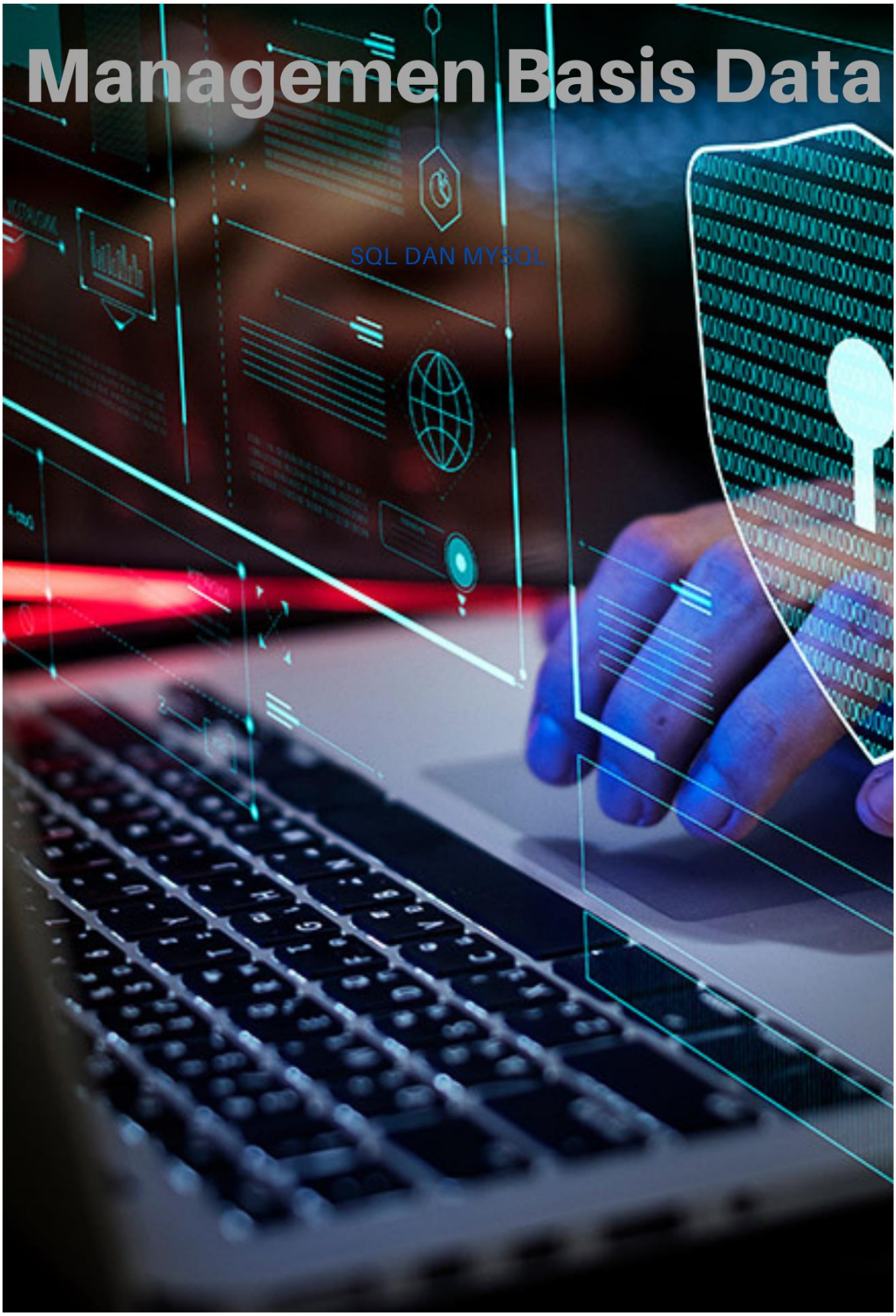
Managemen Basis Data



SQL DAN MYSQL

oleh Ahmad Ashifuddin Aqham, M.M., M.Kom

Managemen Sistem Basis Data (SQL dan MySql)

Ahmad Ashlfuddin Aqham, S.Kom., M.M, M.Kom

YAYASAN PRIMA AGUS TEKNIK

ISBN 978-623-6141-62-5 (PDF)

Managemen Sistem Basis Data (SQL dan MySql)

Penulis: Ahmad Ashifuddin Aqham, S.Kom., M.M, M.Kom

ISBN: 978-623-6141-62-5 (PDF)

Editor:

Bagus Sudirman, M.Kom

Penyunting:

Eko Siswanto, M.Kom

Desain Sampul dan Tata Letak:

Teguh Setiadi, M.Kom

Penerbit:

Yayasan Prima Agus Teknik

Redaksi:

Jln Majapahit No 605 Semarang

Tlpn. (024) 6723456

Fax . 024-6710144

Email: penerbit_ypat@stekom.ac.id

Distributor Tunggal:

UNIVERSITAS STEKOM

Jalan Majapahit No. 605 Semarang

Tlpn. (024) 6723456

Fax. 024-6710144

Email: info@stekom.ac.id

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa atas berkat, rahmat dan karunia-Nya sehingga buku ajar untuk mata system basis data ini telah berhasil diselesaikan. Buku ajar ini diperuntukkan sebagai pegangan bagi mahasiswa program studi Komputerisasi Akuntansi yang dirancang secara praktis agar mahasiswa dapat memahami dan mempraktekkan DBMS dengan mudah.

Buku ajar ini disusun dengan tujuan menyediakan materi pembelajaran system basis data yang lengkap dan praktis untuk mahasiswa sesuai dengan standar kurikulum yang telah ditentukan dan disepakati bersama. Materi dan tugas pembelajaran dikembangkan dengan prinsip-prinsip tutorial untuk secara terintegrasi untuk mengembangkan kompetensi mahasiswa.

Harapan ke depannya, buku ajar ini merupakan bahan dan sumber belajar dan kompetensi mahasiswa yang sesuai sebagai bekal di dunia kerja nantinya, maka dari itu agar isi dari buku ajar ini selalu relevan dengan kondisi terkini dengan dinamika perubahan teknologi yang terus terjadi, selalu dibutuhkan masukan untuk bahan perbaikan atau revisi.

Demikian, semoga buku ajar ini dapat bermanfaat bagi anda semua khususnya mahasiswa UNIVERSITAS STEKOM Semarang atau praktisi yang sedang mengembangkan bahan ajar untuk kegiatan perkuliahan.

Semarang, Juni 2021

Penulis

DAFTAR ISI

BAB 1
PENGANTAR SISTEM MANAJEMEN BASIS DATA

A. DEFINISI BASIS DATA

Lemari arsip yang di simpan dan di susun, di kelompokkan berdasarkan urutan tertentu seperti abjad atau urutan secara kronologis biasa di sebut atau di bayangkan sebagai Basis Data atau Database. Hal ini dilakukan sebagai upaya untuk mempermudah pencarian arsip, karena arsip yang tersusun dengan rapi maka proses pencarian arsip dapat lebih cepat.

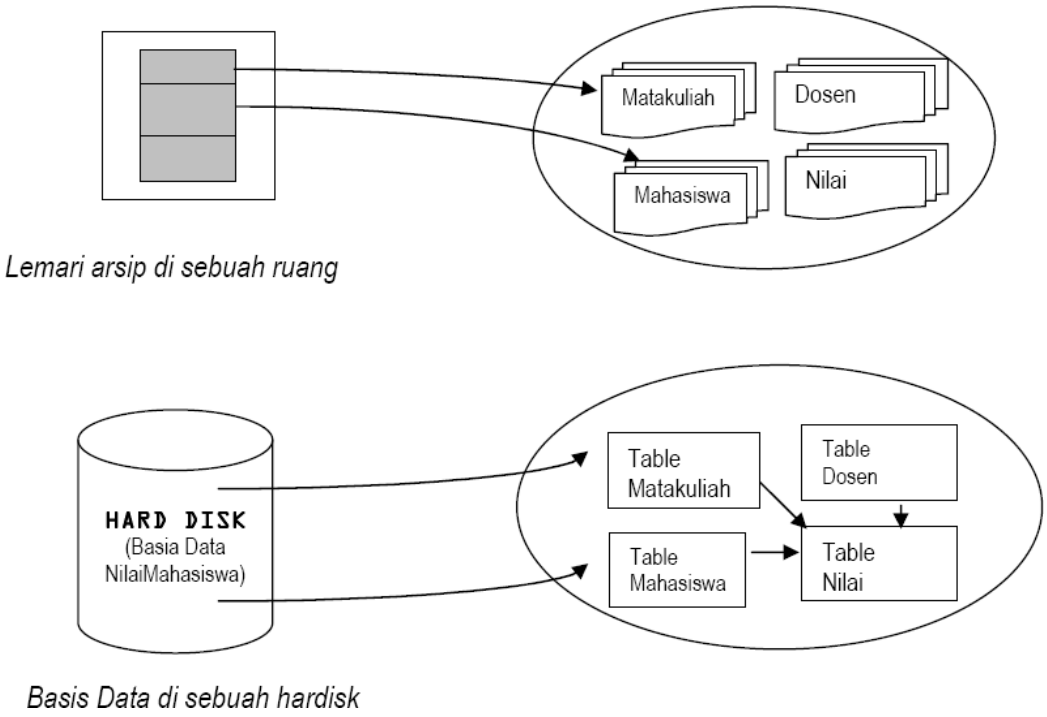
Basis Informasi terdiri dari 2 kata, ialah Basis serta Informasi, yang bisa didefinisikan selaku berikut:

- Basis: Markas ataupun gudang, tempat bersarang/ berkumpul.
- Data: Reperesentasi dunia nyata yang mewakili sesuatu objek semacam manusia(pegawai, siswa, pembeli, pelanggan), benda, hewan, kejadian, konsep, kondisi, serta sebagainya, yang direkam dalam wujud simbol, bacaan, foto, bunyi, ataupun kombinasinya.

Basis informasi sendiri bisa didefiniskan dalam beberapa pemikiran selaku berikut:

- Himpunan kelompok informasi(arsip) yang silih berhubungan yang diorganisasi sedemikian rupa supaya nanti bisa dimanfaatkan kembali dengan kilat serta gampang.
- Kumpulan informasi yang silih berhubungan yang ditaruh secara bersama sedemikian rupa serta tanpa pengulangan(redundansi) yang tidak butuh, buat penuhi bermacam kebutuhan.
- Kumpulan file/ tabel/ arsip yang silih berhubungan ditaruh dalam media poenyimpanan elektronis.

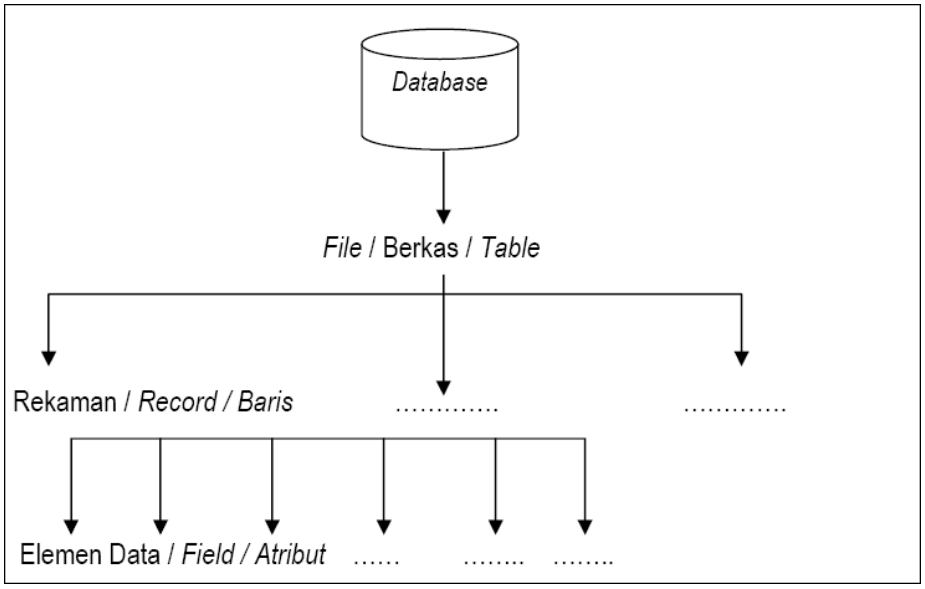
Prinsip utama basis informasi merupakan pengaturan informasi/ arsip. Tujuan utama basis informasi merupakan kemudahan serta kecepatan dalam pengambilan kembali informasi/ arsip. Esensi dari suatu basis informasi bukan cuma media penyimpanannya yang berbentuk media elektronis, namun didalamnya ada pengaturan/ pemilahan/ pengelompokkan/ pengorganisasian datayang ditaruh cocok dengan guna serta jenisnya. Pemilahan/ pengelompokkan/ pengorganisasian ini bisa berupa beberapa file/ tabel terpisah ataupun dalam wujud pendefinisian kolom-kolom/ field- field informasi dalam tiap file/ tabel.



Gambar 1. Lemari Arsip dan Basis Data

B. HIRARKI DATA

Hirarki informasi bisa dikelompokkan jadi 3 bagian ialah file, record serta elemen informasi, semacam foto berikut::



Gambar 2. Hirarki Data

Penafsiran dari foto di atas merupakan selaku berikut:

- Elemen/ Field/ Atribut merupakan satuan informasi terkecil yang tidak bisa dipecah lagi jadi unit lain yang bermakna. Pada informasi mahasiswa, field/ atribut informasinya bisa berbentuk nim, nama_m, tpt_lhr_m, alm_m serta atribut yang lain yang berhubungan dengan mahasiswa tersebut. Sebutan lain dari elemen informasi merupakan medan/ field, kolom item, serta atribut. Sebutan yang universal dipakai merupakan field, atribut ataupun kolom.
- Rekaman/ Record/ Baris merupakan gabungan beberapa elemen informasi yang silih terpaut. Contohnya merupakan nim, nama_m, tpt_lhr_m, tgl_lhr_m, alm_m serta atribut yang lain dari seseorang mahasiswa bisa dikumpulkan dalam suatu record ataupun baris.

  Contoh:

  Record seseorang mahasiswa:

  Nim: 01031417; nama_m: Mulyanti; tpt_lhr_m: Bogor; tgl_lhr_m: 1/ 10/ 76; j_kelamin: Perempuan; alm_m: Perum Telaga Murni RT 07/ 02 N0. 1; kota_m: Bogor; agama_m: Islam; kode_jur: MI.
- Berkas/ File/ Tabel merupakan kumpulan record sejenis yang memiliki panjang atribut/ field sama, tetapi berbeda isi informasinya.

Berikut ilustrasi dari ketiga pengertian di atas :

nim	nama_m	tpt_lhr_m	tgl_lhr_m	j_kelamin	alm_m	kota_m	agama_m	kode_jur
01031417	Mulyanti	Bogor	1/10/76	Wanita	Perum Telaga Murni RT 07/02 No.1	Bogor	Islam	MI
01031013	Ahmad Sofyan	Surabaya	2/13/77	Pria	Jl. SMP I Nurul Huda RT 002/02 No.4	Bekasi Timur	Islam	KA
01031023	Ani Lusiamah	Bandung	10/12/77	Wanita	Kav. Bulak Sentul No. 34 RT 011/03	Cibinong	Islam	MI
01031043	Cecep Iwan Kurnia	Solo	10/10/78	Pria	Taman Tytan Indah RT 001/10 No.31	Bekasi Barat	Kristen	KA
01031044	Chandra Khirana	Cirebon	2/10/79	Pria	Jl. Masjid Al-Hidayah RT 01/02 No.7	Bekasi Barat	Hindu	TK
01031046	Darmiyati	Bekasi	12/10/87	Wanita	Jl. Dahlia I Blok BC 2/3 RT 004/10	Bekasi Utara	Hindu	TI
01031050	Deni Hermawan	Jakarta	1/17/80	Pria	DS Sukadanau RT 001/02 No.3	Cikarang	Islam	TI
01031091	Fitria Choirunissa	Bekasi	1/21/79	Wanita	Jl. Gandaria Selatan I RT 002/24 No.1	Jakarta Timur	Islam	TK
01031178	Reni Atika	Bogor	8/18/81	Wanita	Jl. Cikarang Baru No.2 RT 01/05	Cikarang	Islam	TI
01031219	Sylvia Dwita Ningrum	Semarang	10/19/81	Wanita	Jl. Sultan Agung No.23 RT 05/03	Bekasi Barat	Budha	TI
01031341	Ira Sulistyaningsih	Bekasi	12/1/79	Wanita	Jl. Purna 11/B-1 No. 7 RT 005/08	Tambun	Budha	SI
01031365	Muhammad Fahrurozi	Jakarta	10/13/80	Pria	Jl. Dewi Sartika No.83 RT 03/08	Jakarta Timur	Islam	TK
01031452	Susilo Wahono	Jakarta	12/21/79	Pria	Jl. Kaliabang Tengah RT 06/04 No.12	Jakarta Timur	Islam	TI
01031480	Yonita Veronika	Bekasi	10/24/80	Wanita	Jl. Bunda Harapan No.36 Rt 001/08	Bekasi Barat	Kristen	TI
01031484	Yudi Ridwan	Subang	1/19/80	Pria	Jl. Kebun Kelapa RT 04/01 No.8	Cibitung	Hindu	SI
01031487	Yuni Nurwati	Jakarta	11/14/78	Wanita	Kp. Pisangan RT 001/01 No.12	Bekasi Utara	Budha	TK

Gambar 3. Contoh File, Field, Record dan Data Value.

Data value / Isi data :

Pada record pertama : 01031417 adalah isi data untuk kolom nim, Mulyanti untuk kolom nama_m dan seterusnya

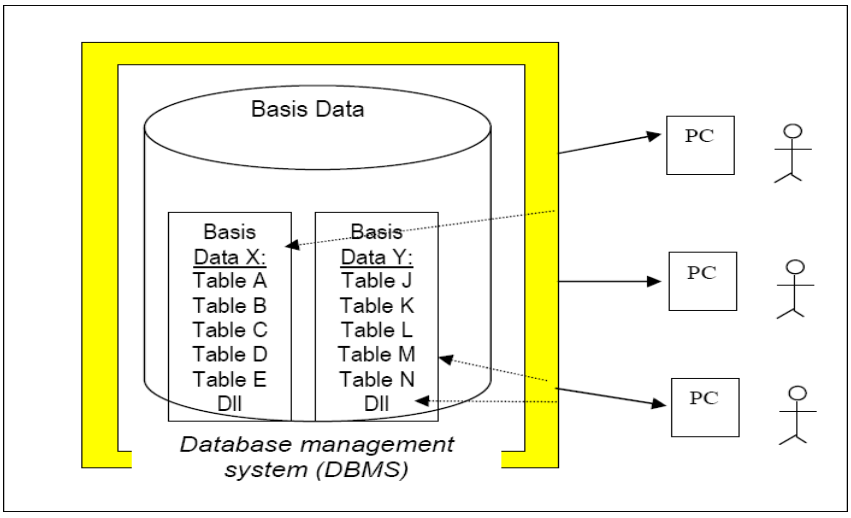
C. SISTEM BASIS DATA

Sistem merupakan suatu tatanan(keterpaduan) yang terdiri atas beberapa komponen fungsional(dengan satuan guna/ tugas spesial) yang silih berhubungan serta secara bersama- sama bertujuan buat penuhi sesuatu proses ataupun pekerjaan tertentu.

Basis informasi cumalah suatu objek yang pasif/ mati. Dia terdapat sebab terdapat pembuatnya. Dia tidak hendak bermanfaat bila tidak terdapat pengelola/ penggeraknya. Yang jadi pengelola/ penggeraknya secara langsung merupakan program/ aplikasi/ aplikasi. Gabungan keduanya(basis informasi serta pengelolanya) menciptakan suatu sistem. Hingga secara universal, suatu Sistem Basis Informasi ialah suatu sistem yang terdiri atas kumpulan file(tabel) yang silih berhubungn(dalam suatu basis informasi disebuah sistem pc) serta sekumpulan program(DBMS) yang membolehkan sebagian pemakai serta ataupun program lain buat mengakses serta memanipulasi file/ file(tabel- tabel) tersebut.

Berikut ini merupakan komponen- komponen sistem basis informasi:

1. Fitur keras(hardware)
2. Sistem Pembedahan(Operating System)
3. Basis Informasi(Database)
4. Sistem(Aplikasi/ Fitur Lunak) pengelola basis informasi(DBMS)
5. Pemakai(user)
6. Aplikasi(fitur lunak) lain(bertabiat opsional).



Gambar 4. Sistem Basis Dat

1. Perangkat Keras

Perangkat keras yang biasanya terdapat salam sebuah sistem adalah :

a. Komputer (Sesuatu sistem yang stand- alone ataupun lebih dari satu buat sistem jaringan)
b. Memori sekunder yang on- line(Harddisk)
c. Memori sekunder yang off- line(Tape ataupun removable disk buat keperluan backup informasi)
d. Media ataupun fitur komunikasi(buat sistem jaringan)

2. Sistem Operasi (Operating System)

Ialah program yang mengaktifkan/ memfungsikan sistem pc, mengatur segala sumber energi(resource) dalam pc serta melaksanakan operasi- operasi dasar dalam pc(pembedahan I/ O, pengelolaan file, serta lain- lain)

Program pengelola basis informasi(DBMS) hendak aktif(running) bila sitem operasoi yang dikehendakinya(cocok) sudah aktif.

Beberapa Sistem Pembedahan yang banyak digunakan merupakan: MS-DOS, MS- Windows(3. 1, 95, 98 serta sebagainya) buat yang stand alone, serta Ms- Windows(2000 Server, UNIX, LINUX, Novel- Netware serta lain sebagainya) buat jaringan.

3. Basis Data (Database)

Suatu sistem basis informasi bisa mempunyai sebagian basis informasi. Tiap basis informasi bisa berisi/ mempunyai beberapa objek basis informasi(semacam file/ tabel, indeks serta lain- lain). Disamping berisi/ menaruh informasi, tiap basis informasi pula memiliki/ menaruh definisi struuktur(baik buat basis informasi ataupun objek- objeknya secara perinci).

4. Sistem Pengelola Basis Data (DBMS)

Pengelolaan basis informasi secara raga tidak dicoba oleh pemakai secara langsung, namun ditangani oleh suatu Fitur Lunak(Sistem) yang khusu/ khusus. Fitur lunak ini diucap DBMS, yang hendak memastikan gimana informasi diorganisasi, ditaruh, diganti, serta diambil kembali. Fitur ini pula menerapkanmekanisme pengamanan informasi, pemekaian informasi secara bersama, pemaksaan keakuratan/ konsistensi informasi, serta sebagainya.

Fitur lunak yang tercantum DBMS semacam dBase IV, FoxBase, MS-Access, Clipper serta Borland- Paradox buat kelas yang simpel, ataupun Borlad-

Interbase, MS- SQLServer, CA- Open Ingres, Oracle, Informix serta Sybase buat kelas lingkungan ataupun berat.

5. Pemakai (User)

Terdapat sebagian tipe/ jenis pemakai terhadap sesuatu sistem basis informasi yang dibedakan bersumber pada metode berhubungan terhadap sistem, ialah:

• Programmer Aplikasi

Pemakai yang berhubungan dengan basis informasi lewat Informasi Manipulation Language(DML), yang disertakan(embedded) dalam program yang ditulis dalam bahasa pemrograman induk(semacam C, Pascal, Cobol, serta lain- lain).

• User Mahir(Casual user)

Pemakai yang berhubungan dengan sistem tanpa menulis materi program. Mereka melaporkan query(buat akses informasi) dengan bahasa query yang disediakan oleh DBMS.

• User Universal(End User/ Naive User)

Pemakai yang berhubungan dengan sistem basis informasi lewat pemanggilan satu program aplikasi permanen(executable program) yang sudah ditulis ataupun disediakan tadinya.

• User Spesial(Specialized User)

Pemakai yang menulis aplikasi basis informasi non konvensional buat keperluan spesial semacam aplikasi AI, Sistem Ahli, Pengolahan Citra, serta lain- lain yang dapat saja mengakses basis informasi dengan ataupun tanpa DBMS yang bersangkutan.

Buat suatu sistem basis informasi yang stand- alone, hingga pada sesuatu dikala cuma terdapat satu pemakai yang bisa bekerja. Sebaliknya buat sistem basis informasi dalam jaringan, hingga pada sesuatu dikala terdapat banyak pemakai yang bisa berhubungan serta memakai basis informasi yang sama. Opsi buat stand- alone ataupun jaringan(multiuser) bergantung pada ataupun didetetapkan oleh kebutuhan pemakai, fitur keras yang ada, sistem pembedahan yang digunakan dan DBMS yang diseleksi.

• Aplikasi ataupun Fitur Lunak Lain

Aplikasi ini bertabiat opsional, maksudnya terdapat ataupun tidaknya tergantung kebutuhan. DBMS yang digunakan lebih berfungsi dalam pengorganisasian informasi dalamn basis informasi, sedangkan selaku pemakai basis informasi(spesialnya yang jadi end- user/ naive user) bisa dibuatkan ataupun disediakan

program spesial buat melaksanakan pengisian, pengubahan, serta pengambilan informasi. Program ini terdapat yang telah disediakan bersama dengan DBMS-nya, tetapi terdapat pula yang wajib terbuat sendiri denga memakai aplikasi lain yang spesial buat itu(development tools).

D. Sistem Manajemen Basis Data (DBMS)

DBMS merupakan koleksi terpadu dari program- program(sistem fitur lunak) yang digunakan buat mendefinisikan, menghasilkan, mengakses serta menjaga basis informasi(database). Tujuannya merupakan sediakan area yang gampang serta nyaman buat pemakaian serta perawatan basis informasi. Contoh dari DBMS merupakan MS- Access, MS- SQLServer serta Oracle.

BAB 2
OPERASI BASIS DATA

A. OPERASI DASAR BASIS DATA

Didalam suatu disk(hard disk), basis informasi bisa diciptakan serta bisa pula ditiadakan. Didalam suatu disk bisa ditempatkan bebrapa(lebih dari satu) bassi informasi. Sedangkan dalam suatu basisdata bisa ditempatkan satu ataupun lebih file/ tabel. Pada file/ tabel inilah sebetulnya informasi ditaruh ataupun ditempatkan. Misalnya, dalam suatu basis informasi penjualan, terdiri dari tabel benda, faktur, pelanggan serta transaksi benda.

Operasi- operasi dasar yang bisa dicoba berkenaan dengan basis informasi merupakan selaku berikut:

- Pembuatan Basis Informasi Baru(create database), identik dengan pembuatan lemari arsip yang baru
- Penghapusan basis informasi(drop database), identik dengan peluluhlantahkan lemari arsip, tercantum isinya bila terdapat.
- Pembuatan file/ tabel baru ke sesuatu basis informasi(drop table), identik dengan peluluhlantahkan map arsip lama yang terdapat di lemari arsip.
- Penambahan/ pengisian informasi baru ke suatu file/ tabel di suatu basis informasi(insert), identik dengan akumulasi lembaran arsip ke suatu map arsip.
- Pengambilan informasi dari suatu file/ tabel(retrieve/ search), identik dengan pencarian lembaran arsip dari suatu map arsip.
- Pengubahan informasi dari suatu file/ tabel(pembaharuan), identik dengan revisi isi lembaran arsip yang terdapat disebuah lemari arsip.
- Penghapusan informasi dari suatu file/ tabel(delete), identik dengan penghapusan suatu lembaran arsip yang terdapat pada suatu lemari arsip.

B. KEGUNAAN BASIS DATA

Penataan sesuatu basis informasi digunakan buat menanggulangi masalah-masalah pada penataan informasi, antara lain:

- Redundansi serta inkonsistensi data
- Kesulitan pengaksesan data

- Isolasi data untuk standarisasi
- Multiple User (banyak pemakai)
- Masaalah keamana (security)
- Masalah integrasi (kesatuan)
- Masalah Data Independence (kebebasan data)

1. Redundansi dan Inkonsistensi Data

Redundansi terjadi jika suatu informasi disimpan di beberapa tempat. Misalnya, ada data mahasiswa yang memuat NIM, nama, alamat dan telp. Sementara pada data KHS mahasiswa, yang isinya terdapat NIM, nama, alamat, telp, mata kuliah dan nilai. Pada kedua tabel tersebut kita temukan ada atribut yang sama, seperti tabel berikut :

Tabel 2.1 Mahasiswa

NIM	Nama	Alamat	Telp
101	Agung	Kendal	112233
102	Ratna	Weleri	123456

Tabel 2.2 KHS

NIM	Nama	Alamat	Telp	Mata Kuliah	Nilai
102	Ratna	Weleri	123456	Basis Data	A
101	Agung	Kendal	112233	Struktur Data	C
102	Ratna	Weleri	123456	Struktur Data	B
102	Ratna	Weleri	123456	Agama	A

Pada tabel 2.1 dan 2.2, informasi nama seorang mahasiswa disimpan dibeberapa tempat. Penyimpanan dibeberapa tempat untuk datayang sama disebut redudansi yang mengakibatkan pemborosan ruang penyimpanan dan juga biaya untuk akses yang lebih tinggi.

Akibat dari redudansi adalah inkonsistensi data atau data yang tidak konsisten. Ini dapat terjadi bila suatu ketika mahassiwa tersebut pindah alamat, maka seluruh tabel yang mengandung data mahasiswa tersebut harus diubah/update. Bila hanya satu tabel yang di update, maka data menjadi tidak konsisten.

2. Kesulitan dalam Pengaksesan Data

Bila suatu saat dibutuhkan mencari suatu data dengan kriteria tertentu, maka akan muncul kesulitan dalam pencarian data, karena data harus dipilah satu persatu untuk mendapatkan data sesuai dengan kriteria. Hal ini disebabkan karena belum tersedia program khusus untuk mencari data berdasarkan kriteria tertentu.

Misalnya jika pada suatu saat dibutuhkan untuk mencari semua data mahasiswa yang alamatnya berada di Bantul, maka seluruh data mahasiswa harus dipilah-pilah untuk mendapatkan semua data mahasiswa yang berasa dari Bantul. Untuk itu maka diperlukan sebuah DBMS yang mampu mengambil data secara langsung dengan bahasa yang familiar dan mudah digunakan (user friendly).

3. Isolasi Data Untuk Standarisasi

Jika data tersebar dalam beberapa file/tabel dalam bentuk format yang tidak sama, maka ini akan menyulitkan dalam menulis program aplikasi untuk mengambil dan menyimpan data. Maka haruslah data dalam satu basis data dibuat satu format, sehingga mudah dibuat program aplikasinya.

4. Multiple User (Banyak Pemakai)

Basis data memungkinkan penggunaan data bersama-sama oleh banyak pengguna pada saat yang bersamaan atau pada saat yang berbeda. Dengan meletakkan basis data pada bagian server yang bisa diakses dari banyak client, maka basis data dapat diakses oleh semua client yang terhubung ke server basis data. Dimana pengaksesan data disesuaikan dengan hak akses masing-masing pengguna. Misalnya, sebuah perguruan tinggi memiliki data tentang mahasiswa, pembayaran dan lain-lain, yang diletakkan pada sebuah basis data. Bagian akademik akan bisa mengakses data-data akademik mahasiswa, bagian keuangan akan diijinkan mengakses data pembayaran mahasiswa, sementara mahasiswa hanya bisa melihat status akademik atau keuangan yang berhubungan dengan dirinya saja.

5. Masalah Keamanan (Security)

Tidak semua pemakai sistem basis data diperbolehkan untuk mengakses semua data. Misalkan data yang berkaitan dengan data gaji karyawan hanya

hanya boleh akses oleh bagian keuangan dan personalia, tidak pada bagian gudang untuk membaca dan mengubahnya.

6. Masalah Integritas (Kesatuan)

Basis data berisi file/tabel yang saling terkait untuk menjaga konsistensi data. Sehingga perubahan data pada suatu file master akan mengakibatkan perubahan data pada file lain yang saling terkait dengan file master tersebut.

Untuk mengaitkan atau menghubungkan antara satu tabel dengan tabel yang lain, maka dibutuhkan suatu suatu field/atribut kunci yang mengaitkan atau merelasikan tabel tersebut.

7. Masalah Independence (Kemandirian Data)

Independence atau kemandirian data dimana perubahan yang terjadi pada tingkat yang lebih rendah tidak mempengaruhi tingkat yang lebih tinggi. Ini berarti bahwa perintah-perintah dalam paket DBMS bebas terhadap basis data. Apapun perubahan dalam basis data semua perintah akan mengalami kestabilan tanpa mengalami perubahan.

C. KEUNTUNGAN SISTEM BASIS DATA

1. Kecepatan dan Kemudahan (Speed)

Dengan menggunakan basis data, pengambilan informasi dapat diakukan dengan cepat dan mudah. Basis data memiliki kemampuan dalam mengelompokkan, mengurutkan bahkan perhitungan dengan matematika. Dengan perancangan yang benar, maka penyajian informasi akan dapat dilakukan dengan cepat dan mudah.

2. Kebersamaan Pemakai (Sharability)

Sebuah basis data dapat digunakan oleh banyak user dan banyak aplikasi. Untuk data-data yang diperlukan oleh banyak orang/bagian, tidak perlu dilakukan pencatatan oleh masing- masing bagian, tetapi cukup dengan satu basis data untuk dipakai bersama.

Contohnya data mahasiswa dalam suatu perguruan tinggi, dibutuhkan oleh banyak bagian, diantaranya : bagian akademik, bagian keuangan, bagian kemahasiswaan, dan perpustakaan. Tidak harus semua bagian ini memiliki catatan data mahasiswa. Data cukup disediakan oleh sebuah basis data dan semua bagian bisa mengakses data tersebut dengan keperluannya.

3. Pemusatan Kontrol Data

Karena cukup dengan satu basis data untuk banyak keperluan, pengontrolan terhadap data juga cukup dilakukan di satu tempat saja. Jika ada perubahan data alamat mahasiswa misalnya, maka tidak perlu kita meng-update semua data di masing-masing bagian, tetapi cukup hanya di satu basis data.

4. Efisiensi Ruang Penyimpanan (Space)

Dengan pemakaian bersama, kita tidak perlu menyediakan tempat penyimpanan di berbagai tempat, tetapi cukup satu saja, sehingga ini akan menghemat ruang penyimpanan yang dimiliki oleh sebuah organisasi. Dengan teknik perancangan basis data yang benar, kita akan dapat menyederhanakan penyimpanan sehingga tidak semua dapat harus disimpan.

Misalnya ada data pengambilan mata kuliah oleh mahasiswa, dimana yang dicatat adalah: NIM, Nama, Jurusan, Alamat, Kode, NamaMatkul dan SKS.

Tabel 2.3 KRS

NIM	Nama	Jurusan	Alamat	Kode	NamaMatkul	SKS
101	Anto	SI	Klaten	A	Agama	3
101	Anto	SI	Klaten	B	Bahasa	4
101	Anto	SI	Klaten	C	Kalkulus	2
102	Fika	MI	Bantul	A	Agama	3
102	Fika	MI	Bantul	B	Bahasa	4
102	Fika	MI	Bantul	C	Kalkulus	2

Jika kita memiliki data 4000 mahasiswa dengan rata-rata setiap mahasiswa mengambil 5 mata kuliah, maka data seorang mahasiswa akan kita tulis 5 kali dan data mengenai mata kuliah akan ditulis 4000 kali. Andaikan tiap data bernilai 1 byten (diabaikan tipe datanya) maka kita akan membutuhkan ruang penyimpanan sebesar jumlah field x jumlah mhs x jumlah matkul (7x4000x5 = 140000)

Dengan basis data yang dirancang dengan benar, data ini bisa dibagi menjadi 3, seperti tampak pada tabel 2.4,.2.5,.2.6.

Tabel 2.4 Mahasiswa

NIM	Nama	Jurusan	Alamat
101	Anto	SI	Klaten
102	Fika	SI	Bantul

Tabel 2.5 Mata Kuliah

Kode	NamaMatkul	SKS
A	Agama	3
B	Bahasa	4
C	Kalkulus	2

Tabel 2.6 KRS

NIM	Kode
101	A
101	B
101	C
102	A
102	B
102	C

Kita membutuhkan ruang penyimpanan untuk tabel mahasiswa seperti tampak pada tabel 2.4 sebesar jumlah kolom x jumlah mhs (4x400=16000). Untuk mata kuliah tabel 2.5 jumlah kolom x jumlah matkul (3x5=15), dan untuk tabel KRS jumlah kolom x jumlah mhs x jumlah matkul (2x4000x5 = 40000), maka ketiga tabel tersebut jika dijumlahkan akan membutuhkan ruang sebesar 1600+16+40000=16015 byte. Bila dibandingkan dengan penyimpanan data pada tabel 2.3 sebesar 140000, jelas relasi antar tabel pada basis data bisa memperkecil ruang penyimpanan.

5. Keakuratan (Accuracy)

Penerapan secara ketat aturan tipe data, domain data, keuinikan data, hubungan antar data dan lain-lain dapat menekan ketidakakuratan dalam pemasukan atau penyimpanan data.

6. Ketersediaan (availability)

Dengan basis data kita dapat membackup data, memilah-milah data mana yang masih diperlukan dan data mana yang perlu kita simpan di tempat lain. Hal ini mengingat pertumbuhan transaksi suatu organisasi dari waktu ke waktu

membutuhkan media penyimpanan yang semakin besar.

7. **Keamanan (Security)**

   Kebanyakan DBMS dilengkapi dengan fasilitas manajemen pengguna. Pengguna diberikan hak akses yang berbeda-beda sesuai dengan kepentingan dan posisinya. Basis data bisa diberikan password untuk membatasi orang yang mengaksesnya.

8. **Kemudahan dalam pembuatan program aplikasi baru**

   Penggunaan basis data bagian dari perkembangan teknologi. Dengan adanya basis data pembuatan aplikasi bisa memanfaatkan kemampuan dari DBMS, sehingga pembuat aplikasi tidak perlu mengatur penyimpanan data, cukup mengatur interface untuk pengguna.

9. **Pemakaian Secara Langsung**

   Basis data memiliki fasilitas untuk melihat datanya secara langsung dengan tool yang disediakan DBMS. Untuk melihat data dapat langsung ke tabel atau menggunakan query. Biasanya yang menggunakan fasilitas ini adalah user ang sudah ahli atau database adminsitrator.

10. **Kebebasan Data (Data Independece)**

    Jika sebuah program telah selesai dibuat, dan ternyata ada perubahan isi/struktur data, maka dengan basis data perubahan ini hanya perlu dilakukan pada level DBMS tanpa harus membongkar kembali aplikasinya.

11. **User View**

    Basis data menyediakan pandangan yang berbeda-beda untuk tiap-tiap pengguna, sesuai dengan posisi dan kepentingannya di dalam organisasi.

    Contohnya data pada perusahaan yang bergerak dibidang retail. Data yang ada berupa data barang, penjualan, dan pembelian. Ada beberapa jenis pengguna yang memerlukan informasi yang terkait dengan data perusahaan tersebut, yaigtu pelanggna, kasir, bagian gudang, bagian akuntansi, dan juga manajer. Tidak semua data boleh diakses oleh semua pengguna. Misalnya kasir, hanya berhak untuk melihat informasi nama barang dan harga jualnya. Sementara dia tidak berhak untuk memasukkan data penjualan. Berbeda dengan pelanggan yanghanya boleh melihat keberadaan barang dan harga jual, tetapi tidak berhak mamsukkan atau merubah data. Sementara itu bagian akuntansi berhak melihat harga beli dan harga jual dari setiap barang, bahkan berhak melihat keuntungan dari tiap-tiap barang untuk menganalisis data akuntansinya.

D. KEKURANGAN SISTEM BASIS DATA

Kerugian-kerugian yang ada dengan diterapkannya basis data pada suatu perusahaan adalah sebagai berikut :

1. Diperlukan hardware (perangkat keras tambahan) :CPU yang lebih kuat, terminal yang lebih banyak, alat komunikasi.
2. Biaya Performance yang lebih besar : listrik, personil yang lebih tinggi klasifikasinya, biaya telekomunikasi antar lokasi.
3. Rawannya keberhasilan operasi : gangguan listrik dan komunikasi.
4. Sistem lebih kompleks : banyaknya aspek yang harus diperhatikan.

E. CONTOH DATABASE

Berikut ini adalah contoh database Perpustakaan, dimana pada database tersebut terdiri dari file/tabel Anggota, Buku, Peminjaman

1. Anggota

   Anggota={noAngg,NamaAngg,Jkelamin,Tgllhr,Alamat,NoTelp} Dengan data value sebagai berikut :

NoAngg	NamaAngg	JKelamin	Tgllhr	Alamat	NoTelp
11001	Rosa Subagja	Wanita	11/12/1980	Jl.A.Yani No. 10 Kendal	0812123123
11002	Dika Pratama	Pria	01/03/1982	Jl. Cempaka No. 100 Kendal	0711343536
11003	Aisha Putri	Wanita	20/10/1985	Jl. Mayor Ruslan No.21 Kendal	0816123124
11004	Suparjo	Pria	22/04/1983	Jl. Rambutan No.33 Kendal	0852676866
11005	Martina Saleh	Wanita	06/09/1984	Jl. Kebun Jahe No. 15 Kendal	0821222234
11006	Arif Hasan	Pria	09/04/1988	Jl. Karet No. 12 Kendal	0711565755
11007	Gunawan	Pria	14/03/1988	Jl. Kelapa Sawit N0.45 Kendal	0825656567

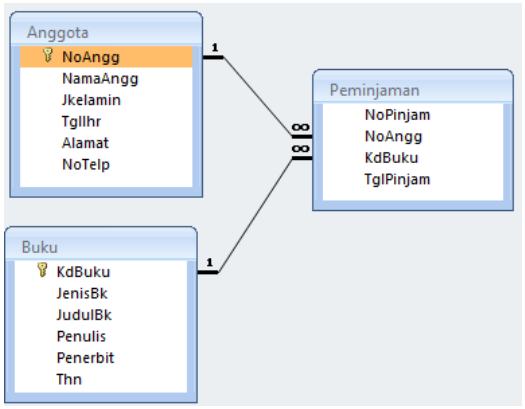
2. Buku Buku={KdBuku,JenisBk,JudulBk,Penulis,Penerbit,Thn} Dengan data value sebagai berikut :

KdBuku	JenisBk	JudulBk	Penulis	Penerbit	Thn
TK001	Teks Komputer	Sistem Manajemen Basis Data	Ir. Bambang Hariyanto, MT	Informatika	2015
JR001	Jaringan	Konsep Teknologi Seluler	Uke Kurniawan, dkk	Informatika	2015
AK001	Akuntansi	Akuntansi Proses Penyusunan Laporan Keuangan	Sutrisno	Ekonisia	2017
AK002	Akuntansi	Analisis SWOT: Teknik Membedah Kasus Bisnis	Freddy Rangkuti	Gramedia Pustaka Utama	2019

3. Peminjaman Peminjaman={NoPinjam,NoAngg,KdBuku,TglPinjam}

NoPinjam	NoAngg	KdBuku	TglPinjam
P001	11001	NV001	01/11/2019
P001	11001	AK002	01/11/2019
P002	11003	JR001	01/11/2019
P002	11003	NV001	01/11/2019
P003	11005	NV002	02/11/2019

Implementasi hubungan (relasi) antar tabel pada basis data perpustakaan di atas adalah sebagai berikut :



Gambar 2.5 Relasi Database Perpustakaan

Pada gambar di atas terlihat bahwa tabel peminjaman terhubung dengan tabel Anggota dan Buku, dikarenakan pada tabel Peminjaman membutuhkan data-data yang ada pada kedua tabel tersebut, yang artinya :

- Seorang Anggota bisa meminjam lebih dari 1 buku
- Sebuah Buku bisa dipinjam lebih dari 1 kali

Operasi manipulasi yang bisa dilakukan pada basis data ini adalah :

1. Insert

   Melakukan pemasukan data-data baru pada file/tabel Anggota, Buku dan Peminjaman.

2. Delete

   Melakukan penghapusan terhadap data yang telah ada pada file/tabel Anggota, Buku dan Peminjaman.

3. Update

   Melakukan perubahan data seperti alamat anggota, jika anggota tersebut pindah alamat, atau perubahan terhadap data lain pada database tersebut sesuai dengan kebutuhan.

4. Retrieve

   Menampilkan informasi mengenai Anggota berdasarkan jenis kelamin, informasi buku berdasarkan jenis buku, informasi buku yang sedang dipinjam dan sebagainya.

F. ABSTRAKSI DATA

Salah satu tujuan dari DBMS adalah menyediakan fasilista antar muka (interface) dalam melihat/menikmati data (yang lebih ramah/user oriented) kepada pemakai/user. Abstraksi data merupakan tingkatan/level dalam bagaimana melihat data dalam sebuah basis data.

Ada 3 level abstraksi data :

1. Level Fisik (Physical Level)

   Merupakan level terendah dalam abstraksi data, yang menunjukkan bagaimana (how) data disimpan dalam kondisi sebenarnya. Level ini menyajikan representasi fisik dari pengorganisasin data, seperti struktur data dan isi dari data itu sendiri. Level ini cukup kompleks, dimana level ini hanya digunakan oleh programmer, yang digunakan untuk melakukan pemrogaman dengan menggunakan database dan DBMS tertentu sesuai deng kebutuhan end-user.

2. Level Lojik/Konseptual (Conceptual Level)

   Level abstraksi data yang lebih tinggi yang menggambarkan data apa

(*what*) yang disimpan dalam basis data, dan hubungan relasi yang terjadi antar data. Level ini menggambarkan keseluruhan basis data. Pemakai tidak memperdulikan kerumitan dalam struktur level phisik lagi, penggambaran cukup dengan memakai kotak, garis dan keterangan secukupnya. Level ini digunakan oleh *database administrator*, yang memutuskan informasi apa yang akan dipelihara dalam satu *database.*

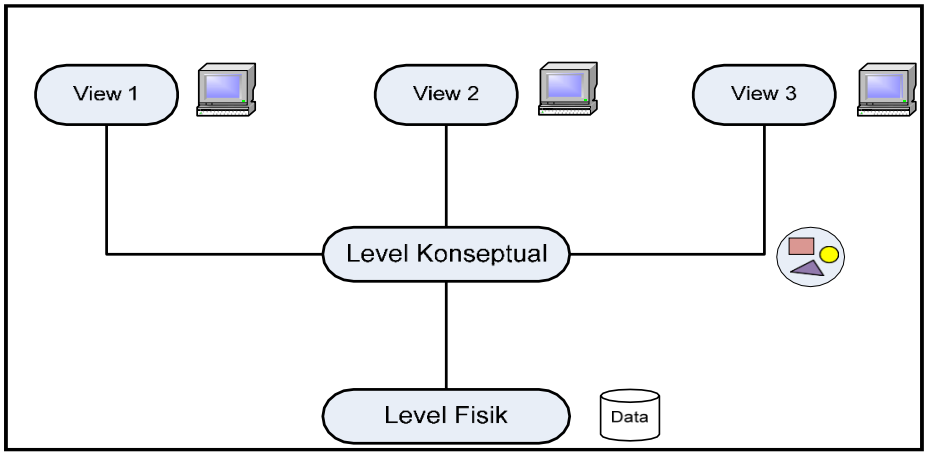
3. Level Penampakan/Pandangan Pemakai (View Level)

   Merupakan level abstraksi data yang paling tinggi yang hanya menunjukkan sebagian dari basis data. Bila pada level konseptual data merupakan suatu kumpulan besar dan kompleks. Pada level ini hanya sebagian saja yang dilihat dan dipakai. Hal ini disebabkan beberapa pemakai database tidak membutuhkan semua isi database. Level ini sangat dekat dengan pemakai (user), dan setiap user kemungkinan hanya membutuhkan sebagian dari database. Ada beberapa kelompok user dengan pandangan berbeda butuh data dalam database, jadi pada level ini yang memakai adalah pemakai akhir atau end- user.

   Misalkan pemakai akhir pada bagian keuangan hanya memakai data untuk *file /table* pembayaran, mahasiswa dan karyawan, tetapi tidak membutuhkan *file / table* buku dan nilai. Demi kemudahan interaksi antara pemakai dengan sistem, maka view level ini didefinisikan. Jadi ada beberapa pandangan disusun untuk mengakses satu sistem *database* yang sama.

   Data yang ‘dinikmati’ pemakai juga bahkan sama sekali berbeda dengan representasi fisiknya, misalnya untuk data yang bisa divisualkan dengan gambar, data yang dapat diperdengarkan dengan suara, dan sebagainya. Data yang diperlihatkan juga bisa saja tidak berasal dari hanya sebuah tabel, tapi mewakili relasi antar tabel, tapi bagi pemakai yang menggunakannya terasa sebagai satu kesatuan data yang kompak.

Bagian Keuangan Bagian Perpustakaan Bagian Akademik



Gambar 2.6 Level Abstraksi Data

BAB 3
NORMALISASI DATA

A. DESKRIPSI NORMALISASI

Normalisasi merupakan salah satu metodologi untuk menciptakan struktur tabel (relasi) dalam basis data dengan tujuan untuk mengurangi kemubaziran data. Selain itu, normalisasi juga sering digunakan sebagai pernagkat verifikasi terhadap tabel yang dihasilkan dari metodologi lain, misalnya tabel hasil ERD. Normalisasi memberikan panduan yang sangat membantu bagi pengembang untuk mencegah penciptaan struktur tabel yang kurang fleksibel atau mengurangi ketidakefisienan.

Kroenke dalam Kadir (65) mendefinisikan normalisasi sebagai proses untuk mengubah relasi yang memilikimasalah tertentu ke dalam dua buah relasi atau lebih yang tak memiliki masalah tersebut. Masalah yang dimaksud Kroenke disebut dengan Anomali.

B. ANOMALI

Anomali adalah proses pada basis data yang memberikian efek samping yang tidak diharapkan (misalnya menyebabkan ketidakkonsistenan data atau menyebabkan susatu data menjadi hilang ketika data lain dihapus).

Anomali trebagi dalam 3 jenis, yaitu :

1. Anomali peremajaan
2. Anomali penghapusan
3. Anomali penyisipan

a. Anomali Peremajaan

Anomali ini terjadi bila ada perubahan terhadap sejumlah data yang mubazir, tetapi tidak seluruhnya diubah, sehingga menyebabkan ketidakkonsistenan data.

NIM	Nama	Alamat	Telp	MataKuliah	Nilai
102	Ratna	Bantul	123456	Basis Data	A
101	Agung	Klaten	112233	Struktur Data	C
102	Ratna	Bantul	123456	Struktur Data	B
102	Ratna	Bantul	123456	Agama	A

Contohnya tabel KHS di atas, dimana tabel ini terdiri atas field NIM, Nama, Alamat, Telp, MataKuliah dan Nilai. Seandainya jika Mahasiswa Ratna dengan NIM 102 berpindah ke kota lain, misalnya Solo, dan pengubahan hanya dilakukan pada data pertama, maka hasilnya akan terlihat sebagai berikut :

NIM	Nama	Alamat	Telp	MataKuliah	Nilai
102	Ratna	Solo	123456	Basis Data	A
101	Agung	Klaten	112233	Struktur Data	C
102	Ratna	Bantul	123456	Struktur Data	B
102	Ratna	Bantul	123456	Agama	A

Terlihat ada ketidakkonsistenan data. Pada Fakta pertama menyatakan bahawa Mahasiswa Ratna dengan NIM 102 beralamat di Solo, sedangkan fakta ke tiga dan kekempat menyatakan bahwa Ratna dengan NIM 102 beralamat di Bantul.

b. Anomali Penyisipan

Anomali ini terjadi jika hendak dilakukan penambahan data baru dan ternyata ada elemen data yang masih kosong dan elemen tersebut justru menjadi kunci. Seperti contoh berikut, terdapat tabel Kursus berikut :

noSiswa	namaKursus	Biaya
10	Bahasa Inggris	60.000
10	Bahasa Prancis	80.000
10	Bahasa Mandarin	60.000
15	Bahasa Inggris	60.000
20	Bahasa Jepang	65.000

Tabel diatas mencatat noSiswa yang mengambil kursus serta biaya kursusnya. Masalh akan timbul jika dibuka kursus baru, misalnya kursus Bahasa Arab dengan biaya 75.000. Karena kursus tersebut masih baru, maka belum ada siswa yang mengambil kursus tersebut, akibatnya data kursus baru tersebut tidak dapat dicatat.

c. Anomali Penghapusan

Anomali penghapusan terjadi sekiranya suatu baris yang tak terpakai dihapus, maka akan menyebabkan data yang lain hilang.

Contohnya pada tabel kursus di atas, jika data siswa dengan noSiswa yang mengambil kursus Bahasa Jepang dihapus,maka mengakibatkan data kursus Bahasa Jepang dengan biaya 65.000 juga akan terhapus.

C. DEPENDENSI (KETERGANTUNGAN)

Dependensi merupakan konsep yang mendasari normalisasi. Dependensi menjelaskan hubungan antar atribut. Dependesni menjelaskan nilai suatu atribut yang menentukan nilai atribut yang lainnya.

Macam-macam dependensi :

1. Dependensi fungsional
2. Dependensi fungsional sepenuhnya
3. Dependensi total
4. Dependensi transitif

a. Dependensi Fungsional

Dependensi fungsional didefinisikan sebagai berikut :

Suatu atribut Y mempunyai dependensi fungsional terhadap atribut X jika dan hanya jika setiap nilai X berhubungan dengan sebuah nilai Y.

Notasi :

X --□ Y

Dibaca :

X secara fungsional menentukan Y

Contoh :

Tabel Pemasok_Barang.

No_Pemasok	Nama_Pemasok
P01	Gunawan
P02	Tuti
P03	Karina

Pada tabel di atas, No_Pemasok secara fungsional menentukan nama_pemasok Ketergantungan fungsional pada tabel di atas adalah :

No_Pemasok --□

Catatan :

Bagian yang terletak disebelah kiri panah biasanya disebut penentu (determinan) dan bagian yang terletak di sebelah kanan tanda panah disebut yang tergantung (dependen)

Tanda { } biasa digunakan kalau ada lebih dari satu atribut, baik pada penentu maupun yang tergantung.

Contoh lain :

Tabel Pembeli

PEMBELI	**KOTA**	**BARANG**	**JUMLAH**
P1	Yogya	B1	10
P1	Yogya	B2	5
P2	Solo	B1	7
P2	Solo	B2	6
P2	Solo	B3	6
P3	Klaten	B3	7
P3	Klaten	B4	6

Ketergantungan Fungsional

Tabel berikut : PEMBELI --

□ KOTA

{PEMBELI, BARANG} --□ JUMLAH

{PEMBELI, BARANG} --□ KOTA

{PEMBELI, BARANG} --□ {JUMLAH, KOTA}

b. Dependensi Fungsional Sepenuhnya

Definisi dependensi fungsional sepenuhnya :

Suatu atribut Y mempunyai dependensi fungsional penuh terhadap atribut X jika : Y mempunyai dependensi fungsional terhadap X Y tidak memiliki dependensi terhadap bagian dari X Contoh :

Tabel Kirim_Barang

No_Pemasok	Nama_Pemasok	No_brg	Jumlah
P01	Bahana	B01	1000
P01	Bahana	B02	1400
P01	Bahana	B03	2000
P02	Sinar Mulia	B03	1000
P03	Harapan	B02	2000

Ketergantungan Fungsional sepenuhnya pada tabel Kirim_Barang di atas :

{No_Pemasok, No_brg} -- Jumlah

Jumlah memiliki ketergantungan fungsional sepenuhnya terhadap No_Pemasok dan No_brg. Dimana Jumlah tergantung dari No_Pemasok dan No_brg, tidak bisa hanya tergantung pada No_Pemasok saja atau No_brg saja.

c. Dependensi Total

Definisi Dependensi Total adalah :

Suatu atribut Y mempunyai dependensi total terhadap atribut X jika : Y memiliki dependensi fungsional tehadap X

Contohnya :

Tabel Pemasok

No_Pemasok	Nama_Pemasok
P01	Bahana
P02	Sinar Mulia
P03	Harapan

Dependensi total pada tabel diatas :

No_Pemasok, Nama_Pemasok

Dengan asumsi bahwa tidak ada nama Pemasok yang sama.

d. Dependensi Transitif

Definisi dependensi transitif adalah :

Atribut Z mempunyai dependensi transitif terhadap X bila :

Y memiliki dependensi fungsional terhadap X Z memiliki dependensi fungsional terhadap Y Notasi :

R (X □ Y, Y □ Z, maka X □ Z)

Dibaca : Atribut Z pada relasi R dikatakan tergantung transitif pada atribut X , jika atribut Y tergantung pada atribut X pada relasi R dan atribut Z tergantung pada atribut Y pada relasi.

Contoh :

Kuliah	Ruang	Tempat	Waktu
Jaringan Komputer	Merapi	Gedung Utara	Senin, 08.00-09.50
Pengantar Basis Data	Merbabu	Gedung Utara	Selasa, 08.00-09.50
Matematika I	Rama	Gedung Selatan	Rabu, 10.00-11.50
Sistem Pakar	Sinta	Gedung Selatan	Kamis, 08.00-09.50
Kecerdasan Buatan	Merapi	Gedung Utara	Selasa, 10.00-11.50

Pada relasi ini :

Kuliah { Ruang, waktu} Ruang Tempat

Terlihat bahwa :

Kuliah - Ruang - Tempat

D. DIAGRAM DEPENDENSI FUNGSIONAL

Diagram dependensi fungsional adalah diagram yang digunakan untuk menggambarkan dependensi fungsional. Diagram ini menunjukkan hubungan antara atribut yang menjadi penentu atribut lainnya, hubungan yang dinyatakan dengan tanda panah.

E. DEKOMPOSISI TAK HILANG

Pada Proses normalisasi seringkali terjadi pemecahan sebuah relasi menjadi dua relasi atau lebih. Proses pemecahan ini biasa disebut dengan dekomposisi. Dekomposisi yang dilakukan adalah dekomposisi tak hilang,yang artinya bahwa tidak ada informasi yang hilang ketika relasi dipecah menjadi relasi-relasi yang lain.

Contoh :

Bentuk Relasi semula :

NIM	NAMA	PROGRAMSTUDI
101	BUDI	TI
102	ARI	TI
103	BUDI	SI

1. Contoh Dekomposisi Tak Hilang

Relasi : NIM_NAMA

NIM	NAMA
101	BUDI
102	ARI
103	BUDI

Relasi : NIM_PROG

NIM	PROGRAMSTUDI
101	TI
102	TI
103	SI

2. Contoh Dekomposisi Hilang

Relasi: NIM_NAMA

NIM	NAMA
101	BUDI
102	ARI
103	BUDI

Relasi : NAMA_PROG

NAMA	PROGRAMSTUDI
BUDI	TI
ARI	TI
BUDI	SI

Pada dekomposisi di atas :

1) Pada kasus (a), dekomposisi bersifat tak hilang. Berdasarkan kedua relasi hasil dekomposisi (terkadang disebut proyeksi), relasi semula bisa diperoleh kembali.
2) Pada kasus (b), terdapat suatu kerancuan, nama dengan NIM 101 adalah BUDI, tetapi BUDI dengan NIM 101 mengambil PROGRAMSTUDI apa tidak jelas pada tabel NAMA_PROG.

F. BENTUK NORMAL

Bentuk normal adalah suatu aturan yang dikenakan pada relasi-relasi dalam basis data dan harus dipenuhi oleh relasi-relasi tersebut pada level-level normalisasi.

Suatu relasi atau tabel dikatakan berada dalam bentuk normal tertentu jika memenuhi kondisi- kondisi tertentu.

Beberapa level yang biasa digunakan pada normalisasi adalah :

- ✓ Bentuk Normal Pertama (1NF)
- ✓ Bentuk Normal Kedua (2NF)
- ✓ Bentuk Normal Ketiga (3NF)
- ✓ Bentuk Normal Boyce-Codd (BCNF)
- ✓ Bentuk Normal Keempat (4NF)
- ✓ Bentuk Norma Kelima (5NF)

1. Bentuk Normal Pertama

Bentuk normal pertama biasa dikenakan pada tabel yang belum ternormalisasi. Tabel yang belum ternormalisasi adalah tabel yang memiliki atribut yang berulang. Contoh :

NIP	Nama	Jabatan	Keahlian	Pengalaman(Tahun)
107	Erika	Analis Senior	Java PHP	6 1
108	Erin	Analis Junior	Delphi Java	2 2
109	Paijo	Programmer	Delphi Java PHP	1 1 1

Pada contoh di atas, Field Keahlian menyatakan atribut yang berulang. Misalnya Rudianto memiliki dua keahlian, Firmansyah memiliki 3 keahlian.

Definisi Bentuk Normal Pertama :

- Bentuk normal 1NF terpenuhi jika sebuah tabel tidak memiliki atribut bernilai banyak (multivalued attribute), atribut composite atau kombinasinya dalam domain data yang sama.
- Setiap atribut dalam tabel tersebut harus bernilai atomic (tidak dapat dibagi-bagi lagi)

Suatu relasi dikatakan dalam bentuk normal pertama jika dan hanya jika setiap atribut bernilai tunggal untuk setiap baris. Data yang tak ternormalisasi pada tabel di atas diubah kebentuk normal pertama dengan cara membuat setiap baris berisi kolom dengan jumlah yang sama dan stiap kolom hanya mengandumg satu nilai.

Nip	Nama	Jabatan	Keahlian	Pengalaman
107	Paijo	Analis Senior	Java	6
107	Paijo	Analis Senior	PHP	1
108	Erika	Analis Junior	Delphi	2
108	Erika	Analis Junior	Java	2
109	Erin	Programmer	Delphi	1
109	Erin	Programmer	Java	1
109	Erin	Programmer	PHP	1

2. Bentuk Normal Kedua

Bentuk normal kedua didefinisikan berdasarkan dependensi fungsional. Suatu relasi berada dalam bentuk normal kedua jika dan hanya jika:

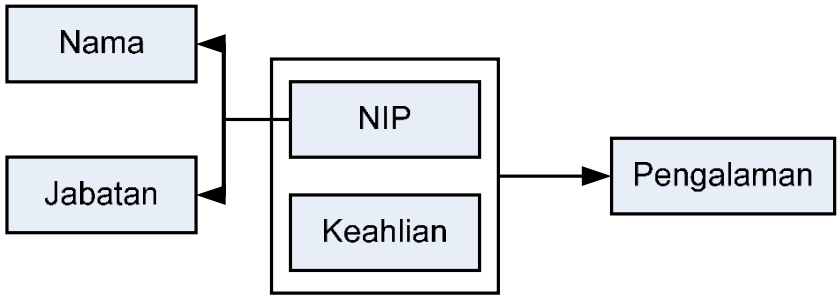
- Bentuk normal 2NF terpenuhi dalam sebuah tabel jika telah memenuhi bentuk 1NF, dan semua atribut selain primary key, secara utuh memiliki Functional Dependency pada primary key
- Sebuah tabel tidak memenuhi 2NF, jika ada atribut yang ketergantungannya (Functional Dependency) hanya bersifat parsial saja (hanya tergantung pada sebagian dari primary key)
- Jika terdapat atribut yang tidak memiliki ketergantungan terhadap primary key, maka atribut tersebut harus dipindah atau dihilangkan

Atau dengan kata lain, bentuk normal kedua mensyaratkan setiap atribut bergantung pada kunci primer. Untuk mengubah suatu relasi bentuk normal pertama ke bentuk normal kedua perlu dilakukan dekomposisi.

Contoh pada tabel 1NF diatas, Nama dan jabatan mempunyai dependensi sepenuhnya terhadap NIP (sebab untuk setiap NIP yang sama, Nama dan Jabatan juga sama).

Sedangkan Keahlian dan pengalaman tidak demikian.

Untuk mengubah suatu relasi dari bentuk 1NF, ke bentuk 2NF maka harus dilakukan dekomposisi. Proses dekomposisi dapat dilakukan dengan menggambarkan diagram dependensi fungsional terlebih dahulu.



Berdasarkan diagram di atas maka NIP – {Nama, Jabatan}

{NIP, Keahlian} – Pengalaman

Berdasarkan diagram dependensi fungsional diatas maka pendekomposisian menghasilkan dua buah tabel :

NNJ (<u>NIP</u>, Nama, Jabatan)

NKP (<u>NIP</u>, Keahlian,Pengalaman)

Tanda garis bawah menggambarkan kunci primer masing-masing relasi/tabel.

BAB 4
MODEL ENTITY RELATIONSHIP

Model ER (Entity Relationship) merupakan model data konseptual tingkat tinggi untuk perancangan basis data. Model data konseptual adalah model himpunan konsep yang mendeskripsikan struktur basisdata, transaksi pengambilan dan pembaruan basisdata. Model ER tidak bergantung pada DBMS dan platform perangkat keras tertentu. Model ER adalah persepsi terhadap dunia nyata yang terdiri atas objek-objek dasar yang disebut entitas dan keterhubungan (relationship) antar entitas-entitas itu. Konsep paling dasar model ER adalah entitas, relationship dan atribut.

A. ENTITAS (ENTITY)

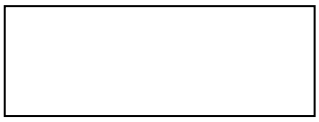
Sebuah entiti adalah sebuah benda (thing) atau objek (object) di dunia nyata yang dapat dibedakan dari semua objek lainnya. Himpunan entitas (entity sets) adalah sekumpulan entiti yang mempunyai tipe yang sama. Kesamaan tipe ini dapat dilihat dari atribut/properti yang dimiliki setiap entiti.

Pemilihan entitas adalah langkah utama dalam permodelan ER yang akan digunakan untuk memodelkan suatu sistem. Entitas dapat berupa objek real dalam dunia nyata misalnya Mahasiswa, Pekerja, Mobil dan sebagainya, namun entitas dapat berupa objek abstrak seperti rekening.

Contohnya :

- Kumpulan orang yang menyimpan uang pada sebuah bank dapat didefinisikan sebagai sebuah entiti set nasabah.
- Kumpulan orang yang belajar di perguruan tinggi didefinisikan sebagai mahasiswa.

Entity set dilambangkan dengan bentuk persegi seperti gambar berikut :



Gambar Lambang Entity Set

1. Atribut

Atribut merupakan pendeskripsian karakteristik dari entitas. Atribut adalah properti atau ciri atau karakteristik dari tipe entitas yang dipentingkan di satu sistem/organisasi. Setiap atribut entitas menspesifikasikan properti tertentu dari entitas. Atribut digambarkan dalam bentuk lingkaran atau elips. Atribut yang menjadi kunci entitas atau key diberi garis bawah.

Contohnya :

Penduduk memiliki atribut serupa seperti nama, TanggalLahir, JenisKelamin, alamat dan sebagainnya.

Domain attribut adalah himpunan nilai yang dapat diberikan ke suatu atribut, menyatakan himpunan dimana nilai atribut berasal. Masing-masing atribut diasosiasikan dengan satu himpunan nilai atau domain. Domain mendefenisikan nilai-nilai yang mempunyai potensi diberikan ke atribut. *Domain* adalah satu kumpulan nilai-nilai yang valid untuk atribut dengan karakteristik :

a. Tipe data
b. Panjang
c. Format (mask)
d. Nilai yang diijinkan
e. Konstrain(misalnya range)
f. Arti
g. Keunikan
h. Dukungan terhadap
i. Nilai *default*

2. Kunci Entitas

Konsep kunci pada himpunan entitas dianalogikan dengan kunci relasi di model relasional. Namun, nilai atribut di entitas dapat berupa nilai himpunan atau nilai majemuk, bukan hanya nilai tungggal. Atribut bernilai himpunan ini dapat menjadi bagian dari kunci.

Contoh :

Kunci pada himpunan entitas S, adalah himpunan atribut, A, sehingga

a. Tidak ada dua entitas S yang mempunyai nilai sama untuk tiap atribut di A
b. Tidak ada *subset* di A yang dapat menjadi kunci di S. Dengan demikian, kunci mempunyai properti minimal.

Entitas yang tidak mempunyai atribut kunci. Entitas lemah diidentifikasikan dengan menghubungkan entitas tertentu dari tipe entitas yang lain ditambah atribut dari entitas lemah.

Kunci Super (*Super Key*)

Himpunan satu atribut atau lebih yang memungkinkan identifikasi secara unik entitas pada himpunan entitas itu.

Calon Kunci (*candidate key*)

Kunci *super* yang minimal

Kunci Utama (*primary key*)

Calon kunci yang dipilih perancang basisdata sebagai alat utama untuk mengidentifikasi entitas pada himpunan entitas.

Kunci Alternatif (*alternate key*)

Calon kunci yang tidak dipilih sebagai kunci utama.

Entitas yang tidak memiliki kunci utama disebut entitas lemah, entitas yang mempunyai kunci utama disebut entitas kuat.

3. Entitas Kuat dan Entitas Lemah

a. Entitas Kuat

Entitas yang mempunyai atribut kunci. Entitas ini bersifat mandiri, keberadaanya tidak bergantung pada entitas lainnya. Percepatan entitas kuat selalu memiliki karakteristik yang unik disebut identifier (sebuah atribut tunggal atau gabungan atribut-atribut yang secara unik dapat digunakan untuk membedakannya dari entitas kuat yang lain).

b. Entitas Lemah

Entitas yang tidak mempunyai atribut kunci. Entitas lemah diidentifikasikan dengan menghubungkan entitas tertentu dari tipe entitas yang lain ditambah atribut dari entitas lemah.Dimana himpunan entitas yang keberadaannya bergantung dengan keberadaan himpunan entitas lain.

Contohnya :

1) Entitas Transaksi bergantung adanya entitas Rekening.
2) Entitas Pekerjaan (*task*) bergantung adanya Proyek.

4. Skema Entitas

Skema relasi yang lengkap berisi; Nama relasi yang uniq dalam sebuah basisdata (database).

Nama-nama attribut di relasi diasosiasikan sebagai nama-nama domain. Attribut merupakan nama yang diberikan kepada kolom di suatu relasi. Semua kolom harus diberi nama yang uniq. Nama domain hanya nama yang diberikan ke suatu himpunan nilai yang terdefinisi bagus.

Konstrain-konstrain integritas yang berupa batasan-batasan pada relasi, yaitu batasan-batasan pada tupel-tupel yang muncul di relasi. Skema relasi disebut legal, bila telah memenuhi semua konstrain integritas yang diasosiasikan sebagai sebuah skema.

Contoh Skema Relasi;

Dosen (ID:CHAR[10], Nama:CHAR[30], Alamat:CHAR[25], Kota:CHAR[15],
KodePos:CHAR[6], Telp:CHAR[15], Fax:CHAR[15])

Skema relasi tersebut memiliki jumlah attribut (derajat relasi) 7.

Semua attributnya diasosiasikan bertipe CHAR, namun memiliki nama yang berbeda-beda. Tipe-tipe yang sama diantara attributnya bukan berarti memiliki domain yang sama.

B. Representasi Entitas di Diagram ER

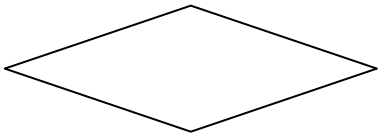
Relasi merupakan representasi data secara logik bukan representasi secara fisik. Relasi mendeskripsikan struktur data tanpa memperhatikan cara pengaksesan dan penyimpanannya secara fisik, sehingga cara analisa dan perancangannya pun tidak perlu memusingkan. Tipe entitas di representasikan sebagai persegi dan atribut-atributnya direpresentasikan sebagai bulat telur (oval) yang diikatkan dengan persegi panjang dengan sesuatu garis. Atribut-atribur yang bernilai himpunan direpresentasikan dengan oval yang bergaris ganda.

C. Representasi Entitas di Model Relational

Masing-masing tipe entitas dapat dikonversi menjadi relasi dan masing-masing atributnya dikonversi menjadi atribut relasi.

D. Keterhubungan (*Relationship*)

Relasi adalah hubungan di antara beberapa entiti. *Relationship set* adalah sekumpulan relasi yang mempunyai tipe yang sama. *Relationship sets* digambarkan dengan bentuk diamond seperti gambar berikut :



Tiga tipe *Relationship* yaitu :

1. *Relationship* keberadaan (misalnya karyawan mempunyai anak)
2. *Relationship* fungsional (misalnya professor mengajar mahasiswa)
3. *Relationship* kejadian (misalnya pembeli memberi pesanan)

Sebuah Relationship memiliki derajat (*degree*) berdasarkan jumlah entitas yang terhubung antara lain: Unary, Binary, Ternary dan Quaternary.

a. Unary

Tipe hubungan ini berbarti hubungan yang terjadi antara dirinya sendiri dlam sebuah *entitas*; contohnya: entitas Pegawai yang memiliki Relationship Supervisor. Pegawai yang menjadi Supervisor juga berasal dar entitas yang sama yaitu Pegawai sedangkan entitas yang dituju juga pada entitas Pegawai.

b. Binary

Tipe *hubungan* ini antara dua entitas;

contohnya: entitas Pegawai dan entitas Kantor Cabang.

c. Ternary

Tipe hubungan ini antara tiga entitas;

contohnya :entitas Sales, Produk dan Pelanggan yang memiliki sebuah Relationship yang bernama Penjualan.

E. Atribute Relationalship

Sebagaimana entitas, *relationship* juga dapat mempunyai atribut-atribut. Hubungan atribut mendefinisikan sifat-sifat asosiasi yang ada (termasuk apakah atribut lain dapat diakses melalui atribut yang diberikan) antara atribut tertentu dan atribut lain. Hubungan atribut secara otomatis didefinisikan antara atribut kunci dan setiap atribut non-kunci.

F. Kunci Relationalship

Kunci adalah sekumpulan atribut yang secara unik mengidentifikasi entitas : Namun, atribut- atribut di relationship tidak secara penuh dapat mencirikan *relationship*. Peran-peran yang dimainkan partisipan-partisipan harus juga terlibat dalam pembentukan kunci *relationship*.

G. Konstrain Relationalship

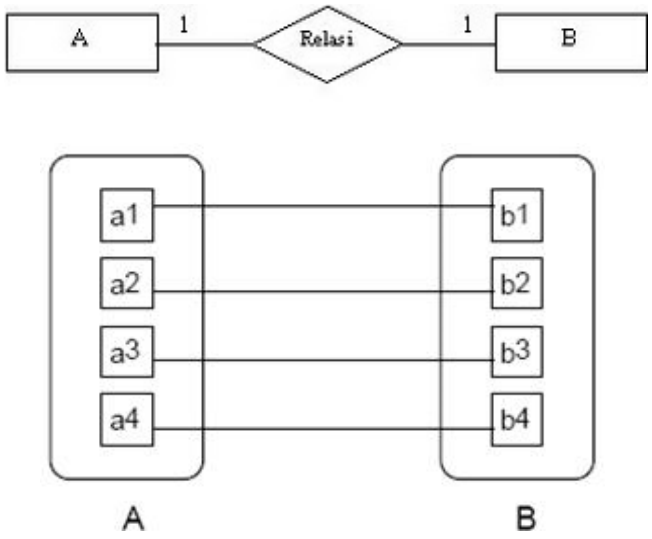
Derajat *relationship* adalah jumlah entitas di *relationship*. Entitas-entitas yang terlibat di *relationship* disebut derajat *relationship*. *Relationship* berderajat dua disebut biner. *Relationship* berderajat tiga disebut *tenary*. *Relationship* berderajat empat disebut *quartenary*. Terdapat dua tipe batasan pada *relationship* yaitu konstrain kardinalitas dan partisipasi.

H. Konstrain Kardinalitas

Konstrain kardinalitas menyatakan rasio jumlah entitas terhadap entitas lain yang diasosiasikan pada *relationship*, yaitu :

1. **Derajat hubungan 1:1**

   Derajat hubungan 1:1 terjadi bila setiap anggota entitas A hanya boleh berpasangan dengan satu anggota dari entitas B, dan sebaliknya tiap anggota entitas B hanya boleh berpasangan dengan satu anggota dari entitas A.

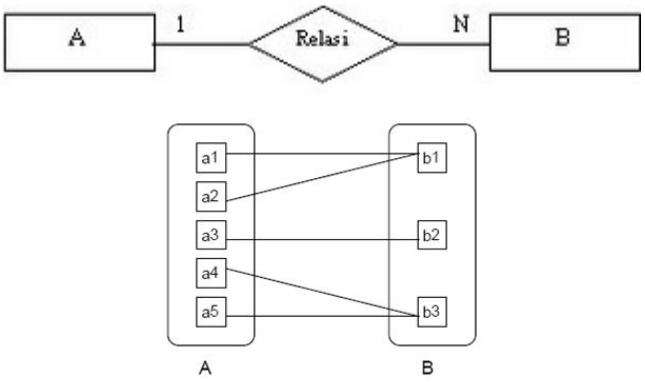


Keterangan :

Sebuah entity A diasosiasikan pada sebuah entity B, dan sebuah entityB diasosiasikan dengan paling banyak sebuah entity A.

2. **Satu ke banyak (one to many) atau 1:N**

Derajat hubungan ini terjadi bila tiap anggota entitas A boleh berpasangan dengan lebih dari satu anggota entitas B. Sebaliknya tiap anggota entitas B hanya boleh berpasangan dengan satu anggota entitas A.

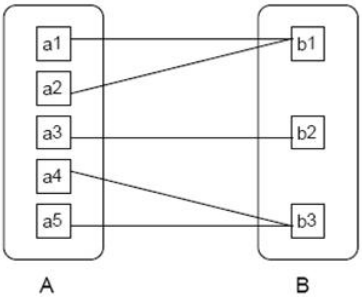


Keterangan :

Sebuah entity A diasosiasikan dengan sejumlah entity B, tetapi entity Bdapat diasosiasikan paling banyak satu entity A.

3. **Banyak ke banyak (many to many) atau M:N**

Derajat hubungan antar entitas m:n terjadi bila tiap anggota entitas A dapat berpasangan dengan lebih dari satu anggota entitas B. Sebaliknya setiap anggota entitas B juga dapat berpasangan dengan lebih dari satu anggota entitas



A.

Keterangan :

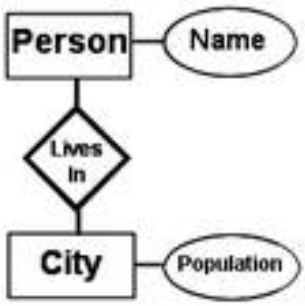
Suatu entity A dapat diasosiasikan dengan sejumlah entity B dan entity B dapat diasosiasikan dengan sejumlah entity di A.

I. ENTITY RELATIOANAL DIAGRAM (ERD)

ERD merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi. Diagram ER sering menggunakan simbol-simbol untuk mewakili tiga jenis informasi yang berbeda. Kotak (*box*) biasanya digunakan untuk mewakili entitas. Berlian (*Diamonds*) biasanya digunakan untuk mewakili hubungan dan oval digunakan untuk mewakili atribut.

Ada tiga elemen dasar dalam model ER:

1. Entitas adalah "sesuatu" tentang apa yang kita mencari informasi.
2. Atribut adalah data yang kami kumpulkan tentang entitas.
3. Hubungan menyediakan struktur yang dibutuhkan untuk menarik informasi dari beberapa entitas.



Keterangan :

Perhatikan contoh diatas ; sebuah database yang berisi informasi tentang penduduk kota. Para diagram ER ditunjukkan pada gambar di atas berisi dua entitas - orang dan kota-kota. Ada satu "Lives In" hubungan. Dalam contoh diatas hanya ada satu atribut yang terkait dengan setiap entitas. Orang-orang memiliki nama dan kota memiliki populasi.

BAB 5
STRUKTUR QUERY LANGUAGE (SQL)

Sistem basis data seharusnya menggunakan sebuah kalimat yang mudah agar dapat dipahami saat digunakan atau yang lebih dikenal dengan *user friendly*. SQL adalah sebuah bahasa untuk membuat sebuah sistem basis data sebagai bahasa *query* yang lebih *marketable*. Beberapa kelebihan dari bahasa SQL adalah karena SQL menggunakan kombinasi aljabar relational dan kalkulus relational. SQL mempunyai kemampusn untuk mendefinisikan struktur data, modifikasi data dalam basis data dan menentukan konstrain sekuriti.

Structured Query Language (SQL) adalah bahasa standard untuk melakukan berbagai operasi data pada database, diantaranya mendefinisikan tabel, menampilkan data dengan kriteria tertentu, menambahkan data hingga menghapus data tertentu. Penggunaan SQL pada beberapa bahasa pemrograman secara umum relatif sama.

A. PENGENALAN MYSQL

MySQL adalah Sebuah program database server yang mampu menerima dan mengirimkan datanya sangat cepat, multi user serta menggunakan peintah dasar SQL (Structured Query Language).

MySQL merupakan dua bentuk lisensi, yaitu FreeSoftware dan Shareware. MySQL yang biasa kita gunakan adalah MySQL FreeSoftware yang berada dibawah Lisensi GNU/GPL (General Public License).

MySQL Merupakan sebuah database server yang free, artinya kita bebas menggunakan database ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensinya. MySQL pertama kali dirintis oleh seorang programmer database bernama ***Michael Widenius*** **.** Selain database server, MySQl juga merupakan program yang dapat mengakses suatu database MySQL yang berposisi sebagai Server, yang berarti program kita berposisi sebagai Client. Jadi MySQL adalah sebuah database yang dapat digunakan sebagai Client mupun server.

Database MySQL merupakan suatu perangkat lunak database yang berbentuk database relasional atau disebut Relational Database Management System (RDBMS) yang menggunakan suatu bahasa permintaan yang bernama SQL (Structured Query Language).

Kelebihan dan Keuntungan memakai MySQL :

Jika kita menggunakan sistem database, ada beberapa pilihan, antara lain :

- MySQL
- Oracle
- PostreSQL
- mSQL
- Microsoft SQL Server

Ketika di bandingkan antara MySQL dengan system managemen database yang lain, perlu di perhitungkan apa yang paling penting untuk kita. Apakah performa, support, fitur-fitur SQL, kondso keamanan dalam license atau masalah harga. Dengan pertimbangan tersbut, database MySQL memiliki beberapa kelebihan dan keuntungan dibanding database lain, di antaranya adalah :

- Banyak ahli berpendapat MySQL merupakan server tercepat.
- MySQL merupakan sistem DBMS yang terbuka atau lebih di kenal denan OpenSource (kode sumbernya terbuka), yaitu software ini bersifat free atau bebas digunakan oleh perorangan atau instansi tanpa harus membeli atau membayar kepada permbuatannya.
- MySQL mempunyai performa yang tinggi tapi simpel
- Database MySQLmengerti bahasa SQL
- MySQL dapat di akses melalui protokol ODBC buatan Microsoft.
- Semua klien dapat mengakses server dalam satu waktu tanpa harus menunggu yang lain untuk mengakses database.
- Database MySQL dapat di akses dari semua tempat di internet dengan hak akses tertentu.
- MySQL dapat berjalan di berbagai operating sistem seperti Linux, Windows, Solari dan lain-lain.
-

B. PENGENALAN SQL

SQL sebagai bahasa pemersatu dalam dunia database, dan SQL server beperan sebagai penampung data untuk keperluan penyimpanan dengan daya tampung besar, dapat di contohkan jika suatu data yang di tampung dalam text, Ms. Excel atau Ms. Acces memiliki kapasitas atau batasannya untuk menampung data, dengan menggunakan SQL Server data yang di simpan dapat dikatakan tidak terbatas, disamping itu kemudahan terhadap koneksi dengan program aplikasi yang akan di buat.

Apa yang dapat di lakukan oleh bahasa SQL :

- SQL mampu mengeksekusi query di dalam database
- SQL mampu mengambil data dari sebuah database
- SQL mampu menyisipkan catatan di sebuah database
- SQL mampu menghapus catatan di sebuat database
- SQL mampu membuat database baru
- SQL mampu membuat tabel baru di sebuah database
- SQL mampu mengatur hak akses pada tabel, prosedur dan tampilan
- SQL mampu mengatur hak akses pada tabel, prosedur dan tampilan
- Dan asih banyak lagi yang dapat di lakukan oleh SQL

C. SEJARAH SQL

Sejarah SQL dimulai artikel seorang peneliti dari IBM bernama EF Cod yang membahas tentang ide pembuatan basis data relasional pada bulan Juni 1970. Artikel ini juga membahas kemungkinan pembuatan bahasa standar untuk mengakses data dalam basis data tersebut. Bahasa tersebut kemudia diberi nama SEQUEL (Structured English Quedy Language).

SQL merupakan bahasa pemrograman yang di rancng khusus untuk mengirimkan suatu perintah query (pengaksesan data berdasarkan pengalaman tertentu) terhadap sebuah database. Kebanyakan software database yang ada saat ini dapat di akses melalui SQL. Setiap aplikasi yang spesifik dapat mengimplementasikan SQL. Setiap aplikasi yang spesifik dapat mengimplementasikan SQL secara sedikit berbeda, tetapi seluruh database SQL mendukung subset standar yang ada.

SQL (*Structured Query Language*) adalah satu bahasa generasi level -4 yang awalnya di kembangkan oleh IBM di San Jose Research Laboratory. Berbeda dengan bahasa pemrograman level ke 3. SQL adalah bahasa yang bersifat *request oriented* dan bersifat dipelajari karena sintaksis yang digunakan hampir menyerupai bahasa yang di gunakan oleh manusia untuk berkomunikasi. Oleh karena itu, SQL lebih fleksibel dalam penggunaannya. Selain itu, SQL jga bersifat non case sensitif. Banyak vendor pembuat DBMS yang saat ini menggunakan SQL sebagai standarisasi dalam produk mereka, seperti ORACLE, Micorosoft SQL Server, PostGreSQL, dan MySQL.

D. KELOMPOK PERNYATAAN SQL

1. DDL (Data Definition Language)

Yaitu bahasa yang memiliki kemampuan untuk mendefinisikan data yang berhubungan dengan pembuatan dan penghapusan objek seperti tabel, indeks, bahkan basis datanya sendiri. Misalnya, CREATE, DROP dan ALTER.

Data Definition Language (DDL) merupakan sub bahasa SQL yang di gunakan untuk membangun kerangka database. Ada tiga perintah yang termasuk dalam DDL, yaitu sebagai berikut,

CREATE : Perintah ini di gunakan untuk membuat, termasuk di antaranya membuat database baru, tabel baru, view baru, dan kolom.

ALTER : Perintah ini di gunakan untuk mengubah struktur tabel yang telah dibuat. Pekerjaannya mencakup mengganti nama tabel, menambah kolom, mengubah kolom, menghapus kolom, maupun memberikan atribut pada kolom.

DROP : perintah ini digunakan untuk menghapus databse dan tabel.

2. DML (Data Manipulation Language)

Data Manipulation Language (DML) merupakan sub bahasa SQL yang di gunakan untuk memanipulasi data dalam database yang telah dibuat, perintah yang di gunakan di antaranya sebagai berukut :

INSERT : perintah ini di gunakan untuk menyisipkan atau memasukkan data baru ke dalam tabel. Penggunannya setelah database dan tabel selesai dibuat.

UPDATE : perintah ini digunakan untuk memperbarui data lama menjadi data terkini. Jika anda memiliki data yang salah atau kurang Up To Date dengan kondisi sekarang maka dapat diubah isi datanya dengan menggunakan perintah UPDATE.

SELECT : Perintah ini di gunakan untuk mengambil data atau menampilkan data dari satu tabel atau beberapa tabel dalam relasi. Data yang diambil dapat kita tampilkan dalam layar prompt MySQL secara langsung maupun ditampilkan pada tampilan aplikasi.

DELETED : perintah ini di gunakan untuk menghapus data dari tabel biasanya data yang dihapus adalah data yang tidak di perlukan lagi. Pada saat

menghapus data, perintah yang telah dijalankan tidak dapat digagalkan sehingga data yang telah hilang tidak dapat di kembalikan lagi.

Selain untuk mengambil inforasi dari database. Anda juga dapat menggunakan perintah SQL untuk memanipulasi data. Proses tersebut meliputi menambah, menghapus dan mengedit data.

Perintah memanipulasi data sangat sering di gunakan dalam aplikasi database dan bahkan di katan menjadi inti sebuah aplikasi. Sebuah tabel dapat diisi dengan data, dihapus, maupun diedit datanya. Perintah-perintah tersebut dilaksanakan berdasarkan kriteria tertentu menggunakan keyword WHERE, BETWEEN maupun LIKE.

BAB 6
BEKERJA DENGAN MYSQL

MySQL adalah RDBMS yang cepat dan mudah digunakan, serta sudah banyak di gunakan untuk berbagai kebutuhan. MySQL dikembangkan oleh MySQL AB Swedia. Berikut ini hal-hal yang menyebabkan MySQL menjadi begitu populer :

- Berlisensi open source, sehingga powerful dan menyediakan fitur yang lengkap.
- Menggunakan bentuk standart bahasa data SQL
- Dapat bekerja dengan banyak sistem operasi dan dengan bahasa bahasa pemrograman seperti PHP, PERL, C, C++, JAVA, dan lain-lain.
- Bekerja dengan cepat dan baik, bahkan dengan data set yang banyak.
- Sangat mudah digunakan dengan PHP untuk pengembangan aplikasi web.
- Mendukung banyak database, sampai 50 juta baris atau lebih dalam suatu tabel.
- Dapat dikostumisasi sesuai dengan keinginan anda.

1. Membuat Tabel di dalam MySql menggunakan cmd

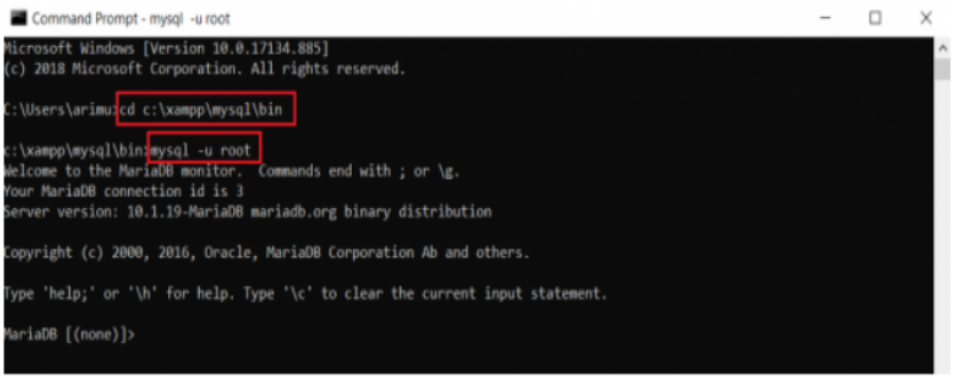
Perintah mysql –u root adalah perintah untuk masuk ke database dengan use root.

A. Mengaktifkan Direktory MySQL

Run → **CMD**

Setelah CMD terbuka ketik : **cd c:\xampp\mysql\bin** kemudian tekan enter

Lalu selanjutnya ketik : **mysql -u root**



Perintan MySQL –u root adalah perintah untuk masuk ke database mysql dengan user root

B. Membuat Database

Membuat database dengan menggunakan perintah :

Create DATABASE database_name;

```
MariaDB [(none)]> create database akademik;
Query OK, 1 row affected (0.00 sec)
```

C. Quey untuk melihat daftar database.

Show database;

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| akademik           |
| csv_db             |
| data2018           |
| db_ecommerce       |
| db_select          |
| db_smk             |
| ecommerce          |
| information_schema |
| kafilaexpress      |
| mysql              |
| performance_schema |
| phpmyadmin         |
| se                 |
| tes                |
| test               |
| tugasakhir_gis     |
+--------------------+
16 rows in set (0.00 sec)
```

Untuk menggunakan database (Akademik) yang baru saja kita gunakan perintah.

Use akademik;

```
MariaDB [(none)]> use akademik;
Database changed
```

Untuk melihat tabel di dalam database akademik yang baru saja kita buat gunakan perintah.

Show tables;

```
MariaDB [akademik]> show tables;
Empty set (0.00 sec)
```

Terlihat bahwa tidak ada tabel yang ada di dalam database akademik

(kosong)

D. Membuat tabel mahasiswa

Selanjutnya adalah perinta query untuk membuat tabel di dalam mysql perintahnya adalah sebagai berikut

```
create table mahasiswa (
    nik char(10) not null primary key,
    nama varchar(50) not null,
    jk char(1),
    tanggal_lhr date,
    jurusan varchar(30),
    umur int
    );
```

```
MariaDB [akademik]> create table mahasiswa(
    -> nik char(10) not null primary key,
    -> nama varchar(50) not null,
    -> jk char(1),
    -> tanggal_lhr date,
    -> jurusan varchar(30),
    -> umur int
    -> );
Query OK, 0 rows affected (0.16 sec)
```

Adapun penjelasan dari perintah di atas adalah :

Pada perintah dia tas kita membuat tabel dengan nama mahasiswa pada database akademik. Adapaun struktur dari tabel mahasiswa tersebut terdapat 7 field :

Nik

Field nik menggunakan tipe data char dengan panjang 10 karakter, dengan default data tidak boleh kosong. Fiekd nik ini kita jadikan sebagai primary key.

Nama

Field nama menggunakan tipe data varchar dengan panjang 50 karakter dengan status keterisian data tidak boleh ksosong.

Jk

Jenis Kelamin atau biasa di singkat dengan JK ini menggunakan tipe data char dengan panjang 1 karakter. Nantinya nilai pada field ini akan mendefinisikan jenis kelamin seseorang misalnya 1 untuk laki laki dan 2

untuk perempuan.

Tanggal_lhr

Tanggal_lahir ini menggunakan tipe data date, tipe ini terdiri dari tahun bukan dan tanggal secara default dari mysql.

Jurusan

Field jurusan menggunakan tipe data varchar dengan panjang karakter 30

umur

field umur menggunakan tipe data integer (bilangan bulat)

E. Melihat struktur table di MySQL

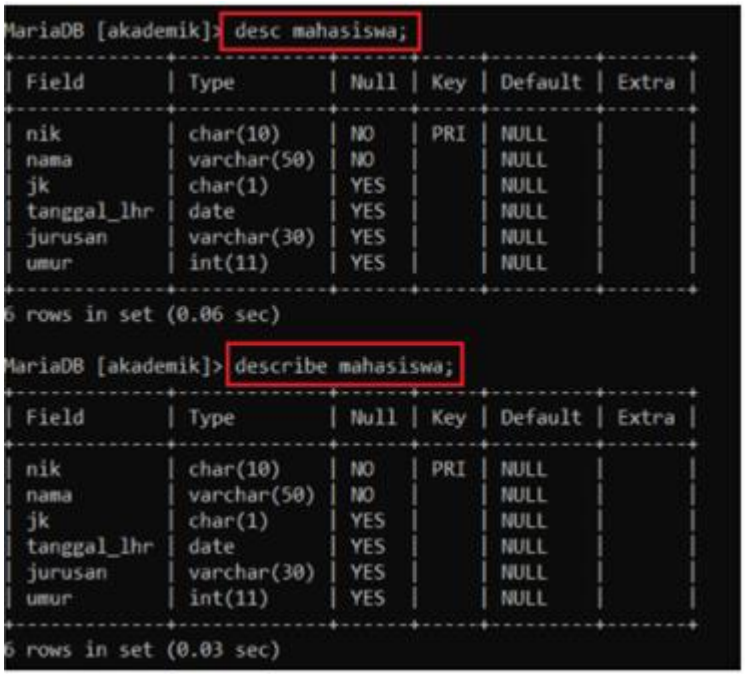
Jadi untuk melihat deskripsi dari struktur tabel mmahasiswa yang baru saja kita buat menggunakan perintah :

```
desc mahasiswa;
```

Atau juga bisa dengan menggunakan :

```
describe mahasiswa;
```

Dan hasilnya adalah sebagai berikut :



```
MariaDB [akademik]> desc mahasiswa;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| nik         | char(10)    | NO   | PRI | NULL    |       |
| nama        | varchar(50) | NO   |     | NULL    |       |
| jk          | char(1)     | YES  |     | NULL    |       |
| tanggal_lhr | date        | YES  |     | NULL    |       |
| jurusan     | varchar(30) | YES  |     | NULL    |       |
| umur        | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
6 rows in set (0.06 sec)

MariaDB [akademik]> describe mahasiswa;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| nik         | char(10)    | NO   | PRI | NULL    |       |
| nama        | varchar(50) | NO   |     | NULL    |       |
| jk          | char(1)     | YES  |     | NULL    |       |
| tanggal_lhr | date        | YES  |     | NULL    |       |
| jurusan     | varchar(30) | YES  |     | NULL    |       |
| umur        | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
6 rows in set (0.03 sec)
```

Jadi pada dasarnya kedua perintah di atas menghasilkan output yang sama

F. Menambahkan data ke dalam tabel di MySQL

Selanjutnya adalah insert data kedalam tabel mahasiswa :

```
insert into mahasiswa
values ('135410154','Setiawan Dimas A','1','1995-05-24','Teknik Informatika',21),
('12548595','Safitri A','2','1996-05-04','Sistem Informasi',20);
```

```
MariaDB [akademik]> insert into mahasiswa
    -> values ('135410154','Setiawan Dimas A','1','1995-05-24','Teknik Informatika',21),
    -> ('12548595','Safitri A','2','1996-05-04','Sistem Informasi',20);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

Pada perintah diatas tidak menuliskan perintah nama field atau kolom tabel mahasiswa tersebut, sekarang kita lihat pada perintah di bawah ini yang menggunakan nama field atau kolom :

```
insert into mahasiswa (nik,nama,jk,tanggal_lhr,jurusan,umur)
values ('135410154','Setiawan Dimas A','1','1995-05-24','Teknik Informatika',21),
('12548595','Safitri A','2','1996-05-04','Sistem Informasi',20);
```

```
MariaDB [akademik]> insert into mahasiswa (nik,nama,jk,tanggal_lhr,jurusan,umur)
    -> values ('135410154','Setiawan Dimas A','1','1995-05-24','Teknik Informatika',21),
    -> ('12548595','Safitri A','2','1996-05-04','Sistem Informasi',20);
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

G. Melihat dat dalam tabel

Untuk menampilkan data yang telah kita tambahkan bisa kita menggunakan perintah sebagai berikut :

```
select * from mahasiswa;
```

```
MariaDB [akademik]> select * from mahasiswa;
+-----------+------------------+------+-------------+--------------------+-------+
| nik       | nama             | jk   | tanggal_lhr | jurusan            | umur  |
+-----------+------------------+------+-------------+--------------------+-------+
| 12548595  | Safitri A        | 2    | 1996-05-04  | Sistem Informasi   |    20 |
| 135410154 | Setiawan Dimas A | 1    | 1995-05-24  | Teknik Informatika |    21 |
+-----------+------------------+------+-------------+--------------------+-------+
2 rows in set (0.00 sec)
```

H. Menghapus Database

Dalam administrasi database, penghapusan record/baris yang dianggap sudah tidak diperlukan lagi merupakan hal yang sering dilakukan. MySQL menyediakan beberapa cara untuk menghapus record dari tabel.

TABEL DAFTAR DOSEN			
NIP	**NAMA DOSEN**	**NO HP**	**ALAMAT**
160436012	Sabrina Sari	0812349900	Pekanbaru
260432002	Maya Ari Putri	0812345234	Palembang
275430005	Susi Indriani	0812656532	Bogor
480432066	Tia Santrini	0812451177	Padang
576431001	Edi Siswanto	0812979005	Jakarta

Untuk penghapusan baris (atau disebut juga dengan *record*) dalam MySQL, kita memerlukan beberapa syarat, yaitu nama tabel dimana baris atau record tersebut berada, dan kondisi untuk penghapusan tersebut.

Format dasar penulisan query DELETE adalah:

DELETE FROM nama_tabel WHERE kondisi

- *nama_tabel* adalah nama dari tabel yang record/barisnya akan dihapus
- *kondisi* adalah kodisi baris yang akan dihapus, misalnya hapus kolom dimana nama dosen = *Sabrina Sari*
- Langsung saja dengan contoh, misalkan kita ingin menghapus record/baris untuk nama dosen *Sabrina Sari* maka querynya:

```
mysql> DELETE from daftar_dosen WHERE nama_dosen = 'Sabrina Sari';
Query OK, 1 row affected (1.73 sec)
```

Kunci dari query DELETE adalah pernyataan kondisi setelah WHERE, yaitu dimana kita memberitahukan kepada MySQL syarat/kondisi yang harus dipenuhi untuk menghapus suatu record. Kondisi ini juga mendukung logika OR atau AND untuk penghapusan yang lebih kompleks, seperti contoh berikut:

```
mysql> DELETE from daftar_dosen WHERE nama_dosen = 'Maya Ari Putri'
OR nama_dosen='Tia Santrini';
Query OK, 2 rows affected (0.15 sec)

mysql> SELECT * FROM daftar_dosen;
+------------+---------------+------------+---------+
| NIP_dosen  | nama_dosen    | no_hp      | alamat  |
+------------+---------------+------------+---------+
| 0275430005 | Susi Indriani | 0812656532 | Bogor   |
| 0576431001 | M. Siddiq     | 0812979005 | Jakarta |
| 0770435006 | Rubin Hadi    | 0812567678 | Papua   |
| 0869437003 | Mustalifah    | 0812338877 | Aceh    |
| 1080432007 | Arif Budiman  | 0812456345 | Makasar |
+------------+---------------+------------+---------+
5 rows in set (0.06 sec)
```

Cara Menghapus Record dengan Query DELETE..ORDER BY..LIMIT

MySQL menyediakan kondisi tambahan dengan perintah opsional ORDER BY dan LIMIT. Perintah ORDER BY dan LIMIT ini sama dengan pembahasan kita pada Tutorial MySQL: Cara Menampilkan data dari tabel dengan query SELECT. ORDER BY digunakan untuk mengurutkan data, sedangkan LIMIT digunakan untuk membatasi hasil record.

Perintah opsional ORDER BY dan LIMIT disediakan untuk perintah penghapusan yang lebih spesifik, misalkan: hapus 10 baris terakhir dari tabel dimulai dari tanggal transaksi paling awal. Hal ini biasanya dilakukan jika kita tabel yang regular di hapus jika telah melebihi batas maksimum record.

Sebagai contoh untuk tabel daftar_dosen, misalkan kita ingin menghapus 3 baris terakhir dari tabel daftar_dosen yang dimulai dari abjad terakhir dari nama dosen. Untuk keperluan ini, pertama-tama kita harus mengurutkan tabel daftar_dosen dengan menggunakan kolom nama dosen secara terbalik, lalu menghapus 3 record paling awal.

Cara Menghapus Seluruh Isi dari Tabel

Untuk keperluan penghapusan seluruh isi dari tabel, cukup menggunakan query DELETE tanpa mencantumkan kodisi WHERE. Misalkan untuk menghapus keseluruhan tabel daftar_dosen, querynya adalah:

```
mysql> DELETE FROM daftar_dosen;
Query OK, 5 rows affected (0.04 sec)
```

Dari query diatas, MySQL juga menampilkan jumlah record/baris yang telah dihapus, hal ini cukup berguna sebagai informasi kepada pengguna.

Namun jika informasi tersebut tidak kita butuhkan, MySQL menyediakan

query TRUNCATE seperti contoh berikut:

```
mysql> TRUNCATE TABLE daftar_dosen;
Query OK, 0 rows affected (0.04 sec)

mysql> SELECT * FROM daftar_dosen;
Empty set (0.00 sec)
```

Secara internal, TRUNCATE dijalankan dengan cara menghapus tabel menggunakan DROP, lalu membuat ulang tabel itu kembali. Cara ini akan lebih cepat dari pada perintah DELETE yang menghapus baris satu persatu. Perbedaan kecepatan eksekusi ini baru terasa jika tabel tersebut memiliki jumlah record yang banyak.

I. Cara Mengubah data Tabel MySQL (query UPDATE)

Tentang query UPDATE yang digunakan untuk mengubah atau memperbarui data dalam sebuah tabel MySQL.

Mempersiapkan Tabel Sample: daftar_dosen

Untuk tabel contoh, saya masih menggunakan tabel daftar_dosen yang kita buat pada tutorial query SELECT MySQL. Tetapi jika anda mengikuti tutorial tentang query DELETE sebelum tutorial ini, maka tabel daftar_dosen telah kosong disebabkan query TRUNCATE sebelumnya.

TABEL DAFTAR DOSEN			
NIP	**NAMA DOSEN**	**NO HP**	**ALAMAT**
160436012	Sabrina Sari	0812349900	Pekanbaru
260432002	Maya Ari Putri	0812345234	Palembang
275430005	Susi Indriani	0812656532	Bogor
480432066	Tia Santrini	0812451177	Padang
576431001	Edi Siswanto	0812979005	Jakarta

Cara Mengubah Data Tabel Menggunakan Query UPDATE

Query UPDATE digunakan untuk melakukan perubahan data pada tabel MySQL, yakni update baris atau record. Format dasar query UPDATE adalah sebagai berikut:

```
UPDATE nama_tabel SET nama_kolom = data_baru WHERE kondisi
```

- *nama_tabel* adalah nama dari tabel yang record/barisnya akan diperbaharui (update)

- *nama_kolom* adalah nama kolom dari tabel yang akan diupdate.
- *data_baru* adalah nilai data yang akan diinput sebagai nilai baru dari kolom
- *kondisi* adalah kodisi atau syarat dari baris yang akan diubah, misalnya jika kolom nama dosen= *Sabrina Sari* maka lakukan update.

Sebagai contoh, jika saya ingin merubah no HP dari dosen *Sabrina Sari* menjadi *085298710065* dari tabel daftar_dosen, maka querynya adalah sebagai berikut:

```
mysql> SELECT * FROM daftar_dosen WHERE nama_dosen='Sabrina Sari';
+------------+--------------+------------+-----------+
| NIP_dosen  | nama_dosen   | no_hp      | alamat    |
+------------+--------------+------------+-----------+
| 0160436012 | Sabrina Sari | 0812349900 | Pekanbaru |
+------------+--------------+------------+-----------+
1 row in set (0.08 sec)

mysql> UPDATE daftar_dosen SET no_hp ='085298710065'
WHERE nama_dosen='Sabrina Sari';
Query OK, 1 row affected (0.07 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM daftar_dosen WHERE nama_dosen='Sabrina Sari';
+------------+--------------+--------------+-----------+
| NIP_dosen  | nama_dosen   | no_hp        | alamat    |
+------------+--------------+--------------+-----------+
| 0160436012 | Sabrina Sari | 085298710065 | Pekanbaru |
+------------+--------------+--------------+-----------+
1 row in set (0.00 sec)
```

Dari query diatas dapat dilihat bahwa kolom **no_hp** untuk nama dosen *Sabrina Sari* telah diupdate menjadi nomor baru.

J. Cara Mengupdate Lebih dari 1 Baris

Untuk query yang lebih rumit, kita bisa merubah beberapa kolom sekaligus. Syarat untuk ***kondisi*** juga dapat menggunakan operator logika seperti OR atau AND sekaligus.

```
mysql> SELECT * FROM daftar_dosen;
+------------+-----------------+------------+-----------+
| NIP_dosen  | nama_dosen      | no_hp      | alamat    |
+------------+-----------------+------------+-----------+
| 0160436012 | Sabrina Sari    | 0812349900 | Pekanbaru |
| 0260432002 | Maya Ari Putri  | 0812345234 | Palembang |
| 0275430005 | Susi Indriani   | 0812656532 | Bogor     |
| 0480432066 | Tia Santrini    | 0812451177 | Padang    |
| 0576431001 | M. Siddiq       | 0812979005 | Jakarta   |
| 0770435006 | Rubin Hadi      | 0812567678 | Papua     |
| 0869437003 | Mustalifah      | 0812338877 | Aceh      |
| 1080432007 | Arif Budiman    | 0812456345 | Makasar   |
+------------+-----------------+------------+-----------+
8 rows in set (0.00 sec)
```

```
mysql> UPDATE daftar_dosen SET alamat ='Surabaya' WHERE
nama_dosen='Rubin Hadi' OR NIP_dosen='1080432007'
OR no_hp='0812345234';
Query OK, 3 rows affected (0.16 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> SELECT * FROM daftar_dosen;
+------------+----------------+------------+-----------+
| NIP_dosen  | nama_dosen     | no_hp      | alamat    |
+------------+----------------+------------+-----------+
| 0160436012 | Sabrina Sari   | 0812349900 | Pekanbaru |
| 0260432002 | Maya Ari Putri | 0812345234 | Surabaya  |
| 0275430005 | Susi Indriani  | 0812656532 | Bogor     |
| 0480432066 | Tia Santrini   | 0812451177 | Padang    |
| 0576431001 | M. Siddiq      | 0812979005 | Jakarta   |
| 0770435006 | Rubin Hadi     | 0812567678 | Surabaya  |
| 0869437003 | Mustalifah     | 0812338877 | Aceh      |
| 1080432007 | Arif Budiman   | 0812456345 | Surabaya  |
+------------+----------------+------------+-----------+
8 rows in set (0.00 sec)
```

Dari query diatas terlihat bahwa 3 baris/record telah berubah alamatnya menjadi *Surabaya*. Dalam satu statement UPDATE, kita membuat 3 buah logika OR.

K. Cara Mengupdate Record dengan Query UPDATE..ORDER BY..LIMIT

Sama seperti query DELETE, query UPDATE juga memiliki perintah opsional ORDER BY..LIMIT untuk pemrosesan tingkat lanjut. Perintah ORDER BY..LIMIT digunakan untuk membatasi perintah UPDATE dengan batas tertentu.

Misalkan kita ingin mengubah alamat dari 5 baris awal tabel daftar_dosen yang diurutkan berdasarkan nama, maka querynya:

```
mysql> SELECT * FROM daftar_dosen ORDER BY nama_dosen;
+------------+----------------+------------+-----------+
| NIP_dosen  | nama_dosen     | no_hp      | alamat    |
+------------+----------------+------------+-----------+
| 1080432007 | Arif Budiman   | 0812456345 | Surabaya  |
| 0576431001 | M. Siddiq      | 0812979005 | Jakarta   |
| 0260432002 | Maya Ari Putri | 0812345234 | Surabaya  |
| 0869437003 | Mustalifah     | 0812338877 | Aceh      |
| 0770435006 | Rubin Hadi     | 0812567678 | Surabaya  |
| 0160436012 | Sabrina Sari   | 0812349900 | Pekanbaru |
| 0275430005 | Susi Indriani  | 0812656532 | Bogor     |
| 0480432066 | Tia Santrini   | 0812451177 | Padang    |
+------------+----------------+------------+-----------+
8 rows in set (0.00 sec)
```

```
mysql> UPDATE daftar_dosen SET alamat ='Bali' ORDER BY nama_dosen LIMIT 5;
Query OK, 5 rows affected (0.09 sec)
Rows matched: 5  Changed: 5  Warnings: 0

mysql> SELECT * FROM daftar_dosen ORDER BY nama_dosen;
+------------+----------------+------------+-----------+
| NIP_dosen  | nama_dosen     | no_hp      | alamat    |
+------------+----------------+------------+-----------+
| 1080432007 | Arif Budiman   | 0812456345 | Bali      |
| 0576431001 | M. Siddiq      | 0812979005 | Bali      |
| 0260432002 | Maya Ari Putri | 0812345234 | Bali      |
| 0869437003 | Mustalifah     | 0812338877 | Bali      |
| 0770435006 | Rubin Hadi     | 0812567678 | Bali      |
| 0160436012 | Sabrina Sari   | 0812349900 | Pekanbaru |
| 0275430005 | Susi Indriani  | 0812656532 | Bogor     |
| 0480432066 | Tia Santrini   | 0812451177 | Padang    |
+------------+----------------+------------+-----------+
8 rows in set (0.06 sec)
```

Dari hasil query tersebut kita dapat melihat 5 baris awal dari tabel daftar_dosen, alamatnya telah berubah menjadi Bali.

L. Cara Mengupdate Seluruh Kolom dari Tabel MySQL

Jika kita tidak hati-hati dan lupa memberikan kodisi pada perintah WHERE, maka query UPDATE kita akan merubah seluruh kolom dari tabel tersebut.

```
mysql> SELECT * FROM daftar_dosen;
+------------+----------------+------------+-----------+
| NIP_dosen  | nama_dosen     | no_hp      | alamat    |
+------------+----------------+------------+-----------+
| 0160436012 | Sabrina Sari   | 0812349900 | Pekanbaru |
| 0260432002 | Maya Ari Putri | 0812345234 | Bali      |
| 0275430005 | Susi Indriani  | 0812656532 | Bogor     |
| 0480432066 | Tia Santrini   | 0812451177 | Padang    |
| 0576431001 | M. Siddiq      | 0812979005 | Bali      |
| 0770435006 | Rubin Hadi     | 0812567678 | Bali      |
| 0869437003 | Mustalifah     | 0812338877 | Bali      |
| 1080432007 | Arif Budiman   | 0812456345 | Bali      |
+------------+----------------+------------+-----------+
8 rows in set (0.00 sec)

mysql> UPDATE daftar_dosen SET no_hp='085278790005';
Query OK, 8 rows affected (0.06 sec)
Rows matched: 8  Changed: 8  Warnings: 0

mysql> SELECT * FROM daftar_dosen;
+------------+----------------+--------------+-----------+
| NIP_dosen  | nama_dosen     | no_hp        | alamat    |
+------------+----------------+--------------+-----------+
| 0160436012 | Sabrina Sari   | 085278790005 | Pekanbaru |
| 0260432002 | Maya Ari Putri | 085278790005 | Bali      |
| 0275430005 | Susi Indriani  | 085278790005 | Bogor     |
| 0480432066 | Tia Santrini   | 085278790005 | Padang    |
| 0576431001 | M. Siddiq      | 085278790005 | Bali      |
| 0770435006 | Rubin Hadi     | 085278790005 | Bali      |
| 0869437003 | Mustalifah     | 085278790005 | Bali      |
| 1080432007 | Arif Budiman   | 085278790005 | Bali      |
+------------+----------------+--------------+-----------+
8 rows in set (0.00 sec)
```

Dapat dilihat dengan mengeliminasi kondisi WHERE, mengakibatkan seluruh kolom tabel akan diupdate. Terkadang hasil seperti ini memang yang kita harapkan, namun seperti contoh diatas, kesalahan dalam membuat logika WHERE akan berdampak fatal terhadap keseluruhan tabel.

BAB 7
QUERY ANTAR TABEL

A. JOIN ANTAR TABEL

Menggabungkan tabel dengan beberapa metode yang disediakan ileh SQL yaitu **INNER, OUTER, LEFT, RIGHT dan FULL JOIN.** Sebelum kita kita akan melihat struktur data bata base yang saya gunakan terlebih dahulu dibawah ini.

Pertama kita akan membuat database dengan nama db_daerah kemudian kita buat 2 tabel :

Tb_kota

id_kota	nama_kota	id_provinsi
1	Jakarta	1
2	Semarang	2
3	Pati	2
4	Bandung	4
5	Surabaya	5
6	Medan	6

Tb_provinsi

id_provinsi	nama_provinsi
1	DKI Jakarta
2	Jawa Tengah
3	Papua Barat
4	Jawa Barat
5	Jawa Timur

Perlu di ketahui bahwa untuk field yang berelasi antara dua tabel diatas tidak harus sama. Kemudian masuk ke menu SQL di phpmyadmin untuk menulis perintah SQL yang akan menampilkan data-data dalam 2 tabel diatas :

1. Pertama yaitu kita akan menggunakan perintah INNER JOIN

```
SELECT *
FROM tb_kota
INNER JOIN tb_provinsi
ON
tb_kota.id_provinsi = tb_provinsi.id_provinsi;
```

Keterangan :

Pada perintah ON kota.id_propinsi = propinsi.id ini mempunyai arti untuk menampilkan data-data yang mempunyai nilai sama antra id_propinsi pada tabel tb_kota dan id_provinsi pada tabel tb_provinsi dimana itu mempunyai arti berelasi.

id_kota	nama_kota	id_provinsi	id_provinsi	nama_provinsi
1	Jakarta	1	1	DKI Jakarta
2	Semarang	2	2	Jawa Tengah
3	Pati	2	2	Jawa Tengah
4	Bandung	4	4	Jawa Barat
5	Surabaya	5	5	Jawa Timur

Dari sini juga bisa kita lihat hanya terdapat data-data yang mempunyai nilai sama antara id_provinsi pada tabel tb_kota dan id_provinsi pada tabel tb_provinsi yang akan tampil.

Jika kita custom field yang akan ditampilkan hanya nama_kota dan nama_provinsi saja, maka hasilnya seperti di bawah ini.

```
SELECT nama_kota, nama_provinsi
FROM tb_kota
INNER JOIN tb_provinsi
ON
tb_kota.id_provinsi = tb_provinsi.id_provinsi;
```

Hasil

nama_kota	nama_provinsi
Jakarta	DKI Jakarta
Semarang	Jawa Tengah
Pati	Jawa Tengah
Bandung	Jawa Barat
Surabaya	Jawa Timur

2. Kedua kita menggunakan perintah LEFT JOIN

```
SELECT *
FROM tb_kota
LEFT JOIN tb_provinsi
ON
tb_kota.id_provinsi = tb_provinsi.id_provinsi;
```

Berbeda dengan perintah yang pertama, pada LEFT JOINT akan menampilkan data yang tidak berelasi. Pada tabel tb_provinsi data yang tidak berelasi akan bernilai NULL

id_kota	nama_kota	id_provinsi	id_provinsi	nama_provinsi
1	Jakarta	1	1	DKI Jakarta
2	Semarang	2	2	Jawa Tengah
3	Pati	2	2	Jawa Tengah
4	Bandung	4	4	Jawa Barat
5	Surabaya	5	5	Jawa Timur
6	Medan	6	*NULL*	*NULL*

3. Ketiga kita menggunakan perintah RIGHT JOINT

```
SELECT *
FROM tb_kota
RIGHT JOIN tb_provinsi
ON
tb_kota.id_provinsi = tb_provinsi.id_provinsi;
```

Sama halnya dengan ketika kita menggunakan LEFT JOIN, RIGHT JOIN akan menampilkan data yang tidak berlasi, tetapi disini kebalikan dari LEFT JOIN pada tabel tb_kota data yang tidak berelasi akan bernilai NULL.

id_kota	nama_kota	id_provinsi	id_provinsi	nama_provinsi
1	Jakarta	1	1	DKI Jakarta
2	Semarang	2	2	Jawa Tengah
3	Pati	2	2	Jawa Tengah
4	Bandung	4	4	Jawa Barat
5	Surabaya	5	5	Jawa Timur
NULL	*NULL*	*NULL*	3	Papua Barat

Jadi pada dasarnya ketiga JOIN yang telah di bahas diatas di gunakan pada case yang berbeda beda, tergantung kebutuhan. Namun biasanya yang lebih sering di gunakan ialan INNER JOIN dalam artian bisa menampilkan data yang hanya berelasi.

.

DAFTAR PUSTAKA

Ariwibowo, Budi, 2021, "*Dasar – Dasar SQL MariaDB*", CV. Diandra Primamitra Media", Yogyakarta.

Dewi, Ketut dan Sumiari, Kadek, 2018, "*Teori Basis Data*", CV. Andi Offset, Yogyakarta.

Diat, Prasojo Lantip, 2014, "*Perancangan Database Sistem Informasi Manajemen Pendidikan Dengan DBMS Microsoft (Acces Dan Sql Server),* UNY Press, Yogyakarta.

Henderi, 2020, "*SISTEM BASIS DATA (Model Relational, SQL, dan Objek Oriented Database*", CV. Bintang Surya Madani, Yogyakarta.

Jatnika, Hendra, 2013, "*Pengantar Sistem Basis Data, Memahami Konsep Dasar dan Tuntunan Praktis Perancangan Database*", CV. ANDI OFFSET, Yogyakarta.

Modul Praktikum, 2018, "*Perancangan Basis Data*:, Universitas AMIKOM YOGYAKARTA.

Setiadi, Didik, 2010, "*Sistem Basis Data*", STIMIK Eresha, Jakarta.

Widodo, Wahyu, Agus dan Kurniaingtyas, Diva, 2017, "*Sistem Basis Data*", UB Press, Malang.

Yanto, Robbi, 2016, "*Managemen Basis Data Menggunakan MySQL*", CV. Budi Utama, Yogyakarta.