BUKU AJAR
BASIS DATA

Fitria Nur Hasanah, M.Pd
Rahmania Sri Untari, M.Pd

BUKU AJAR MATA KULIAH
BASIS DATA

Oleh
Fitria Nur Hasanah, M.Pd
Rahmania Sri Untari, M.Pd

Diterbitkan oleh
UMSIDA PRESS
Jl. Mojopahit 666 B Sidoarjo

ISBN : 978-602-5914-89-8

Buku Ajar
BASIS DATA

Penulis :
Fitria Nur Hasanah, M.Pd
Rahmania Sri Untari, M.Pd

ISBN : 978-602-5914-89-8

Editor :
Septi Budi Sartika, M.Pd
M. Tanzil Multazam , S.H., M.Kn.

Copy Editor :
Fika Megawati, S.Pd., M.Pd.

Design Sampul dan Tata Letak :
Mochamad Nashrullah, S.Pd

Penerbit :
UMSIDA Press

Redaksi :
Universitas Muhammadiyah Sidoarjo
Jl. Mojopahit No 666B
Sidoarjo, Jawa TImur

Cetakan pertama, September 2019

KATA PENGANTAR

Alhamdulillah Puji Syukur Penulis sampaikan kehadirat Allah SWT, yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Buku Ajar Basis Data dengan baik. Buku ajar ini ditujukan kepada mahasiswa yang mengampuh mata kuliah Basis data. Dengan dibuatnya Buku Ajar ini penulis berharap agar dapat bermanfaat dan membantu dalam memahami materi Basis Data yang semakin berkembang.

Ucapkan terima kasih penulis ucapkan kepada :

1. Dr. Hidayatulloh, M.Si., Rektor UMSIDA yang memberikan kesempatan luas kepada tim penulis untuk berkarya dan menyumbangkan pikiran sehingga buku ajar ini terselesaikan.
2. Dr. Akhtim Wahyuni, M.PdI Dekan Fakultas Psikologi dan Ilmu Pendidikan yang memberikan arahan dan motivasi kepada penulis dalam menyelesaikan buku ajar Basis Data ini.
3. Rekan-rekan dosen di lingkungan FPIP UMSIDA yang telah berbagi pengalaman dalam penulisan buku ajar.

Penulis menyadari sekali bahwa Buku Ajar ini masih jauh dari kesempurnaan, maka dari itu penulis mengharapkan kritik dan saran pembaca demi kesempurnaan Buku Ajar ini kedepannya. Akhir kata penulis mengucapkan terima kasih, mudah-mudahan bermanfaat bagi para pembaca.

Sidoarjo, 2019
Penulis

DAFTAR ISI

BAB I
PENDAHULUAN

1. Deskripsi

Data merupakan hal yang penting bagi kehidupan manusia, pernyataan ini tidak dapat dipungkiri karena setiap hari manusia memrlukan dan menggunakan data dalam merancang segala sesuatu. Contoh sederhana mahasiswa ketika akan mengerjakan tugas tentu membutuhkan data untuk menyelesaikan tugasnya. Mahasiswa memerlukan data antara lain data tugas, data mata kuliah, data dosen, nilai ujian, dan lain-lain. Sementara itu dapatkan membayangkan kebutuhan mahasiswa atau organisasi lain terhadap data yang dibutuhkan? Saat ini baik instansi pendidikan maupun perusahaan sudah memiliki system informasi dan aplikasi. System informasi dan aplikasi yang terdapat pada instansi memerlukan data. Sehingga antara system informasi dan data memiliki hubungan yang erat dan tidak dapat dipisahkan.

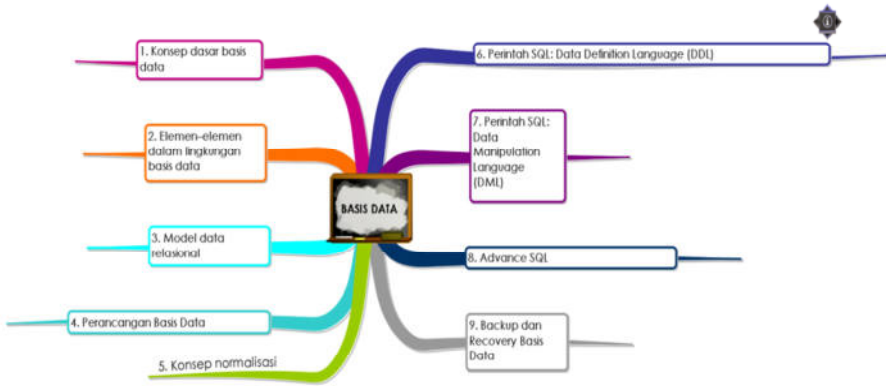
Basis data adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh suatu informasi dari database tersebut. Perangkat lunak yang digunakan untuk mengolah dan mengambil query basis data disebut sistem manajement basis data. Pemrosesan basis data sebagai perangkat andalan sangat diperlukan oleh berbagai institusi dan perusahaan. Dalam pengembangan sistem informasi diperlukan basis data sebagai media penyimpanan data. Kehadiran basis data dapat meningkatkan daya saing perusahaan tersebut.

Basis data dapat mempercepat upaya pelayanan kepada pelanggan, menghasilkan informasi dengan cepat dan tepat sehingga membantu pengambilan keputusan untuk segera memutuskan suatu masalah berdasaran informasi yang ada. Banyak

aplikasi yang dibuat dengan berlandaskan pada basis data antara lain semua transaksi perbankan, aplikasi pemesanan, penjadwalan penerbangan, proses registrasi dan pencatatan data mahasiswa pada perguruan tinggi, aplikasi pemrosesan penjualan, pembelian dan pecatatan data barang pada perusahaan dagang, pencatatan data pegawai beserta akivitasnya termasuk operasi penggajian pada suatu perusahaan, dan sebagainya. Beberapa informasi pada perusahaan retail seperti jumlah penjualan, mencari jumlah stok yang tersedia, barang apa yang paling laku dijual pada bulan ini, dan berapa laba bersih perusahaan dapat diketahui dengan mudah menggunakan basis data.

2. Kemampuan Akhir

Setelah mempelajari uraian materi dalam bab pembelajaran dan kegiatan belajar diharapkan mahasiswa dapat memiliki kompetensi sikap, pengetahuan dan ketrampilan yang berkaitan dengan materi yang ditunjukkan pada Gambar 1.1



Gambar 1.1 Peta Konsep Materi Basis Data

Bab II
KONSEP BASIS DATA

Setelah mempelajari bab ini, mahasiswa diharapkan mampu :

1. Menjelaskan definisi basis data
2. Menjelaskan perbedaan file tradisional dan basis data
3. Mengklasifikasikan komponen basis data
4. Menjelaskan sistem manajemen basis data
5. Menjelaskan tujuan dan manfaat basis data

1. Definisi Basis Data

Pengertian basis data dapat ditinjau dari dua sisi, secara kharfiah dan pengertian secara istilah. Menurut pengertian secara kharfiah, basis data terdiri dari dua kata yaitu basis dan data. Basis dapat diartikan sebagai suatu markas atau gudang, tempat bersarang atau tempat berkumpul. Data dapat diartikan merupakan representasi dari fakta dunia yang mewakili suatu obyek (manusia, barang, peristiwa, keadaan dsb) yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi atau kombinasinya.Adapun menurut pengertian secara istilah, terdapat beberapa definisi yaitu sebagai berikut :

a. Himpunan kelompok data (arsip) yang saling berhubungan yangdiorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah
b. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redundancy) yang tidak perlu, untuk memenuhi berbagai kebutuhan

c. Kumpulan file/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimpanan tertentu.
d. Kumpulan data, yang dapat digambarkan sebagai aktifitas dari satu atau lebih organisasi yang berelasi.

Menurut Elmasri, penggunaan istilah basis data lebih dibatasi pada arti implisit yang khusus mempunyai beberapa pengertian, yaitu :

a. Basis data merupakan penyajian suatu aspek dari dunia nyata (*real word* atau *miniworld*). Misalnya basis data perbankan, perpustakaan, pertanahan, perpajakan
b. Basis data merupakan kumpulan data dari berbagai sumber yang secara logika mempunyai arti implicit. Sehingga apabila data terkumpul secara acak dan tanpa mempunyai arti, tidak dapat disebut basis data.
c. Basis data perlu diancanag, dibangun dan data dikumpulkan untuk suatu tujuan tertentu.
d. Basis data dapat digunakan oleh beberapa pemakai dan beberapa aplikasi yang sesuai dengan kepentingan pemakai.

2. Perbedaan File Tradisional dan Basis Data

File Tradisional adalah file dimana setiap user mengimplementasikan file yang dibutuhkan untuk aplikasi khusus sebagai bagian dari pemrograman aplikasinya. Basis data adalah sekumpulan file-file yang mempunyai kaitan antara satu file dengan file lain sehingga membentuk suatu bangunan data untuk menginformasikan suatu perusahaan atau instansi dalam batasan tertentu.

Perbedaan File Manajemen Tradisional & File Manajemen Basis Data

Perihal	File Tradisional	Manajemen Basis Data
Orientasi program	Sering terjadi kerangkapan data Kaku	Terkontrolnya kerangkapan data Luwes
Kelemahan	Timbulnya data rangkap & ketidak konsistenan Data tidak dapat digunakan bersama-sama Kesukaran dalam pengaksesan data Tidak fleksibel Data tidak standar	Storage yang digunakan besar Dibutuhkan tenaga spesialis Softwarenya mahal Kerusakan pada system database dapat mempengaruhi departemen lain yang terkait

3. Komponen Sistem Manajemen Basis Data

Basis data adalah merupakan suatu sistem yang dibangun oleh beberapa komponen diantaranya ada enam komponen pokok antara lain ialah:

a. Perangkat keras (hardware) dalam sistem komputer. Dalam sistem pengolahan basis data digital perangkat utama sebagai pengolah data dalah komputer.
b. Perangkat Lunak Aplikasi (software) lain yang mendukung dan bersifat opsional. Perangkat lunak digunakan untuk mendukung proses pengelolaan basis data. Misal: bahasa pemrograman C, basic pascal.

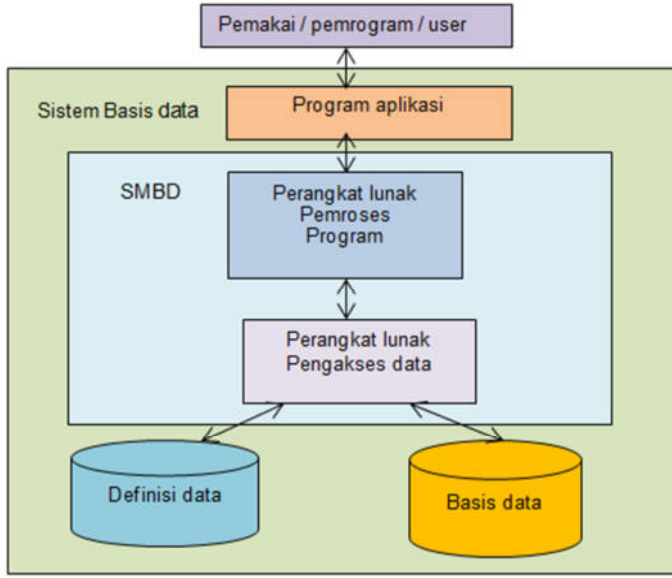
c. Sistem Operasi (operating system). Sistem operasi merupakan perangkat lunak yang digunakan untuk mengelola aplikasi basis data dan penggunaan sumberdaya komputer.
d. Basis data data lain yang mempunyai keterkaitan dan hubungan dengan basis data itu sendiri. Berisi atau memiliki objek-objek basis data seperti file, table, indeks. Mempunyai disfinisi struktur baik untuk basis data maupun objek-objek secara detail.
e. Sistem Pengelola Basis Data Database Management System atau database managemen system (DBMS). Merupakan program aplikasi untuk pengelolaan basis data, seperti Microsoft acces, oracle dan lian-lain
f. Pemakai (user), yaitu pengguna yang terlibat dalam pengelolaan basis dan penggunaan basis data

4. Sistem Manajemen Basis Data

Sistem manajemen basis data adalah merupakan sebuah tatanan (keterpaduan) yang terdiri atas sejumlah komponen-komponen fungsional (komputer) yang saling berhubungan secara bersama-sama, bertujuan untuk memenuhi suatu proses atau pekerjaaan tertentu. Sistem ini merupakan gabungan antara basis data dan kumpulan program atau perangkat lunak DBMS (d*atabase management system).*

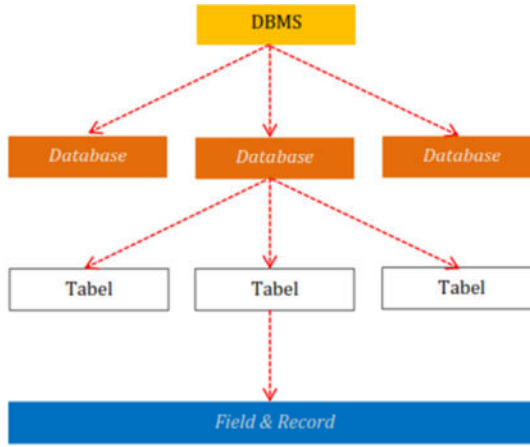
*DBMS a*dalah program aplikasi yang dibuat dan bekerja dalam satu system. *DBMS* didesain untuk membantu dalam hal pemeliharaan dan utilitas kumpulan data dalam jumlah besar. DBMS dapat menjadi alternatif penggunaan secara khusus untuk aplikasi, misalnya penyimpanan data dalam field dan menulis kode aplikasi yang spesifik untuk pengaturannya. Kumpulan file (table) yang saling berhubungan dalam di sebuah komputer dan sekumpulan program yang memungkinkan beberapa pemakai dan

atau program lain untuk mengakses dan memanipulasi file-file atau table-tabel tersebut.



Gambar 2.1. Konsep DBMS

Hubungan DBMS dengan basis data, tabel, field, dan record ditunjukkan pada Gambar 2.2.



Gambar 2.2 Hubungan DBMS dan basis data

5. Tujuan dan Manfaat Basis Data

Kesuksesan suatu organisasi bergantung pada kemampuannya menangkap data secara akurat dan tepat waktu. Hal tersebut berkaitan dengan operasi dan pengaturan data secara efektif, maupun penggunaan data untuk keperluan analisis untuk kebutuhan pendukung keputusan. Kemampuan untuk mengatur atau mengolah sejumlah data, dan kecepatan untuk mencari informasi yang relevan, adalah aset yang sangat penting bagi suatu organisasi.Beberapa tujuan penggunaan basis data adalah sebagai berikut :

1. Kecepatan dan Kemudahan (*Speed*) , melalui basis data diharapkan pengguna dapat melakukan penyimpanan, perubahan dan menampilkan kembali dengan cepat dan mudah.
2. Efisiensi Ruang Penyimpanan (*Space*). Penggunaan basis data mampu mengurangi pengulangan atau redundansi data. Hal ini dapat dilakukan dengan menerapkan sejumlah pengkodean atau

dengan membuat relasi-relasi (dalam bentuk file) antara kelompok data yang saling berhubungan.

3. Keakuratan (*Accuracy*), melalui basis data data keakuratan data lebih terjaga dengan menerapkan aturan dan batasan tertentu (*constraint*), tipe data, domain data dan keunikan data
4. Ketersediaan (*Availability*). Dengan basis data data yang sudah tidak dipakai dapat dipisahkan dari sistem database yang sedang aktif. Hal ini dapat dilakukan dengan cara penghapusan atau memindahkannya ke media backup untuk menghemat ruang penyimpanan. Selain itu dapat memanfaatkan teknologi jaringan komputer agar data yang berada di suatu lokasi atau cabang daat juga diakses oleh lokasi atau cabang lainnya.
5. Kelengkapan (*Completeness*). Agar data yang dikelola senantiasa lengkap baik relatif terhadap kebutuhan pemakai maupun terhadap waktu. Hal ini dapat dilakukan melaluipenambahan record-record data, perubahan struktur basis data, menambah field pada tabel atau menambah tabel baru.
6. Keamanan (*Security*). Walaupun tidak semua sistem basis data menerapkannya, keamanan dalam penggunaan basis data diperlakukan pada sistem yang besar dan serius. Dengan penerapan ini, setiap pengguna dibedakan hak aksesnya; yakni ditentukan obyek-obyek mana saja yang bisa diakses dan proses apa saja yang bisa dia dilakukan.
7. Kebersamaan (*Sharability*). Agar data yang dikelola oleh sistem mendukung lingkungan multiuser (banyak pemakai) dengan menjaga / menghindari munculnya problem baru seperti *inkonsistensi data* (karena terjadi perubahan data yang dilakukan oleh beberapa user dalam waktu yang bersamaan) atau kondisi *deadlock* (karena ada banyak pemakai yang saling menunggu untuk menggunakan data).

EVALUASI

1. Jelaskan beberapa pengertian atau definisi dari basis data!
2. Jelaskan mengapa seseorang memerlukan basis data?
3. Sebutkan beberapa contoh penerapan basis data dalam kehidupan sehari-hari!
4. Jelaskan perbedaan antara basis data dengan system manajemen basis data?

BAB III
ELEMEN LINGKUNGAN BASIS DATA

Setelah mempelajari bab ini, mahasiswa diharapkan mampu :

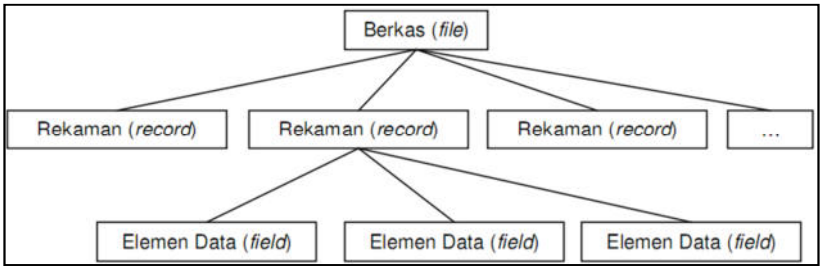
1. Menjelaskan arsitektur basis data
2. Mengklasifikasikan pengguna dalam basis data
3. Menjelaskan file tabel dan file record

1. Arsitektur Basis Data

Arsitektur basis data merupakan serangkaian pengetahuan tentang pemodelan data. Pengetahuan tentang File, table, field, record indeks, abstraksi data dan serangkaian konsep yang digunakan untuk membuat diskripsi struktur basis data. Melalui diskripsi Struktur basis data dapat ditentukan jenis data, hubungan dan konstrain (keterbatasan) data yang ditangani. Dalam basis data, data diorganisasikan kedalam bentuk elemen data (field), rekaman (record), dan berkas (file). Definisi dari ketiganya adalah sebagai berikut:

a. Elemen (kolom atau field) data adalah satuan data terkecil yang tidak dapat dipecah lagi menjadi unit lain yang bermakna. Misalnya data siswa terdiri dari NIS, Nama, Alamat, Telepon atau Jenis Kelamin.
b. Rekaman (record) merupakan gabungan sejumlah elemen data yang saling terkait. Istilah lain dari record adalah baris atau tupel.
c. Berkas(file) adalah himpunan seluruh record yang bertipe sama

Struktur hirarki sebuah database dapat digambarkan dalam diagram hirarki Gambar 3.1



Gambar 3.1. Struktur hirarki sistem basis data

2. Pengguna Basis Data

Pada tingkat pemakai, data base dikelompokkan menjadi beberapa tingkat pemakai yaitu antara lain sebagai berikut:

a. Database Administrator, ialah manusia yang mengorganisasi seluruh sistem basis data. Database adaministrator imemiliki tanggung jawab penuh dalam manajemen database meliputi: pengaturan hak akses, koordinasi dan monitoring serta bertanggung jawab terhadap kebutuhan hardware dan software. Dalam pekerjaannya biasanya dibantu oleh staf Admin.
b. *Database Designer*, adalah manusia yang bertugas merancang dan mengembangkan database. Database designer bertanggung jawab dalam identifikasi data yang tersimpan dalam database, menentukan struktur data yang tepat untuk disimpan dalam database. Database designer memerlukan koordinasi akan kebutuhan user database.
c. *Application Programmer,* ialah penggunaa yang berinteraksi dengan basis data melalui *Data Manipulation Language* (DML). DML meliputi program yang ditulis dalam bahasa pemrograman induk yang dipakai.

d. *End user*, adalah adalah pengguna yang memanfaatkan atau membutuhkan akses ke database melalui query, manambah, merubah menghapus maupun membuat *report database*. *End user*dapat dikategorikan:
   a) *Casual end users* atau pengguna tak tetap atau user mahir. Pengguna yang tidak selalu mengakses database, tapi kadang memerlukan informasi terbaru. Berinteraksi dengan sistem tanpa modul program, hanya menggunakan *query* (untuk akses dan manipulasi data) yang telah disediakan oleh DBMS.
   b) *Natïve* atau *parametric end users* atau user umum. Pengguna yang pekerjaan selalu konstan yaitu melakukan query dan update data. Misalnya:*bank teller*, pegawai reservasi. Pengguna ini berinteraksi dg sistem melalui pemanggilan suatu program aplikasi permanen (executable) yang telah dibuat sebelumnya oleh programmer.
   c) User Khusus (Specialized User). Pengguna yang menulis aplikasi basis data *non konvensional* untuk keperluan khusus yang bisa saja mengakses basis data dengan atau tanpa DBMS yang bersangkutan.
   d) *Sophisticated end users*. pengguna yang melengkapi kebutuhan database user, seperti engineer, scientist, business analyst.
   e) Stand-alone users. penggunauser yang mengelola personal database.
e. *System Analyst*, ialah pengguna yang merencanakan dan menentukan kebutuhan sistem.
f. *Application Programmers* (Software Engineering), ialah pengguna tanggungjawabnya berhubungan dengan kebutuhan koneksi database.

g. *Worker behind the scene, ialah pengguna* yang tidak tertarik pada database, tetapi lebih cenderung pada membangun data base atau kebutuhannya menggunakan alat bantu. Pengguna ini dibedakan menjadi

a) DBMS system designers dan implementer, ialah pengguna yang merancang dan mengimplementasikan modul-modul dan interface menggunakan paket-paket software DBMS. (seperti:. Modul: catalog, *procs query lang., procs interface, access & buffering data, controlling cuncurrency, handling data recovery & security; interfacing: interface for integrated system*).

b) *Tool developers*. Pengguna yang merancang dan mengimplementasikan tools untuk mendukung software DBMS. Seperti Tools untuk meningkatkan performance database, tool untuk monitoring operasional database.

c) *Operators dan maintenance personnel*. Para personel administrator yang bertanggung jawab akan jalannyaoperasional database termasuk maintenance (hardware/software) DBMS.

3. File Table dan File Record

Field adalah kumpulan dari karakter yang membentuk satu arti, maka jika terdapat field misalnya seperti NomerBarang atau NamaBarang, maka yang dipaparkan dalam field tersebut harus yang berkaitan dengan nomer barang dan nama barang. Atau definisi field yang lainnya yaitu tempat atau kolom yang terdapat dalam suatu tabel untuk mengisikan nama-nama (data) field yang akan di isikan.

Record adalah kumpulan field yang sangat lengkap, dan biasanya dihitung dalam satuan baris. Tabel adalah merupakan kumpulan dari beberapa record dan juga field. File adalah terdiri

dari record-record yang menggambarkan dari satu kesatuan data yang sejenis. Misalnya seperti file nama barang berisikan data tentang semua nama barang yang ada. Data adalah kumpulan fakta atau kejadian yang digunakan sebagai penyelesaian masalah dalam bentuk informasi. Pengertian basis data (database) adalah basis data yang terdiri dari dua kata, yaitu kata basis dan data. Basis dapat di artikan markas ataupun gudang, maupun tempat berkumpul.

Kode_pegawai	Nama_depan	Nama_Belakang	Alamat
01	Ani	Mariani	Jl. Mawar no.2
02	Kiki	Aditya	Jl. Kemerdekaan no. 3
03	Fitri	Ginting	Jl. Pembangunan no. 89
04	Hasan	Martono	Jl. Diponegoro no. 10

Gambar 3.2 Field dan record

EVALUASI

1. Jelaskan secara singkat arsitektur atau struktur basis data!
2. Jelaskan secara singkat pengertian struktur fisik basis data!
3. Jelaskan urutan pengguna basis data berdasarkan prioritas tingkat pemakai!

BAB IV
MODEL DATA RELATIONAL

Setelah mempelajari bab ini, diharapkan mahasiswa mampu:
1. Memahami konsep pemodelan data
2. Menjelaskan model data relasional
3. Menjelaskan istilah dalam model data relasional
4. Menjelaskan kunci dalam model data relasional
5. Menjelaskan keuntungan model data relasional

1. Konsep Pemodelan Data

Pemodelan data merupakan sarana untuk melakukan abstraksi data. Merupakan sejumlah konsep untuk membuat diskripsi stuktur basis data. Kebanyakan model data memuat spesifikasi untuk operasi dasar (*basic operation)* dalam pengaksesan dan pembaharuan data. Pada perkembangan terakhir dikenal dengan istilah tabiat data (*data behavior*) pada pemrograman berorientasi object. Terdapat sejumlah cara dalam merepresentasikan model dalam perancangan basis data. Secara umum pemodelan data dapat dikelompokkan menjadi dua yaitu :

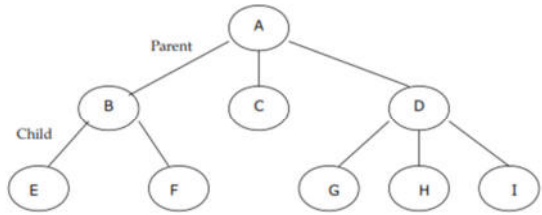
a. *Object based logical model*. Dalam pemodelan ini struktur atau hirarki basis data diilustrasikan berdasarkan object. Model ini meliputi: 1) Model keterhubungan entitas (Entity Relationalship Model atau ERD). 2) Model berorientasi object (Object-Oriented Model). 3) Model Data Semantik(Semantic Data Model). 2) Model data Fungsional (Function Data Model).
b. *Record-based logical model*. Dalam model ini struktur basis data diilustrasikan berdasarkan record. Model ini meliputi: 1) Model

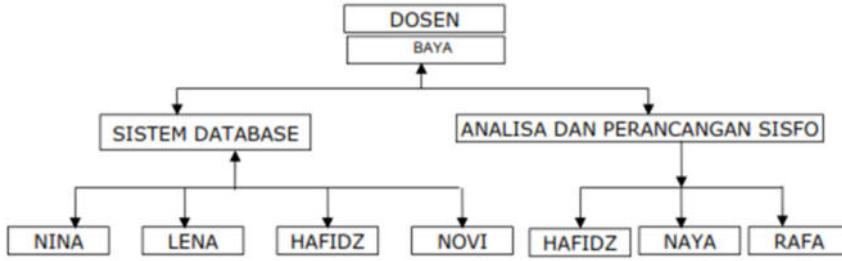
relational (*Relational Model*). 2) Model Herarkis (Hierarchical Model) 3) Model Jaringan (*Network Model).*

Pada record based data model disamping digunakan untuk menguraikan struktur logika keseluruhan dari suatu database, juga digunakan untuk menguraikan implementasi dari system database (*higher level description of implementation*) Terdapat 3 data model pada record based data model :

1) Model Relasional,
   Dimana data serta hubungan antar data direpresentasikan oleh sejumlah table, dan masing -masing table terdiri dari beberapa kolom yang namanya unique. Model ini berdasarkan notasi teori himpunan (set theory), yaitu relation.
   Contoh :
   data base penjual barang terdiri dari 3 tabel :
   – Supllier
   – Path (Suku_cadang)
   – Delivery (pengiriman)

2) Model Hirarki
   Dimana data serta hubungan antar data direpresentasikan dengan record dan link (pointer), dimana record-record tersebut disusun dalam bentuk tree (pohon), dan masing-masing node pada tree tersebut merupakan record/grup data elemen

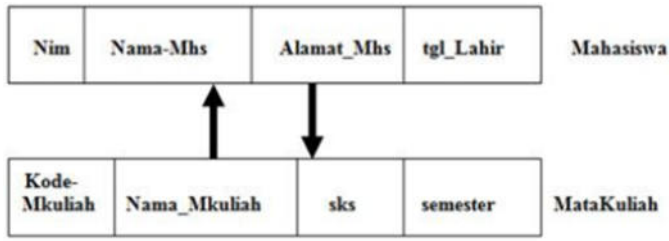


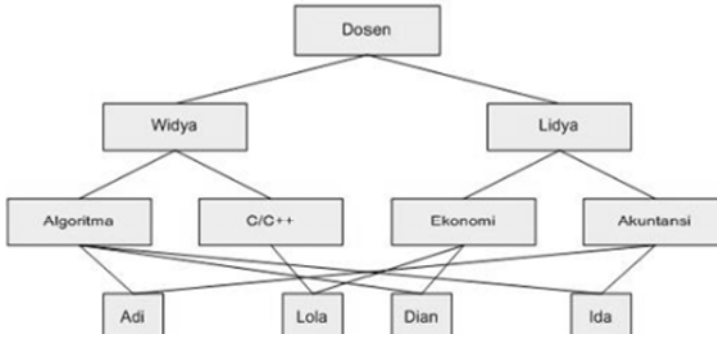


Gambar 4.1. Model Hirarki

3) Model Jaringan

Mirip dengan hirarkical model, dimana data dan hubungan antar data direpresentasikan dengan record dan links. Model data jaringan adalah pengembangan dari model data hirarkis. Dalam model data jaringan, child record diperkenannkan memiliki lebih dari satu parent record. Perbedaannya terletak pada susunan record dan linknya yaitu network model menyusun record-record dalam bentuk graph.





Gambar 4.2. Model data Jaringan

2. Model Data Relational

Model Data Relasional adalah suatu model basis data yang menggunakan tabel dua dimensi, yang terdiri atas baris dan kolom untuk menggambarkan sebuah berkas data. Model ini menunjukkan cara mengelola/mengorganisasikan data secara fisik dalam memory sekunder, yang akan berdampak pula pada bagaimana kita mengelompokkan data dan membentuk keseluruhan data yang terkait dalam sistem yang dibuat. Contoh table dan keterhubungannya:

MHS

NPM	Nama	Alamat
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananingrum	Bogor
50096487	Pipit	Bekasi
21196353	Quraish	Bogor

MKUL

KDMK	MTKULIAH	SKS
KK021	P. Basis Data	2
KD132	SIM	3
KU122	Pancasila	2

NILAI

NPM	KDMK	MID	FINAL
10296832	KK021	60	75
10296126	KD132	70	90
31296500	KK021	55	40
41296525	KU122	90	80
21196353	KU122	75	75
50095487	KD132	80	0
10296832	KD132	40	30

Keuntungan Model Data Relasional

a. Bentuknya sederhana
b. Mudah melakukan berbagai operasi data (query, update/edit, delete).

3. Istilah dalam Model Relasional

Istilah-istilah yang digunakan dalam model data relasional, diantaranya : relasi, atribut, record, cardinalitas.

a. Relasi

   Merupakan sebuah tabel yang terdiri dari beberapa kolom dan beberapa baris.

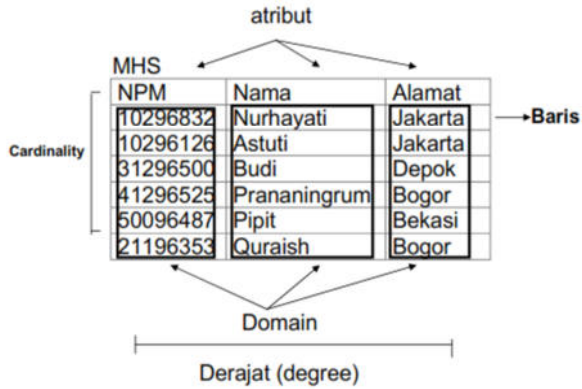
b. Entity atau Entitas

   Merupakan obyek yang mewakili sesuatu dalam dunia nyata dan dapat dibedakan antara satu dengan lainnya (unique)

c. Atribut

   Merupakan karakteristik dari entitas atau relationship, yang menyediakan penjelasan detail tentang entitas atau relationship. Dalam penerapannya (level fisik) atribut merupakan field atau kolom dari sebuah tabel

d. Record
   Merupakan baris pada sebuah relasi
e. Domain
   Merupakan kumpulan nilai yang valid untuk satu atau lebih atribut
f. Derajat (degree)
   Merupakan jumlah atribut dalam sebuah relasi (jumlah field)
g. Cardinality
   Merupakan jumlah tupel dalam sebuah relasi (jumlah record). Model data harus dapat merepresentasikan jumlah peristiwa dari obyek di dalam hubungan yang diberikan.



Gambar 4.3. Ilustrasi Penerapan Istilah

4. Jenis Kunci Relational

Key adalah merupakan suatu atribut yang menandakan kunci dari suatu entitas yang bersifat unik. *Key attribute* adalah satu atau beberapa atribut yang mempunyai nilai unik sehingga dapat digunakan untuk membedakan data pada suatu baris/record dengan baris lain pada suatu entitas. Key attribute dibedakan menjadi tiga yaitu: *Superkey, Candidat Key, Primary* key, dan *Foreign Key.*

a. *Superkey*

Satu atribut/kumpulan atribut yang secara unik mengidentifikasi sebuah tupel di dalam relasi (satu atau lebih field yang dapat dipilih untuk membedakan antara 1 record dengan record lainnya).

Contoh:

Untuk tabel MHS di atas, super key-nya:

- NPM
- NAMA (dengan syarat tidak ada nama yang sama)
- ALAMAT (dengan syarat tidak ada alamat yang sama)
- NPM + NAMA
- NPM + ALAMAT
- NAMA + ALAMAT
- NPM + NAMA + ALAMAT

b. *Candidat Key*

Atribut di dalam relasi yang biasanya mempunyai nilai unik (super key dengan jumlah field yang paling sedikit).

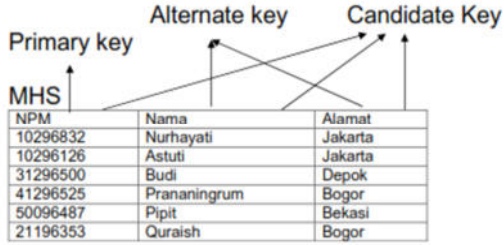
Dari contoh di atas, maka *candidate key*-nya adalah NPM, NAMA dan ALAMAT (karena hanya terdiri dari 1 field saja)

c. *Primary* key.

Candidate key yang dipilih untuk mengidentifikasikan tupel secara unik dalam relasi, maka *primary key* yang dipilih adalah NPM (unik, tidak ada NPM yang sama).

d. Foreign Key

Atribut dengan domain yang sama yang menjadi kunci utama pada sebuah relasi tetapi pada relasi lain atribut tersebut hanya sebagai atribut biasa.



NPM	Nama	Alamat
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananingrum	Bogor
50096487	Pipit	Bekasi
21196353	Quraish	Bogor

5. Keuntungan Model Data Relational

Model Relasional merupakan model data yang paling banyak digunakan saat ini. Hal ini disebabkan oleh bentuknya yang sederhana dibandingkan dengan model jaringan/network atau model hirarki. Bentuk yang sederhana ini membuat pekerjaan seorang programmer menjadi lebih mudah, yaitu dalam melakukan berbagai operasi data (query, insert, update, delete, dan lainnya).

Kelebihan model data relasional, diantaranya

a. Bentuknya sederhana.
b. Data dapat diakses lebih cepat.
c. Struktur basis data mudah diubah.
d. Data lebih akurat.
e. Memudahkan user untuk membangun dan memodifikasi program aplikasi.
f. Memudahkan user dalam membentuk query yang kompleks untuk memanggil kembali (retrieve) data.

EVALUASI

1. Jelaskan perbedaan antara model basis data relasional, hirarki, dan jaringan!
2. Pada model basis data relasional terdapat beberapa *key attribute,* jelaskan perbedaan masing-masing *key attribute tersebut!*

BAB V
PERANCANGAN BASIS DATA DENGAN ERD

Setelah mempelajari materi ini, diharapkan mahasiswa mampu:
1. Menjelaskan pengertian Entity Relationship Diagram (ERD)
2. Memahami konsep dasar ERD
3. Menjelaskan komponen ERD
4. Membuat rancangan ERD

1 Definisi Entity Relationship Diagram (ERD)

Diagram relasi entitas atau *entity-relationship diagram* (ERD) adalah suatu diagram dalam bentuk gambar atau simbol yang mengidentifikasi tipe dari entitas di dalam suatu sistem yang diuraikan dalam data dengan atributnya, dan menjelaskan hubungan atau relasi diantara entitas tersebut. ERD merupakan model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak. ERD berupa model data konseptual, yang merepresentasikan data dalam suatu organisasi.ERD menekankan pada struktur dan relationship data

2 Memahami konsep dasar ERD

ERD digunakan oleh profesional sistem untuk berkomunikasi dengan pemakai eksekutif tingkat tinggi dalam perusahaan atau organisasi yang tidak tertarik pada pelaksanaan operasi sistem sehari-hari, namun lebih menekankan kepada beberapa hal yaitu :

a. Data apa saja yang diperlukan untuk bisnis mereka?
b. Bagaimana data tersebut berelasi dengan data lainnya?
c. Siapa saja yang diperbolehkan mengakses data tsb?

Untuk menggambarkan ER diagaram setidaknya ada empat langkah yang harus dilakukan oleh perancang basis data yaitu:

a. Menemukan atau mendefinisikan Entitas
b. Menemukan atau mendefinisikan atribute
c. Menemukan atau mendefinisikan Relasi
d. Menggambarkan ERD menggunakan notasi-notasi standar

3 Menjelaskan komponen ERD

Untuk dapat membuat entity relasional diagram, maka komponen yang harus terpenuhi adalah:

a. Obyek Data, Atribut dan Hubungan.

Obyek Data Adalah representasi dari hampir semua informasi gabungan yang harus dipahami oleh perangkat lunak. Objek data dapat berupa entitas eksternal (misalkan semua yang menghasilkan informasi), suatu benda (misal laporan atau tampilan), peristiwa (misalnya proses meminjam) atau event, peran (misalnya peminjam), unit organisasi atau suatu struktur. Sebagai contoh : orang atau mobil dapat dipandang sebagai objek data bila salah satu dari mereka dapat didefinisikan dalam bentuk atribut. Deskripsi objek data menghubungkan objek data dengan semua atributnya. Obyek data dihubungkan satu dengan yang lainnya, misalkan seorang dapat memiliki mobil, dimana hubungan "memiliki" mengkonotasikan suatu hubungan khusus antara seorang dengan mobil.

Atribut Menentukan property suatu obyek data dan mengambil salah satu dari tiga karakteristik yang berbeda. Atribut dapat digunakan untuk:

1) Menamai sebuah contoh dari obyek data
2) Menggambarkan contoh

3) Membuat referensi ke contoh yang lain pada tabel yang lain.

Hubungan Obyek data disambungkan satu dengan lainnya dengan berbagai macam cara. Andaikan ada dua objek data, buku dan toko buku, obyek tersebut dapat diwakilkan dengan menggunakan dua notasi sederhana, dibangun suatu hubungan anatar buku dengan toko buku karena kedua obyek data tersebut berhubungan. Hubungan tersebut dapat berupa :

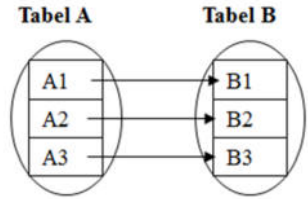
1) Toko buku memesan buku
2) Toko buku menampilkan buku
3) Toko buku menjual buku

Hubungan memesan, menampilkan, menjual mendefinisikan hubungan yang relevan antara buku dan toko buku. Penting untuk dicatat bahwa hubungan obyek mempunyai dua arah, dimana mereka dpaat dibaca dari dua arah, misalnya :toko buku memesan buku atau buku dipesan oleh toko buku.

b. Kardinalitas dan Modalitas Kardinalitas

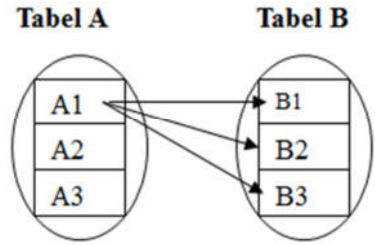
Model data harus dapat merepresentasikan jumlah peristiwa dari obyek di dalam hubungan yang diberikan.

1) Satu ke satu (1:1) Misalnya: seorang suami hanya dapat memiliki satu istri, dan seorang istri hanya mempunyai satu suami. Contoh ilustrasi dapat ditunjukkan pada Gambar 5.1



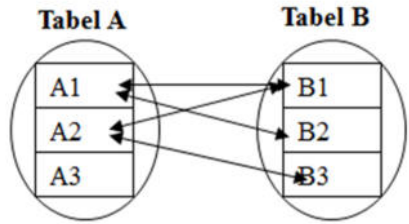
Gambar 5.1 Relasi Satu ke Satu

2) Satu ke banyak (1:N) Misalnya: seorang ibu kandung dapat memiliki banyak anak, tetapi seorang anak hanya dapat memiliki satu ibu kandung. Contoh ilustrasi dapat ditunjukkan pada Gambar 5.2



Gambar 5.1 Relasi Satu ke Banyak

3) Banyak ke banyak (M:N) Misalnya: seorang paman dapat memiliki banyak keponakan, sementara itu seorang keponakan dapat memiliki banyak paman. Contoh ilustrasi dapat ditunjukkan pada Gambar 5.3



Gambar 5.3 Relasi Banyak ke Banyak

Modalitas dari suatu hubungan adalah nol bila tidak ada kebutuhan eksplisit untuk hubungan yang terjadi atau hubungan itu bersifat opsional. Modalitas bernilai satu jika suatu kejadian dari hubungan merupakan perintah.

4 Membuat rancangan ERD

Untuk menggambarkan ERD setidaknya ada tiga langkah yang harus dilakukan oleh perancang basis data yaitu:

a. Menemukan atau mendefinisikan Entitas
b. Menemukan atau mendefinisikan atribute
c. Menemukan atau mendefinisikan Relasi
d. Menentukan kardinalitas

a. Entitas (Entity)

Entitas merupakan obyek yang mewakili sesuatu dalam dunia nyata dan dapat dibedakan antara satu dengan lainnya (unique). Setiap entitas memiliki beberapa atribut yang mendeskripsikan karakteristik dari objek tersebut. Atribut Dapat berupa:

- Fisik (mobil, rumah, manusia, pegawai dsb)
- Abstrak/konsep (department, pekerjaan, mata kuliah dsb)
- Kejadian (pembelian, penjualan, peminjaman, dll)

Entitas dapat dibedakan menjadi dua macam yaitu **Entitas kuat** dan **entitas lemah**. **Entitas lemah** adalah yang keberadaannya tergantung pada entitas lain. Gambar dibawah ini menjelaskan notasi umum entitas kuat dengan nama entitas Anggota dan entitas lemah dengan nama entitas tanggungan. Entitas tanggungan disebut sebagai **entitas lemah** karena jika data seorang pegawai dihapus maka data tanggungannya juga akan terhapus. Keberadaan data tanggungan tergantung pada data di pegawai.

Gambar 5.4 Contoh Notasi

Contoh :

Entitas	Atribut
Petugas	NIP, Nama, Alamat, Agama, Jenis kelamin
Departemen	No, Nama, Lokasi

Gambar 5.5 Contoh Penggunaan Simbol Entitas dan Atribut

Urutan langkah untuk menemukan atau mendefinisikan entitas dalam suatu sistem adalah:

a. Buat ilustrasi/gambaran cerita tentang sistem yang akan dicari entitasnya
b. Tandai setiap objek yang diwakili oleh kata benda yang ada di dalam ilustrasi tersebut
c. Untuk setiap objek tersebut yakinkan bahwa ia memiliki karakteristik yang nanti disebut sebagai atribut
d. Tentukan objek yang merupakan entitas (Jika memang ia memiliki karakteristik jadikan ia sebagai entitas)
e. Menggambarkan entitas beserta atributnya menggunakan notasi simbol yang telah ditentukan.

Contoh cara menentukan entitas:

Langkah 1. *Deskripsi tentang gambaran sistem (misalnya gambaran tentang system informasi perpustakaan):*

Departemen A mempunyai perpustakaan, perpustakaan ini dijaga oleh dua orang pegawai. Anggota perpustakaan ini adalah seluruh pegawai departemen yang sudah terdaftar menjadi anggota perpustakaan. Koleksi buku yang dimiliki ada sekitar 200 eksemplar, dengan berbagai jenis buku, pengarang dan penerbit. Untuk melakukan peminjaman buku, anggota melakukan proses peminjaman ke petugas dengan memberikan kartu anggota perpustakaan, lama peminjaman adalah dua minggu, proses pengembalian dilakukan dengan memberikan buku dan kartu anggota untuk pengecekan. Jika anggota terlambat mengembalikan maka akan dikenakan denda Rp.1000, 00 per hari.

Langkah 2. *Tandai setiap objek yang diwakili oleh kata benda yang ada di dalam ilustrasi tersebut.*

Departemen A mempunyai perpustakaan, perpustakaan ini dijaga oleh dua orang **pegawai. Anggota** perpustakaan ini adalah seluruh pegawai departemen yang sudah terdaftar menjadi anggota perpustakaan. Koleksi buku yang dimiliki ada sekitar 200 eksemplar, dengan berbagai **jenis buku, pengarang** dan **penerbit**. Untuk melakukan peminjaman buku, anggota melakukan proses **peminjaman** ke petugas dengan memberikan kartu anggota perpustakaan, lama peminjaman adalah dua minggu, proses **pengembalian** dilakukan dengan memberikan buku dan kartu anggota untuk pengecekan. Jika anggota terlambat mengembalikan maka akan dikenakan **denda** Rp.1000, 00 per hari.

Langkah 3. *Untuk setiap objek tersebut yakinkan bahwa ia memiliki karakteristik yang nanti disebut sebagai atribut*

Pegawai : id, nama, Username, Password
Anggota : No_anggota, Nama, alamat, Nomor_telp, jenis_kelamin
Buku : Kode, kategori,Judul,Jumlah_halaman, ISBN, Pengarang, penerbit
Jenis buku : Kode, Jenis
Pengarang : Kode, Nama
Penerbit : Kode, Nama
Peminjaman : Kode_peminjaman, Id_Petugas, No_anggota, Kode_Buku, Tgl_pinjam, Tgl_kembali
Pengembalian : Kode_pengembalian, No_anggota, tgl_kembali, denda

b. Atribut

Atribut adalah sifat-sifat atau karakteristik pada suatu entitas. Nama atribut ini identik dengan nama kolom atau field pada suatu tabeldalam basis data.Candidat Key adalah merupakan superkey yang jumlah atributnya paling sedikit.
Misalnya *candidat key* untuk entitas petugas antara lain:

1) Id_Pegawai
2) Nama (jika dapat dijamin kalau tidak ada nama yang sama antara satu baris dengan baris yang lain)

Primary key adalah suatu *candidat key* yang dipilih menjadi kunci utama karena sering dijadikan acuan untuk mencari informasi, ringkas, menjadi keunikan suatu baris. MisalnyakodeBuku antara buku yang satu dengan buku yang lain pasti berbeda, dalam hal ini kodeBuku dapat digunakan sebagai

suatu key. Gambar 5.6 menjelaskan simbol atau notasi primary key.

Gambar 5.6 Primary key

c. Relasi (Relationship)

Relasi menyatakan hubungan antara dua atau beberapa entitas.Setiap relasi mempunyai batasan (constraint) terhadap kemungkinan kombinasi entitas yang berpartisipasi. Batasan tersebut ditentukan dari situasi yang diwakili relasi tersebut. Ragam atau jenis relasi dibedakan menjadi beberapa macam antara lain adalah.

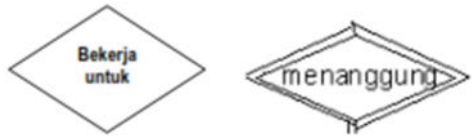
a) Relasi Binary. Relasi ini merupakan relasi antara 2 entitas. Relasi ini dibedakan menjadi :
   - Relasi One-to-one (notasi 1:1)
   - Relasi One-to-many (notasi 1:N) atau many-to-one (notasi N:1)
   - Relasi Many-to-many (notasi M:N)

b) Relasi Ternary. Relasi ini merupakan relasi antara 3 entitas atau lebih.

   Dalam Relasi One-to-one (1:1) setiap atribute dari satu entitas berpasangan dengan satu attribute dari entitas yang direlasikan.Dalam relasi One-to-many (1:N) atau many-to- one (N:1) satu atribute berelasi dengan beberapa atribute dari entitas yang direlasikan.Dalam Many-to-many (M:N) satu atribute berelasi dengan beberapa atribute dari entitas yang direlasikan, begitu pula sebaliknya.

Notasi Relasi

Sebagaimana entias dalam relasi juga dapat dibedakan menjadi relasi kuat dan relasi lemah. gambar dibawah ini menjelaskan notasi umum untuk relasi kuat dan relasi lemah.



Gambar 5.5 Relasi kuat dan relasi lemah

Menemukan Relasi

Beberapa langkah yang dapat dilakukan untuk menemukan atau mengidentifikasi relasi yaitu antara lain sebagai berikut:

1) Dari gambaran cerita sistem, tandai setiap hubungan yang diwakili oleh kata kerja yang ada di dalam ilustrasi tersebut beserta entitas yang berhubungan
2) Identifikasikan rasio kardinalitas dari setiap hubungan
3) Identifikasikan batasan partisipasi dari setiap hubungan yang ada berikut kemungkinan atribut yang muncul dari setiap hubungan

Gambarkan hubungan tersebut dalam bentuk notasi diagram dan gabungkan dengan notasi Entitas dan atribut yang dibuat sebelumnya. Sebagai contoh tentukan relasi untuk sistem informasi perpustakaan dengan melihat deskripsi sistem di atas.
Langkah-langkah penyelesaian adalah :

Langkah 1:

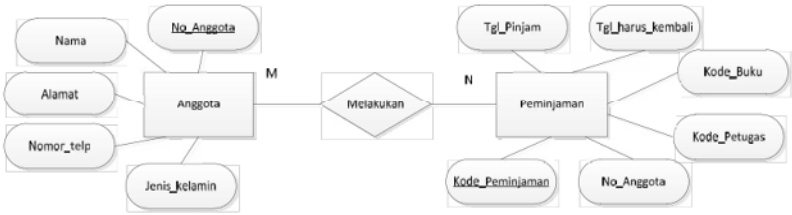
Dari gambaran cerita sistem, tandai dan tentukan setiap hubungan yang diwakili oleh kata kerja yang ada di dalam

ilustrasi dan entitas yang berhubungan. Identifikasi hubungan antara entitas

Tabel Entitas dan Relasi

Entitas 1	Hubungan	Entitras 2
Anggota	Melakukan	Peminjaman
Pegawai	Mengelola	Peminjaman
Pengarang	Menulis	Buku
Penerbit	Menerbitkan	Buku
Anggota	Melakukan	Pengembalian
Pegawai	Mengelola	Pengembalian

Dari tabel di atas maka berikut relasinya :



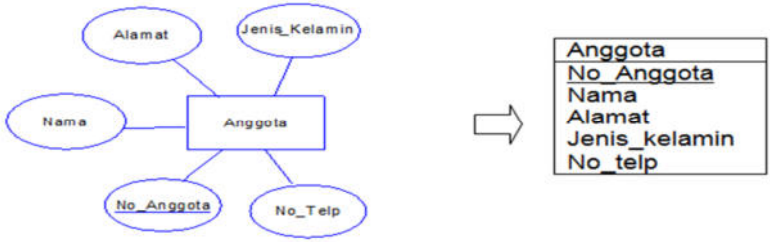
Gambar 5.6 Notasi hubungan entitas dan relasi

Langkah 2.

Mapping ER diagram ke tabel. Didalam data base yang menjadi pusat perhatian dan intisari dari sistem database itu adalah table dan relasinya. Tabel ini sama artinya dengan entitas pada model data pada level konseptual. Setiap orang bisa membuat table tetapi membuat table yang baik tidak semua orang dapat melakukannya. Kebutuhan akan membuat tabel yang baik ini

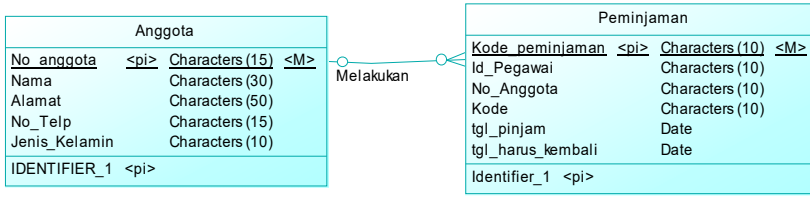
maka muncul teori beberapa teori atau metode yaitu mapping ERD to table.

Contoh:
Dari relasi yang telah dibuat pada sistem informasi perpustakaan, pada setiap entitasnya pilih satu atribut kunci sebagai primary key (atribut harus uniqe).



Gambar 5.7. Maping Notasi ke Diagram ER

Dengan cara yang sama dapat dilakukan mapping ERD to table pada semua entitas. Setelah semua entitas selesai dibuat, tentukan relasi antar entitas sesuai dengan tabel hubungan antar entitas di atas. Misalnya relasi antara entitas anggota dan Peminjamanya adalah melakukan



Gambar 5.8 Entity Relationship Diagram

EVALUASI

1. Jelaskan secara singkat tentang ERD yang kalian ketahui!
2. Tugas identifikasi entitas dan atribut
   Lakukan observasi ke perusahaan yang ada di sekitar anda, terkait kebutuhan sistem basis data, susun narasi gambaran sistem dari perusahaan tersebut serta identifikasi terkait :
   a. Entitas
   b. Atribut
   c. Relasi antar entitas
   d. Gambaran ERD

Bab VI
KONSEP NORMALISASI

Setelah mempelajari materi ini, diharapkan mahasiswa mampu:
1. Memahami konsep normalisasi
2. Menjelaskan aturan normalisasi
3. Menerapkan proses normalisasi
4. Merancang bentuk normal pertama (1NF)
5. Merancang bentuk normal tahap kedua (2NF)
6. Merancang bentuk normal tahap ketiga (3NF)

1. Konsep Normalisasi

Normalisasi diartikan sebagai suatu teknik yang menstrukturkan atau mendekomposisi atau memecah data menggunakan cara–cara tertentu untuk mencegah timbulnya permasalahan pengolahan data dalam basis data. Permasalahan yang dimaksud adalah berkaitan dengan penyimpangan–penyimpangan (*anomalies*) yang terjadi akibat adanya kerangkapan data dalam relasi dan inefisiensi pengolahan. Anomali merupakan proses pada basis data yang memberikan efek samping yang tidak diharapkan (misalnya terjadinya redudansi). Bila ada anomali maka relasi mungkin perlu dipecah menjadi beberapa tabel lagi agar diperoleh database yang optimal.

Proses normalisasi dilakukan jika terjadi dependensi (ketergantungan). Dependensi menjelaskan nilai sebuah atribut yang menentukan nilai atribut lainnya. Terdapat beberapa jenis dependensi, diantaranya dependensi fungsional dan dependensi transitif.

a. Dependensi Fungsional, suatu atribut Y mempunyai dependensi fungsional terhadap atribut X jika dan hanya jika setiap nilai nilai X berhubungan dengan sebuah nilai Y ($X \rightarrow Y$).

b. Dependensi Transitif, atribut Z mempunyai dependensi transitif terhadap X jika atribut Y memiliki dependensi fungsional terhadap X dan jika atribut Y memliki dependesi fungsional terhadap Y.

Proses normalisasi akan menghasilkan relasi yang optimal, yaitu :

a. Memiliki struktur *record* yang mudah untuk dimengerti.
b. Memiliki struktur *record* yang sederhana dalam pemeliharaan.
c. Memiliki struktur *record* yang mudah untuk ditampilkan kembali untuk memenuhi kebutuhan pemakai.
d. Minimalisasi kerangkapan data guna meningkatkan kinerja system

Beberapa konsep yang harus dipahami sebelum mengimplementasikan teknik normalisasi data antara lain ialah: 1) ketergantungan fungsional. 2) Domain dan tipe data. 3) Konsep key atribut (Field/atribute kunci).

2. Aturan Normalisasi

Sebuah basis data dapat dikatakan baik, jika setiap tabel yang menjadi unsur pembentuk basis data tersebut juga telah berada dalam keadaan baik atau normal. Selanjutnya, sebuah tabel dapat dikategorikan baik (*efisien*) atau normal, jika telah memenuhi 3 (tiga) kriteria berikut :

a. Jika ada *dekomposisi* (penguraian) tabel, maka dekomposisinya harus dijamin aman (*Lossless–Join Decomposition*).
b. Terpeliharanya ketergantungan fungsional pada saat perubahan data (*DependencyPreservation*).
c. Tidak melanggar *Boyce–Code Normal Form* (BCNF)

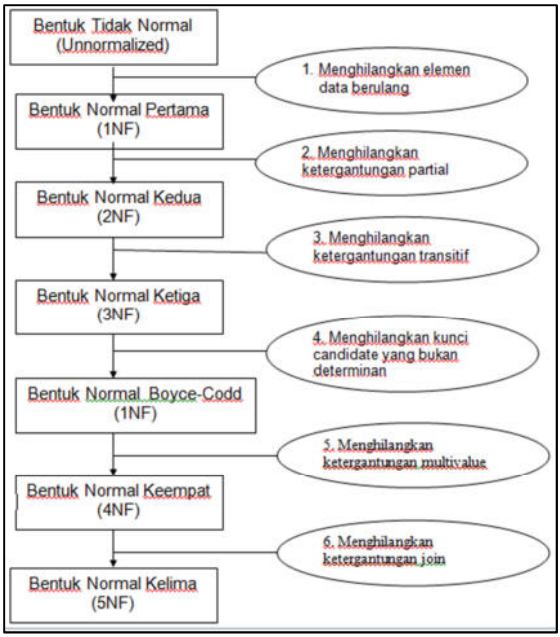
3. Proses Normalisasi

Hasil dari proses normalisasi adalah tabel–tabel data dalam bentuk normal (*normal form*), yaitu tabel–tabel data yang terhindar dari dua hal yaitu: Pengulangan informasi dan Potensi *inkonsistensi* data pada operasi pengubahan. Terdapat enam bentuk normal (normal form) dalam teknik normalisasi data, keenam bentuk tersebut adalah :

a. Bentuk Normal Tahap pertama (1^{st} NF)
b. Bentuk Normal Tahap Kedua (2^{nd} NF)
c. Bentuk Normal Tahap Ketiga (3^{rd} NF)
d. Bentuk Normal Boyce - Code (BCNF)
e. Bentuk Normal Tahap Keempat (4^{rd} NF)
f. Bentuk Normal Tahap Kelima (5^{rd} NF)

Dalam proses normalisasi, data diuraikan dalam bentuk tabel, selanjutnya dianalisis berdasarkan persyaratan tertentu ke beberapa tingkat. Apabila tabel yang diuji belum memenuhi persyaratan tertentu, maka tabel tersebut perlu dipecah menjadi beberapa tabel yang lebih sederhana sampai memenuhi bentuk yang optimal.

Langkah yang dilakukan untuk melakukan normalisasi ditunjukkan pada Gambar 6.1



Gambar 6.1 Langkah Proses Normalisasi Data

4. Merancang bentuk normal pertama (1NF)

Bentuk normal ke satu (1NF) dilakukan penghapusan beberap grup elemen yang berulang agar tidak terjadi redudansi atau agar menjadi satu nilai tunggal yang berinteraksi di antara setiap baris pada tabel. Pada 1NF Setiap data disajikan dalam bentuk flat file (tabular/ tabel), seluruh atribut kunci terdefinisikan, tidak ada pengulangan group pada table, dan semua atribut bergantung pada Primary Key.

Bentuk normal ke satu 1 NF ini mempunyai beberapa ciri antara lain yaitu:

a. Setiap data dibentuk dalam flat file (file data/ rata)

b. Data dibentuk dalam satu record demi satu record dan nilai dari field field berupa "atomic value", tidak dapat dibagi-bagi lagi.
c. Tidak ada set atribute yang berulang ulang atau atribute bernilai ganda (multivalue).
d. Tidak ada set atribut *composite* atau kombinasinya dalam domain data yang sama.
e. Tiap field hanya satu pengertian, bukan merupakan kumpulan kata yang mempunyai arti mendua, hanya satu arti saja dan juga bukanlah pecahan kata sehingga artinya lain.

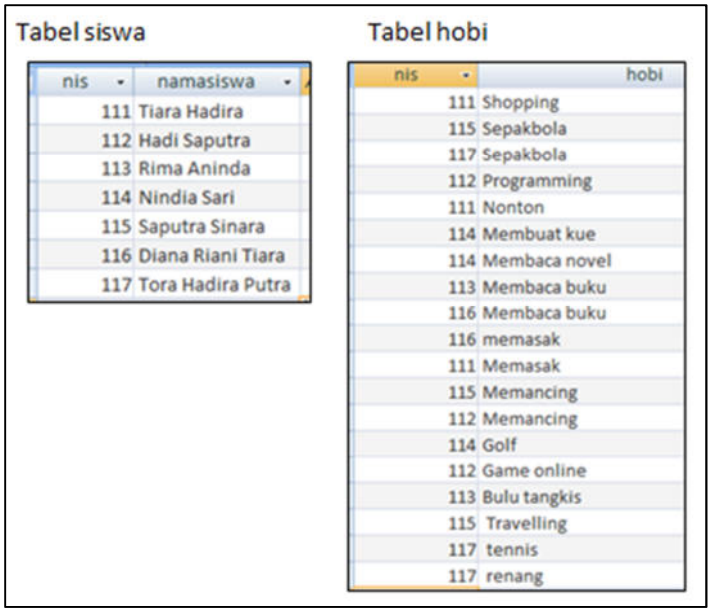
Contoh table yang belum memenuhi bentuk 1NF:

nis	namasiswa	hobi
111	Tiara Hadira	Memasak, Nonton, shopping
112	Hadi Saputra	Programming, game online, Memancing
113	Rima Aninda	Membaca buku, bulu tangkis
114	Nindia Sari	Membuat kue, membaca novel, golf
115	Saputra Sinara	Memancing, travelling, sepakbola
116	Diana Riani Tiara	Membaca buku, memasak, bowling
117	Tora Hadira Putra	Sepakbola, tennis, renang

data pada tabel yang belum memenuhi 1NF, bentuk lain dapat dijabarkan seperti berikut:

nis	namasiswa	hobi1	hobi2	hobi3
111	Tiara Hadira	Memasak	Nonton	shopping
112	Hadi Saputra	Programming komputer	game online	Memancing
113	Rima Aninda	Membaca buku	bulu tangkis	
114	Nindia Sari	Membuat kue	membaca novel	golf
115	Saputra Sinara	Memancing	travelling	sepakbola
116	Diana Riani Tiara	Membaca buku	memasak	bowling
117	Tora Hadira Putra	Sepakbola	tennis	renang

Untuk dapat memenuhi aturan bentuk 1NF, dilakukan dekomposisi menjadi 2 entitas, yakni tabel siswa dan tabel hobi seperti berikut :



Bentuk Normal Pertama (1NF) :

a. Setiap data disajikan dalam bentuk flat file (tabular/ tabel)
b. Seluruh atribut kunci terdefinisikan
c. Tidak ada pengulangan group pada tabel
d. Semua atribut bergantung pada Primary Key

5. Merancang bentuk normal tahap kedua (2NF)

Pada bentuk normal ke dua (2NF) syarat yang harus dipenuhi adalah:

a. Bentuk data telah memenuhi kriteria bentuk normal kesatu.
b. Atribute bukan kunci haruslah bergantung secara fungsi pada kunci utama atau primary key.
c. Sudah ditentukan kunci kunci field, dimana kunci field haruslah unik dan dapat mewakili atribute lain yang menjadi anggotanya

Sebagai contoh ditentukan sebuah tabel siswa sebagai berikut :

NIS	Nama_siswa	Alamat	Kode_ Mapel	Nama_Mapel	Nama_Guru	Nilai

Tabel di atas telah memenuhi 1NF, namun belum memenuhi 2NF. Tabel di atas perlu didekomposisi menjadi beberapa tabel untuk memenuhi syarat 2NF. sebagai berikut :

Tabel Nilai : (NIS, Kode_mapel, Nilai)

Tabel Siswa : (NIS, Nama_siswa, Alamat)

Tabel Mapel : (Kode_mapel, Nama_mapel, Nama_Guru)

6. Merancang bentuk normal tahap ketiga (3NF)

Untuk menjadi bentuk normal ketiga (3 NF) suatu tabel harus mempunyai ciri-ciri sebagai berikut:

a. Memenuhi bentuk 2 NF (normal kedua)
b. Atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci utama atau primary key.
c. Setiap atribute bukan kunci haruslah bergantung hanya pada primary key dan pada primary key secara menyeluruh

Berikut ini adalah contoh relasi yang telah memenuhi bentuk 2 NF, tetapi belum memenuhi bentuk 3 NF :

NIS	Nama_siswa	Alamat_jln	Alamat_kota	Alamat_prov	Kodepos

Pada relasi di atas, masih terdapat atribut non primary key (yakni Alamat_kota dan Alamat_Prov) yang memiliki ketergantungan terhadap atribut non primary key yang lain, yaitu Kode_pos.

Kodepos : {Alamat_kota, Alamat_prov}

Untuk memenuhi syarat 3NF, maka relasi tersebut harus didekomposisi sebagai berikut :

Siswa : (NIS, Nama_siswa, Alamat_jn, Kodepos)

Kodepos : (Kodepos, Alamat_kota, Alamat_prov)

7. Bentuk Normal Boyce - Code (BCNF)

BCNF merupakan kasus khusus 3NF, table atau relasi table berada dalam bentuk BCNF jika:

a. Setiap penentu (determinan) pada tabel adalah sebuah kunci kandidat (candidate key)
b. Jika tabel hanya mengandung satu kunci kandidat maka bentuk 3NF sama dengan BCNF.

8. Bentuk Normal Tahap Keempat (4rd NF)

Tabel pada basis data berada dalam bentuk 4NF jika dan hanya jika telah berada dalam bentuk 3NF dan setiap sebuah ketergantungan multi nilai yang rumit X→→Y, dimana X merupakan superkey yang berarti X merupakan candidate key.

9. Bentuk Normal Tahap Kelima (5rd NF)

Table pada basis data berada dalam bentuk 5NF jika, telah berada dalam bentuk 4NF dan setiap ketergantungan join berhubungan dengan candidate key secara tidak langsung.

EVALUASI

Perhatikan table di bawah ini :

NO_PROY	NAMA_PROY	NRP	NAMA_PEG	KEAHLIAN	UPAH_HARI	HARI_KERJA
12	Train A	105	Sambudi	Mandor	50000	25
		107	Tantowi	Tkg. Las	40000	24
		108	Subardi	Tkg. Las	40000	25
		110	Ferry	Tkg. Listrik	40000	20
		115	Triyadi	Buruh kasar	25000	21
		116	Wagino	Buruh kasar	25000	24
16	Train C	102	Surono	Mandor	50000	25
		103	Reynaldi	Tkg. Las	40000	22
		109	Rachmat	Tkg. Las	40000	25
		112	Syamsul	Tkg. Listrik	40000	22
		114	Ngadiyo	Buruh kasar	25000	25
		120	Tugino	Buruh kasar	25000	25
		122	Wawan	Buruh kasar	25000	25

Buatlah bentuk 1NF, 2NF, dan 3 NF (jika memungkinkan) dari table tersebut dan buat relasinya.

Bab VII
PERINTAH SQL: Data Definition Language (DDL)

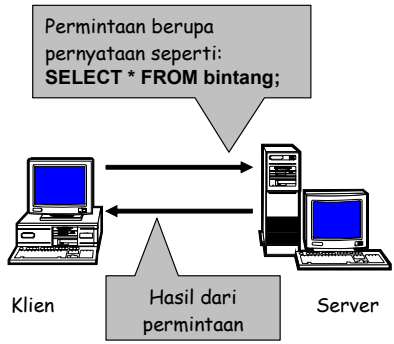
Structure Query Language (SQL) telah menjadi bagian dari arsitektur aplikasi. SQL DDL memungkinkan objek basis data, seperti skema, domain, table, view, dan index untuk dibuatkan dan dihapuskan.

Setelah mempelajari materi ini, diharapkan mahasiswa mampu:

1. Menjelaskan definisi SQL
2. Menjelaskan type data SQL
3. Menerapkan pembuatan basis data
4. Menerapkan pembuatan table dengan query
5. Menerapkan relasi antar table dengan query
6. Memodifikasi table dengan query

1 Menjelaskan definisi SQL

SQL *(Structured Query Language)* adalah sebuah bahasa yang digunakan untuk mengakses data dalam software DBMS. Bahasa ini merupakan bahasa standar yang digunakan dalam manajemen basis data relasional. Saat ini hampir semua server basis data mendukung bahasa ini untuk melakukan pengelolaan datanya. Untuk melakukan pengaksesan data, digunakan MySQL. MySQL merupakan basis data server yang bersifat open source, multiplatform, dan berbasis relasional. Ilustrasi penggunaan SQL ditunjukkan pada Gambar 7.1



Gambar 7.1 Ilustrasi proses akses data di SQL

Instruksi – instruksi atau pernyataan SQL dapat dikelompokkan menjadi DDL (data definition language) dan DML (data manipulation language).

a. DDL (*Data Definition Language*)

DDL adalah sebuah metode Query SQL yang berguna untuk mendefinisikan struktur data pada sebuah basis data, Query yang dimiliki DDL adalah :

CREATE	:	Membuat Database dan Tabel
Drop	:	Menghapus Tabel dan Database
Alter	:	Melakukan perubahan struktur tabel yang telah dibuat, baik menambah Field (Add), mengganti nama Field (Change) ataupun menamakannya kembali (Rename)

2 Menjelaskan type data SQL

Tipe data adalah suatu bentuk pemodelan data yang

dideklarasikan pada saat melakukan pembuatan tabel. Tipe data ini akan mempengaruhi setiap data yang akan dimasukkan ke dalam sebuah tabel. Data yang akan dimasukkan harus sesuai dengan tipe data yang dideklarasikan. Berbagai type data dapat dilihat pada Tabel berikut.

Tabel 7.1. Type Data untuk Bilangan (Number)

Type Data	Keterangan
TINYINT	Ukuran 1 byte. Bilangan bulat terkecil, dengan jangkauan untuk bilangan bertanda: -128 sampai dengan 127 dan untuk yang tidak bertanda : 0 s/d 255.
SMALLINT	Ukuran 2 Byte. Bilangan bulat dengan jangkauan untuk bilangan bertanda : -32768 s/d 32767 dan untuk yang tidak bertanda : 0 s/d 65535
MEDIUMINT	Ukuran 3 byte. Bilangan bulat dengan jangkauan untuk bilangan bertanda : -8388608 s/ d 8388607 dan untuk yang tidak bertanda : 0 s/d 16777215
INT	Ukuran 4 byte. Bilangan bulat dengan jangkauan untuk bilangan bertanda : -2147483648 s/d 2147483647 dan untuk yang tidak bertanda : 0 s/d 4294967295
BIGINT	Ukuran 8 byte. Bilangan bulat terbesar dengan jangkauan untuk bilangan bertanda : -9223372036854775808 s/d 9223372036854775807 dan untuk yang tidak bertanda : 0 s/d 18446744073709551615
FLOAT	Ukuran 4 byte. Bilangan pecahan
DOUBLE	Ukuran 8 byte. Bilangan pecahan
REAL	Ukuran 8 byte. Sinonim dari DOUBLE

DECIMAL (M,D)	Ukuran M byte. Bilangan pecahan, misalnya DECIMAL(5,2 dapat digunakan untuk menyimpan bilangan -99,99 s/d 99,99

Table 7.2. Type Data untuk Waktu

Type Data	Keterangan
DATETIME	Ukuran 8 byte. Kombinasi tanggal dan jam, dengan jangkauan dari '1000-01-01 00:00:00' s/d '9999-12-31 23:59:59'
DATE	Ukuran 3 Byte. Tanggal dengan jangkauan dari '1000-01-01' s/d '9999-12-31'
TIMESTAMP	Ukuran 4 byte. Kombinasi tanggal dan jam, dengan jangkauan dari '1970-01-01 00:00:00' s/d '2037'
TIME	Ukuran 3 byte. Waktu dengan jangkauan dari '839:59:59' s/d '838:59:59'
YEAR	Ukuran 1 byte. Data tahun antara 1901 s/d 2155

Table 7.3. Type Data untuk Karakter dan Lain-lain

Type Data	Keterangan
CHAR	Mampu menangani data hingga 255 karakter.
VARCHAR	Mampu menangani data hingga 255 karakter. Tipe data VARCHAR tidak mengharuskan untuk memasukkan data yang telah ditentukan oleh kita.
TINYBLOB, TINYTEXT	Ukuran 255 byte. Mampu menangani data sampai 2^8-1 data.
ENUM('nilai1','nilai2',…,'nilaiN')	Ukuran 1 atau 2 byte. Tergantung jumlah nilai enumerasinya (maksimum 65535 nilai)

3 Menciptakan Basis Data

Database adalah sebuah media utama yang harus dibuat dalam membangun sebuah basis data agar nantinya dapat kita letakkan beberapa tabel dengan field-fieldnya. MySQL adalah database berbasis relasional, struktur data diatur melalui pembuatan table-table yang saling berkaitan (mempunyai relasi).

Tiga elemen yang merupakan model fundamental dari relasi adalah:

c. Struktur Data (Table)

Terdiri dari baris (*row* atau *record*) dan setiap baris terdiri dari kolom-kolom (*column* atau *field*) yang terdefinisi melalui tipe data pada kolom tersebut.

d. Integritas Data

Isi data sesuai kondisi sebenarnya, misalkan field "tinggi_badan" tidak boleh negative, "jenis_kelamin" hanya mempunya nilai 'L' dan 'P'. Kesesuaian data dengan nilai sebenarnya ini disebut juga "batasan nilai untuk integritas data" atau "integrity constraints".

e. Manipulasi Data

Data yang tersimpan dapat dimanipulasi dengan bahasa query seperti SQL.

Perintah yang digunakan untuk menciptakan database pada MySQL dengan Syntax berikut :

create database nama_database;

Contoh :

```
1 create database Perpus;
```

```
create database Perpus;
/* Affected rows: 1  Baris ditemukan: 0  Peringatan: 0  Durasi untuk 1 query: 0,015 sec. */
```

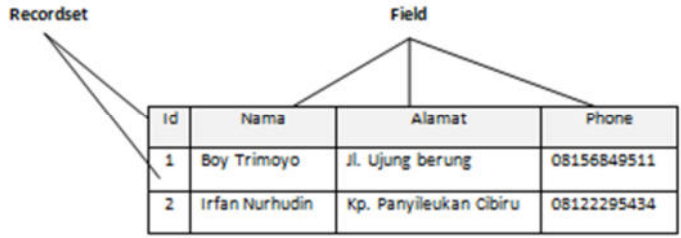
Pada contoh diatas, query OK menyatakan bahwa pembuatan database dengan nama perpus berhasil dibuat.

4 Membuat Desain Tabel dengan Query

Suatu file database (*.mdb, *.accdb) terdiri dari satu atau lebih table, index dan komponen lain. Sedangkan dalam satu tabel bisa terdiri dari satu atau lebih record data masing-masing berisi informasi yang sejenis. Dalam tabel terdapat baris dan kolom. Baris diistilahkan dengan recordset dan kolom dengan field.

f. Membuat Tabel

Tabel merupakan komponen utama pembentuk database (basis data). Tabel terletak pada sebuah database, sehingga pembuatan tabel dilakukan setelah sebuah database telah dibuat. Dalam tabel terdapat bari dan kolom. Seperti yang telah dijelaskan pada pembehasan sebelumnya baris diistilahkan dengan recordset dan kolom dengan field. Ilustrasi ditunjukkan pada Gambar 7.2



Gambar 7.2 Ilustrasi baris dan kolom

Perintah yang digunakan untuk menciptakan tabel dengan syntax query berikut :

```
CREATE TABLE nama_tabel
(
      field1   tipe-data(length),
      field2   tipe-data(length),
      …
      fieldn   tipe-data(length)
)
```

Keterangan :

Komponen	: Keterangan
Tabel	: Nama dari tabel yang akan dibuat.
Field1, Field2	: Nama dari masing-masing field yang akan digunakan pada tabel yang baru dibuat. Anda harus membuat minimal satu field.
Tipe_data	: Tipe data dari field yang digunakan pada tabel baru.
length	: Ukuran dari field dalam karakter. Digunakan hanya untuk tipe data Text.

Contoh :

Membuat table dengan nama Tabel petugas

Field : Id_petugas, Nama, Username, dan password.

Maka Query yang dibuat untuk membentuk Tabel petugas adalah sebagai berikut:

```
create table petugas (
Id_petugas varchar(20),
Nama varchar(50),
Username varchar(15),
Password varchar (15)
);
```

Beberapa elemen umum yang harus ditentukan dalam membuat sebuah tabel:

a. Nama dari tabel harus Unique untuk setiap file database, tidak diperkenankan dalam satu folder terdapat lebih dari satu nama file database yang sama.
b. Nama dari field (kolom) harus bersifat Unique untuk setiap tabel (tidak boleh sama).
c. Tipe data dan ukuran masing-masing field (kolom) harus disesuaikan dengan kondisi data yang akan disimpan.
d. Pemakaian Constraint yang diikutkan dalam pembentukan suatu tabel, terdiri dari Null, Not Null, Primary Key, Unique dan Foreign Key atau gabungan beberapa Constraint yang ada.

Primary key atau ***unique key*** adalah suatu nilai di basis data yang digunakan untuk mengidentifikasi keunikan baris-baris di dalam table. Dalam membuat sebuah database, kita akan menemukan sebuah record yang data nya tidak boleh sama dengan record yang lain. Agar data tidak terjadi redudansi data maka harus membuat sebuah kolom yang di deklarasikan sebagai kunci primer (*primary key*).

Syntax untuk menciptakan *primary key* ada beberapa cara, yaitu :

```
CREATE TABLE nama_tabel
(
     field1  tipe-data(length)PRIMARY KEY,
     field2  tipe-data(length),
     …
     fieldn  tipe-data(length)
)
```

Contoh :

Buat table Anggota dengan menggunakan query dari Table 4

Tabel 7.4. Tabel Anggota

Field	Type	Length	Keterangan
NIS	Varchar	15	Primary key
Nama	Varchar	50	
Prodi	Varchar	15	
Jk	Varchar	10	
alamat	Varchar	50	

Maka query yang digunakan untuk membentuk table anggota adalah :

```
create table anggota (
NIS varchar(15) Primary key ,
Nama varchar(50),
prodi varchar(15),
Jk varchar(10),
alamat varchar(50)
);
```

Jika table berhasi dibuat maka akan muncul pesan seperti script berikut:

```
create table anggota (  NIS varchar(15) Primary key ,  Nama varchar(50),  prodi varchar(15),  Jk varchar(10),  alamat varchar(50)  );
/* Affected rows: 0  Baris ditemukan: 0  Peringatan: 0  Durasi untuk 1 query: 0,514 sec. */
```

Altrernatif lain penulisan Constraint dengan ***Primary Key*** NIS dimana constraintnya diletakkan pada bagian terakhir:

```
Create Table anggota (
NIS Varchar (15),
Nama Varchar (50),
Prodi Varchar (15),
Jk Varchar (10),
Alamat Varchar (50),
Constraint PK_Anggota Primary Key (NIS) );
```

jika ingin menambahkan ***Primary Key*** pada tabel yang sudah terbentuk tetapi belum mempunyai Primary Key, format penulisannya seperti berikut:

```
Alter Table anggota
 Add Constraint PK_Anggota Primary Key (NIS)
```

g. Menghapus Tabel

Menghapus data adalah menghilangkan satu atau beberapa record data dari suatu tabel. Perintah query yang digunakan untuk menghapus adalah Delete, hanya dapat digunakan untuk menghapus record (baris) dan idak dapat digunakan untuk menghapus field (kolom). Untuk menentukan record yang akan dihapus dapat dilakukan perintah "Where". Jika tidak menggunakan perintah ini maka seluruh record yang ada pada tabel yang bersangkutan akan terhapus semua.

Untuk menghapus Tabel yang telah dibuat dapat menggunakan query berikut :

DROP TABLE nama_tabel;

Contoh :

Perintah untuk menghapus table petugas : **Drop table petugas;**

h. Relasi Antar Tabel dengan Query

Relasi tabel adalah hubungan sebuah tabel dengan tabel lainnya. Sehingga tabel tidak lagi berdiri sendiri, melainkan dapat dihubungkan antara satu dengan yang lainnya dan menjadi satu kesatuan. Terdapat dua buah kolom yang diperlukan untuk menghubungkan sebuah tabel dengan tabel lainnya.

1) Kolom yang pertama, yaitu kolom **primary key** (kunci utama) pada tabel yang satu.

2) Kolom yang kedua adalah **foreign key** (kunci asing) pada tabel lainnya.

Foreign Key adalah kolom atau field pada suatu tabel yang berfungsi sebagai kunci tamu dari tabel lain. **Foreign Key** sangat berguna bila kita bekerja dengan banyak tabel yang saling berelasi satu sama lain. Contoh dari Foreign key adalah sebagai berikut.

Setelah membuat sebuah hubungan kita wajib menentukan aturan yang dikenakan padanya. Aturan dasar yang telah baku adalah :

1) **Field yang dihubungkan** dari tabel utama **haruslah** bersesuaian berupa sebuah **Primary Key** dan **Foreign Key**.
2) Kedua field yang saling terhubung harus memiliki **jenis data yang sama**.

Sebuah tabel hanya boleh memiliki satu buah primary key (kunci utama). Namun, sebuah tabel boleh memiliki lebih dari satu buah foreign key (kunci asing). Oleh karena itu, pilihlah satu buah kolom pada tabel yang akan dijadikan primary key yang dapat mewakili kolom lainnya dan nilainya pun unik, misalnya kolom NIM, kode_buku, dan lainnya.

Sintaks relasi antar tabel dengan query:

```
CREATE TABLE nama_tabel (
  field-1 type(length) PRIMARY KEY,
  field-2 type(length),➔ kolom yang akan direlasikan
  field-3 type(length),
  …… ….(….)
  foreign key (field-2) references table_yg_direlasikan
  (primary_key_pd_table_yg_direlasikan)  );
```

Contoh :

Database Perpustakaan terdiri dari 3 tabel : table kategori, table penerbit, dan table buku

Tabel Kategori

Field	Type	Length	Keterangan
Kode_kategori	Varchar	20	Primary key
Nama_kategori	Varchar	20	

Tabel Penerbit

Field	Type	Length	Keterangan
kode_penerbit	Varchar	20	Primary key
Nama_penerbit	Varchar	25	
Alamat_penerbit	Varchar	25	

Tabel Buku

Field	Type	Length	Keterangan
kode_buku	Varchar	20	Primary key
kode_kategori	**varchar**	20	Foreign key
kode_penerbit	**varchar**	20	Foreign key
judul	**varchar**	50	
jumhal	**Int**		
tahun_terbit	**year**		

Pembuatan **table** dan **relasi tabel** dengan query, langkah awal yang harus dilakukan adalah dengan menentukan kolom yang uniq (*primary key*), langkah selanjutnya menghubungkan atau merelasikan primary key dan foreign key pada table yang akan direlasikan.

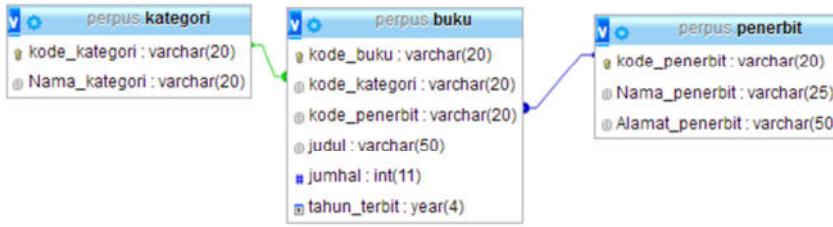
Mengacu pada table table kategori, table penerbit, dan table buku, contoh query yang digunakan untuk merelasikan tiga table tersebut adalah :

```
create table kategori (
kode_kategori varchar (20) primary key,
Nama_kategori varchar(20)
);

create table penerbit(
kode_penerbit varchar(20) primary key,
Nama_penerbit varchar(25),
Alamat_penerbit varchar(50)
);

create table buku(
kode_buku varchar(20) primary key,
kode_kategori varchar (20),
kode_penerbit varchar(20),
judul varchar(50),
jumhal int,
tahun_terbit year,
foreign key (kode_kategori) references kategori (kode_kategori),
foreign key (kode_penerbit) references penerbit (kode_penerbit)
);
```

Hasil relasi table yang telah dibuat:



i. Memodifikasi Tabel

Perubahan tabel yang telah dibuat akan selalu dilakukan mengingat perkembangan database, termasuk diantaranya menambahkan beberapa field pada tabel, mengganti nama field maupun table.

a) Mengganti Nama Table

Query SQL untuk merubah nama tabel dengan menggunakan **RENAME**, sintax seperti berikut :

RENAME TABLE tabel_lama **TO** tabel_baru;

Contoh : merubah nama table petugas menjadi admin

```
rename table petugas to admin;
```

b) Menambah field pada table

Menambah kolom dapat diartikan sebagai langkah untuk menyisipkan field baru pada sebuah tabel. Untuk melakukan penambahan Field maka **ALTER** spesifikasi yang digunakan adalah **ADD**. Sintax yang digunakan adalah:

ALTER TABLE nama_tabel **ADD** nama_field Type_data(length);

Contoh : menambahkan kolom Jenis kelamin pada table admin

```
ALTER TABLE admin ADD Jk varchar(10);
```

c) Menghapus field pada table

Pada pembuatan database pasti terdapat kesalahan seperti pada field tabel yang berlebihan dan lain-lain. Untuk melakukan Penghapusan Field maka ALTER spesifikasi yang digunakan adalah **DROP**. Sintax yang digunakan adalah :

ALTER TABLE nama_tabel **DROP** nama_field;

Contoh : menghapus kolom Jk pada table admin

```
ALTER TABLE admin DROP Jk;
```

Beberapa ketentuan yang harus diperhatikan dalam melakukan penghapusan tabel:

- Sebelum tabel dihapus, Anda harus menutup terlebih dahulu tabel tersebut.

- Suatu tabel yang telah dihapus tidak dapat dikembalikan seperti semula.
- Data record yang ada pada tabel yang dihapus juga akan terhapus.
- Tidak ada pesan/persetujuan terlebih dahulu selama proses penghapusan tabel dilaksanakan.

EVALUASI

1. Susun DDL untuk membuat table **mahasiswa** dengan spesifikasi sebagai berikut

Field	Type	Keterangan
Nim	Char(20)	Primary Key, panjang harus tepat 20 karakter
Nama	Varchar(30)	Tidak boleh kosong
JK	Char(1)	Jenis Kelamin, hanya mempunyai nilai L untuk laki-laki dan P untuk perempuan
GolDarah	Char(2)	Golongan Darah: A, B, AB, O
Ipk	Numeric	Interval 0 sampai 4

2. Buat table peminjaman dan tabel pengembalian untuk database perpustakaan dan relasikan antar table.

Bab VIII
PERINTAH SQL: Data Manipulation Language (DML)

Pada bab ini membahas mengenai sintaks dalam SQL Data manipulation Language dan beberapa contoh detail.

Setelah mempelajari materi ini, diharapkan mahasiswa mampu:

1. Menjelaskan pengertian DML
2. Menerapkan perintah insert dalam perancangan basis data
3. Menerapkan perintah select dalam perancangan basis data
4. Menerapkan perintah update dalam perancangan basis data
5. Menerapkan perintah delete dalam perancangan basis data

1. Pengertian DML

DML adalah sebuah metode Query yang dapat digunakan apabila DDL telah terjadi, sehingga fungsi dari Query DML ini untuk melakukan pemanipulasian database yang telah dibuat. Query yang dimiliki DML adalah :

INSERT	:	Memasukkan data pada Tabel Database
UPDATE	:	Pengubahan terhadap data yang ada pada Tabel Database
DELETE	:	Penghapusan data pada tabel Database

2. Perintah insert dalam perancangan basis data

Memasukkan data atau entry data, dalam semua program yang menggunakan query SQL sebagai standar permintaannya, digunakan perintah INSERT. Syarat untuk memasukkan data adalah

telah terciptanya tabel pada sebuah database. Sintax yang digunakan adalah :

INSERT INTO nama_tabel **VALUES** (‘isi_field1’, ‘isi_field2’, ‘isi_field3’,….., ‘isi_fieldN’);

Contoh :

```
INSERT INTO admin  VALUES ('p01', 'Edward', 'admin','admin');
```

Atau bisa dengan menyertakan nama field pada table

```
INSERT INTO admin (Id_petugas,Nama, Username,Password )  VALUES ('p02', 'Jenifer', 'admin','admin');
```

Hasilnya :

Id_petugas	Nama	Username	Password
p01	Edward	admin	admin
p02	Jenifer	admin	admin

3. Perintah select dalam perancangan basis data

Menampilkan data adalah hal yang sangat penting karena kita harus melihat dan menyeleksi suatu data dalam table maupun antar table. Untuk Melihat data atau *Selection,* Query yang digunakan adalah **SELECT** yang diikuti beberapa pernyataan khusus berkenaan dengan tabel yang diseleksi. Pemakaian perintah **Select** digunakan hanya untuk melakukan proses pengambilan data, bukan digunakan untuk mengubah data di dalam database.

a) **Menampilkan Data dari Sebuah Tabel**

Untuk menampilkan dari sebuah tabel dapat menggunakan Sintax berikut:

SELECT * FROM nama_tabel;

Query diatas mengartikan bahwa data dari seluruh Field yang terdapat dalam tabel akan ditampilkan.

Contoh :

```
39 select * from admin;
```

admin (4×2)

Id_petugas	Nama	Username	Password
p01	Edward	admin	12345
p02	Amin	admin	admin

Atau

SELECT (Field1, field2, ……, FieldN) **FROM** nama_tabel;

Query diatas mengartikan bahwa data yang akan ditampilkan didalam tabel hanya filed – filed tertentu.

Contoh :

```
37 select Id_petugas, Nama from admin ;
```

admin (2×2)

Id_petugas	Nama
p01	Edward
p02	Amin

b) **Menampilkan Data dengan Perintah Kondisi (Where)**

WHERE yang artinya dimana, untuk menampilkan data menggunakan perintah where (dimana) dapat menggunakan perintah berikut :

SELECT * FROM nama_tabel **WHERE** kondisi

Contoh :

Jika ingin menampilkan data dari admin dengan kriteria nama **Amin** maka sintak yang digunakan adalah:

```
39 select * from admin where Nama='amin';
```

admin (4×1)

Id_petugas	Nama	Username	Password
p02	Amin	admin	admin

c) **Menampilkan Data dengan Perintah Like**

Perintah Like kadang dibutuhkan dalam pembuatan database yaitu dalam menampilkan data tertentu yang hanya berkaitan dengan kata-kata yang diinginkan.

Query yang digunakan adalah :

SELECT * FROM nama_tabel
WHERE Kondisi **LIKE '%**nama_kaitan**%'**;

Contoh :
Jika ingin menampilkan judul buku dari table buku yang mengandung kata joomla.

```
42 SELECT * FROM buku
43    WHERE buku_judul LIKE '%joomla%';
```

buku (9×2)

buku_kode	kategori_kode	penerbit_kode	buku_judul	buku_jumlah	buk
B002	K2	P1	Step by Step Joomla	45	CM
B004	K1	P2	Joomla Powerful Extensions	20	Jo

d) **Menampilkan Data dengan Pengurutan Sorting (Order By)**

Fungsi ini digunakan untuk melakukan pengurutan data, sehingga data dari sebuah atau beberapa tabel dapat tampil berurutan sesuai keinginan.

Pengurutan data terbagi menjadi dua :

- *ASC* (pengurutan dengan Ascending)
- *DESC* (pengurutan dengan Descending)

Sintax yang digunakan adalah :
SELECT * FROM nama_tabel **ORDER BY** kolom Type

Contoh :
Jika ingin menampilkan nama admin dari urutan abjad

```
45 select * from admin order by Nama desc;
```

admin (4×2)

Id_petugas	Nama	Username	Password
p01	Edward	admin	12345
p02	Amin	admin	admin

e) **Menampilkan Data dengan Pengelompokkan Data (Group By)**

Group By adalah fungsi untuk mengelompokkan data dalam sebuah kolom yang ditunjuk. Fungsi ini akan menghasilkan kelompok data dengan menghilangkan data yang sama dalam satu tabel. Maka apabila dalam satu kolom terdapat beberapa data yang sama maka data yang akan ditampilkan hanya salah satu. Sintax yang digunakan seperti berikut :

SELECT * FROM nama_tabel **GROUP BY** nama_kolom;

Contoh :

```
46 SELECT * FROM admin GROUP BY Nama;
```

admin (4×2)

Id_petugas	Nama	Username	Password
p02	Amin	admin	admin
p01	Edward	admin	12345

f) **Menampilkan Data dari Beberapa Tabel**

Perintah **SELECT** mempunyai banyak sekali variasi. Mungkin bisa disebut perintah yang mempunyai variasi paling banyak di antara perintah-perintah lainnya.

Seperti yang kita ketahui bahwa MySql adalah sebuah **RDBMS (Relational Database Management System)** yang artinya bahwa tabel-tabel tersebut **bisa ber-relasi/tidak ber-relasi dengan tabel lainnya**. Perintah **SELECT** adalah untuk menampilkan data yang berada di dalam tabel. Pertanyaannya adalah bagaimana jika ingin melihat isi tabel yang ber-relasi dengan tabel lainnya. Terdapat banyak cara untuk menggabungkan beberapa table yang memiliki relasi pada database. Pada workshop ini akan dibahas salah satu cara, yaitu menggunakan **inner join.**

Dengan inner join, tabel akan digabungkan dua arah, sehingga tidak ada data yang NULL di satu sisi. Sebagai contoh, kita akan menggabungkan/ mengambil data dari tabel buku dan table kategori dimana kita akan menampilkan daftar buku dengan kategori tertentu.

Misalkan isi tabel buku dan kategori adalah sebagai berikut :

perpus.kategori: 3 total baris (approximately)

kode_kategori	Nama_kategori
K01	Komik
K02	Fiksi
K03	Mapel

perpus.buku: 2 total baris (approximately)

kode_buku	kode_kategori	kode_penerbit	judul	jumhal	tahun_terbit
B01	K03	P02	Cisco dan CCNA Jaringan Komputer	557	2014
B02	K03	P01	Database System	349	2015

Jika ingin mengambil data buku dengan kategori tertentu maka menggunakan **Inner Join dengan WHERE.** Penggabungan dengan klausa WHERE memiliki bentuk umum sebagai berikut:

SELECT tabel1.*, tabel2.* FROM tabel1, tabel2
WHERE tabel1.PK=tabel2.FK;

Berikut ini perintah SQL untuk menggabungkan tabel buku dan kriteria:

```
1 select B.kode_buku, K.Nama_kategori, B.judul
2 from buku B, kategori K
3 where B.kode_kategori=K.kode_kategori;
```

Hasil #1 (3×2)

kode_buku	Nama_kategori	judul
B01	Mapel	Cisco dan CCNA Jaringan Komputer
B02	Mapel	Database System

4. Perintah update dalam perancangan basis data

Memperbarui isi data atau update data adalah sebuah proses meremajakan data lama menjadi data yang lebih baru. Namun tidak semua data dalam database yang perlu diremajakan, melainkan sebagian data yang dianggap perlu untuk diremajakan. Query SQL yang digunakan adalah UPDATE.

UPDATE nama_tabel
SET field_1 = 'data_baru', field_2 ='data_baru',
…… , field_N ='data_baru'
where field_uniq='data';

Contoh :
Jika ingin mengubah data petugas dengan id_petugas "P01" maka sintak yang digunakan adalah:

```
UPDATE admin SET Nama ='Edward', Username ='admin', Password = '12345'
where Id_petugas='P01';
```

Id_petugas	Nama	Username	Password
p01	Edward	admin	12345
p02	Jenifer	admin	admin

5. Perintah delete dalam perancangan basis data

Untuk menghapus data, MySQL memiliki query bernama DELETE. Berikut Sintax untuk menghapus semua data yang terdapat pada table.

DELETE FROM nama_tabel;

Sedangkan berikut sintax untuk menghapus data yang diinginkan dari sebuah table.

DELETE FROM nama_tabel **WHERE** kondisi;

Contoh :

Jika akan menghapus data admin yang mempunyai Id_petugas = P02

```
DELETE FROM admin WHERE Id_petugas='P02';
```

Untuk menghapus seluruh record :

```
Delete from nama_table
```

EVALUASI

Perhatikan Tabel propinsi dan Tabel kota berikut :

Tabel Propinsi

kode_prop	nama_prop	jumlah_penduduk
jabar	jawa barat	0
jak	dki jakarta	0
jateng	jawa tengah	0
jatim	jawa timur	0
yog	di yogyakarta	0

Tabel Kota

kode_kota	kode_prop	nama_kota
ban	jatim	banyuwangi
ban	yog	bantul
bks	jabar	bekasi
blr	jateng	blora
byl	jateng	boyolali
kdr	jatim	kediri
kul	yog	kulon progo
mlg	jatim	malang
sby	jatim	surabaya
sle	yog	sleman

Kerjakan soal berikut berdasarkan Tabel Propinsi dan Tabel Kota:

1. Gunakan perintah select untuk menampilkan **nama_propinsi** dari kota **blora** dan **malang**
2. Tambahkan total **jumlah_penduduk** pada **Tabel Propinsi**

Bab IX
ADVANCE SQL

Setelah mempelajari materi ini, diharapkan mahasiswa mampu:
1. Menerapkan fungsi Agregat dalam perancangan basis data
2. Menerapkan Sub Query dalam perancangan basis data
3. Menerapkan View dalam perancangan basis data
4. Menerapkan Trigger dalam perancangan basis data

1. Fungsi Agregat dalam perancangan basis data

Fungsi agregat merupakan fungsi yang menerima koleksi nilai dan mengembalikan nilai tunggal sebagai hasilnya. Menurut Indrajani (2014) Standar ISO mendefinisikan lima fungsi agregat yang ditunjukkan pada Tabel 9.1.

Tabel 9.1 Fungsi Agregat

Fungsi	Deskripsi
COUNT	Mengembalikan jumlah (banyaknya atau kemunculannya) nilai di suatu kolom
SUM	Mengembalikan jumlah (total atau *sum*) nilai di suatu kolom
AVG	Mengembalikan rata-rata (*average*) nilai di suatu kolom
MIN	Mengembalikan nilai terkecil (*minimal*) di suatu kolom
MAX	Mengembalikan nilai terbesar (*maximal*) di suatu kolom

Keyword DISTINCT dapat dimanfaatkan untuk mengeliminasi duplikasi kemunculan data yang sama.

Sintaks *keyword* **DISTINCT** diperlihatkan sebagai berikut:

```
SELECT DISTINCT A1, A2, ..., An
FROM r1, r2, r3, ..., rm
WHERE P
```

Contoh penggunaan fungsi aggregate, perhatikan Tabel 9.2 .

Tabel 9.2 Penggunaan Fungsi Agregat

kode_mk	nama_mk	sks	semester
PTI447	Praktikum Basis Data	1	3
TIK342	Praktikum Basis Data	1	3
PTI333	Basis Data Terdistribusi	3	5
TIK123	Jaringan Komputer	2	5
TIK333	Sistem Operasi	3	5
PTI123	Grafika Komputer	3	5
PTI777	Sistem Informasi	2	3

a) Contoh Penggunaan COUNT(*)

Fungsi Count yang mempunyai bentuk COUNT(field) digunakan untuk menghitung banyaknya data dalam suatu table dari hasil query. Dengan fungsi ini dapat dihitung jumlah data yang diperoleh atas dasar field tertentu atau seluruh field pada query. Penggunaan COUNT untuk mendapatkan jumlah data, maka sintak yang digunakan adalah:

```
1 SELECT COUNT(*) AS jumlah_record
2 FROM matakuliah;
```

Maka hasil yang ditampilkan adalah “7” karena jumlah total mata kuliah yang terdapat pada table adalah sejumlah 7 mata kuliah.

b) Contoh Penggunaan SUM

Fungsi sum mempunyai bentuk SUM(x) digunakan untuk menghasilkan nilai penjumlahan dari suatu data bilangan dimana x adalah nilai numerik atau nama field yang memiliki nilai numeric. Penggunaan SUM untuk mendapatkan jumlah total sks dari table matakuliah, maka sintak yang digunakan adalah:

```
1 SELECT SUM(sks) AS total_sks FROM matakuliah;
```

c) Contoh Penggunaan MIN, MAX, AVG

Penggunaan MIN untuk mendapatkan sks terendah dari table matakuliah, maka sintak yang digunakan adalah:

```
1 SELECT MIN(sks) AS min_sks FROM matakuliah;
```

Penggunaan MAX untuk mendapatkan sks tertinggi dari table matakuliah, maka sintak yang digunakan adalah:

```
1 SELECT MAX(sks) AS min_sks FROM matakuliah;
```

Fungsi AVG yang mempunyai bentuk AVG(x) digunakan untuk menghasilkan nilai rata-rata dari suatu data bilangan. Dimana x adalah nilai numerik atau nama field yang memiliki nilai numerik atau rumus yang menghasilkan nilai numerik. Penggunaan AVG untuk mendapatkan rata-rata sks dari table matakuliah, maka sintak yang digunakan adalah:

```
1 SELECT AVG(sks) AS rata_rata FROM matakuliah;
```

d) Pengelompokan Data

Pengelompokan data berdasarkan semester dari table matakuliah, sintak yang digunakan adalah:

```
SELECT semester, COUNT(semester) AS jumlah
FROM matakuliah
GROUP BY(semester)
```

e) Menyaring Hasil Fungsi Agregat

Penyeleksian data sks dengan jumlah sks lebih besar dari 3 dan dikelompokkan berdasarkan semester dari table matakuliah, sintak yang digunakan adalah:

```
SELECT semester, COUNT(semester) AS jumlah
FROM matakuliah
GROUP BY(semester)
HAVING jumlah>3
```

Aturan-aturan yang berlaku :

a. SUM dan AVG digunakan untuk field numeric, sedangkan COUNT, MIN, dan MAX dapat digunakan untuk field numeric dan non numeric
b. COUNT (*) berfungsi untuk menghitung seluruh baris dalam table walaupun terdapat NULL atau duplikasi.
c. Menggunakan DISTINCT sebelum nama kolom berfungsi untuk menghilangkan duplikasi
d. DISTICT tidak berpengaruh terhadap operasi MIN atau MAX, tetapi berpengaruh pada SUM atau AVG.
e. Fungsi aggregate dapat digunakan dalam SELECT dan HAVING

2. **Sub Query**

Subquery (disebut juga subselect atau nested select/query atau inner- select) adalah query SELECT yang ada di dalam perintah SQL lain— misalnya SELECT, INSERT, UPDATE, atau DELETE. Keberadaan subquery secara nyata mampu

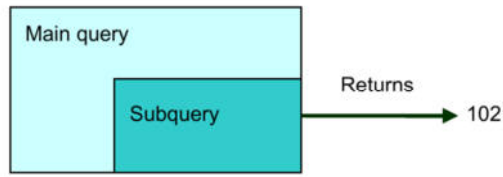
menyederhanakan persoalan- persoalan rumit berkaitan query data.

Sebagai contoh, terdapat pernyataan sebagai berikut: "**Dapatkan data mahasiswa yang alamatnya sama dengan mahasiswa dengan nim 104".** Secara normal, diperlukan dua tahapan untuk menyelesaikan permasalahan tersebut. Langkah awal yang harus dilakukan adalah mendapatkan alamat dari mahasiswa yang memiliki nim 104, selanjutnya dengan menggunakan operator bias diseleksi data mahasiswa yang alamatnya sama dengan mahasiswa yang mempunyai nim 104. Penyelesaian dengan sub query, dapat dilakukan dengan satu tahapan.

Bentuk umum dari sub query adalah :

```
SELECT A1, A2, ..., An
FROM r1, r2, r3, ..., rm
WHERE P
   (SELECT A1, A2, ..., An
   FROM r1, r2, r3, ..., rm
   WHERE P)
```

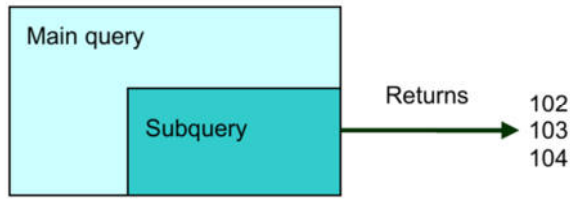
Terdapat 3 jenis sub query diantaranya scalar sub query, multiple-row sub query, dan multiple-column sub query. **Subquery baris tunggal (scalar)** hanya mengembalikan hasil satu baris data. Operator yang digunakan adalah =, >, >=, <, <=, atau <>. Ilustrasi scalar sub query ditunjukkan pada Gambar 9.1



Gambar 9.1 Ilustrasi Scalar Sub Query

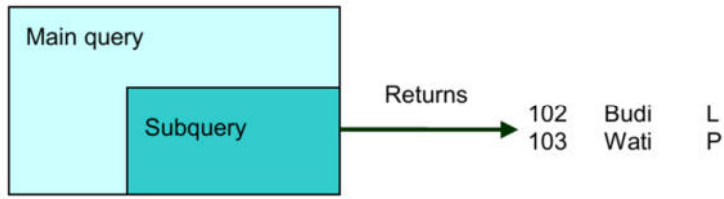
Subquery baris ganda (multiple-row) mengembalikan lebih dari satu baris data. Subquery baris ganda dapat menggunakan

operator komparasi IN, ANY/SOME, atau ALL. Operator IN memiliki arti sama dengan anggota di dalam list. Operator ANY/SOME digunakan untuk membandingkan suatu nilai dengan setiap nilai yang dikembalikan oleh sub query. Operator ALL digunakan untuk membandingkan nilai dengan semua nilai yang dikembalikan oleh query. Ilustrasi multiple-row sub query ditunjukkan pada Gambar 9.2



Gambar 9.2 Ilustrasi multiple-row Sub Query

Subquery kolom ganda (multiple-column) mengembalikan lebih dari satu baris dan satu kolom data. Ilustrasi multiple-column sub query ditunjukkan pada Gambar 9.3



Gambar 9.3 Ilustrasi multiple-column Sub Query

Sebagai contoh, perhatikan table mahasiswa, table ambil_mk, table mata kuliah, table dosen, dan table jurusan sebagai berikut:

Tabel mahasiswa

nim	nama	jenis_kelamin	alamat
101	Arif	L	Jl. Kenangan
102	Budi	L	Jl. Jombang
103	Wati	P	Jl. Surabaya
104	Ika	P	Jl. Jombang
105	Tono	L	Jl. Jakarta
106	Iwan	L	Jl. Bandung
107	Sari	P	Jl. Malang

Tabel ambil_mk

nim	kode_mk
101	PTI447
103	TIK333
104	PTI333
104	PTI777
111	PTI123
123	PTI999

Tabel matakuliah

kode_mk	nama_mk	sks	semester	kode_dos
PTI447	Praktikum Basis Data	1	3	11
TIK342	Praktikum Basis Data	1	3	11
PTI333	Basis Data Terdistribusi	3	5	10
TIK123	Jaringan Komputer	2	5	33
TIK333	Sistem Operasi	3	5	10
PTI123	Grafika Multimedia	3	5	12
PTI777	Sistem Informasi	2	3	99

Tabel dosen

kode_dos	nama_dos	alamat_dos
10	Suharto	Jl. Jombang
11	Martono	Jl. Kalpataru
12	Rahmawati	Jl. Jakarta
13	Bambang	Jl. Bandung
14	Nurul	Jl. Raya Tidar

Tabel jurusan

kode_jur	nama_jur	kode_dos
TE	Teknik Elektro	10
TM	Teknik Mesin	13
TS	Teknik Sipil	23

Contoh penerapan Scalar Sub Query

Untuk mendapatkan data mahasiswa yang alamatnya sama dengan mahasiswa dengan nim 104 bisa digunakan scalar sub query sebagai berikut:

```
SELECT * FROM mahasiswa
WHERE alamat = (SELECT alamat
                FROM mahasiswa
                WHERe nim=104)
```

Jika di eksekusi maka hasil yang ditampilkan adalah :

```
+-----+------+---------------+-------------+
| nim | nama | jenis_kelamin | alamat      |
+-----+------+---------------+-------------+
| 102 | Budi | L             | Jl. Jombang |
| 104 | Ika  | P             | Jl. Jombang |
+-----+------+---------------+-------------+
2 rows in set (0.00 sec)
```

Contoh penerapan Multiple-Row Sub Query

Penggunaan **Operator IN**, sebagai contoh jika ingin mendapatkan data mahasiswa yang mengambil mata kuliah.

```
SELECT nim, nama FROM mahasiswa
WHERE nim IN (SELECT nim FROM ambil_mk)
```

Jika di eksekusi maka hasil yang ditampilkan adalah :

```
+-----+------+
| nim | nama |
+-----+------+
| 101 | Arif |
| 103 | Wati |
| 104 | Ika  |
+-----+------+
3 rows in set (0.00 sec)
```

Penggunaan **Operator ANY/SOME**, sebagai contoh jika ingin mendapatkan data matakuliah yang memiliki sks lebih kecil dari sembarang sks matakuliah di semester 5.

```
SELECT * FROM matakuliah
WHERE sks < ANY
    (SELECT sks FROM matakuliah WHERE semester=5)
```

Jika di eksekusi maka hasil yang ditampilkan adalah :

```
+---------+----------------------+-----+----------+----------+
| kode_mk | nama_mk              | sks | semester | kode_dos |
+---------+----------------------+-----+----------+----------+
| PTI447  | Praktikum Basis Data |   1 |        3 |       11 |
| PTI777  | Sistem Informasi     |   2 |        3 |       99 |
| TIK123  | Jaringan Komputer    |   2 |        5 |       33 |
| TIK342  | Praktikum Basis Data |   1 |        3 |       11 |
+---------+----------------------+-----+----------+----------+
4 rows in set (0.00 sec)
```

Penggunaan operator ALL, sebagai contoh jika ingin mendapatkan data matakuliah yang memiliki sks yang lebih kecil dari semua sks mata kuliah di semester 5

```
SELECT * FROM matakuliah
WHERE sks < ALL
    (SELECT sks FROM matakuliah WHERE semester=5)
```

Jika di eksekusi maka hasil yang ditampilkan adalah :

```
+---------+----------------------+-----+----------+----------+
| kode_mk | nama_mk              | sks | semester | kode_dos |
+---------+----------------------+-----+----------+----------+
| PTI447  | Praktikum Basis Data |   1 |        3 |       11 |
| TIK342  | Praktikum Basis Data |   1 |        3 |       11 |
+---------+----------------------+-----+----------+----------+
2 rows in set (0.16 sec)
```

Contoh penerapan Multiple-column Sub Query

Penggunaan multiple-column sebagai contoh jika ingin menampilkan data mata kuliah yang semester dan sksnya sesuai dengan semester dan sks mata kuliah dengan kode “PTI333”.

```
SELECT * FROM matakuliah
WHERE (semester, sks) IN
              (SELECT semester, sks FROM matakuliah
              WHERE kode_mk='PTI333')
```

Jika di eksekusi maka hasil yang ditampilkan adalah :

kode_mk	nama_mk	sks	semester	kode_dos
PTI123	Grafika Multimedia	3	5	12
PTI333	Basis Data Terdistribusi	3	5	10
TIK333	Sistem Operasi	3	5	10

3 rows in set (0.00 sec)

Operasi pada **sub query** dapat juga dikombinasikan dengan **fungsi agregat,** sebagai contoh jika ingin mendapatkan data matakuliah yang memiliki sks sama dengan sks terbesar.

```
SELECT * FROM matakuliah
WHERE sks = (SELECT MAX (sks) FROM matakuliah)
```

Jika di eksekusi maka hasil yang ditampilkan adalah :

kode_mk	nama_mk	sks	semester	kode_dos
PTI123	Grafika Multimedia	3	5	12
PTI333	Basis Data Terdistribusi	3	5	10
TIK333	Sistem Operasi	3	5	10

3 rows in set (0.06 sec)

3. **View**

View merupakan *virtual table* yang dibangun dari satu atau beberapa table. View tidak membuat penyimpanan secara fisik seperti table, namun hanya menyimpan referensi/pointer ke record pada table-table yang berkaitan. Yang tersimpan dalam view hanyalah perintah seleksi data, namun bentuk fisik view dapat diperlakukan sebagaimana sebuah table.

Keuntungan view diantaranya :

a. Kompleksitas pengambilan data dapat disembunyikan dalam view
b. User hanya perlu tahu nama dari view
c. Data yang direferensi oleh view dapat juga di select dengan perintah lainnya dan hasilnya kemudian di filter
d. Isi data di view dapat diupdate (data hasil update akan diteruskan pada isi data tabel yang direference oleh view)

Struktur umum sebuah view adalah :

```
CREATE VIEW namaView (kolom1, kolom2, …) AS
SELECT kolom_a, kolom_b, …
FROM namaTable
WHERE predikat
```

Melalui view dapat dilakukan INSERT, UPDATE dan DELETE dengan beberapa batasan. Salah satu contoh penggunaan view untuk menampilkan kode dan nama provinsi, perhatikan table berikut :

Tabel propinsi

kode_prop	nama_prop	jumlah_penduduk
jabar	jawa barat	234987
jak	dki jakarta	342561
jateng	jawa tengah	374653
jatim	jawa timur	236457
yog	di yogyakarta	276357

maka contoh sintak perintah view sebagai berikut :

```
1  CREATE VIEW vw_propinsi AS
2  SELECT kode_prop, nama_prop
3  FROM propinsi
```

Untuk menampilan atau mengakses data view dapat menggunakan perintah **select** sebagai berikut:

```
SELECT * FROM vw_propinsi
```

Perintah view dapat digunakan untuk menyeleksi data dari beberapa table, seperti contoh sebagai berikut.

Perhatikan **table propinsi** sebelumnya dan **table kota** berikut:

Tabel kota

kode_kota	kode_prop	nama_kota
ban	jatim	banyuwangi
ban	yog	bantul
bks	jabar	bekasi
blr	jateng	blora
byl	jateng	boyolali
kdr	jatim	kediri
kul	yog	kulon progo
mlg	jatim	malang
sby	jatim	surabaya
sle	yog	sleman

Perintah yang digunakan untuk menampilkan kota tertentu termasuk dalam propinsi mana, maka digunakan sintak :

```
1  CREATE VIEW vw_kota AS
2  SELECT k.kode_kota, k.nama_kota,
3         p.kode_prop, p.nama_prop
4  FROM kota k, propinsi p
5  WHERE k.kode_prop=p.kode_prop
```

Contoh perintah view yang digunakan untuk **mengupdate** data jumlah penduduk pada propinsi jawa timur :

```
1  UPDATE vw_propinsi SET jumlah_penduduk =’35000’
2  WHERE kode_prop = ’jatim’
```

Field nama merupakan referensi dari field nama yang ada pada tabel propinsi. Perubahan isi field nama di view vw_propinsi akan mempengaruhi isi field jumlah penduduk di tabel propinsi. Perintah view yang digunakan untuk **menghapus** data pada view hamper sama dengan update data pada view.

4. **Trigger**

Trigger merupakan procedural yang dieksekusi secara otomatis sebagi respon atas suatu kejadian yang berhubungan dengan table. Trigger dieksekusi jika terjadi database event seperti, INSERT, UPDATE, dan DELETE. Trigger INSERT akan aktif pada saat adanya pemasukan record baru, Trigger UPDATE akan aktif pada saat ada perubahan pada sebuah record, dan Trigger DELETE akan aktif pada saat terdapat penghapusan record.

SIntaks umum dari trigger adalah:

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

Keterangan :

a. **trigger_name** : nama trigger.
b. **trigger_time** : kapan kita mengeksekusi trigger, apakah sebelum atau sesudah perubahan pada row data table. Jadi pilihannya adalah **AFTER** atau **BEFORE**.
c. **trigger_event** : merupakan event atau peristiwa yang menyebabkan trigger dilakukan. Pilihan event tersebut adalah **INSERT**, **UPDATE**, **DELETE.**
d. **tbl_name** : nama table.
e. **trigger_body** : *statement-statement* perintah SQL yang akan dilakukan. Jika perintahnya lebih dari satu maka gunakan dalam blok statement **BEGIN ... END**.

f. Jika **DEFINER** dispesifikasikan maka kita memutuskan trigger tersebut dijalankan hanya oleh user tertentu (dalam format penulisan **user@host**). Jika tidak dispesifikasikan, maka user yang melakukan perubahan (CURRENT_USER) adalah pilihan *default*.

Contoh Penggunaan: Trigger After Insert

Pada saat buku dipinjam petugas harus memasukkan data ke dalam tabel pinjam. Pada situasi seperti ini akan menggunakan trigger dengan metode after insert pada tabel peminjaman. Ketika tabel Peminjaman di isi dengan data peminjam maka akan terjadi proses pengurangan stok buku pada tabel buku.

Maka perintah yang digunakan dalam bahasa query adalah:

```
CREATE TRIGGER tr_pinjam AFTER INSERT
   ON Peminjaman FOR EACH ROW
BEGIN
update buku set jml_buku=jml_buku - 1
where kode_buku =new.kode_buku ;
END
```

Pada contoh diatas, pada saat terjadi transaksi peminjaman buku, maka secara otomatis jumlah buku pada table buku berkurang 1. Untuk penerapan Event UPDATE dan DELETE pada trigger menggunakan mekanisme yang sama.

EVALUASI

Untuk menjawab soal, silahkan perhatikan table berikut:

TABEL 1 : Tabel Penjualan Barang

Kode_buku	Jenis_Buku	Nama_buku	Harga	Stok
NL123	Novel	Tere Liye: Rindu	53500	10
NL234	Novel	Rembulan Terang	62200	15
TS456	Teks	Statistika	80500	9
KK678	Komik	Doraemon Vol 2	25500	2
KK321	Komik	Sincan Vol 1	24250	5
TS980	Teks	Basis Data	150000	2
KS222	Kamus	Kamus Bahasa Inggris	35000	18

Berdasarkan Tabel 1

1. Dapatkan harga buku dengan harga termurah dan harga termahal dan kelompokkan berdasarkan jenis bukunya!
2. Dengan menggunakan sub query, dapatkan judul buku dan harga buku dengan stok di atas 10!
3. Buat view **buku mahal** untuk menampilkan Judul buku dan jenis buku dengan harga diatas 50000 dengan nama!
4. Dengan menerapkan Trigger, buat perintah yang digunakan untuk mengurangi stok buku ketika terjadi proses pembelian buku!

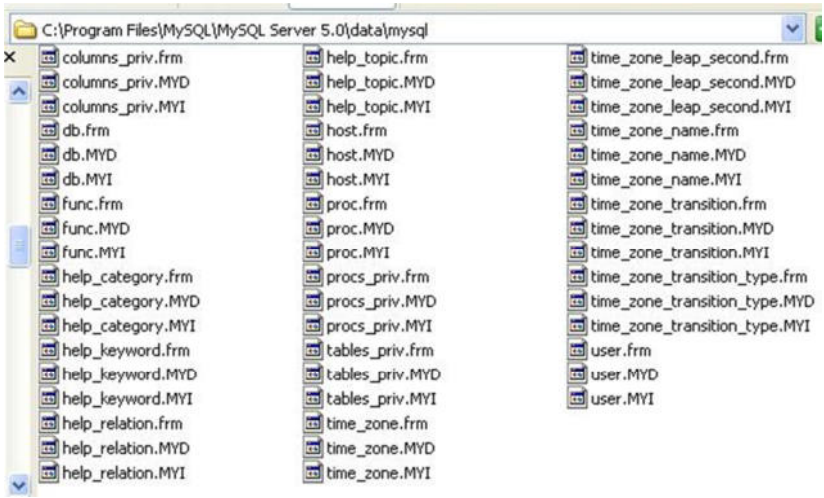
Bab X
BACKUP DAN RECOVERY BASIS DATA

Setelah mempelajari materi ini, diharapkan mahasiswa mampu:
1. Menerapkan Backup Basis Data
2. Menerapkan Recovery Basis Data

Basis data adalah suatu sistem yang harus dapat diandalkan kinerjanya. Tetapi dalam proses pemanfaatan system basis data tidak selamanya dapat berjalan dengan lancar. Adakalanya suatu basis data mengalami gangguan. Gangguan- gangguan itu dapat menyebabkan kerusakan data pada sistem tersebut. Kerusakan data pada sistem basis data dapat dicegah dengan berbagai macam teknik. Untuk pencegahan kerusakan data tersebut dapat dilakukan dengan menggunakan metode backup dan restore. Metode backup dan restore merupakan metode yang sudah lama digunakan untuk menanggulangi kerusakan data.

Metode backup dan restore basis data ini juga digunakan untuk membuat salinan data yang ada pada server secara berkala untuk proses maintenance. Server pada MySQL memiliki berbagai jenis table, diantaranya MyISAM, ISAM, InnoDB, dan DBD. Jenis table tersebut dapat memiliki beberapa file yang membentuk kesatuan table. Sebagai contoh, tipe tabel MyISAM memiliki tiga buah file, yaitu *file.frm*, *file.myd* dan *file.myi*. File-file tersebut merupakan data file bagi satu buah tabel bertipe MyISAM dapat ditunjukkan pada Gambar 10.1. Sementara pada Tabel InnoDB memiliki file data berupa file tipe.frm dapat ditunjukkan pada Gambar 10.2.

Terdapat banyak cara yang dapat dilakukan untuk melakukan proses backup dan recovery basis data. Salah satu cara paling sederhana dapat dilakukan dengan menyalin data file dari tabel - tabel yang ada pada server MySQL, sebagai contoh, jika ingin membackup file pada tabel berjenis MyISAM, maka dapat meng-copy file-file yang berekstensi *.frm,.myd* dan *.myi*. Contoh file tersebut dapat ditunjukkan pada Gambar 10.1.



Gambar 10.1 File data table MyISAM

Gambar 10.2. File data table InnoDB

1. **Membackup Basis Data**

Proses backup basis data di My SQL terdapat beberapa cara diantaranya dengan menggunakan query, melalui administrator, dan dengan mengakses browser pada localhost.

a. **Proses backup dengan sintaks SQL**

Sintaks yang digunakan untuk melakukan backup dengan query adalah:

```
SELECT INTO OUTFILE
BACKUP TABLE
LOAD DATA INFILE
```

Sebelum proses backup dilakukan, dipastikan tidak ada proses penulisan atau perubahan data dalam table. Sintak yang digunakan untuk melakukan penguncian table dengan perintah:

```
LOCK TABLES Nama_Table WRITE;
```

Sebagai contoh ketika akan melakukan backup pada table peminjaman. Maka tahap awal penguncian table sintaks yang dibuat adalah :

```
LOCK TABLES peminjaman WRITE;
```

Setelah proses penguncian table selesai maka langkah selanjutnya adalah melakukan pengosongan memory (FLUSH). Ini dilakukan untuk memastikan tidak ada proses yang berlangsung terhadap data pada table. Sintaks yang digunakan sebagao proses pengosongan memory adalah:

```
FLUSH Tables;
```

Setelah proses penguncian dan pengosongan table, maka proses backup basis data dapat dilakukan. Proses backup table dapat dilakukan secara keseluruan atau sebagian.

Sintaks untuk melakukan backup table adalah :

```
SELECT * INTO OUTFILE 'backup_peminjaman'
   FROM peminjaman;
```

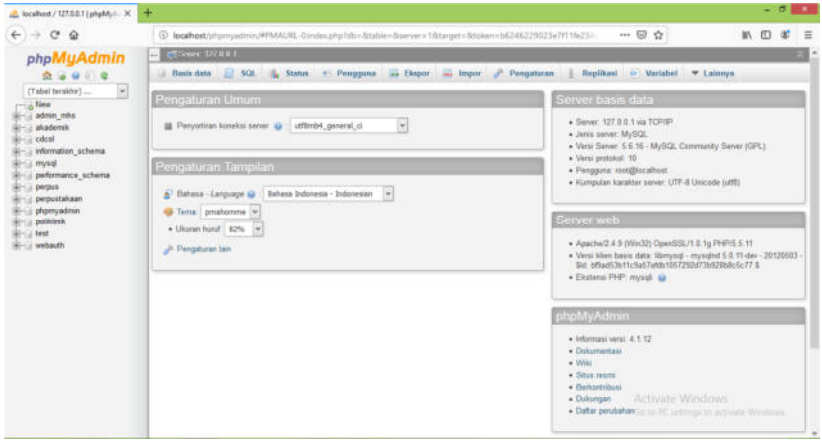
Jika proses backup table berhasil maka aka nada pesan:

```
Query OK, 1 row affected <0,05 sec>
```

Proses pengecekan juga dapat dilakukan dengan melihat apakah file dengan nama **backup_peminjaman** telah terdapat pada lokasi penyimpanan file.

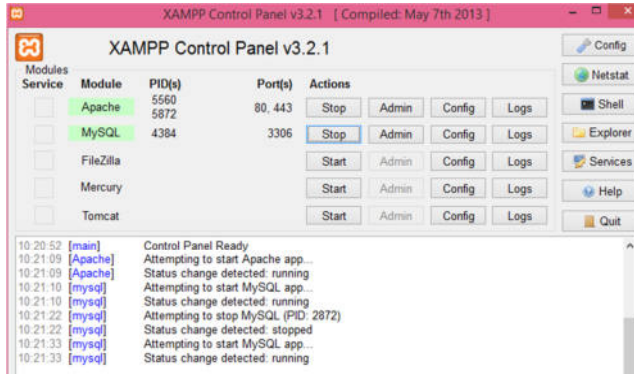
b. Membackup melalui localhost PHPMyAdmin

Proses backup basis data yang kedua dapat dilakukan melalui editor bantu dengan localhost PHPMyAdmin. Aplikasi PHPMyAdmin merupakan free software. Selain untuk melakukan backup dan restore basis data aplikasi ini juga dapat digunakan untuk memanipulasi dan membuat basis data. Tampilan dari PHPMyAdmin ditunjukkan pada Gambar 10.3.



Gambar 10.3 Tampilan Aplikasi PHPMyAdmin

Aplikasi PHPMyAdmin dapat dijalankan melalui browser dengan mengetikkan pada URL ://localhost/PHPMyAdmin. Sebelum mengakses PHPMyAdmin maka Server local (missal : Xampp) harus diaktifkan terlebih dahulu. Server local dapat ditunjukkan pada Gambar 10. 4



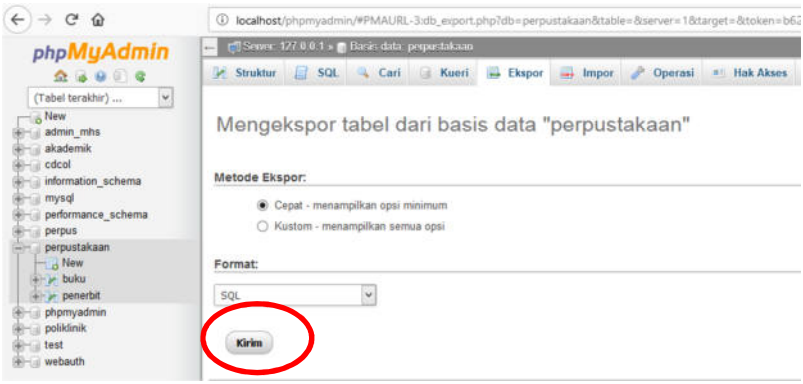
Gambar 10.4 Tampilan server local

Langkah backup basis data melalui PHPMyAdmin adalah sebagai berikut :

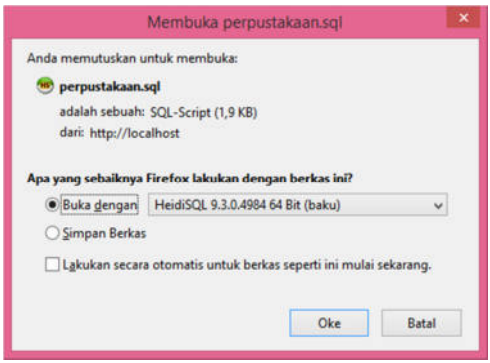
1) Jalankan PHPMyAdmin melalui browser sehingga muncul tampilan seperti Gambar 10.3.
2) Jika ingin melakukan backup terhadap 1 basis data pilih basis data yang akan di backup dan klik menu export pada Gambar 10.5

Gambar 10.5. Menu pada PHPMyAdmin

Setelah tampilan pada menu export keluar, pilih format backup (SQL, Latex, Excel, Word, atau CSV) lalu klik **Kirim** (ditunjukkan pada Gambar 10.6). Setelah proses selesai maka database perpustakaan akan terdownload seperti pada Gambar 10.7.



Gambar 10.6. Tampilan menu Export

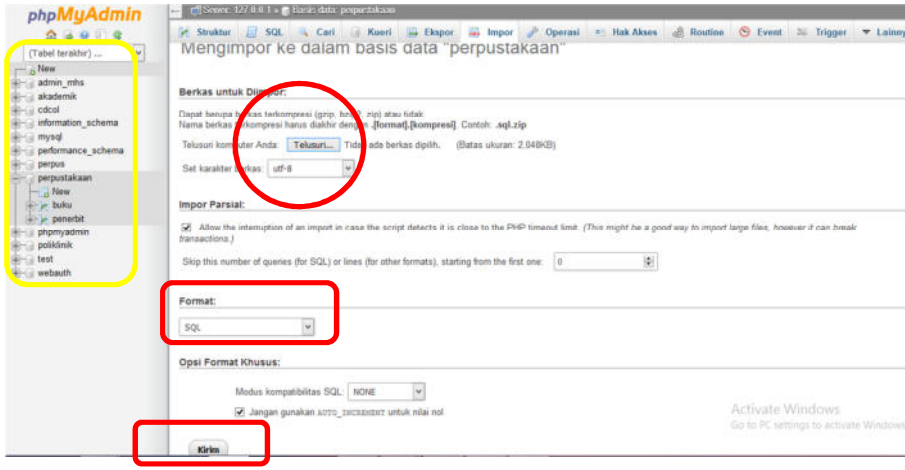


Gambar 10.7 jendela dialog file download

2. Merestore Basis Data

Seperti proses backup basis data, proses restore pada basis data juga terdapat beberapa cara seperti yang telah disebutkan sebelumnya. Pada buku ajar ini akan dijelaskan restore basis data dengan menggunakan aplikasi PHPMyAdmin.

Langkah awal yang dilakukan untuk restore basis data adalah memilih basis data yang akan direstore. Lalu klik menu **IMPORT** yang ada pada Gambar 10. 5. Tampilan menu import ditunjukkan pada Gambar 10.8.



Gambar 10.8 Tampilan menu import

Setelah menu import keluar maka pilih button **Telusuri** untuk mengambil file backup basis data yang telah tersimpan pada computer. Pastikan format yang dipilih sesuai. Jika semua proses selesai pilih button **kirim**. Untuk memastikan apakah basis data yang telah direstore telah masuk, maka periksa pilihan list basis data

pana menu sebelah kiri. Jika basis data telah terpasang, maka proses restore telah berhasil.

EVALUASI

Jawablah pertanyaan di bawah ini :

1. Jelaskan mengapa backup perlu diterapkan pada basis data!
2. Menurut pendapat kalian, kapankah waktu terbaik untuk melakukan backup terhadap suatu basis data?
3. Sebutkan proses backup dan restore yang anda ketahui dalam database MySQL

DAFTAR PUSTAKA

Date, C.J. 2000, *An Introduction to Database System*, Addison Wesley Publishing Company, Vol. 7, New York.

Elmasri dan Navathe. 2007. *Fundamentals of Database Systems, Fifth Edition. Boston: Pearson Education, Inc. Addison Wesley*

Elmasri, Ramez; Navathe, Shamkant B., 2001, F*undamentals of Database Systems*, The Benjamin/ Cummings Publishing Company, Inc., California.

Fatansyah. 2012. Basis Data. Bandung:Informatika

Indrajani. 2015. *Database Design.* Jakarta: Elex Media Komputindo

Kadir, Abdul. 2008. *Belajar Database menggunakan MySQL*. Yogyakarta : Andi Offset

Mardiani, Eri, Rahmansyah, Nur, dkk. 2016. Aplikasi Penggajian Menggunakan Visual Basic, MySQL, dan Data Report. Jakarta: Elex Media Komputindo

Nugroho, Bunafit. 2015. Aplikasi Pemrograman Web Dinamsi dengan PHP dan MySQL. Yogyakarta: Gava Media.

Puspitasari, Dwi. 2016. Normalisasi Tabel pada Basis Data Relasional. Prosiding Sentia. Vol 8. A.341-344.

Shalahudin, M. 2010. Modul Pembelajaran Struktur Data. Bandung: Modula.

BIODATA PENULIS

Fitria Nur Hasanah, M.Pd. dilahirkan di Lamongan, 23 September 1987. Pada tahun 2008, penulis mendapatkan gelar Diploma Manajemen Informatika di Universitas Brawijaya, lulus Sarjana Pendidikan Teknik Informatika di Universitas Negeri Malang tahun 2011, kemudian melanjutkan gelar magister Pendidikan Kejuruan dengan konsentrasi Informatika di Universitas Negeri Malang lulus tahun 2015. Tahun 2011 penulis mengawali karirnya sebagai Guru di SMK Nasional Malang bidang Rekayasa Perangkat Lunak dan Teknik Komputer Jaringan. Selanjutnya tahun 2016 menjadi Dosen tetap di Prodi Pendidikan Teknologi Informasi Universitas Muhammdiyah Sidoarjo . Beberapa penelitian yang pernah dilakukan oleh penulis adalah tentang model pembelajaran dan pengembangan media pembelajaran.

Rahmania Sri Untari, M.Pd dilahirkan di Malang, 19 April 1989. Pendidikan S1 diselesaikan di Program Studi Pendidikan Teknik Informatika Universitas Negeri Malang pada tahun 2011. Pendidikan S2 di Program Pascasarjana Jurusan Pendidikan Kejuruan diselesaikan pada tahun 2015. Pada tahun 2016, penulis melanjutkan Pendidikan S3 Jurusan Pendidikan Kejuruan di Universitas Negeri Malang dan sekarang masih berada pada masa studi. Pada tahun 2011 penulis mengawali karirnya di SMA Negeri 6 Surabaya sebagai guru Teknik Informatika. Selanjutnya, pada tahun 2015 penulis melanjutkan karirnya untuk menjadi Dosen tetap di Prodi Pendidikan Teknologi Informasi Universitas Muhammadiyah Sidoarjo. Beberapa penelitian yang pernah dilakukan penulis adalah tentang model pembelajaran, kurikulum, pengembangan media pembelajaran.