaitu menggantikan karakter berhenti MySQL dari ; menjadi |. Hasilnya MySQL akan mengabaikan karakter ; dan menganggapnya sebagai karakter biasa

5. Menciptakan stored procedure

```
mysql> create procedure p1 () select * from kendaraan; //
Query OK, 0 rows affected (0.17 sec)
```

p1 = nama prosedur

() = daftar parameter

SELECT * FROM kendaraan; = bodi prosedur

6. Memanggil procedure p1

```
mysql> call p1();
+--------------+-----------------+
| no_kendaraan | jenis_kendaraan |
+--------------+-----------------+
| AG3561VA     | sepeda motor    |
| AG351JP      | sepeda motor    |
| AG4444ZA     | mobil           |
| AG4531CA     | sepeda motor    |
| 9887         | pesawat         |
+--------------+-----------------+
5 rows in set (0.00 sec)

Query OK, 0 rows affected (0.03 sec)
```

7. Mengembalikan delimiter dan memulainya

```
mysql> delimiter ;
mysql> delimiter //
```

8. Menghapus delimiter

```
mysql> drop procedure p1;
    -> //
Query OK, 0 rows affected (0.11 sec)
```

9. Buat procedure sederhana yang memiliki fungsi untuk menghitung jumlah record pada tabel kendaraan. Kemudian jumlah kolom tersebut dimasukkan ke dalam variabel param1.

```
mysql> delimiter //
mysql> create procedure simple (out param1 int)
    -> begin
    -> select count(*) into param1 from kendaraan;
    -> end //
Query OK, 0 rows affected (0.05 sec)
```

10. panggil procedure dan lihat hasilnya

```
mysql> call simple (@a) //
Query OK, 0 rows affected (0.03 sec)

mysql> select @a //
+------+
| @a   |
+------+
| 5    |
+------+
1 row in set (0.00 sec)

mysql>
```

select count (*) into param1 from t;

Kemudian sebelum create procedure dijalankan terdapat perintah delimiter. Yaitu menggantikan karakter berhenti MySQL dari ; menjadi |. Hasilnya MySQL akan mengabaikan karakter ; dan menganggapnya sebagai karakter biasa. Semisal perintah delimiter ini tidak ada, maka pada contoh baris ke3 di mana terdapat karakter ; di akhir statement. MySQL akan menganggap baris perintah create procedure berakhir di situ. End ke bawah bukan bagian dari badan program procedure. Akibatnyaperintah create procedure kita salah.

Pada baris ke6, procedure yang telah dibuat dipanggil dengan fungsi call. Jangan lupa untuk memberikan parameter sesuai dengan yang dibutuhkan oleh procedure simple. Pada baris terakhir, variabel parameter tempat menyimpan keluaran dari procedure ditampilkan.

Mengapa variabel parameter untuk procedure di atas memakai tambahan karakter @? Hal ini supaya variabel parameter bisa diberlakukan pada fungsi select. Jika variabel parameter semua diganti tanpa @, maka perintah *select a*| tidak menampilkan hasil apaapa.

3. TUGAS

1. Buatlah sebuah prosedure dengan ketentuan :
    a. Nama Prosedur : latihan1
    b. Tanpa parameter
    c. Isi prosedur : menampilkan isi dari tabel actor (tabel harus ada, field sembarang)
    d. Buatlah cara untuk memanggil prosedur tersebut
2. Lihat isi dari procedure latihan
3. Hapuslah prosedur latihan
4. Penggunaan ekspresi:

Buatlah sebuah prosedur untuk menampilkan: customer_id, rental_id, amount, discount, total

dari tabel payment dimana :

- customer_id ditampilkan sebagai Kode Pelanggan
- Rental_id ditampilkan sebagai Kode Toko
- Amount ditampilkan sebagai Jumlah
- Discount dihitung dari amount * 0.1
- Total dihitung dari amount – discount

MODUL V
STORED PROCEDURE DAN TRANSAKSI

Tujuan :

1. Mampu memahami dan membuat procedure Transaksi dalam basis data
2. Mampu menggunakan perintah-perintah dalam transaksi

Tugas Pendahuluan :

1. Berikan kesimpulan mengenai stored procedure
2. Apa yang anda ketahui tentang transaksi !
3. Sebutkan dan jelaskan perintah-perintah dalam transaksi !

1. Dasar Teori

1.1. Parameter dalam stored procedure

1. Tanpa parameter
CREATE PROCEDURE p5 () ...

2. Satu parameter input
CREATE PROCEDURE p5
([IN] nama tipe-data) ...

3. Satu parameter output
CREATE PROCEDURE p5
(OUT nama tipe-data) ...

4. Satu parameter untuk input dan output
CREATE PROCEDURE p5
(INOUT nama tipe-data) ...

1.2. Arti Transaksi

Transaksi : serangkaian kelompok dari operasi manipulasi database yang dilakukan seolah-olah sebagai satu unit kerja secara individu. Jika sebuah operasi di dalam transaksi gagal dijalankan, maka keseluruhan transaksi juga akan gagal dijalankan.
4 properti standar yang dimiliki dari Transaksi pada MySQL :
(ACID)
Atomicity : memastikan bahwa seluruh operasi dalam unit kerja diselesaikan dengan baik. Jika tidak, transaksi akan dihentikan pada poin kegagalan dan operasi sebelumnya akan dibatalkan sehingga kembali ke keadaan semula.

Consistency : memastikan bahwa database secara tepat mengubah keadaan transksi yang berhasil dijalankan dengan commit

Isolation : memungkinkan transaksi untuk beroperasi secara independent

Durability : memastikan bahwa hasil atau efek dari transaksi dapat bertahan apabila sistem mengalami kegagalan

1.3. Auto Commit, Start Transaction, Commit, Roolback

Fasilitas penanganan kesalahan (error handling) biasa diperlukan untuk mengantisipasi terjadinya kesalahan pada suatu proses transaksi, sehingga programmer bisa mengatur skenario jika suatu operasi gagal sebagian atau seluruhnya.

Secara default skenario dari transaksi adalah AUTO COMMIT. Artinya semua proses yang berhasil dilaksanakan akan secara otomatis secara fisik disimpan dalam database. Jika diinginkan mulai dari posisi tertentu AUTO COMMIT tidak berfungsi, dapat digunakan perintah START TRANSACTION.

Selanjutnya sesuatu perintah sesudah pernyataan START TRANSACTION akan ditunda untuk disimpan, sampai bertemu pernyataan COMMIT yang akan menyimpan seluruh proses yang tertunda atau bertemu pernyataan ROLLBACK yang akan membatalkan seluruh proses yang tertunda.

Akan tetapi perlu diingat ada beberapa perintah yang tidak dapat di ROLL BACK karena mengandung fungsi COMMIT secara implisit.
Perintah tersebut adalah :

- ALTER TABLE
- BEGIN
- CREATE INDEX
- CREATE TABLE
- CREATE DATABASE
- DROP DATABASE
- DROP INDEX
- DROP TABLE
- LOAD MASTER DATA
- LOCK TABLES
- SET AUTOCOMMIT = 1
- START TRANSACTION
- TRUNCATE TABLE
- UNLOCK TABLES

2. Kegiatan Praktikum :

1. Buat Tabel Di bawah ini :

2. CREATE TABLE tabelmhs (
   no_mhs char(4) DEFAULT NULL,
   nama char(25) DEFAULT NULL,
   alamat char(25) DEFAULT NULL
) ENGINE=InnoDB;
3. Untuk membuat transaksi pada mysql gunakan engine = innoDB pada pembuatan/create tabelnya
4. Transaksi penambahan rekaman menggunakan START TRANSACTION

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)
```

5. Penambahan data tabel mhs

```
mysql> insert into tabelmhs values('1005','untung raharjo',
Query OK, 1 row affected (0.00 sec)
```

6. Lihat Hasil Penambahan

```
mysql> select * from tabelmhs;
+-------+----------------+-------------------------+
| nomer | nama           | alamat                  |
+-------+----------------+-------------------------+
| 1001  | Abdur Rahman   | Jl. Mangga 2, Surabaya  |
| 1002  | Fida Arini     | Jl. Rambutan 2., Malang |
| 1003  | Farida Awalia  | Jl. Rangga 8, Surabaya  |
| 1004  | Jaka Sambung   | Jl. Tungga 23, Surabaya |
| 1005  | untung raharjo | Jl. Bandung 23          |
+-------+----------------+-------------------------+
5 rows in set (0.05 sec)
```

7. Mahasiswa untuk sementara sudah direkam
8. Mambatalkan rekaman dengan perintah ROLLBACK;
   Mysql>rollback;
9. Tampilkan semua data tabelmhs setelah di berikan perintah rollback
10. mysql> START TRANSACTION;
    mysql> INSERT INTO mhs VALUES('0005','Untung Raharja','Bandung');
    mysql> INSERT INTO mhs VALUES('0006','Diah Ayu Subekti','Semarang')
    mysql> COMMIT;
11. mysql> rollback;

12. lihat hasilnya apa perbedaanya?

13. Buatlah Stored Procedure : Suatu transaksi penjualan secara sederhana misalnya melibatkan 2 tabel, antara lain tabel barang yang menyimpanan stok dan jenis barang, dan tabel jual merekam transakasi penjualan. Untuk proses merekaman transaksi penjualan menggunakan 2 stored prosedure, yang pertama perekaman jual, dan kedua pemotongan stok pada tabel barang. Adapun langkah-langkahnya sebagai berikut :

    Buatlah struktur tabel BARANG berikut :

```
CREATE TABLE barang (kd_brg CHAR(5),
                     nm_brg CHAR(20),
                     stok int,
                     satuan CHAR(20),
                     harga int,
                    primary key (kd_brg));
```

Lihat isi tabel :

kd_brg	nm_brg	stok	satuan	harga
K0001	Buku	100	Pcs	5000
K0002	Pesil 2B	200	Pcs	3000
K0003	Pengahpus	100	Pcs	1000
K0004	Kertas HVS	200	Rem	30000
K0005	Gunting kecil	10	Pcs	6000

14. Tambahkan isi rekaman sebagai berikut :

```
INSERT INTO barang VALUES('K0001','Buku ',100,'Pcs',5000);
INSERT INTO barang VALUES('K0002','Pesil 2B',200,'Pcs',3000);
INSERT INTO barang VALUES('K0003','Pengahpus',100,'Pcs',1000);
INSERT INTO barang VALUES('K0004','Kertas HVS',200,'Rem',30000);
INSERT INTO barang VALUES('K0005','Gunting kecil',10,'Pcs',6000);
```

15. Buatlah struktur tabel JUAL sepeti pada perintah berikut:

```
CREATE TABLE jual (no_nota char(4),
 tgl date,
 kd_brg CHAR(5),
 jumlah int);
```

16. DELIMITER $$
17. Kemudian buatlah Stored Procedure Simpan_jual(……), isikan parameternya seperti pada skript berikut :
18. DROP PROCEDURE IF EXISTS `akademik`.`Simpan_jual`$$
CREATE PROCEDURE `akademik`.`Simpan_jual`(in_no_nota char(4),
in_tgl date,
in_kd_brg CHAR(5),
in_jumlah int)
BEGIN
INSERT INTO jual VALUES(in_no_nota,in_tgl,in_kd_brg,in_jumlah);
CALL potong_stok(in_kd_brg,in_jumlah);
END$$
DELIMITER ;
19. Lakukan eksekusi
20. Kemudian buatlah Stored Procedure Potong_stok(……), isikan parameternya seperti pada skript berikut :
DELIMITER $$
DROP PROCEDURE IF EXISTS `akademik`.`potong_stok`$$
CREATE PROCEDURE `akademik`.`potong_stok`(in_kd_brg char(5),in_jumlah int)
BEGIN
UPDATE barang SET stok=stok-in_jumlah WHERE kd_brg=in_kd_brg;
END$$
DELIMITER ;
21. Lakukan eksekusi
22. Jalankan Stored Procedure Simpan_jua
23. CALL Simpan_jual('0001',CURRENT_DATE,'K0001',2)
24. Hasil select table jual dan barang

no_nota	tgl	kd_brg	jumlah
0001	2009-10-13	K0001	2

Lihat Stok

```
SELECT * FROM bar
```

kd_brg	nm_brg	stok	satuan	harga
K0001	Buku	98	Pcs	5000
K0002	Pesil 2B	200	Pcs	3000
K0003	Pengahpus	100	Pcs	1000
K0004	Kertas HVS	200	Rem	30000
K0005	Gunting kecil	10	Pcs	6000

3. TUGAS

1. Buat Kesimpulan dari praktikum diatas mengenai arti transaksi mulai autocommit sampai commit, rollback serta mengenai stored procedure
2. Buat procedure untuk perhitungan dari (a*b)+c dengan nilai variable a=9, b=90, c=78
3. Buat stored procedure untuk konversi nilai angka ke nilai huruf dengan kondisi sebagai berikut :
   0-25 : D
   26-50 : C
   51 – 75 : B
   76 – 100 : A
   Nilai angka ditentukan dari sebuah variable statis didalam stored procedure. Gunakan percabangan dalam menentukan konversi nilai.

MODUL VI
TRIGGER

Tujuan :

- Praktikan dapat memahami dan membuat procedure dan function dalam database
- Praktikan dapat memahami dan membuat trigger dalam database

Tugas Pendahuluan

- Apa yang anda ketahui tentang procedure?
- Apa yang anda ketahui tentang function?
- Apa yang anda ketahui tentang trigger?
- Bagaimana membuat trigger?

Dasar Teori

Trigger atau pemicu merupakan store procedures jenis khusus yang menempel pada suatu tabel tertentu dan dieksekusi secara otomatis saat terjadi manipulasi data untuk tabel tersebut. Sebuah trigger dapat ditempelkan pada operasi manipulasi insert, update dan delete. Dalam kata lain, jika kita menempatkan trigger di sebuah tabel, maka setiap kali kita melakukan operasi yang sudah didefinisikan triggernya, maka data yang akan dimanipulasi akan melalui proses yang ada dalam trigger tersebut terlebih dahulu.

Sebuah trigger akan dibutuhkan dalam suatu database yang tabel-tabel didalamnya membutuhkan berbagai macam aturan bisnis yang lebih restricted dan dinamis. Sehingga tidak semua orang (bahkan programmer sekalipun) yang bisa memanipulasi data secara serampangan tanpa mengetahui jalan cerita dari trigger yang ada dalam database tersebut.

Dalam pembuatan trigger harus diperhitungkan berbagai hal diantaranya adalah :

- Pembuatan trigger tidak boleh bertentangan dengan constraint, primary key dan foreign key yang ada dalam sebuah tabel.
- Pembuatan trigger tidak boleh bertentangan dengan referential integrity dan relasi antar tabel.
- Pembuatan trigger juga diperhitungkan dari sisi kompleksitas agar tidak memperlambat proses manipulasi data, khususnya jika dalam sebuah tabel yang diperkirakan akan diakses oleh banyak user dalam satu waktu.
- Jika dalam suatu trigger terdapat kesalahan yang menyebabkan eksekusi suatu manipulasi data terhenti atau dibatalkan, dianjurkan untuk memberikan pesan kesalahan kepada pengguna agar pengguna mengetahui bahwa proses yang dilakukan telah dibatalkan.

Praktikum

Berikut adalah dasar-dasar trigger, function, procedure serta view yang bisa diimplementasikan pada MYSQL 5 keatas. Dengan menggunakan trigger, function, procedure serta view, pemanfaatan MYSQL akan lebih bermanfaat. Contohnya : kita bisa membuat urutan langkah-langkah tertentu setelah suatu even dilakukan (ex: insert, update, delete).

Coba semua contoh operasi di bawah ini :

1. **Procedure dan Function**

1.1 **Procedure**

Syntak untuk membuat procedure dalam MySQL adalah sebagai berikut.

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
```

PROCEDURE *sp_name* ([*proc_parameter*[,...]])
[*characteristic* ...] *routine_body*

1.2 **Function**

Syntaks untuk membuat function dalam MySQL adalah sebagai berikut.

CREATE
[DEFINER = { *user* | CURRENT_USER }]
FUNCTION *sp_name* ([*func_parameter*[,...]])
RETURNS *type*
[*characteristic* ...] *routine_body*

Keterangan syntaks :

1. proc_parameter (hanya pada procedure):
   [IN | OUT | INOUT] *param_name type*

- IN parameter dilewatkan ke dalam procedure tetapi modifikasi nilai dari parameter ini tidak kelihatan setelah procedure tersebut dipanggil.
- Out parameter merupakan parameter yang dilewatkan dari dalam procedure ke pemanggil procedure tersebut. Nilai inisialisasi saat memanggil adalah null dan hasil parameter akan kelihatan setelah procedure dipanggil.
- INOUT parameter diinisilaisai oleh pemanggil procedure, kemudian hasil modifikasinya tersebut akan kelihatan setelah procedure tersebut dipanggil.

2. characteristic
   LANGUAGE SQL
   | [NOT] DETERMINISTIC
   | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
   | SQL SECURITY { DEFINER | INVOKER }
   | COMMENT '*string*'
3. routine_body:

Valid SQL procedure statement

Routine body berisi syntax-syntax sql yang valid, seperti insert atau select. Bisa juga berisi compound. Compount diapit oleh begin… end. Syntax dari compound system adalah sebagai berikut.

[*begin_label*:] BEGIN
[*statement_list*]
END [*end_label*]

Compound ini bisa diisi dengan label. Begin label dan end label harus sama. Di dalam compound juga bisa berisi deklarai variabel, looping atau kontrol program lainnya. Setiap Function harus punya return value untuk mengembalikan nilai yang dihasilkan

2. **Contoh Procedure dan Function MySQL**

Berikut adalah contoh penggunaan procedure. Pertama kita buat terlebih dahulu database dengan nama adakara. Kemudian silakan buat table berikut dengan isinya. Berikut ini adalah hasil dumping dari database adakara.

```
CREATE TABLE `siswa` (
`NIS` varchar(20) collate latin1_general_ci NOT NULL,
  `Nama` varchar(30) collate latin1_general_ci NOT NULL,
  `Angkatan` varchar(9) collate latin1_general_ci NOT NULL,
  PRIMARY KEY  (`NIS`)
)

INSERT INTO `siswa` VALUES ('1024','resika arthana','2005/2006'),
('1','cxfbsdjg','2005/2006'),('2','dshfk','2005/2006'),('3','dsfku','2005/2006'),
('11','Ari damayanti','2005/2006'),('12','wedana','2005/2006'),
('122','andika','2005/2006'),('123','mas ayu','2005/2006');
```

2.2 Procedure

Berikut ini adalah proses pembuatan procedure, dibuat dalam command from.

```
mysql> delimiter //
mysql> CREATE PROCEDURE jumlahSiswa (OUT param1 INT)
    -> BEGIN
    ->   SELECT COUNT(*) INTO param1 FROM siswa;
    -> END;
    -> //
Query OK, 0 rows affected (0.00 sec)
mysql> delimiter ;
```

2.2 Function

Function adalah sebuah prosedur yang bisa kita definisikan dengan perintah CREATE FUNCTION. Bahasa yang digunakan untuk mendefinisikan function dapat ditentukan pada parameter LANGUAGE. Dua buah function dapat

memiliki nama yang sama tapi dengan parameter yang berbeda baik tipe data maupun jumlahnya. Beberapa contoh function (*built-in*) dalam Posgres, dapat dilihat dengan mengetikkan perintah \df pada console posgres.
Contoh:

SELECT UPPER (‘otomatis menjadi capital’);

Syntax Create Function:

CREATE [OR REPLACE] FUNCTION *name* ([*argtype* [, ...]]) RETURNS *rettype* AS '*definition*' LANGUAGE *langname* [WITH (*attribute* [, ...])]

Cobalah contoh sebuah function sederhana berikut ini:

CREATE FUNCTION tambah (int, int) RETURNS int AS ‘ SELECT $1 + $2’ LANGUAGE ‘sql’;

SELECT tambah (10,12);

Untuk menghapus sebuah function, gunakan perintah DROP.

Berikut ini adalah contoh proses pembuatan function di my sql:

mysql> create function tmbhSaudara(nama char(20)) returns char(5)
-> return concat('Saudara ',nama);
Query OK, 0 rows affected (0.36 sec)

3. Pemanggilan Procedure dan Function

Pemanggilan procedure menggunakan syntax

CALL *procedure_atau_function_name*([*parameter*[,...]])

Keterangan :

- *procedure_atau_function_name* adalah nama procedure ata fungsi yang dipanggil dan parameter adalah nama-nama parameter procedure atau fungsi tersebut

3.1 Procedure

Contoh procedure :

```
mysql> call jumlahSiswa(@a);
Query OK, 0 rows affected (0.00 sec)
mysql> select @a;
+------+
| @a   |
+------+
|  8   |
```

```
+------+
1 row in set (0.00 sec)
```

3.2 Function

Contoh function :

```
mysql> select tmbhSaudara(' resika arthana');
+-----------------------------------+
| tmbhSaudara(' resika arthana') |
+-----------------------------------+
| Saudara resika arthana |
+-----------------------------------+
1 row in set (0.00 sec)
```

4. Menghapus Procedure dan Function

Untuk menghapus procedure atau function digunakan perintah sebagai berikut

```
DROP {PROCEDURE | FUNCTION} [IF EXISTS]
procedure_or_function_name
```

5. Trigers

Trigger berfungsi untuk menyisipkan suatu fungsi pada saat suatu record di-INSERT, UPDATE, atau DELETE.

Syntax:

```
CREATE TRIGGER name { BEFORE | AFTER } { event [OR ...] } ON table
FOR EACH { ROW | STATEMENT } EXECUTE PROCEDURE func (
arguments )
```

Atau syntax Umum Triger

```
CREATE
  [DEFINER = { user | CURRENT_USER }]
  TRIGGER trigger_name trigger_time trigger_event
  ON tbl_name FOR EACH ROW trigger_stmt
```

Keterangan :

- DEFINER menunjukkan nama user yang mempunyai hak akses untuk mengakses triger.
- Triger_time menunjukkan saat triger tersebut dijalankan. Terdiri dari Before atau After
- Triger_time menandakan saat keadaan bagaimana triger tersebut aktif. Terdiri dari
  - Insert : Triger aktif saat baris baru dimasukkan ke dalam tabel

 Update : Triger aktif saat ada baris diperbaharui dimasukkan di dalam tabel

 Delete :Triger aktif saat baris dihapus dalam tabel

Contoh Trigers :

```
CREATE TABLE test1(a1 INT);
CREATE TABLE test2(a2 INT);
CREATE TABLE test3(a3 INT NOT NULL AUTO_INCREMENT
PRIMARY KEY);
CREATE TABLE test4(
  a4 INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  b4 INT DEFAULT 0
);
DELIMITER |
CREATE TRIGGER testref BEFORE INSERT ON test1
  FOR EACH ROW BEGIN
    INSERT INTO test2 SET a2 = NEW.a1;
    DELETE FROM test3 WHERE a3 = NEW.a1;
    UPDATE test4 SET b4 = b4 + 1 WHERE a4 = NEW.a1;
  END;
|   |
DELIMITER ;
INSERT INTO test3 (a3) VALUES
  (NULL), (NULL), (NULL), (NULL), (NULL),
  (NULL), (NULL), (NULL), (NULL), (NULL);
INSERT INTO test4 (a4) VALUES
  (0), (0), (0), (0), (0), (0), (0), (0), (0), (0);
Silakan masukkan nilai ke test1
mysql> INSERT INTO test1 VALUES
    -> (1), (3), (1), (7), (1), (8), (4), (4);
Query OK, 8 rows affected (0.01 sec)
Records: 8  Duplicates: 0  Warnings: 0
As a result, the data in the four tables will be as follows:
mysql> SELECT * FROM test1;
+------+
|  a1  |
+------+
|  1   |
|  3   |
|  1   |
```

```
|   7   |
|   1   |
|   8   |
|   4   |
|   4   |
+------+
8 rows in set (0.00 sec)
mysql> SELECT * FROM test2;
+------+
|   a2  |
+------+
|   1   |
|   3   |
|   1   |
|   7   |
|   1   |
|   8   |
|   4   |
|   4   |
+------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM test3;
+----+
| a3 |
+----+
|  2 |
|  5 |
|  6 |
|  9 |
| 10 |
+----+
5 rows in set (0.00 sec)

mysql> SELECT * FROM test4;
+----+------+
| a4 | b4   |
+----+------+
|  1 |    3 |
|  2 |    0 |
```

```
|  3  |   1  |
|  4  |   2  |
|  5  |   0  |
|  6  |   0  |
|  7  |   1  |
|  8  |   1  |
|  9  |   0  |
| 10  |   0  |
+-----+------+
10 rows in set (0.00 sec)
```

Tugas :

1. Buatlah fungsi baru yang berfungsi untuk mengubah semua data nama yang akan di-INSERT ke tabel pegawai (buat juga tabel pegawai dengan dua kolom saja, yaitu id, dan nama) menjadi capital semua.
2. Buatlah trigger untuk memanggil fungsi yang telah dibuat di atas.
3. INSERT-kan ke tabel pegawai data berikut: id=1012 dan nama=owo.
4. SELECT dari tabel pegawai.

MODUL VII
EVALUASI BASIS DATA

Tujuan :

- Peserta memhami tentang normalisasi, stored procedure, trigger, transaksi

Tugas Evaluasi :

1. Tentukan Topik (setiap mahasiswa berbeda topic)
2. Buat desain table Minimal 5 tabel saling terelasi
3. Buat normalisasi dalam bentuk CDM dan PDM
4. Buat Transaksi : penggunaan autocommit, start transaction, operasi dml, commit, rollback
5. Buat Procedure dari table anda
6. Buat Trigger dari table anda

Atau Tugas proyek bisa di tentukan dosen Pengampu.

MODUL VIII
FUNCTION DAN TRIGGER

Tujuan :
Mampu membuat function dalam pengelolahan database dan memahami macam-macam tipe trigger serta membuat database trigger.

PrePraktikum

1. Apakah yang anda ketahui tentang function ?
2. Apakah yang anda ketahui tentang trigger?
3. Sebutkan macam-macam tipe trigger?
4. Trigger dapat dibuat sesuai dengan keperluan. Kapan trigger perlu dibuat dan kapan trigger tidak perlu dibuat?
5. Sebutkan tiga macam trigger timing dan trigger event?

Dasar Teori

1 **Function**

Function adalah suatu blok PL/SQL yang memiliki konsep sama dengan procedure, hanya saja pada function terdapat pengembalian nilai (return value). Karena function dapat mengembalikan sebuah nilai, function dapat diakses seperti layaknya sebuah variabel biasa.

Pembuatan function sangat berguna dalam proses pengelolahan database. Dengan sekali membuat function, anda dapat menggunakannya untuk keperluan yang sama pada blok-blok PL/SQL lainnya, procedure atau bahkan dalam function lainnya. Hal ini sama dengan konsep pembuatan procedure yang tentunya dapat meringankan pekerjaan karena tidak perlu mengulang statemen-statemen yang sama pada setiap blok PL/SQL yang dibuat. Sintax umum untuk membuat sebuah function adalah sebagai berikut:

```
CREATE FUNCTION nama_function ([func_parameter[,...]])
  RETURNS type_data
  [characteristic ...] routine_body
```

2 Trigger

Trigger adalah prosedur tersimpan pada Microsoft SQL Server yang secara otomatis dijalankan apabila data di dalam tabel berubah karena eksekusi perintah SQL (INSERT, UPDATE atau DELETE). Salah satu penggunaannya yang paling umum adalah untuk menerapkan pembatasan yang lebih kompleks dari yang telah diizinkan melalui pembatasan CHECK, yang berfungsi membatasi informasi yang disisipkan ke dalam kolom.

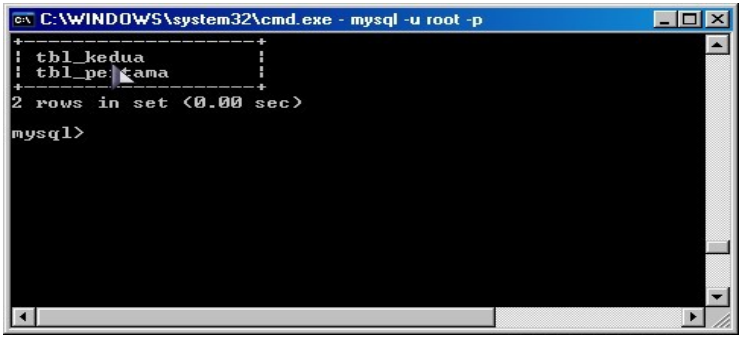
Trigger bisa dibuat bersama dengan perintah INSERT, yang akan melakukan query ke tabel lain dan mengembalikan nilai logik yang membantu membatasi data yang diberikan kepada kolom tertentu. Sebagai contoh: Trigger bisa dibuat untuk menjalankan replikasi, misalnya apabila ada sebuah baris disisipkan ke dalam database Z, sebuah baris dengan informasi yang sama akan ditambahkan ke dalam database Y. Atau Apabila sebuah baris dihapus dari sebuah tabel, Trigger akan menghapus baris lain yang berhubungan dengan baris tersebut pada tabel lain. Trigger dibuat sebagai sebuah transaksi dan bisa dimundurkan apabila ada masalah yang dideteksi. Syntax trigger:

```
CREATE
        [DEFINER = { user | CURRENT_USER }]
        TRIGGER trigger_name trigger_time trigger_event
ON tbl_name FOR EACH ROW trigger_stmt
```

4. **Kegiatan Praktikum**

1. Setting INODB pada MySQL kita jadi terlebih dahulu silakan aktifkan INODB pada MySQL, Setelah aktif barulah kita bisa menggunkan fasilitas trigger pada MySQL. Ok langsung pada tujuan mari kita mulai proyek kita, pertama – tama kita akan membuat 2 buah tabel untuk percobaan kita :

```
1  CREATE TABLE `tbl_pertama` (
2  `kode` char(2) NOT NULL,
3  `nama` varchar(20) default NULL,
4  PRIMARY KEY (`kode`)
5  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
6
7  CREATE TABLE `tbl_kedua` (
8  `kode_kedua` char(2) NOT NULL,
9  `nama_kedua` varchar(20) default NULL,
10 PRIMARY KEY (`kode_kedua`)
11 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

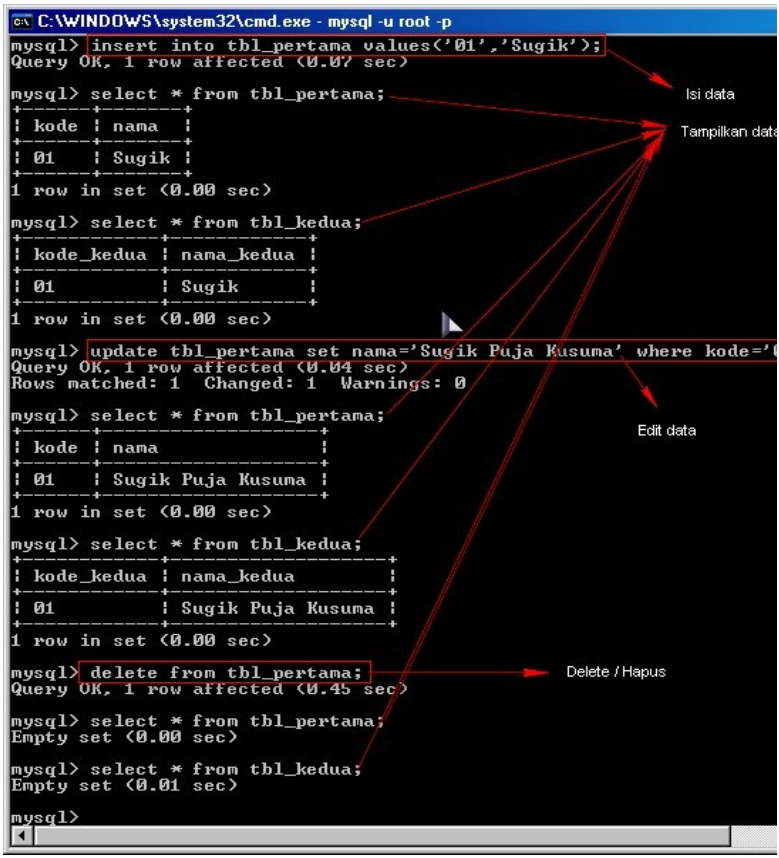


2. Trigger dapat bekerja BEFORE atau AFTER sebuah tabel mengalami tiga kejadian [Insert, Edit, delete].:

```
1  CREATE TRIGGER `tr_insert` AFTER INSERT ON `tbl_pertama`
2  FOR EACH ROW
3  BEGIN
4  INSERT INTO tbl_kedua values(new.kode, new.nama);
5  END;
6
```

```
1  CREATE TRIGGER `tr_update` AFTER UPDATE ON `tbl_pertama`
2  FOR EACH ROW
3  BEGIN
4  update tbl_kedua set kode_kedua=new.kode, nama_kedua=new.nama
5  where kode_kedua=OLD.kode;
6  END;
```

```
1  CREATE TRIGGER `tr_delete` AFTER DELETE ON `tbl_pertama`
2  FOR EACH ROW
3  BEGIN
4  delete from tbl_kedua where kode_kedua=OLD.kode;
5  END;
```



3. Amatilah proses dari kegiatan 1 dan , kemudian apa yang terjadi?
4. Tugas tambahan (ditentukan oleh asisten praktikum)!

5. **Tugas Akhir**

1. Buatlah function dengan nama fungsi test.hello dengan tipe data char dengan kapasitas 20 dan menggabungkan hello, s dan !.
2. Buat terlebih dahulu tabel barang, tabel detail pembelian sesuai dengan struktur yang telah ditetapkan serta trigger untuk kasus penjualan. Tabel identik dengan table penjualan
3. Buatlah sebuah fungsi yang digunakan untuk mendapatkan volume sebuah balok. Ujilah fungsi yang anda buat!

4. Buatlah trigger untuk kasus faktur penjualan, jika kuantitas (jumlah items) yang dimasukkan lebih dari 5 items maka diskonnya sebesar 10 persen, dan jika kurang dari 5 maka akan disikan dengan nilai 0.
5. Buatlah beberapa trigger untuk event pada tabel detailpembelian baik untuk delete, insert maupun update.

MODUL IX
CURSOR I

Tujuan :
Praktikan dapat memanggil sintaks SQL dengan menggunakan cursor

Tugas Pendahuluan

1. Apa yang anda ketahui tentang cursor? Jelaskan!

Dasar Teori

Cursor merupakan pointer yang menunjukan ke suatu bagian memori untuk menyimpan hasil perintah SQL seperti Select, Insert, Update dan Delete. Pemakaian cursor pada SQL Server mendukung struktur pengulangan (loop) diantara result set, membaca setiap baris data satu per satu.

Praktikum :

Beberapa tahapan yang harus dilakukan dalam memakai cursor anatara lain :

1. Deklarasi Cursor
   Sebelum menggunakan cursor pertama-tama yang harus dilakukan adalah mendeklarasikan nama cursor dan variabel yang digunakan.

   DECLARE *cursor_name* CURSOR FOR *select_statement*

2. Open Cursor
   Untuk menggunakan cursor dan fetch data, anda harus mengaktifkan cursor sesuai dengan nama yang telah dideklarasikan sebelumnya.

   ```
   OPEN cursor_name
   ```

3. Membaca baris data dari cursor
   Setelah cursor diaktifkan, SQL Server akan membaca baris data secara berulang-ulang(loop) dari baris data dari suatu tabel.

4. Menutup Cursor
   Menutup cursor dilakukan jika ingin mengunci data setelah tidak digunakan

   CLOSE *cursor_name*

5. Dealokasi cursor
   Dealokasi cursor bertujuan untuk membuang atau menghapus cursor dari memori jika sudah tidak digunakan lagi

Syntax

DECLARE *nama_cursor* [INSENSITIVE][SCROOL] CURSOR FOR *select_statement*
Declare
[FOR {READ ONLY|UPDATE [OF *nama_kolom* [,....n]]}]
OPEN *nama_cursor*
CLOSE *nama_cursor*
DEALLOCATE *nama_cursor*

Contoh :

```
DECLARE Cursor_Barang CURSOR
FOR
  Select Kode, Nama From BARANG
Declare
  @Kode Char(5),
  @Nama Varchar(30)
OPEN Cursor_Barang
  FETCH NEXT From Cursor_Barang
  Into @Kode, @Nama
CLOSE Cursor-Barang
DEALLOCATE Cursor_Barang
```

Tugas :

Buat implementasi dari:
Copy salah satu tabel dari database yang anda buat (hasil modul View) ke sebuah tabel baru

MODUL X
CURSOR II

Tujuan :
Praktikan dapat memanggil sintaks SQL dengan menggunakan cursor

Tugas Pendahuluan

1. Apa yang anda ketahui tentang fungsi Count (Max, Min, Avarage) dengan cursor. Beri contoh scriptnya.

Dasar Teori

Suatu query yang dibuat untuk menyamakan beberapa record tidak harus mempunyai clause into. Jika suatu query diharapkan mengembalikan beberapa record, maka cursor harus dipakai sebagai penggantinya. Cursor dapat digunakan untuk menegaskan satu record pada suatu waktu. Dengan cursor, tiap-tiap record dapat dikembalikan oleh query oleh individual pada suatu waktu.

Berikut contoh perintah select untuk menemukan record dalam basis data.

```
Select * from state
```

Perintah ini akan gagal dijalankan meski program telah benar karena query akan memilih semua nama dalam tabel. Query akan menemukan 50 tuple dalam state basis data.

```
database store
main
select * from state
end main
```

Program ini akan dikompile dan dijalankan. Program tersebut akan berhenti dan menampilkan pesan.

a subquery has return not exactly one value

Hal ini disebabkan perintah select menemukan lebih dari satu record.

Masalah ini dapat diselesaikan dengan cursor, karena cursor dideklarasikan untuk mengoperasikan perintah select. Perintah select digabungkan dengan cursor. Perintah select dapat menyamakan suatu kelompok record dalam basis data. Cursor kemudian dapat digunakan untuk menegaskan masing-masing pilihan record.
Berikut contoh penggabungan perintah select dan cursor.

```
declare c_state cursor for
    select * from state
```

Perintah tersebut membuat cursor dengan nama c_state. Cursor dapat memilih masing-masing record yang dikembalikan oleh perintah select * from state.

Tugas :

1. Tambahkan data ke dalam relasi dengan perintah insert
2. Rubahlah salah satu data pada record data base anda
3. Implementasikan tugas pendahuluan yang sudah anda kerjakan.

MODUL XI
PENGEMBANGAN APLIKASI BASIS DATA

Tujuan :
Memahami langkah-langkah koneksi PHP dengan MySQL.
Memahami perbedaan pengambilan record dari database.

Tugas Pendahuluan
Jelaskan mengenai PHP !
Apa bedanya web server dan web browser?
Sebutkan langkah mengkoneksikan antara php dan database mysql?

TEORI

1.1. PHP

PHP adalah salah satu bahasa pemrograman internet yang mendukung penggunaan database seperti MySQL, PostgreSQL, mSQL, Oracle, dan lain-lain. Untuk dapat menjalankan PHP melalui browser, maka anda diharuskan terlebih dahulu menginstall web server (misalnya Apache, PWS, IIS) lalu menginstall PHP, sedangkan untuk menjalankan MySQL anda tidak perlu menginstall web server, hanya saja jika ingin dijalankan melalui browser, maka anda harus menginstall web server. PHP sebagai alternatif lain memberikan soulsi sangat murah (karena gratis digunakan) dan dapat berjalan diberbagai jenis platform.

PHP (atau resminya PHP : Hypertext Preprosessor) adalah skrip bersifat server-side yang ditambahkan kedalam HTML. PHP sendiri merupakan singkatan dari *Personal Home Page Tools*. Skrip ini akan membuat suatu aplikasi dapat diintegrasikan kedalam HTML sehingga suatu halaman web tidak lagi bersifat statis, namun menjadi bersifat dinamis. Sifat server side berarti pengerjaan skrip dilakukan diserver, baru kemudian hasilnya dikirmkan ke browser.
Cara Penulisan skrip PHP ada dua macam, yaitu *Embedded Script* dan *Non Embedded Script*. Contoh dari jenis skrip PHP ditunjukkan dalam gambar berikut :

```
<html>
<body>
<? php echo "Belajar";
?>
</body>
<html>
```

(a)

```
<?php
    echo "<html>";
    echo "<body>";
    echo "Belajar PHP";
    echo "</body>";
    echo "</html>";
?>
```

(b)

Gambar Skrip dalam PHP: (a) Embedded Script, (b) Non Embedded Script

Gambar diatas menjelaskan bahwa skrip PHP dapat berupa *embedded script* yaitu meletakkan tag PHP diantara tag-tag HTML sedangkan *non embedded script* yaitu semua tag HTML diletakkan dalam tag PHP. Semua Kode PHP menyerupai dengan kode bahasa C, walaupun tidak sepenuhnya sama.
Untuk penulisan tag PHP terdiri dari empat style, yaitu *Style Standart Format*, dengan format: <?php …. ?>, *Short Style* dengan format : <? …. ?>, *Javascript Style* dengan format: <script language="php"> …. </script> dan *ASP style* dengan format : <% …. %>.

1.2. Langkah-langkah koneksi PHP-MySQL

1. Membuka koneksi ke server MySQL
mysql_connect()
Digunakan untuk melakukan uji dan koneksi kepada server database MySQL.

Sintaks :

```
$conn = mysql_connect ("host","username","password");
```

$conn	adalah nama variabel penampung status hasil koneksi kepada database.
host	adalah nama host atau alamat server database MySQL.
username	adalah nama user yang telah diberi hak untuk dapat mengakses server database.
password	adalah kata sandi untuk username untuk dapat masuk ke dalam database.

2. Memilih database yang akan digunakan di server
mysql_select_db()
Digunakan untuk melakukan koneksi kepada database yang dalam server yang berhasil dikoneksi dengan perintah mysql_connect().

Sintaks :

```
$pilih = mysql_select_db(”namadatabase”,$conn);
```

$pilih berisi status koneksi kepada database.
$conn merupakan koneksi kepada server database yang berhasil.
namadatabase adalah nama database yang akan dikenai proses.

3. Mengambil sebuah query dari sebuah database.
mysql_query()
Digunakan untuk melakukan eksekusi perintah SQL untuk memanipulasi database yang berhasil dilakukan koneksinya menggunakan mysql_select_db().

Sintaks :

```
$hasil = mysql_query(”SQL Statement”);
```

$hasil akan berupa record set apabila SQL Statement berupa perintah select.

4. Mengambil record dari database
a. mysql_fetch_array()
Digunakan untuk melakukan pemrosesan hasil query yang dilakukan dengan perintah mysql_query(), dan memasukkannya ke dalam array asosiatif, array numeris atau keduanya.

Sintaks :

```
$row = mysql_fetch_array($hasil);
```

$row adalah array satu record dari record $hasil yang diproses nomor record sesuai dengan nomor urut dari proses mysql_fetch_array yang sedang dilakukan.
$hasil adalah record set yang akan diproses.

b. mysql_fetch_assoc()

Fungsi ini hampir sama dengan fungsi mysql_fetch_array(), hanya saja array yang dihasilkan hanya array asosiatif.

Sintaks :

```
$row = mysql_fetch_assoc($hasil);
```

c. mysql_fetch_row()
Fungsi ini hampir sama dengan fungsi mysql_fetch_array(), hanya saja array yang dihasilkan hanya array numeris.
Sintaks :

```
$row = mysql_fetch_row($hasil);
```

d. mysql_num_rows()
Fungsi ini digunakan untuk menghitung jumlah record yang ada pada database.

Sintaks :

```
$jml = mysql_num_rows($hasil);
```

$jml akan memiliki nilai sesuai dengan jumlah record yang ada.

2. Kegiatan Praktikum :

1. Menguji interkoneksi PHP dengan MySQL.

```
<html>
<head>
  <title>Koneksi Database MySQL</title>
</head>
<body>
<h1>Demo koneksi database MySQL</h1>
<?
$conn=mysql_connect ("localhost","root","");
if ($conn) {
   echo "OK";
} else {
  echo "Server not connected";
}
?>
</body>
</html>
```

2. Melihat perbedaan antara mysql_fetch_array(), mysql_fetch_assoc(), mysql_fetch_row().

a. Buatlah tabel liga berikut ini, dengan 3 field : kode, negara, champion.

```
Create table liga (
        kode char(3) not null,
        negara char(15),
        champion int
);
```

b. Isilah tabel dengan data berikut ini :

```
Insert into liga (kode, negara, champion)
values ('jer','Jerman',4);
Insert into liga (kode, negara, champion)
values ('spa','Spanyol',4);
Insert into liga (kode, negara, champion)
values ('ing','Inggris',3);
Insert into liga (kode, negara, champion)
values ('bel','Belanda',3);
```

c. Akses databases menggunakan mysql_fetch_array()

```
<HTML>
<HEAD>
  <title>Koneksi Database MySQL</title>
</HEAD>
<BODY>
<h1>Koneksi database dengan mysql_fetch_array</h1>
<?
$conn=mysql_connect ("localhost","root","")
     or die ("koneksi gagal");
mysql_select_db("faruq",$conn);
$hasil = mysql_query("select * from liga",$conn);
while ($row=mysql_fetch_array($hasil)) {
   echo "Liga " .$row["negara"]; //array asosiatif
   echo " mempunyai " .$row[2];  //array numeris
   echo " wakil di liga champion <br>";
}
?>
</BODY>
</HTML>
```

d. Akses databases menggunakan mysql_fetch_assoc()

```
<HTML>
<HEAD>
  <title>Koneksi Database MySQL</title>
</HEAD>
<BODY>
<h1>Koneksi database dengan mysql_fetch_assoc</h1>
<?
$conn=mysql_connect ("localhost","root","")
     or die ("koneksi gagal");
mysql_select_db("faruq",$conn);
$hasil = mysql_query("select * from liga",$conn);
while ($row=mysql_fetch_array($hasil)) {
   echo "Liga " .$row["negara"];
   echo " mempunyai " .$row[“champion”];
   echo " wakil di liga champion <br>";
}
?>
</BODY>
</HTML>
```

e. Akses databases menggunakan mysql_fetch_row()

```
<HTML>
<HEAD>
  <title>Koneksi Database MySQL</title>
</HEAD>
<BODY>
<h1>Koneksi database dengan mysql_fetch_assoc</h1>
<?
$conn=mysql_connect ("localhost","root","")
     or die ("koneksi gagal");
mysql_select_db("faruq",$conn);
$hasil = mysql_query("select * from liga",$conn);
while ($row=mysql_fetch_row($hasil)) {
   echo "Liga " .$row[1];
   echo " mempunyai " .$row[2];
   echo " wakil di liga champion <br>";
}
?>
</BODY>
</HTML>
```

a. Buatlah tabel bukutamu yang memiliki 3 field : nama, email, komentar.

```
Create table bukutamu (
        nama char(20) not null,
        email char(20),
        komentar char (40)
);
```

b. Buat form untuk buku tamu, beri nama bukutamu.htm

```
<HTML>
<HEAD>
  <title>Buku Tamu</title>
</HEAD>
<BODY>
<h1>Buku Tamu untuk database MySQL</h1>
<form action="bukutamu_add_form.php" method="post">
Nama     : <input type="text" name="nama" size="35" maxlength="50"> <br>
Email    : <input type="text" name="email" size="35" maxlength="50"> <br>
Komentar : <textarea name="komentar" rows="5" cols="30"></textarea> <br>
<input type="submit" value="Simpan">
<input type="reset" value="Reset">
</form>
</BODY>
</HTML>
```

c. Buat file bukutamu_add_form.php untuk memproses data dari bukutamu.htm dan menambahkan data ke tabel bukutamu.

```
<HTML>
<HEAD>
  <title>Simpan Buku Tamu</title>
</HEAD>
<BODY>
<h1>Simpan Buku Tamu MySQL</h1>
<?
$nama = $_POST["nama"];
$email = $_POST["email"];
$komentar = $_POST["komentar"];
$conn=mysql_connect ("localhost","root","")
     or die ("koneksi gagal");
mysql_select_db("faruq",$conn);
echo "Nama     : $nama <br>";
echo "Email    : $email <br>";
echo "Komentar : $komentar <br>";
$sqlstr="insert into bukutamu (nama,email,komentar)
        values ('$nama','$email','$komentar')";
$hasil = mysql_query($sqlstr,$conn);
echo "Simpan bukutamu berhasil dilakukan";
?>
</BODY>
</HTML>
```

d. Buat file view.php untuk menampilkan isi buku tamu.

```
<?
$conn = mysql_connect("localhost","root","");
mysql_select_db("faruq",$conn);
$hasil = mysql_query("select * from bukutamu",$conn);
$jumlah = mysql_num_rows($hasil);
echo "<center>Daftar Pengunjung</center>";
echo "Jumlah pengunjung : $jumlah";
$a=1;
while($baris = mysql_fetch_array($hasil))
{
  echo "<br>";
  echo $a;
  echo "<br>";
  echo "Nama : ";
  echo $baris[0];
  echo "<br>";
  echo "Email : ";
  echo $baris[1];
  echo "<br>";
  echo "Komentar : ";
  echo $baris[2];
  $a++;
}
?>
```

4. Membuat program searching database dengan menggunakan tabel no 3a

a. Buat file search.htm

```
<HTML>
<HEAD>
  <title>Cari Database</title>
</HEAD>
<BODY>
<h1>Searching Buku Tamu untuk database MySQL</h1>
<form action="hasilsearch.php" method="post">
<select name="kolom">
<option value="nama">nama</option>
<option value="email">email</option>
</select>
Masukkan kata yang anda cari
<input type ="text" type ="text" name="cari">
<input type ="submit" value="cari" >
</form>
</BODY>
</HTML>
```

b. Buat file hasilsearch.php untuk menampilkan data

```
<?
 $kolom=$_POST['kolom'];
 $cari=$_POST['cari'];
 $conn=mysql_connect("localhost","root","");
 mysql_select_db("faruq", $conn);
 $hasil=mysql_query("select * from bukutamu where  $kolom like '%$cari%'", $conn);
 $jumlah=mysql_num_rows($hasil);
 echo "<br>";
 echo "Ditemukan: $jumlah";
 echo "<br>";
 while($baris=mysql_fetch_array($hasil))
   {
   echo "Nama : ";
   echo $baris[0];
   echo "<br>";
   echo "Email : ";
   echo $baris[1];
   echo "<br>";
   echo "Komentar :";
   echo $baris[2];
}
?>
```

3. TUGAS :
1. Buat Kesimpulan dari praktikum diatas.

2. Modifikasilah program diatas, sehingga anda memiliki 3 tombol yaitu, Bukutamu untuk inputan data, Tampilan untuk menampilkan database bukutamu dalam bentuk tabel, Search untuk melakukan searching database bukutamu dalam bentuk tabel.

MODUL XII
PROJECT TUGAS AKHIR

Tujuan :

- Peserta dapat mengimplementasikan semua modul dalam suatu proyek untuk mengkoneksikan database dengan program aplikasi

Tugas Proyek :

1. Tentukan Topik Project (setiap mahasiswa berbeda topic)
2. Buat desain table Minimal 5 tabel saling terelasi
3. Buat normalisasi dalam bentuk CDM dan PDM
4. Koneksikan table dengan program aplikasi berbasis WEB
5. Buat desain yang sesuai.

Atau Tugas proyek bisa di tentukan dosen Pengampu.