ISBN:

BUKU AJAR

MATA KULIAH

BASIS DATA

PENYUSUN:

Dewi Soyusiawaty, S.T., M.T.

Sri Winiarti, S.T., M.Cs.

Miftahurrahmah Rosyda, S.Kom., M.Eng.

Jefree Fahana, S.T., M.Kom.

Program Studi Teknik Informatika

HAK CIPTA

BUKU AJAR MATA KULIAH BASIS DATA

Diterbitkan oleh:
Program Studi Teknik Informatika
Fakultas Teknologi Industri
Universitas Ahmad Dahlan
Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166

Penulis	: Dewi Soyusiawaty, S.T., M.T. Sri Winiarti, S.T., M.Cs. Miftahurrahmah Rosyda, S.Kom., M.Eng. Jefree Fahana, S.T., M.Kom.
Editor	: Laboratorium Teknik Informatika, Universitas Ahmad Dahlan
Desain sampul	: Laboratorium Teknik Informatika, Universitas Ahmad Dahlan
Tata letak	: Laboratorium Teknik Informatika, Universitas Ahmad Dahlan
Ukuran/Halaman	: 21 x 29,7 cm / 161 halaman

Didistribusikan oleh:

Laboratorium Teknik Informatika
Universitas Ahmad Dahlan
Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166
Indonesia

KATA PENGANTAR

Puji dan syukur kita panjatkan kepada Allah SWT yang telah memberi rahmat dan hidayahNya sehingga revisi penyusunan diktat Basis Data ini akhirnya bisa diselesaikan. Diktat ini disusun untuk menunjang perkuliahan dalam mata kuliah Basis Data di lingkungan Program Studi Teknik Informatika Universitas Ahmad Dahlan.

Materi yang disajikan sudah diurutkan disesuaikan dengan perencanaan mata kuliah tersebut, sehingga insyaAllah mahasiswa dapat dengan mudah memahami. Pada beberapa bab diberikan dengan soal-soal sebagai latihan, agar mahasiswa dapat berlatih sendiri, baik sebagai tugas atau untuk belajar secara mandiri.

Penulis menyadari masih banyak ketidaksempurnaan pada penulisan diktat ini, baik isi maupun redaksinya, oleh karenanya kritik dan saran yang membangun diharapkan dapat memperbaiki diktat ini untuk tahun-tahun berikutnya.

Terima kasih kepada semua pihak yang telah membantu baik secara langsung ataupun tidak terhadap terselesaikannya diktat ini. Akhir kata, insyaAllah diktat ini dapat bermanfaat bagi siapa saja yang membutuhkannya.

Yogyakarta, September 2020

Tim Penyusun

DAFTAR ISI

BAB 1. PENDAHULUAN

1.1. Tujuan Instruksional

A. Tujuan Instruksional Umum

Mahasiswa memahami konsep dasar Basis Data

B. Tujuan Instruksional Khusus

Mahasiswa dapat mengidentifikasi data yang diperlukan dalam suatu kasus

1.2. Analogi Konsep

Suatu Basis data merupakan kumpulan data terpusat dan terstruktur yang disimpan dalam sistem komputer, dimana sistem tersebut menyediakan fasilitas untuk mengambil, menambahkan, memodifikasi dan menghapus data saat diperlukan. Sistem juga menyediakan fasilitas untuk dapat mengubah data yang diambil menjadi informasi yang berguna.

Konsep basis data dapat dianalogikan sebagai berikut:

Basis data = Lemari Arsip = Lemari Pakaian = Rak Buku = Rak Piring = dan lain sebagainya yang pasti dikelola untuk menyimpan barang-barang tertentu. Beberapa hal yang dilakukan untuk mengelola tempat penyimpanan tersebut, yaitu:

a. Memberi sampul/map pada kumpulan arsip yang disimpan
b. Menentukan kelompok/jenis arsip
c. Memberi penomoran dengan pola tertentu yang nilainya unik pada setiap sampul/map maupun penomoran kelompok atau jenis arsip
d. Menempatkan arsip tersebut dengan cara/urutan tertentu di lemari arsip

Tujuannya adalah jika pada saat data atau arsip tersebut dibutuhkan, maka dapat ditemukan dan diperoleh dengan cepat dan mudah.

1.3. Data dan Informasi

Dalam basis data, data dan informasi merupakan bagian yang penting, sehingga data dan informasi memiliki pengertian:

a. Data merupakan nilai (*value*) yang merepresentasikan deskripsi dari suatu objek atau kejadian (*event*)
b. Informasi merupakan hasil dari pengolahan data dalam suatu bentuk yang lebih berguna dan lebih berarti bagi penerimanya serta menggambarkan suatu kejadian-

kejadian yang nyata (*fact*) yang digunakan untuk pengambilan keputusan

c. Data lebih bersifat historis, sedangkan informasi mempunyai tingkatan yang lebih tinggi, lebih dinamis, serta mempunyai nilai sangat penting

1.4. Sistem Informasi/SI

a. Sistem Informasi / SI adalah suatu sistem dalam suatu organisasi yang merupakan kombinasi dari orang-orang, fasilitas, teknologi, media, prosedur dan pengendalian untuk mendapatkan jalur komunikasi penting, memproses tipe transaksi rutin tertentu, memberi sinyal kepada manajemen dan yang lainnya terhadap kejadian-kejadian internal dan eksternal yang penting dan menyediakan suatu dasar informasi untuk pengambilan keputusan

b. Sistem Informasi Manajemen adalah sekumpulan elemen yang saling berhubungan, saling berinteraksi dan bekerjasama antara berbagai bagian dengan cara-cara tertentu untuk melakukan fungsi pengolahan data, pemasukan data, dan menghasilkan keluaran berupa informasi yang berguna dan mempunyai nilai nyata, sebagai dasar pengambilan keputusan, mendukung kegiatan manajemen dan operasional dengan memanfaatkan berbagai sumber daya yang ada bagi proses tersebut guna mencapai tujuan organisasi

1.5. Komponen Sistem Informasi

Sistem Informasi terdiri dari beberapa komponen, antara lain :

a. Hardware : CPU, Disk, Terminal, Printer
b. Software : Sistem operasi, sistem basis data, program aplikasi
c. Personil : Operator sistem, Penyedia masukan, Pengguna keluaran
d. Data : data yang tersimpan dalam jangka waktu tertentu
e. Prosedur : instruksi dan kebijaksanaan untuk mengoperasikan sistem

1.6. Basis data

Basis Data adalah suatu kumpulan data terhubung yang disimpan secara bersama-sama pada suatu media, yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, dan dengan *software* untuk melakukan manipulasi untuk kegunaan tertentu. Operasi dasar basis data:

a. Create database
b. Drop database
c. Create table
d. Drop table

e. Insert
f. Retrieve / Search
g. Update
h. Delete

1.7. Objektif Basis Data

Pemanfaatan basis data dilakukan untuk memenuhi sejumlah tujuan (objektif), yaitu:

a. Speed (Manipulasi terhadap data dengan cepat dan mudah)
b. Efficiency (Efisiensi penggunaan ruang penyimpanan)
c. Accuracy (Menentukan kualitas informasi : akurat, tepat waktu dan relevan)
d. Availability (Ketersediaan)
e. Completeness (Kelengkapan)
f. Security (Keamanan)
g. Sharability (Kebersamaan pemakaian)

1.8. Kriteria Basis Data

Dari pengertian basis data yang telah dijelaskan, dapat disimpulkan bahwa basis data mempunyai beberaa kriteria penting yaitu:

a. Bersifat *data oriented* dan bukan *program oriented*
b. Dapat digunakan oleh beberapa program aplikasi tanpa mengubah basis datanya
c. Dapat berkembang dengan mudah, baik volume maupun strukturnya
d. Dapat digunakan dengan cara berbeda-beda
e. Kerangkapan data minimal

1.9. Sejarah Basis Data

Berikut sejarah perkembangan basis data yang dirangkum dalam bentuk tabel

Tabel 1.1 Evolusi Basis Data

Masa	**Perkembangan Basis Data**
1960-an	Sistem Pemrosesan Berkas
	DBMS
	Layanan informasi secara online berbasis manajemen teks
1970-an	Penerapan sistem pakar pada SPK
	Basis data berorientasi objek
1980-an	Sistem hypertext yang memungkinkan untuk melihat basis data secara acak menurut suatu kata kunci
1990-an	Sistem basis data cerdas

	Sistem basis data multimedia cerdas

1.10. Contoh Penerapan Basis Data

Bidang fungsional banyak memanfaatkan basis data demi ke-efisiensian, akurasi dan kecepatan operasi dalam menjalankan prosedur, contoh pengelolaan basi data dalam bidang fungsional yaitu:

a. Kepegawaian (dengan banyak pegawai)
b. Pergudangan/inventory (pabrik, grosir, apotik dan lain-lain)
c. Akuntasi (perusahaan)
d. Reservasi (hotel, pesawat, kereta api dan lain-lain)
e. Layanan pelanggan (perusahaan dengan banyak pelanggan)

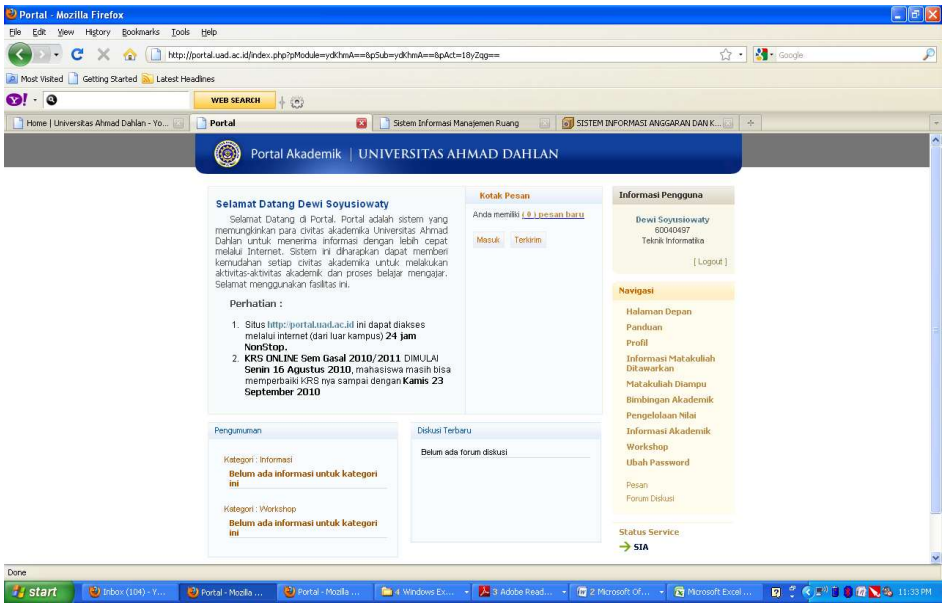
Bentuk organisasi/perusahaan yang memanfaatkan basis data (sebagai komponen sistem informasi organisasi/perusahaan) dapat berupa:

a. Perbankan (Pengelolaan data nasabah, data tabungan, data pinjaman, pembuatan laporan akuntansi, dan lain-lain)
b. Asuransi (pengelolaan data nasabah, data pembayaran premi, pemrosesan pengajuan klaim dan lain-lain)
c. Rumah Sakit (pengelolaan data pasien, histori penyakit, data dokter, data jadwal praktek dokter, data pembayaran perawatan dan lain-lain)
d. Produsen Barang (pengelolaan data keluar masuk barang, stok barang, dan lain-lain)
e. Industri Manufaktur (pengelolaan pesanan barang, data karyawan, dan lain-lain)
f. Pendidikan/Sekolah (data siswa, jadwal kuliah, nilai, dan lain-lain)
g. Telekomunikasi (data administrasi, data pelanggan, dan lain-lain)

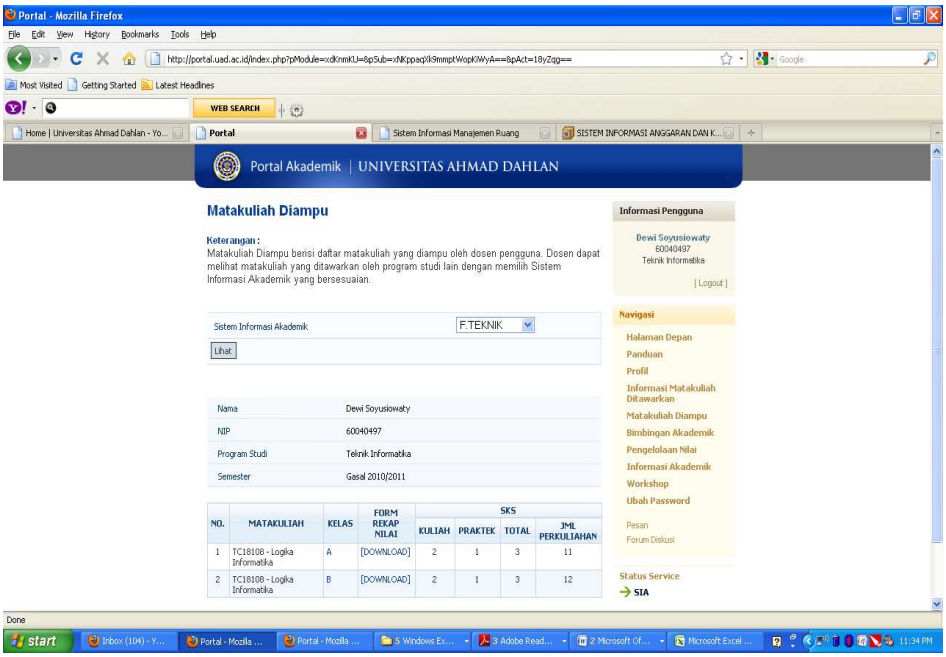
Salah satu vontoh nyata penerapan pengolahan basis data yang bisa dialami secara langsung adalah pada poratl akademik.

Portal akademik merupakan salah satu contoh sistem informasi yang memanfaatkan basis data dalam penyampaian informasinya. Banyak user yang dapat memanfaatkan informasi yang ada di portal, mahasiswa, dosen, kaprodi, kepala tata usaha dan lain-lain. Gambar 1.1. menunjukkan halaman portal dengan login sebagai dosen. Beberapa informasi yang diberikan untuk dosen meliputi info mata kuliah yang diampu, daftar mahasiswa bimbingan akademik, pengelolaan nilai dsb. Informasi ini tentu didapat dari sekumpulan

data yang disimpan. Sedangkan gambar 1.2 menunjukkan halaman mata kuliah yang diampu.

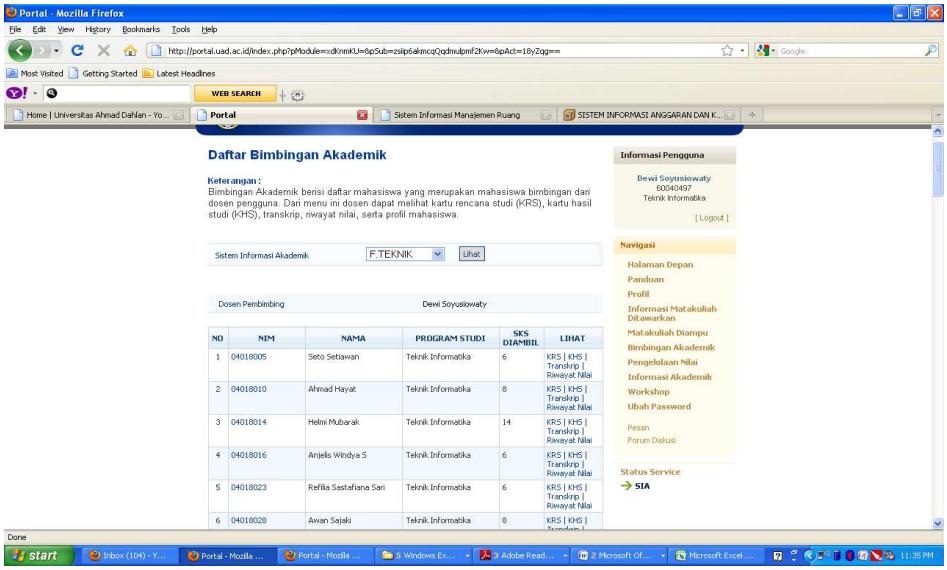


Gambar 1.1. Portal Akademik UAD



Gambar 1.2. Halaman Mata Kuliah Diampu – Portal Akademik

Gambar 1.3 menunjukkan daftar mahasiswa bimbingan akademik, berupa NIM, nama mahasiswa, program studi dan juga termasuk dosen bimbingan akademik dapat melihat transkrip nilai dari tiap mahasiswa bimbingan akademiknya.



Gambar 1.3. Halaman Daftar Bimbingan Akademik – Portal Akademik

Tabel 1.2 berikut adalah contoh-contoh penggunaan basis data dalam berbagai sistem informasi dan aplikasi:

Tabel 1.2. Contoh Penggunaan Basis Data

Penggunaan basis data dalam sistem informasi dan aplikasi	**Interaksi basis data dengan sistem informasi dan aplikasi**	**Data yang disimpan dalam basis data**
Sistem Informasi Perpustakaan	Jika kita akan meminjam buku di perustakaan, kita melakukan registrasi terlebih dahulu. Kemudian, data peminjam akan disimpan oleh sistem dalam basis data. Setelah itu baru kita dapat melakukan berbagai transaksi yaitu peminjaman buku, perpanjangan peminjaman buku, perpanjangan peminjaman buku, pengembalian buku, dan pembayaran denda. Seluruh transaksi yang dilakukan disimpan dalam basis data dan	Data buku, misalnya kode buku, judul buku, pengarang, penerbit, dan tahun terbit. Data peminjam, misalnya nomor peminjam, nama peminjam, alamat, nomor telepon, dan email Data petugas perpustakaan, misalnya nomor induk karyawan, jabatan dan level. Data transaksi peminjaman, misalnya tanggal pinjam, kode buku, jumlah buku, nomor peminjam, dan lama pinjaman.

	dapat diakses kembali oleh sistem sesaui dengan kebutuhan.	
	Peminjaman dapat mencari buku yang akan dipinjam dengan menekan tombol find atau search. Setelah itu sistem informasi akan berinteraksi dengan basis data perpustakaan untuk mencari buku tersebut. Lalu hasil pencarian akan ditampilkan pada layar. Melalui interaksi yang terjadi antara sistem informasi dengan basis data, petugas perpustakaan dapat mengetahui secara otomatis jika peminjaman buku lebih dari ketentuan yang berlaku, peminjam belum mengembalikan buku tetapi akan meminjam lagi buku yang lain, dan pemberitahuan ada yang kena denda pada hari H.	Data transaksi perpanjangan peminjaman, misal tgl perpanjangan, kode buku, jml buku diperpanjang, no peminjam, perpanjangan ke Data transaksi pengembalikan buku, misal tgl pengembalian, kode buku, jumlah buku yang dikembalikan, nomor peminjam. Data transaksi pemb denda, misal tgl denda, kode buku, jml buku yang kena denda, nomor peminjam, status lunas.
Sistem Informasi Akademis	Dalam sistem informasi akademis ini terjadi proses antara lain penjadwalan kuliah, penjadwalan dosen, pencatatan absensi mahasiswa, pencatatan absensi dosen, penjadwalan ujian, dan pendataan nilai mahasiswa. Seluruh proses tersebut akan menyimpan datanya dalam basis data. Petugas akademis dapat mengetahui jadwal kuliah kelas tertentu dengan mengakses basis data melalui menu inquiry jadwal kuliah yang terdapat pada sistem informasi. Selain itu juga jika dosen menanyakan jadwal mengajar pada petugas akademis, maka petugas akademis tersebut akan mengakses	Data mahasiswa, misalnya nomor induk mahasiswa, nama mahasiswa, alamat, telepon rumah, nomor HP, dan email. Data dosen, misalnya kode dosen, nama dosen, alamat, dan nomor HP. Data mata kuliah, misalnya kode mata kuliah, nama mata kuliah, dan sks. Data kelas, misalnya kode kelas dan penanggung jawab kelas. Data ruangan, misalnya kode ruang, nama ruang, dan kapasitas ruang. Data transaksi absensi, misalnya nomor transaksi absen, nomor induk mahasiswa, kode dosen, kode mata kuliah, dan kehadiran. Data transaksi jadwal kuliah, misalnya nomor induk mahasiswa, kode mata kuliah, dan hari kuliah

	basis data melalui menu transaksi jadwal dosen pada sistem informasi akademis itu.	
		Data transaksi jadwal ujian, misalnya nomor induk mahasiswa, kode mata kuliah, hari ujian, dan kehadiran. Data transaksi nilai, misalnya nomor induk mahasiswa, kode mata kuliah, nilai ujian tengah semester, nilai tugas mandiri, dan nilai ujian akhir semester.
Sistem Informasi Hotel	Pada sistem ini terdapat proses antara lain reservasi kamar, reservasi ruangan, jasa *laundry*, *restaurant,* dan jasa *travel agent.* Jika kita akan melakukan reservasi pada suatu hotel maka petugas reservasi akan memasukkan data kita pada sistem informasi hotel tersebut, kemudian memasukkan juga data reservasi yang kita inginkan. Kemudian petugas hotel tersebut akan mencari ketersediaan kamar atau ruangan melalui sistem informasi hotel. Jika tersedia maka reserasi diterima dan datanya akan disimpan dalam basis data. Saat anda datang ke hotel tersebut (check in), maka anda juga dapat menggunakan jasa *laundry*, *restaurant,* dan jasa *travel agent.* Petugas hotel akan memasukkan seluruh data jasa yang digunakan selama menginap di hotel tersebut dan tentunya disimpan dalam basis data. Yang nantinya akan digunakan kembali dalam pencetakan tagihan.	Data kamar Data ruangan Data jasa *laundry* Data jasa *restaurant* hotel Data jasa *travel agent* Data pengunjung Data karyawan Data kartu kredit Data transaksi reservasi kamar Data transaksi reservasi ruangan Data transaksi jasa hotel Data transaksi *check in* dan *checkout* Data transaksi pembayaran

	Demikian pula saat anda *check out,* maka kasir akan mencetak tagihan untuk anda. Sistem informasi akan mengakses basis data dan mengambil data-data yang diperlukan untuk pencetakan tagihan.	
Sistem Informasi Rumah Makan	Proses yang terdapat pada sistem informasi rumah makan antara lain adalah pemesanan makanan, pembatalan pesanan makanan, dan pembayaran atas pesanan makanan. Saat pelanggan memesan makanan, maka pelayan rumah makan akan memasukkan pesanan melalui sistem informasi rumah makan. Kemudian data tersebut disimpan dalam basis data. Demikian pula saat kasir mencetak tagihan, maka melalui sistem informasi tersebut, kasir dapat mengakses basis data untuk menghasilkan tagihan. Kemudian pelayan rumah makan memberikan tagihan tersebut kepada pelanggan, setelah pelanggan membayar tagihan, maka data pembayaran akan disimpan kembali dalam basis data.	Data makanan Data minuman Data pelayan rumah makan Data transaksi pemesanan makanan Data transaksi pembayaran Data transaksi pembatalan pesanan makanan

1.11. Latihan dan Evaluasi

Tugas Pribadi

Menuliskan refleksi tentang materi pertemuan hari ini yaitu :

1. Seperti apa kegiatan perkuliahan hari ini.
2. Materi apa yang saya dapat hari ini.
3. Materi apa yang belum saya pahami
4. Apa yang akan saya lakukan setelah ini.

Tugas Kelompok

Menentukan topik penerapan basis data yang akan dikembangkan untuk tiap kelompok yang terdiri atas :

1. Topik penerapan basis data
2. Uraian proses yang terjadi dari unit atau organisasi yang dipilih
3. Data dan nilai yang mewakili

BAB 2. SISTEM BASIS DATA

2.1. Tujuan Instruksional

A. Tujuan Instruksional Umum

Mahasiswa mampu menjelaskan konsep dasar Sistem Basis Data

B. Tujuan Instruksional Khusus

Mahasiswa mampu mengidentifikasi, mengenai integrasi data, relasi data, dan komponen system basisdata

2.2. Sistem

Pengertian dari sistem adalah sebuah tatanan/keterpaduan yang terdiri atas sejumlah komponen fungsional (dengan satuan fungsi/tugas khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses/pekerjaan tertentu.

2.3. Sistem Basis Data/SBD

Sistem basis data memiliki beberapa definisa antara lain:

a. Sistem Basisi Data merupakan sekumpulan basis data dengan para pemakai yang menggunakan basis data secara bersama-sama, personil yang merancang dan mengelola basis data, teknik-teknik untuk merancang dan mengelola basis data, serta sistem komputer yang mendukungnya.
b. Sistem yang terdiri atas kumpulan file/tabel yang saling berhubungan (dalam sebuah basis data di sebuah sistem komputer) dan sekumpulan program (DBMS) yang memungkinkan beberapa pemakai dan/atau program lain untuk mengakses dan memanipulasi file-file (tabel-tabel) tersebut.

2.4. Komponen sistem basis data

Sistem basis data memiliki komponen-komponen penting yaitu:

a. Perangkat keras
b. Sistem operasi
c. Basis data
d. Sistem pengelola basis data (DBMS)
e. Pemakai (Programmer, User mahir, user umum, user khusus)

2.5. DBMS (Data Base Management System)

Program komputer yang digunakan untuk memasukkan, mengubah, menghapus, memanipulasi, dan memperoleh data informasi dengan praktis dan efisien termasuk juga

mengatur mekanisme pengamanan data, pemakaian data bersama, pemaksaan keakuratan/konsistensi data dan sebagainya. Tabel 2.1 berisi kelemahan dan kelebihan DBMS.

Tabel 2.1. Kelemahan dan Kelebihan DMBS

Kelemahan DBMS	Kelebihan DBMS
Harga DBMS mahal. Ada pendapat, ada uang ada barang. Teknologi baru tentunya lebih mahal daripada teknlogi yang terdahulu.	**Mengontrol redundansi data.** Dengan adanya integrasi *file* ini maka berbagai duplikasi data yang terjadi dihilangkan.
Ukuran. Kerumitan dan banyaknya fungsi yang ada pada DBMS menyebabkan DBMS memerlukan banyak *software* pendukung yang mengakibatkan penambahan tempat penyimpanan dan memori.	**Konsistensi Data.** Jika ada perubahan yang terjadi dalam DBMS karena proses tambah, ubah, atau hapus data, maka pengguna-pengguna DBMS akan dapat mengakses nilai terbaru dalam DBMS secara cepat.
Kompleksitas. Pada DBMS terdapat pengaturan fungsi-fungsi sehingga DBMS menjadi *software* yang cukup rumit dan kompleks. Aturan fungsi-fungsi tersebut harus diketahui oleh pengguna DBMS dengan baik. Jika tidak maka pengguna DBMS tidak akan mendapat manfaat dari implemenasi DBMS.	**Informasi yang lebih dari sejumlah data yang sama.** Contoh kasus adalah sebagai berikut: Pada FBS adalah hal yang cukup sulit untuk mendapatkan informasi pemasok yang barangnya terlaris dijual, sedangkan pada DBMS, mendapatkan informasi tersebut sangatlah mudah, mengingat seluruh data dalam DBMS telah terintegrasi.
Penambahan biaya perangkat keras.	**Pemakaian data bersama.** Dalam FBS, *file* dimiliki oleh bagian tertentu. Namun dalam DBMS, konsep demikian tidak pernah. Karena *database* dimiliki oleh perusahaan atau organisasi, bukan oleh bagian tertentu.

Adanya biaya konversi. Biaya konversi ini akan digunakan untuk proses konversi FBS ke DBMS.	**Meningkatnya Integritas Data.** Dalam DBMS, terdapat fasilitas yang mengatur akses misalnya otorisasi untuk mengakses, menambah, mengubah, dan menghapus. Dengan demikian setiap pengguna DBMS tidak dapat melakukan sesuaut yang bukan menjadi haknya. Contohnya jika bagian pembelian hanya diberi akses untuk mengambil data barang, maka dia tidak dapat menambah, mengubah, dan menghapus data barang.
Dampak yang lebih tinggi pada suatu kegagalan. Jika terjadi kerusakan pada DBMS, maka akan berdampak pada seluruh pengguna dan sistem informasi yang mengakses DBMS.	**Meningkatnya standarisasi.** Dengan adanya pemakaian data besama-sama, maka penamaan table, *field,* tipe data, hak akses, dan sebagainya harus dibuat standard an dokumentasinya. Hal ini bertujuan untuk memudahkan pengguna DBMS.
	Meningkatkan skala ekonomi. Adanay integrasi data seluruh perusahaan atau organisasi ini menjadikan pengurangan biaya, yang akhirnya dapat meningkatkan skala ekonomi.
	Keseimbangan konflik kebutuhan. Pengguna atau suatu bagian dalam perusahaan mungkin memiliki kebutuhan yang tidak sama dengan kebutuhan pengguna lainnya. Dengan DBMS, kita dapat membuat keputusan tentang desain dan penggunaan operasional basis data secara keseluruhan.
	Meningkatnya akses data dan tanggapan. Integrasi menghilangkan batasan-batasan dasar dari seluruh bagian-bagian atau departemen-

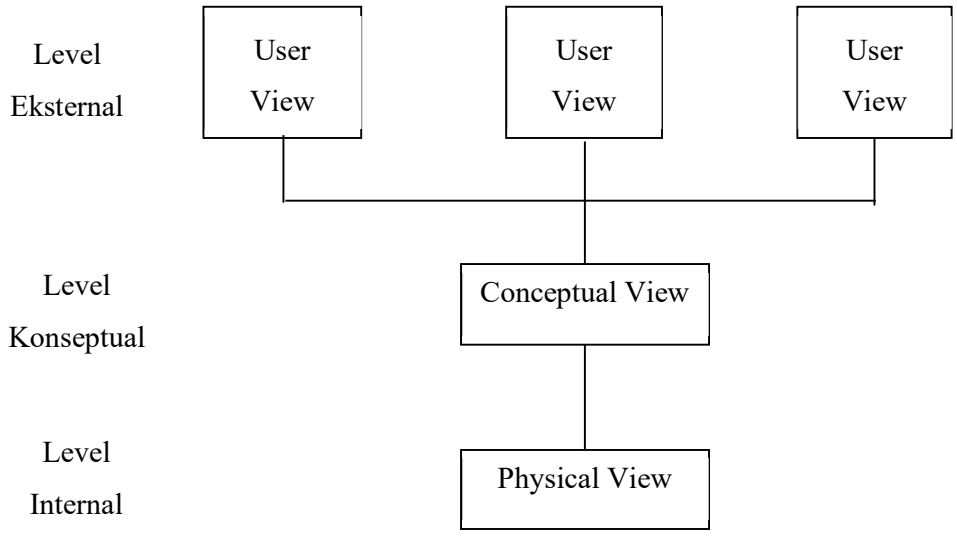
	departemen dalam perusahaan sehingga dapat diakses secara langsung oleh seluruh pengguna DBMS. DBMS juga menyediakan bahasa *query* pembuatan laporan yang mengijinkan pengguna DBMS untuk meminta pertanyaan khusus dan untuk memperoleh informasi dengan segera.
	Meningkatnya produktivitas. DBMS menyediakan banyak fungsi baku di mana *programmer* dapat menuliskan fungsi-fungsi baku tersebut dalam suatu instruksi pada program aplikasi. Ditingkat paling dasar, DBMS menyediakan seluruh rutin *low-level file-handling program.* Fungsi ini menjadikan *programmer* lebih berkonsentrasi pada kemampuan fungsi spesifik yang diinginkan oleh pengguna tanpa takut untuk melakukan implementasi pada tingkat rendah secara detail.
	Meningkatnya pemeliharaan karena independensi data Pada sistem FBS, rincian data dan logika untuk mengakses data dibuat di dalam program aplikasi masing-masing, sehingga terjadi ketergantungan data terhadap program. Suatu perubahan terhadap cara data disimpan dalam *disk,* akan memerlukan perubahan dalam program yang mendefinisikan data tersebut. Sedangkan pada DBMS, terjadi pemisahan data dengan aplikasi program dan akan kebal terhadap perubahan data. Hal ini dikenal dengan istilah independensi data.

	Meningkatnya konkurensi Dalam FBS, jika dua atau lebih pengguna diijinkan utnuk mengakses *file* yang sama secara bersamaan, memungkinkan terjadi pertentangan satu sama lain, kehilangan informasi, dan hilangnya integritas. DBMS dapat mengatur akses data yang dilakukan secara bersamaan.
	Meningkatnya *service backup* dan *recovery* DBMS menyediakan fasilitas untuk mengurangi kegagalan sistem atau aplikasi program yaitu fasilitas *backup* dan *restore.*

2.6. Abstraksi data

a. Sistem basis data biasanya menyembunyikan detil tentang bagaimana data disimpan dan diperlihara. Oleh karena itu, seringkali data yang terlihat oleh pemakai sebenarnya berbeda dengan yang tersimpan secara fisik

b. Abstraksi data merupakan level dalam bagaimana melihat data dalam sebuah sistem basis data



Gambar 2.1. Abstraksi Data

a. **Conceptual view** merupakan pandangan yang berkaitan dengan permasalahan data apa saja yang diperlukan untuk disimpan dalam basis data dan penjelasan mengenai hubungan antar data yang satu dengan lainnya. Data pegawai disimpan dalam beberapa file/tabel, seperti file pribadi, file pendidikan, file pekerjaan, file keluarga, dan sebagainya.
b. **Physical view** merupakan level terendah dalam abstraksi data, yaitu pandangan tentang bagaimana data disimpan dalam media penyimpan data. Pemakai melihat data sebagai gabungan dari struktur dan datanya sendiri. Pemakai juga berkompeten mengetahui bagaimana representasi fisik dari penyimpanan/pengorganisasian data.
c. **User view** merupakan level tertinggi yang hanya menunjukkan sebagian dari basis data. Aplikasi ini mengkonversi data asli/fisik menjadi data bermakna/lojik ke pemakai

2.7. Bahasa basis data

Bahasa basis data merupakan perantara bagi pemakai dengan basis data dalam berinteraksi, yang telah ditetapkan oleh pembuat DBMS

a. Data Definition Language (DDL)
   1) Dengan bahasa ini kita dapat membuat tabel baru, membuat indeks, mengubah tabel, menentukan struktur tabel, dll.
   2) Hasil dari kompilasi perintah DDL menjadi Kamus Data, yaitu data yang menjelaskan data sesungguhnya
   3) Contoh: Create
b. Data Manipulation Language (DML)
   1) Berguna untuk melakukan manipulasi dan pengambilan data pada suatu basis data, yang berupa insert, update, delete, dll.
   2) Ada 2 jenis yaitu prosedural (ditentukan data yang diinginkan dan cara mendapatkannya) dan non-prosedural (tanpa menyebutkan cara mendapatkannya)
   3) Contoh : dbase 3+, foxbase, SQL, QBE

2.8. Pengguna basis data

1) Database Administrator

a. Orang yang memiliki kewenangan untuk melakukan pengawasan baik data maupun program
b. Fungsi DBA adalah :

a) Mendefinisikan pola struktur basis data
b) Mendefinisikan struktur penyimpanan dan metode akses
c) Memodifikasi pola dan organisasi fisik
d) Memberikan kewenangan pada user untuk mengakses data
e) Menspesifikasikan keharusan integritas data

2) Database User

Ada 4 pemakai basis data, yaitu :

a) Programmer aplikasi, Merupakan pembuat program aplikasi
b) Casual user / Naïve User, Pemakai yang sudah mahir, berinteraksi dengan sistem tanpa menulis program, tapi menggunakan query
c) End user, Pemakai yang belum mahir tinggal menjalankan aplikasi yang sudah dibuat oleh programmer aplikasi
d) Specialized user, Pemakai khusus yang menuliskan aplikasi database tidak dalam kerangka pemrosesan data, namun untuk keperluan khusus seperti CAD, AI, ES, dll

BAB 3. PEMODELAN DATA

3.1. Tujuan Instruksional

A. Tujuan Instruksional Umum

Mahasiswa memahami konsep pemodelan data

B. Tujuan Instruksional Khusus

Mahasiswa dapat memodelkan data.

3.2. Model Data

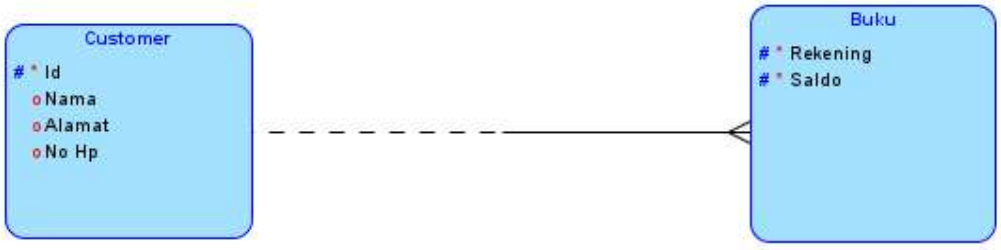
Model data merupakan suatu cara untuk menjelaskan bagaimana pemakai dapat melihat data secara logik.

3.3. Model data berbasis objek

Himpunan data dan relasi yang menjelaskan hubungan logik antar objek. Terdiri dari 2 jenis :

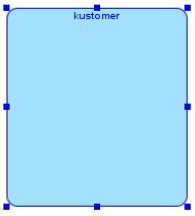
3.3.1. Entity Relationship model

Menjelaskan hubungan antar data dalam basis data berdasarkan persepsi bahwa *real world* (dunia nyata) terdiri dari objek-objek dasar yang mempunyai hubungan/ relasi antara objek tersebut.



Gambar 3.1. Model Data Objek - ERD

Arti simbol :



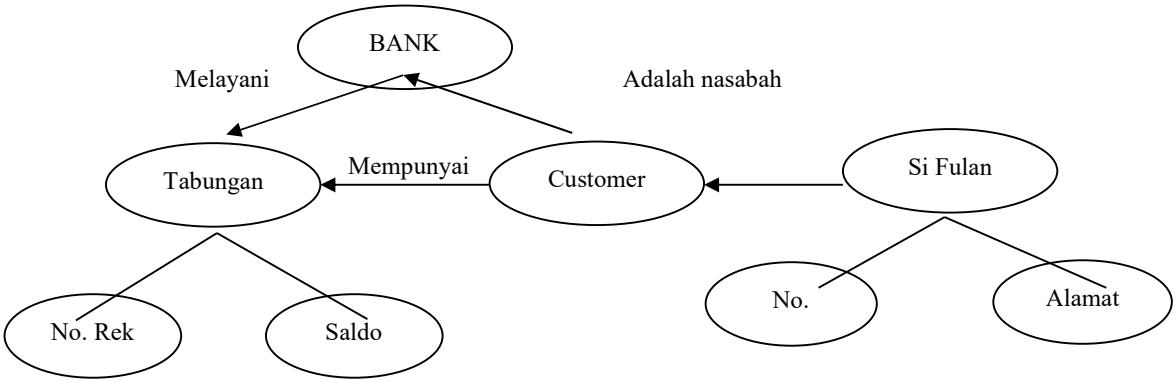
Entitas

Atribut

Relasi

3.3.2. Semantik model

Relasi antar objek dinyatakan dengan kata-kata (*semantic*)



Gambar 3.2. Model Data Objek - Semantik

Arti tanda :

⟶	Menunjukkan adanya relasi
———	Menunjukkan atribut

3.4. Model data berbasis record

Model ini mendasarkan pada record untuk menjelaskan kepada user tentang hubungan logik antar data dalam basis data. Ada 3 jenis yaitu :

3.4.1. Relational Model

Menjelaskan hubungan logik antar data dalam basis data dengan cara memvisualisasikan ke dalam bentuk tabel dua dimensi yang terdiri dari sejumlah baris dan kolom yang menunjukan atribut-atribut. Dalam relational database model, sebuah database adalah kumpulan relasi yang saling terhubung satu sama lainnya. Relasi adalah istilah dalam relational database, tapi lebih *familiar* disebut sebagai tabel. Selayaknya tabel yang memiliki kolom dan baris, kolom (*column*) disebut atribut, sedangkan baris (*row*) disebut *tuple*. Hal ini hanya sekedar penamaan, dan agar lebih gampang. Istilah-istilah dalam model basis data relasional:

a) Record, sebuah baris dalam suatu relasi. Disebut juga tuple
b) Cardinality, banyaknya record dalam sebuah relasi
c) Atribut, suatu kolom dalam sebuah relasi
d) Domain, batasan nilai dalam atribut dan tipe datanya
e) Derajat / degree, banyaknya kolom dalam relasi

f) Super Key

g) Candidate Key

Atribut atau sekumpulan atribut yang unik yang dapat digunakan untuk membedakan suatu record . Database dalam relational database dapat diserhanakan sebagai *sekumpulan tabel yang saling terhubung*. Setiap baris dari dalam tabel setidaknya harus memiliki sebuah kolom yang *unik*. Unik disini maksudnya tidak boleh sama. Contohnya, dalam tabel data_mahasiswa, kolom NIM (Nomor Induk Mahasiswa) akan menjadi kandidat yang bagus, karena tidak mungkin ada 2 mahasiswa yang memiliki NIM yang sama. NIM di sini disebut juga dengan *Candidate Key.* Nomor KTP juga merupakan candidate key yang bagus, setidaknya setiap orang akan memiliki nomor KTP yang berbeda-beda. Namun dalam beberapa kasus, nomor KTP tidak selalu ada, karena bisa saja seseorang belum memiliki KTP karena sesuatu dan lain hal. Beberapa karakteristik Candidate key: unik (tidak boleh berulang), tidak boleh memiliki nilai null (kosong), nilai dari candidate key akan sangat jarang berubah.

BAB 4. ENTITY RELATIONSHIP DIAGRAM

4.1. Tujuan Instruksional

A. Tujuan Instruksional Umum

Mahasiswa memahami konsep ERD

B. Tujuan Instruksional Khusus

Mahasiswa dapat menggambarkan ERD

4.2. Model ER - Pemodelan Database

a) Mode ER mode dikembangkan untuk memfasilitasi desain database dengan memungkinkan spesifikasi skema perusahaan yang mewakili struktur logis keseluruhan dari database.

b) Model ER sangat berguna dalam memetakan makna dan interaksi perusahaan dunia nyata ke skema konseptual. Karena kegunaan ini, banyak alat perancangan basis data menarik konsep dari model ER.

c) Model data ER menggunakan tiga konsep dasar:

1. set entitas,
2. hubungan set,
3. atribut.

d) Model ER juga memiliki representasi diagram yang terkait, diagram ER, yang dapat mengekspresikan struktur logis keseluruhan dari sebuah database secara grafis.

4.2.1. Set Entitas

a. Entitas adalah objek yang ada dan dapat dibedakan dari objek lain.

Contoh: orang tertentu, perusahaan, acara, pabrik

b. Kumpulan entitas adalah sekumpulan entitas dengan jenis yang sama yang berbagi properti yang sama.

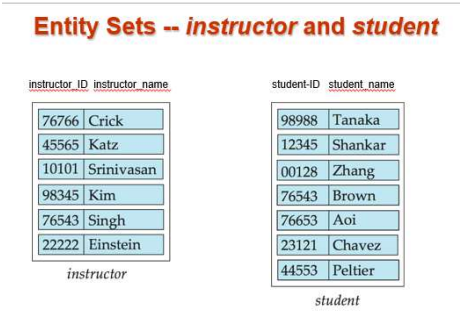
Contoh: kumpulan semua orang, perusahaan, pohon, hari libur

c. Suatu entitas diwakili oleh seperangkat atribut; yaitu, properti deskriptif yang dimiliki oleh semua anggota kumpulan entitas.

Contoh: instructor = (ID, name, street, city, salary)

1. course= (course_id, title, credits)

d. Bagian dari atribut membentuk kunci utama dari kumpulan entitas; yaitu, mengidentifikasi setiap anggota kumpulan secara unik.



4.2.2. Relationship Sets

Hubungan adalah asosiasi di antara beberapa entitas

Contoh : 44553 (Peltier) *advisor* 22222 (Einstein)

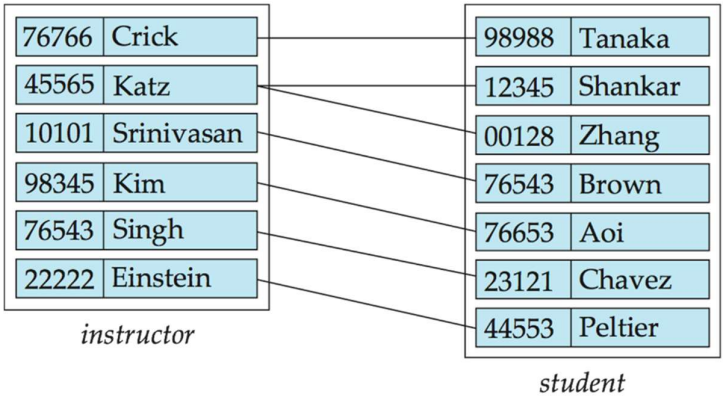
student entity relationship set *instructor* entity

a) Suatu himpunan relasi adalah hubungan matematis antara entitas n ≥ 2, yang masing-masing diambil dari kumpulan entitas.

$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$

dimana $(e_1, e_2, \ldots, e_n)$ adalah hubungan.

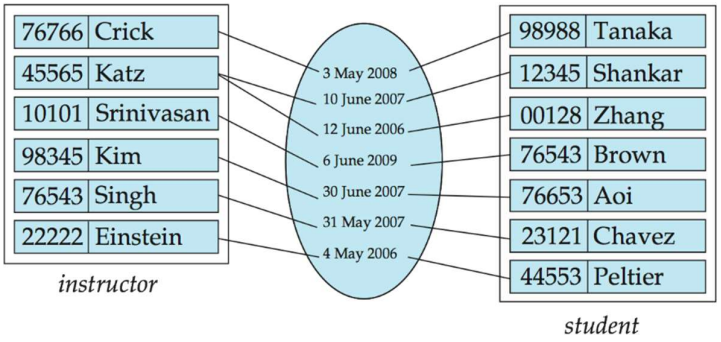
Contoh : (44553,22222) ∈ *advisor*



Relationship Set advisor

b) Atribut juga dapat dikaitkan dengan himpunan relasi.

Misalnya, hubungan advisor yang ditetapkan antara instruktur set entitas dan siswa dapat memiliki tanggal atribut yang melacak ketika siswa mulai dikaitkan dengan penasihat.



4.2.3. Degree of a Relationship Set

a) Hubungan biner

- melibatkan dua set entitas (atau gelar dua).
- sebagian besar himpunan relasi dalam sistem basis data adalah biner.

b) Hubungan antara lebih dari dua set entitas jarang terjadi. Sebagian besar hubungan bersifat biner. (Lebih lanjut tentang ini nanti.)

Contoh: siswa bekerja pada proyek-proyek penelitian di bawah bimbingan seorang instruktur.

hubungan proj_guide adalah hubungan terner antara instruktur, siswa, dan proyek.

4.2.4. Mapping Cardinality Constraints

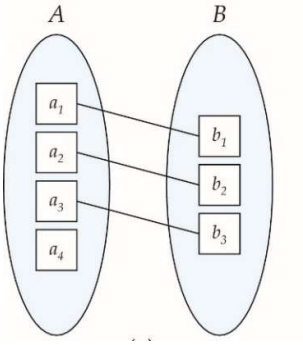
a) Ekspresikan jumlah entitas tempat entitas lain dapat dikaitkan melalui himpunan relasi.

b) Paling berguna dalam menggambarkan himpunan relasi biner.

c) Untuk hubungan biner, pengaturan kardinalitas pemetaan harus salah satu dari jenis berikut:

a. One to one (Satu ke satu)

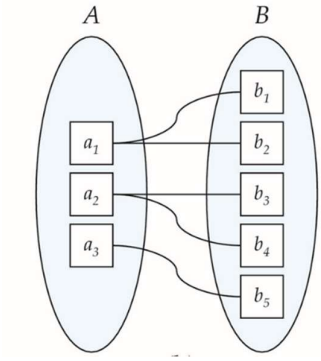
Setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B, dan begitu sebaliknya setiap entitas pada himpunan entitas B berhubungan dengan paling banyak satu entitas pada entitas A.



One to one

b. One to many (Satu ke banyak)

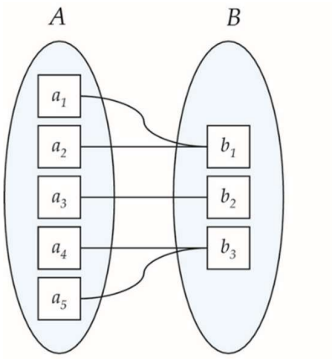
Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, dan tidak sebaliknya dimana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak satu entitas pada himpunan entitas A.



One to many

c. Many to one (Banyak ke Satu)

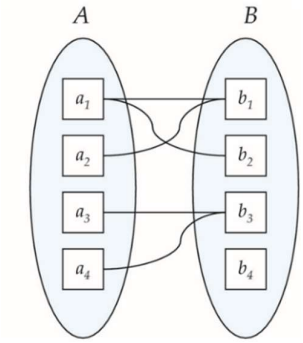
Setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B, dan tidak sebaliknya dimana setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.



Many to one

d. Many to many(Banyak ke Banyak)

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, dan sebaliknya dimana setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.



Many to many

4.2.5. Atribut Kompleks

- Simple and composite attributes.
- Single-valued and multivalued attributes

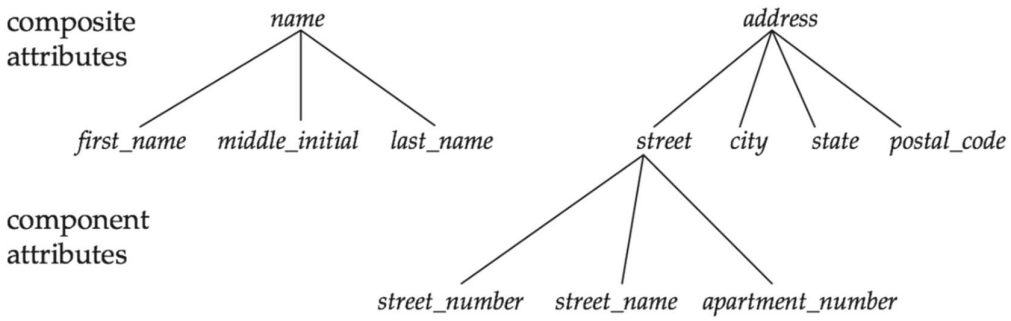
Example: multivalued attribute: *phone_numbers*

- Derived **attributes**

Dapat dihitung dari atribut lain

Example: age, given date_of_birth

- Domain **– himpunan nilai yang diizinkan untuk setiap atribut**



Gambar : Composite Attributes

4.2.6. Atribut Redundant

a) Misalkan kita memiliki set entitas:

   - instructor, with attributes: ID, name, dept_name, salary
   - department, with attributes: dept_name, building, budget

b) mencontohkan fakta bahwa setiap instruktur memiliki departemen terkait menggunakan hubungan menetapkan inst_dept.

c) Atribut dept_name muncul di kedua set entitas. Karena ini adalah kunci utama untuk departemen set entitas, itu mereplikasi informasi yang ada dalam hubungan dan karena itu berlebihan dalam mengatur instruktur entitas dan perlu dihapus.

d) TAPI: ketika mengkonversi kembali ke tabel, dalam beberapa kasus atribut akan diperkenalkan kembali, seperti yang akan kita lihat nanti.

4.2.7. Entitas Lemah

a) Pertimbangkan entitas bagian, yang diidentifikasi secara unik oleh course_id, semester, year , dan sec_id.

b) Jelas, entitas bagian terkait dengan entitas saja. Misalkan kita membuat hubungan mengatur sec_course antara bagian set entitas dan tentu saja.

c) Perhatikan bahwa informasi di sec_course redundan, karena bagian sudah memiliki atribut course_id, yang mengidentifikasi kursus yang terkait dengan bagian tersebut.

d) Salah satu pilihan untuk menangani redundansi ini adalah menyingkirkan hubungan sec_course; namun, dengan melakukannya hubungan antara bagian dan jalur menjadi tersirat dalam suatu atribut, yang tidak diinginkan.

e) Cara alternatif untuk menangani redundansi ini adalah dengan tidak menyimpan atribut course_id di entitas bagian dan hanya menyimpan atribut yang tersisa

section_id, year, dan semester. Namun, bagian kumpulan entitas tidak memiliki atribut yang cukup untuk mengidentifikasi entitas bagian tertentu secara unik; Meskipun masing-masing bagian entitas berbeda, bagian untuk kursus yang berbeda dapat berbagi section_id, tahun, dan semester yang sama.

f) Untuk mengatasi masalah ini, kami memperlakukan hubungan sec_course sebagai hubungan khusus yang memberikan informasi tambahan, dalam hal ini, course_id, yang diperlukan untuk mengidentifikasi bagian entitas secara unik.

g) Gagasan dari entitas yang lemah mengatur meresmikan intuisi di atas. Satu set entitas lemah adalah entitas yang keberadaannya bergantung pada entitas lain, yang disebut entitas identifikasinya; alih-alih mengasosiasikan kunci utama dengan entitas yang lemah, kami menggunakan entitas pengidentifikasi, bersama dengan atribut tambahan yang disebut diskriminator untuk mengidentifikasi entitas lemah secara unik. Satu set entitas yang bukan kumpulan entitas yang lemah disebut kumpulan entitas yang kuat.

h) Setiap entitas lemah harus dikaitkan dengan entitas pengidentifikasi; yaitu, kumpulan entitas yang lemah dikatakan keberadaan tergantung pada set entitas yang mengidentifikasi. Set entitas yang mengidentifikasi dikatakan memiliki entitas lemah yang ditetapkan yang diidentifikasinya. Hubungan yang menghubungkan entitas lemah yang ditetapkan dengan himpunan entitas pengidentifikasi disebut hubungan identifikasinya.

i) Perhatikan bahwa skema relasional yang akhirnya kita buat dari bagian kumpulan entitas memang memiliki atribut course_id, untuk alasan yang akan menjadi jelas nanti, meskipun kita telah menjatuhkan course_id atribut dari bagian set entitas.

4.3. E-R Diagrams

a) ERD merupakan notasi grafis dalam pemodelan data konseptual yang mendeskripsikan hubungan antara penyimpanan

b) ERD digunakan untuk memodelkan struktur data dan hubungan antar data

c) Dengan ERD kita dapat menguji model dengan mengabaikan proses yang harus dilakukan. Dengan ERD kita mencoba menjawab pertanyaan seperti:

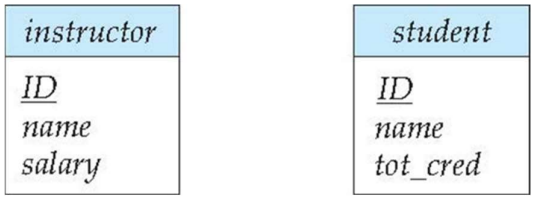
   - Data apa yang diperlukan ?
   - Bagaimana data yang satu berhubungan dengan yang lain ?

4.3.1. Entity

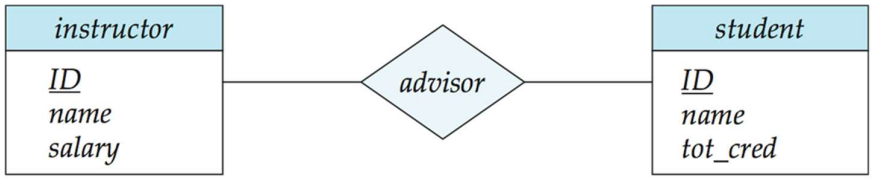
a) Suatu objek yang dapat diidentifikasi dalam lingkungan pemakai, sesuatu yang penting bagi pemakai dalam konteks sistem yang akan dibuat.

b) Individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain.

c) Entitas dapat direpresentasikan secara grafis sebagai berikut:
   - Rectangles mewakili set entitas.
   - Atribut yang tercantum di dalam persegi panjang entitas
   - Garis bawah menunjukkan atribut kunci primer

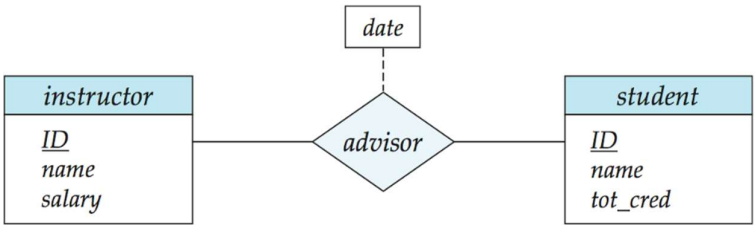


Gambar : contoh entitas

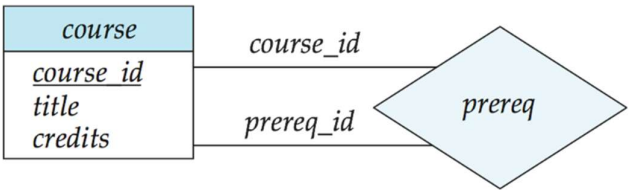


Gambar : hubungan entitas

Keterangan : bentuk Diamond mewakili set hubungan



Gambar : Hubungan entitas dengan atribut



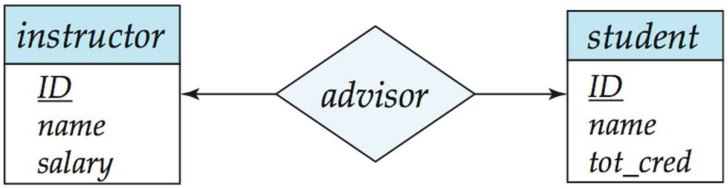
Gambar : roles

Keterangan :

a. Kumpulan entitas dari suatu hubungan tidak perlu berbeda

   Setiap kemunculan set entitas memainkan "peran" dalam hubungan

b. Label "course_id" dan "prereq_id" disebut sebagai peran.



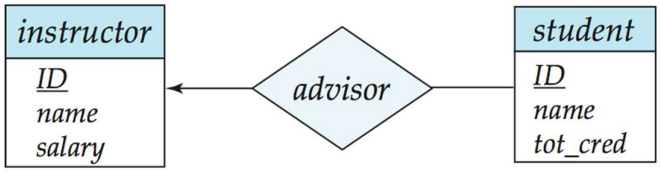
Gambar : Cardinality Constraints

Keterangan :

a. Kami mengungkapkan batasan kardinalitas dengan menggambar garis yang diarahkan (→), menandakan "satu," atau garis tidak terarah (-), menandakan "banyak," antara himpunan relasi dan kumpulan entitas.

b. Hubungan satu-ke-satu antara seorang instruktur dan seorang siswa:

   - Seorang siswa dikaitkan dengan paling banyak satu instruktur melalui penasihat hubungan
   - Seorang siswa dikaitkan dengan paling banyak satu departemen melalui stud_dept

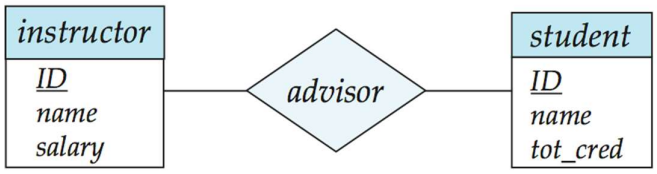


Gambar : Hubungan One-to-Many

Keterangan :

a. hubungan satu-ke-banyak antara seorang instruktur dan seorang siswa.

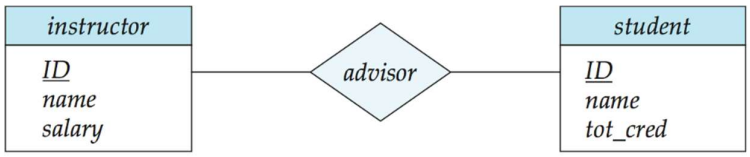
- instruktur dikaitkan dengan beberapa siswa (termasuk 0) melalui penasihat
- eorang siswa dikaitkan dengan paling banyak satu instruktur melalui penasihat



Gambar : Hubungan Many-to-One

Keterangan :

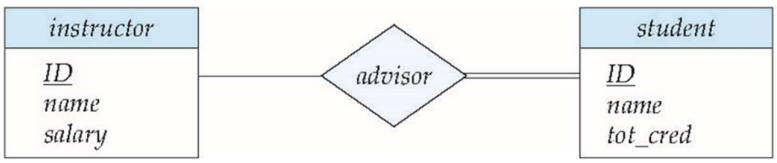
a. Dalam hubungan banyak-ke-satu antara seorang instruktur dan seorang siswa,
   - instruktur dikaitkan dengan paling banyak satu siswa melalui penasihat,
   - dan seorang siswa dikaitkan dengan beberapa (termasuk 0) instruktur melalui penasihat



Gambar : Hubungan Many-to-Many

Keterangan :

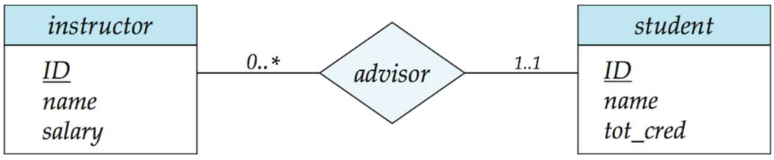
a. Seorang instruktur dikaitkan dengan beberapa siswa (mungkin 0) melalui penasihat
b. Seorang siswa dikaitkan dengan beberapa instruktur (mungkin 0) melalui penasihat



Gambar : Total and Partial Participation

Keterangan :

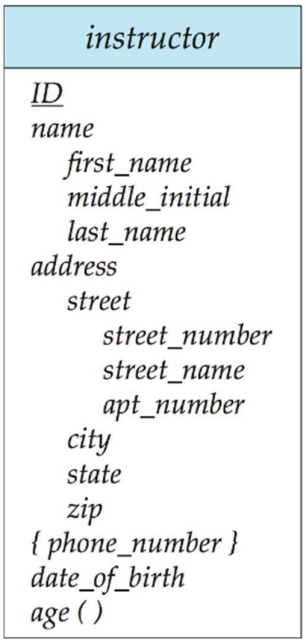
a. Partisipasi total (ditunjukkan oleh garis ganda): setiap entitas dalam kumpulan entitas berpartisipasi dalam setidaknya satu hubungan dalam himpunan relasi.
b. Partisipasi siswa dalam hubungan penasihat adalah total
   - setiap siswa harus memiliki instruktur yang terkait
c. Partial participation: beberapa entitas tidak dapat berpartisipasi dalam hubungan apa pun dalam himpunan relasi
   - Contoh: partisipasi instruktur dalam penasihat bersifat parsial



Gambar : Notation for Expressing More Complex Constraints

Keterangan :

a. Garis mungkin memiliki kardinalitas minimum dan maksimum terkait, ditunjukkan dalam bentuk l..h, di mana l adalah minimum dan h kardinalitas maksimum.
   - Nilai minimum 1 menunjukkan total partisipasi.
   - Nilai maksimum 1 menunjukkan bahwa entitas berpartisipasi dalam paling banyak satu hubungan
   - Nilai maksimum * menunjukkan tidak ada batasan.
b. Instruktur dapat memberi saran kepada 0 atau lebih siswa. Seorang siswa harus memiliki 1 penasihat; tidak dapat memiliki banyak penasihat.



Gambar : Notation to Express Entity with Complex Attributes

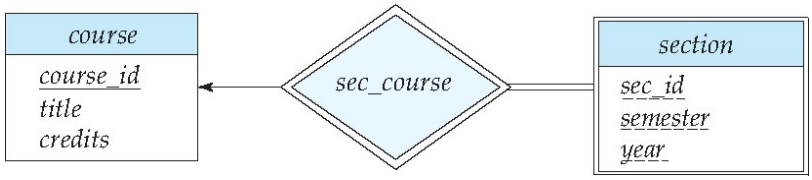
4.3.2. Mengekspresikan Entitas Lemah

a) Dalam diagram E-R, himpunan entitas yang lemah digambarkan melalui persegi panjang ganda.

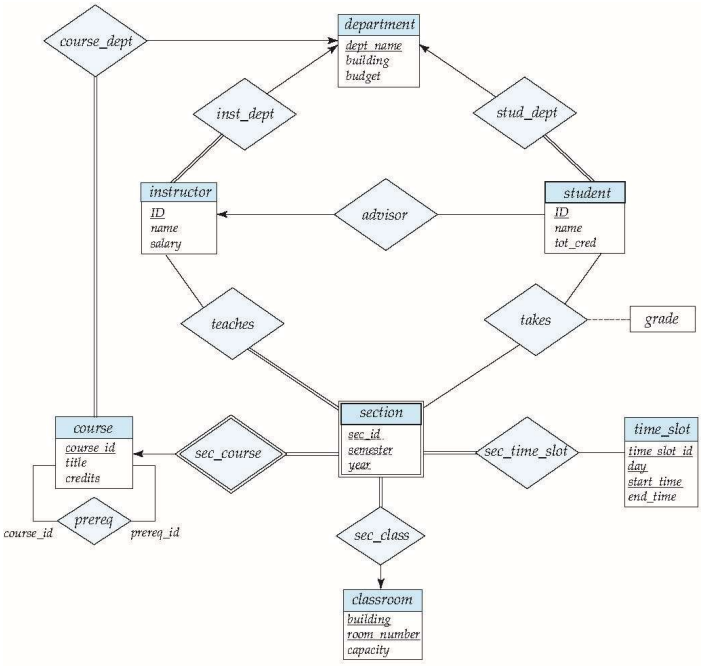
b) Kami menggarisbawahi diskriminator dari entitas lemah yang ditetapkan dengan garis putus-putus.

c) Hubungan yang mengatur menghubungkan entitas lemah yang ditetapkan ke kumpulan entitas kuat yang mengidentifikasi dilukiskan oleh berlian ganda.

d) Kunci utama untuk bagian - (course_id, sec_id, semester, year).



Gambar : skema entitas lemah



Gambar : E-R Diagram for a University Enterprise

4.4.Reduction to Relation Schemas(Penguranga n Skema Relasi)

a) Set entitas dan himpunan relasi dapat dinyatakan secara seragam sebagai skema relasi yang mewakili isi dari basis data.

b) Database yang sesuai dengan diagram E-R dapat diwakili oleh kumpulan skema.

c) Untuk setiap himpunan entitas dan himpunan hubungan ada skema unik yang diberi nama himpunan entitas terkait atau himpunan relasi.

d) Setiap skema memiliki sejumlah kolom (umumnya sesuai dengan atribut), yang memiliki nama unik.

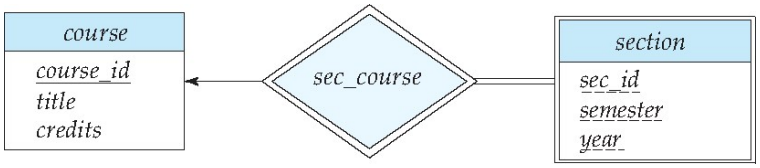
4.4.1. Menggambarkan Entitas (Representing Entity Sets)

a) Set entitas yang kuat mengurangi ke skema dengan atribut yang sama

student(ID, name, tot_cred)

b) Set entitas yang lemah menjadi tabel yang mencakup kolom untuk kunci primer dari kumpulan entitas kuat yang mengidentifikasi

bagian (course_id, sec_id, sem, year)

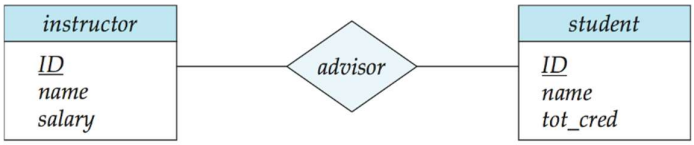


4.4.2. Menggambarkan Hubungan (Representing Relationship Sets)

a) Kumpulan hubungan banyak-ke-banyak direpresentasikan sebagai skema dengan atribut untuk kunci utama dari dua set entitas yang berpartisipasi, dan atribut deskriptif dari himpunan relasi.

Contoh: skema untuk advisor yang ditetapkan hubungan

advisor = (s_id, i_id)



4.4.3. Representasi Entitas dengan Atribut Komposit

a) Atribut komposit diratakan dengan membuat atribut terpisah untuk setiap atribut komponen

Contoh: diberikan entitas mengatur instruktur dengan nama atribut komposit dengan atribut komponen first_name dan last_name skema yang sesuai dengan kumpulan entitas memiliki dua atribut name_first_name dan name_last_name

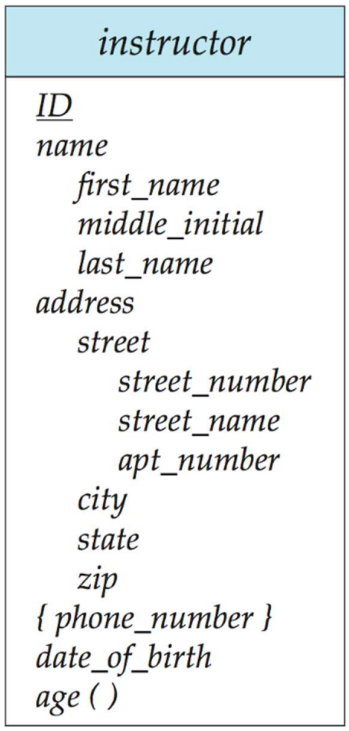
Prefix dihilangkan jika tidak ada ambiguitas

(name_first_name bisa menjadi first_name)

b) Mengabaikan atribut multinilai, skema instruktur diperpanjang.

instructor(ID,

first_name, middle_initial, last_name,
street_number, street_name,
apt_number, city, state, zip_code,
date_of_birth)



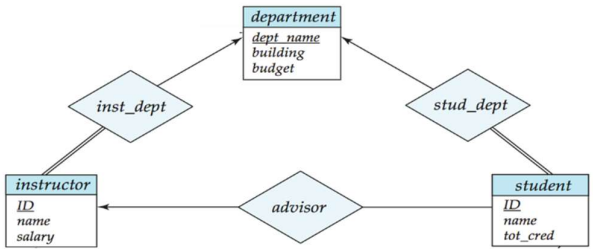
4.4.4. Representasi Entitas dengan Atribut Multinilai

a) Atribut multinilai M dari suatu entitas E diwakili oleh skema EM terpisah
b) Skema EM memiliki atribut yang bersesuaian dengan kunci primer E dan atribut yang sesuai dengan atribut multinilai M
c) Contoh: Atribut nomor ponsel multinilai diwakili oleh skema :? inst_phone = (ID, phone_number)
d) Setiap nilai dari atribut multinilai memetakan ke tuple terpisah dari relasi pada skema EM
    - Misalnya, entitas instruktur dengan kunci utama 22222 dan nomor telepon 456-7890 dan 123-4567 memetakan ke dua tupel:(22222, 456-7890) dan (22222, 123-4567)

4.4.5. Redundansi Skema (Redundancy of Schemas)

a) Set hubungan banyak-ke-satu dan satu-ke-banyak yang bersifat total pada banyak sisi dapat diwakili dengan menambahkan atribut ekstra ke sisi "banyak(many)", yang berisi kunci utama dari sisi "satu(one)"

b) Contoh: Daripada membuat skema untuk hubungan mengatur inst_dept, tambahkan atribut dept_name ke skema yang timbul dari instruktur set entitas.



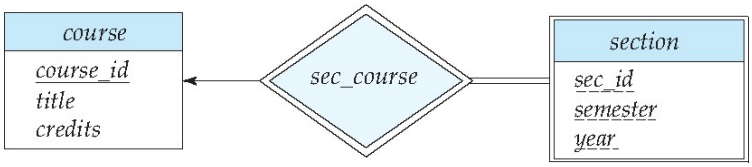
c) Untuk pengaturan hubungan satu-ke-satu, kedua sisi dapat dipilih untuk bertindak sebagai sisi "banyak(many)".

Yaitu, atribut ekstra dapat ditambahkan ke salah satu tabel yang terkait dengan dua set entitas.

d) Jika partisipasi bersifat parsial pada sisi "banyak(many)", mengganti skema dengan atribut ekstra dalam skema yang sesuai dengan sisi "banyak(banyak)" dapat mengakibatkan nilai nol.

e) Skema yang terkait dengan himpunan hubungan yang menghubungkan entitas lemah yang diset ke kumpulan entitas kuat yang mengidentifikasi bersifat berlebihan.

f) Contoh: Skema bagian sudah berisi atribut yang akan muncul di skema sec_course

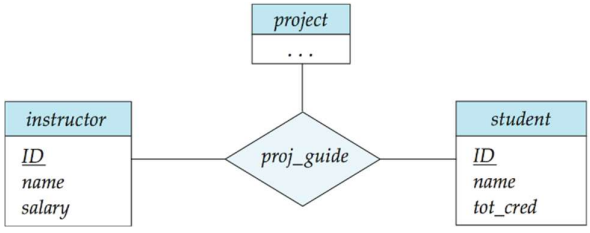


4.4.6. Kumpulan Hubungan Non-biner (Non-binary Relationship Sets)

a) Sebagian besar himpunan relasi adalah biner.

b) Ada saat-saat ketika lebih nyaman untuk merepresentasikan hubungan sebagai non-biner.

c) Diagram E-R dengan Hubungan Ternary



4.4.7. Batasan Kardinalitas pada Ternary Relationship (Cardinality Constraints on Ternary Relationship)

a) Kami mengizinkan paling banyak satu panah keluar dari hubungan terner (atau lebih tinggi) untuk menunjukkan kendala kardinalitas

b) Sebagai contoh, panah dari proj_guide ke instruktur menunjukkan setiap siswa memiliki paling banyak satu panduan untuk sebuah proyek

c) Jika ada lebih dari satu anak panah, ada dua cara untuk mendefinisikan artinya.

- Sebagai contoh, hubungan terner R antara A, B dan C dengan panah ke B dan C dapat berarti
    a. Setiap entitas A dikaitkan dengan entitas unik dari B dan C atau
    b. Setiap pasangan entitas dari (A, B) dikaitkan dengan entitas C yang unik, dan setiap pasangan (A, C) dikaitkan dengan B yang unik

d) Setiap alternatif telah digunakan dalam berbagai formalisme

e) Untuk menghindari kebingungan, kami melarang lebih dari satu panah

4.4.8. Spesialisasi (Specialization)

a) Proses desain top-down; kami menetapkan sub-pengelompokan dalam kumpulan entitas yang berbeda dari entitas lain dalam kumpulan.

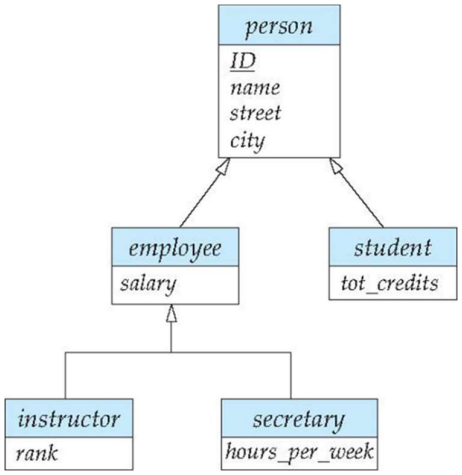
b) Sub-pengelompokan ini menjadi kumpulan entitas tingkat rendah yang memiliki atribut atau berpartisipasi dalam hubungan yang tidak berlaku untuk kumpulan entitas tingkat yang lebih tinggi.

c) Digambarkan oleh komponen segitiga berlabel ISA (mis., Instruktur "adalah" orang).

d) Attribute inheritance - set entitas level yang lebih rendah mewarisi semua atribut dan partisipasi hubungan dari entitas level yang lebih tinggi yang disetel ke mana ia terhubung.

e) Contoh Specialization:

- Overlapping – employee and student
- Disjoint – instructor and secretary
- Total and partial



4.4.8.1.Spesialisasi melalui Skema (Representing Specialization via Schemas)

a) Method 1:

1. Bentuk skema untuk entitas tingkat yang lebih tinggi
2. Membentuk skema untuk setiap set entitas level yang lebih rendah, termasuk kunci primer dari set entitas level yang lebih tinggi dan atribut lokal

schema	attributes
person	ID, name, street, city
student	ID, tot_cred
employee	ID, salary

3. Kelemahan: mendapatkan informasi tentang, karyawan memerlukan akses dua hubungan, yang sesuai dengan skema tingkat rendah dan yang sesuai dengan skema tingkat tinggi

b) Method 2:

1. Bentuk skema untuk setiap entitas yang ditetapkan dengan semua atribut lokal dan yang diwariskan.

schema	attributes
person	ID, name, street, city
student	ID, name, street, city, tot_cred
employee	ID, name, street, city, salary

2. Kelemahan: nama, jalan, dan kota dapat disimpan secara berlebihan untuk orang-orang yang merupakan siswa dan karyawan

4.4.9. Generalisasi (Generalization)

a) Proses desain bottom-up - menggabungkan sejumlah set entitas yang berbagi fitur yang sama ke dalam set entitas tingkat yang lebih tinggi.
b) Spesialisasi dan generalisasi adalah inversi sederhana satu sama lain; mereka direpresentasikan dalam diagram E-R dengan cara yang sama.
c) Istilah spesialisasi dan generalisasi digunakan secara bergantian.

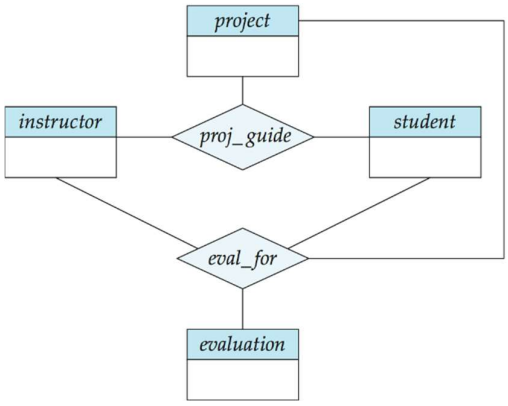
4.4.9.1. Desain pada Spesialisasi / Generalisasi

a) Batasan komplet - menentukan apakah entitas dalam set entitas level yang lebih tinggi harus termasuk paling tidak satu set entitas level bawah dalam suatu generalisasi.
    1. total: entitas harus milik salah satu set entitas tingkat lebih rendah
    2. parsial: entitas tidak perlu menjadi milik salah satu set entitas tingkat lebih rendah
b) Generalisasi parsial adalah default. Kita dapat menentukan generalisasi total dalam diagram ER dengan menambahkan total kata kunci dalam diagram dan menggambar garis putus-putus dari kata kunci ke kepala panah hampa terkait yang berlaku (untuk total generalisasi), atau ke rangkaian panah hampa -kursi yang berlaku (untuk generalisasi yang tumpang tindih).
c) Generalisasi siswa adalah total: Semua entitas siswa harus baik lulusan atau sarjana. Karena set entitas level yang lebih tinggi tiba melalui generalisasi umumnya terdiri dari hanya entitas-entitas dalam set entitas tingkat yang lebih rendah, batasan kelengkapan untuk set entitas tingkat lebih tinggi yang umum biasanya total.

4.4.10. Aggregation

a) Pertimbangkan hubungan proj_guide, yang kita lihat sebelumnya.

b) Misalkan kita ingin merekam evaluasi siswa dengan panduan tentang proyek.



c) Hubungan menetapkan eval_for dan proj_guide mewakili informasi yang tumpang tindih.

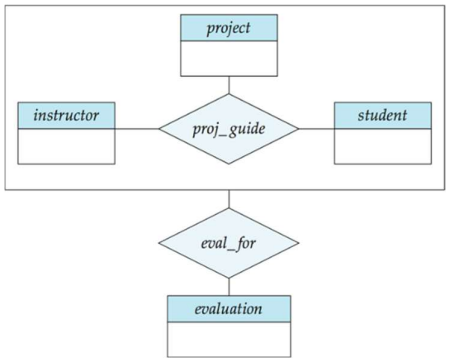
1. Setiap hubungan eval_for sesuai dengan hubungan proj_guide
2. Namun, beberapa hubungan proj_guide mungkin tidak sesuai dengan hubungan eval_for apapun
   - Jadi kita tidak dapat membuang hubungan proj_guide

d) Hilangkan redundansi ini melalui agregasi

1. Perlakukan hubungan sebagai entitas abstrak
2. Memungkinkan hubungan antar hubungan
3. Abstraksi hubungan menjadi entitas baru

e) Hilangkan redundansi ini melalui agregasi tanpa memperkenalkan redundansi, diagram berikut mewakili:

1. Seorang siswa dipandu oleh instruktur tertentu pada proyek tertentu
2. Seorang siswa, instruktur, kombinasi proyek dapat memiliki evaluasi yang terkait



4.4.10.1. Agregasi melalui Skema (Representing Aggregation via Schemas)

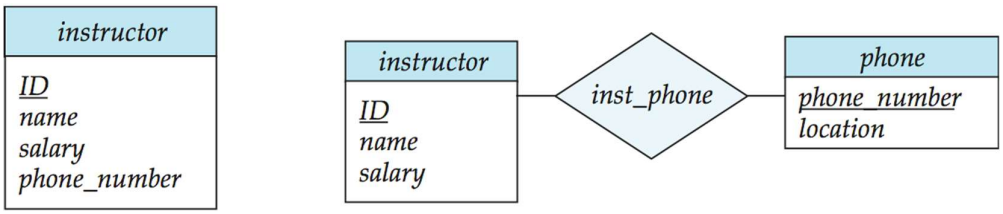
a) Untuk mewakili agregasi, buat skema yang berisi
   1. Kunci utama dari hubungan gabungan,
   2. Kunci utama dari kumpulan entitas terkait
   3. Atribut deskriptif apa pun

b) Dalam contoh kita:
   1. Skema eval_for adalah:
      eval_for (s_ID, project_id, i_ID, evaluation_id)
   2. Skema proj_guide adalah mubazir.

4.5.Design Issues

4.5.1. Entitas vs Atribut

a) Penggunaan set entitas vs. Atribut

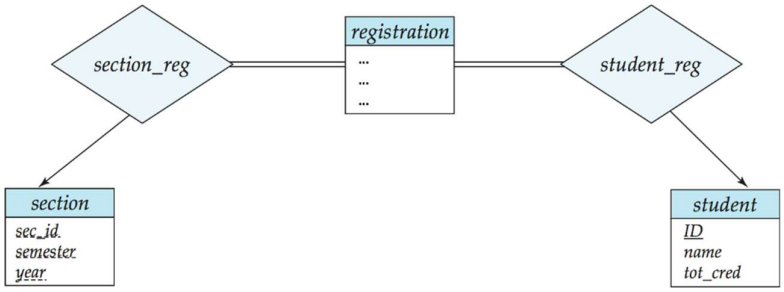


b) Penggunaan ponsel sebagai entitas memungkinkan informasi tambahan tentang nomor telepon (ditambah beberapa nomor telepon)

4.5.2. Entitas vs Set Hubungan

a) Penggunaan set entitas vs. set hubungan

Panduan yang memungkinkan adalah untuk menetapkan suatu hubungan yang ditetapkan untuk menggambarkan suatu tindakan yang terjadi antar entitas



b) Penempatan atribut hubungan

Misalnya, tanggal atribut sebagai atribut advisor atau sebagai atribut student.

4.5.3. Binary vs hubungan Non-Binary

a) Meskipun dimungkinkan untuk mengganti hubungan non-biner (n-ary, untuk n> 2) yang ditetapkan oleh sejumlah himpunan relasi biner yang berbeda, himpunan relasi n-ary menunjukkan lebih jelas bahwa beberapa entitas berpartisipasi dalam satu hubungan.

b) Beberapa hubungan yang tampak non-biner mungkin lebih baik diwakili menggunakan hubungan biner

- Sebagai contoh, hubungan orangtua terner, yang menghubungkan seorang anak dengan ayah dan ibunya, paling baik digantikan oleh dua hubungan biner, ayah dan ibu
  - Menggunakan dua hubungan biner memungkinkan informasi parsial (misalnya, hanya ibu yang diketahui)
- Tetapi ada beberapa hubungan yang secara alami tidak biner
  - Contoh: proj_guide

4.5.4. Konversi Hubungan Non-Biner ke Bentuk Biner

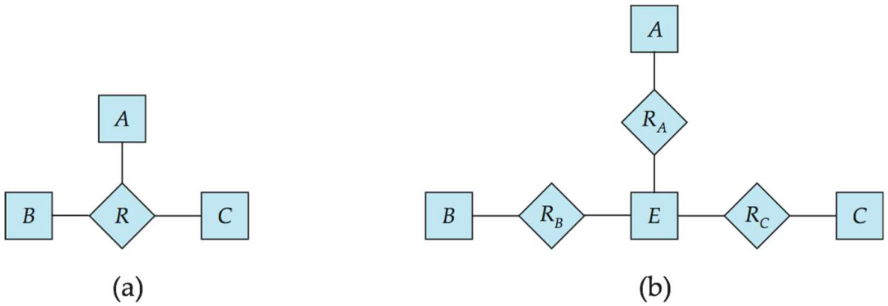
a) Secara umum, setiap hubungan non-biner dapat diwakili menggunakan hubungan biner dengan membuat kumpulan entitas buatan.

- Ganti R antara entitas set A, B dan C oleh entitas mengatur E, dan tiga set hubungan:

  1. R_A, terkait E dan A
  2. RB, terkait E dan B
  3. R_C, menghubungkan E dan C

- Buat atribut identifikasi untuk E dan tambahkan atribut apa pun dari R ke E
- Untuk setiap hubungan (ai, bi, ci) di R, buat

1. entitas baru ei dalam entitas mengatur E
2. menambahkan (ei, ai) ke RA
3. tambahkan (ei, bi) ke RB
4. tambahkan (ei, ci) ke RC

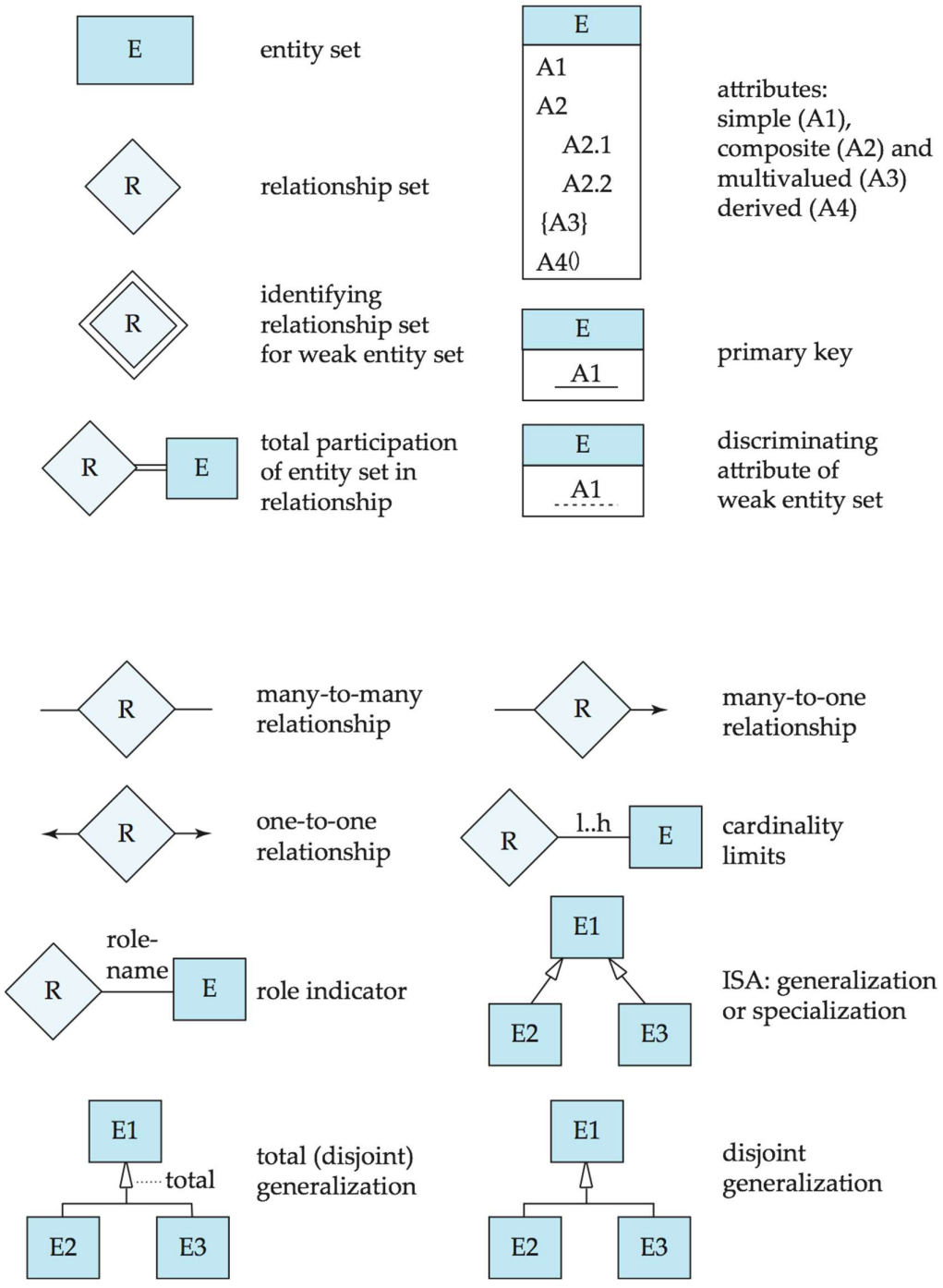


b) Juga perlu menerjemahkan kendala
   - Menerjemahkan semua kendala mungkin tidak dapat dilakukan
   - Mungkin ada contoh dalam skema terjemahan yang "tidak dapat berhubungan dengan contoh R
     - Latihan: tambahkan kendala pada hubungan RA, RB dan RC untuk memastikan bahwa entitas yang baru dibuat sesuai dengan tepat satu entitas di setiap entitas set A, B dan C
   - Kita dapat menghindari menciptakan atribut identifikasi dengan membuat E kumpulan entitas lemah (dijelaskan secara singkat) yang diidentifikasi oleh tiga himpunan relasi.

4.5.5. Keputusan Desain E-R

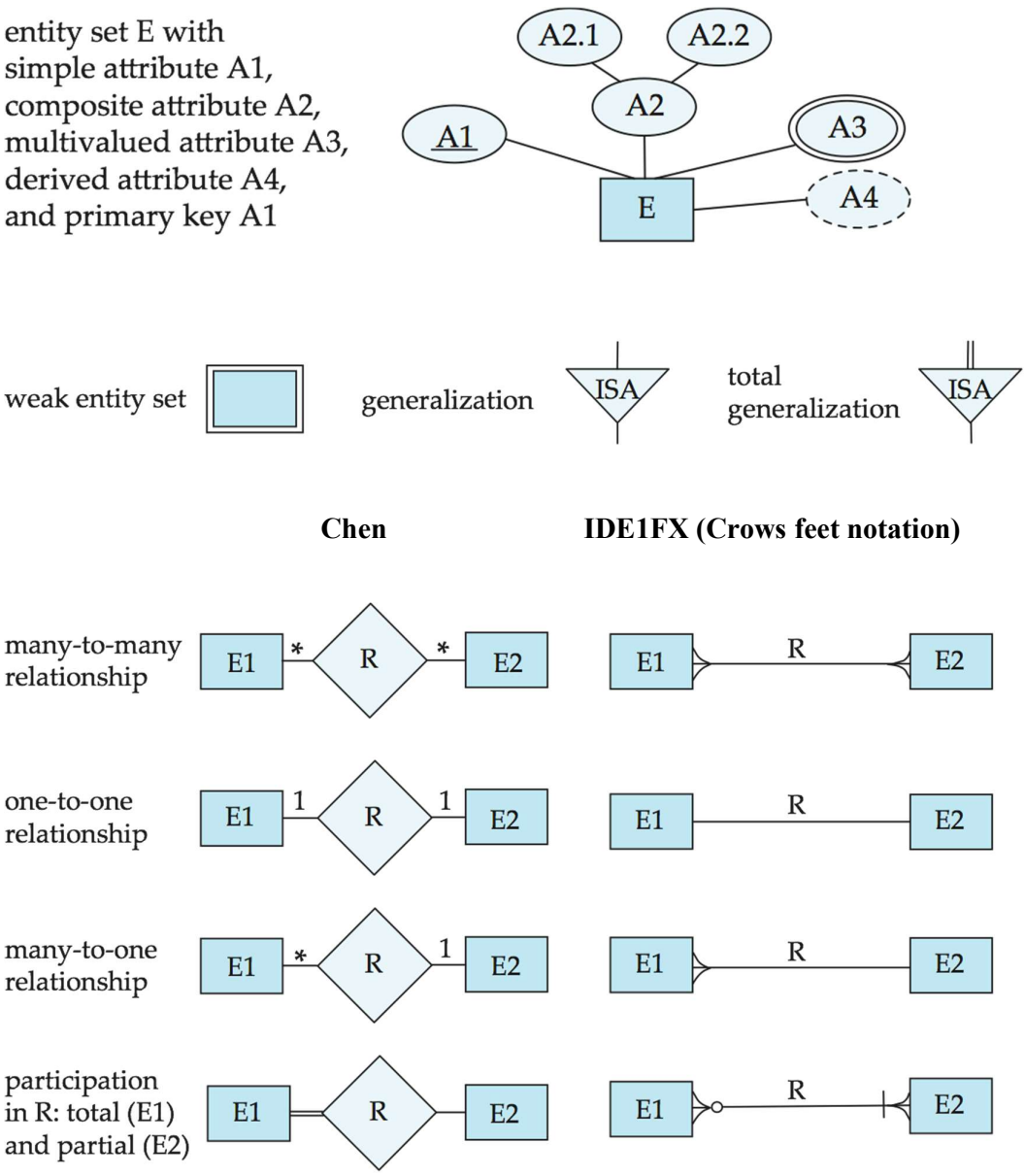
a) Penggunaan atribut atau entitas yang ditetapkan untuk mewakili suatu objek.
b) Apakah konsep dunia nyata paling baik diekspresikan oleh suatu himpunan entitas atau himpunan relasi.
c) Penggunaan hubungan terner versus sepasang hubungan biner.
d) Penggunaan set entitas yang kuat atau lemah.
e) Penggunaan spesialisasi / generalisasi - berkontribusi pada modularitas dalam desain.
f) Penggunaan agregasi - dapat memperlakukan entitas agregat yang ditetapkan sebagai satu kesatuan tanpa memperhatikan detail struktur internalnya.

4.5.6. Simbol yang Digunakan dalam Notasi E-R



4.5.7. Notifikasi ER Alternatif

a) Chen, IDE1FX, …



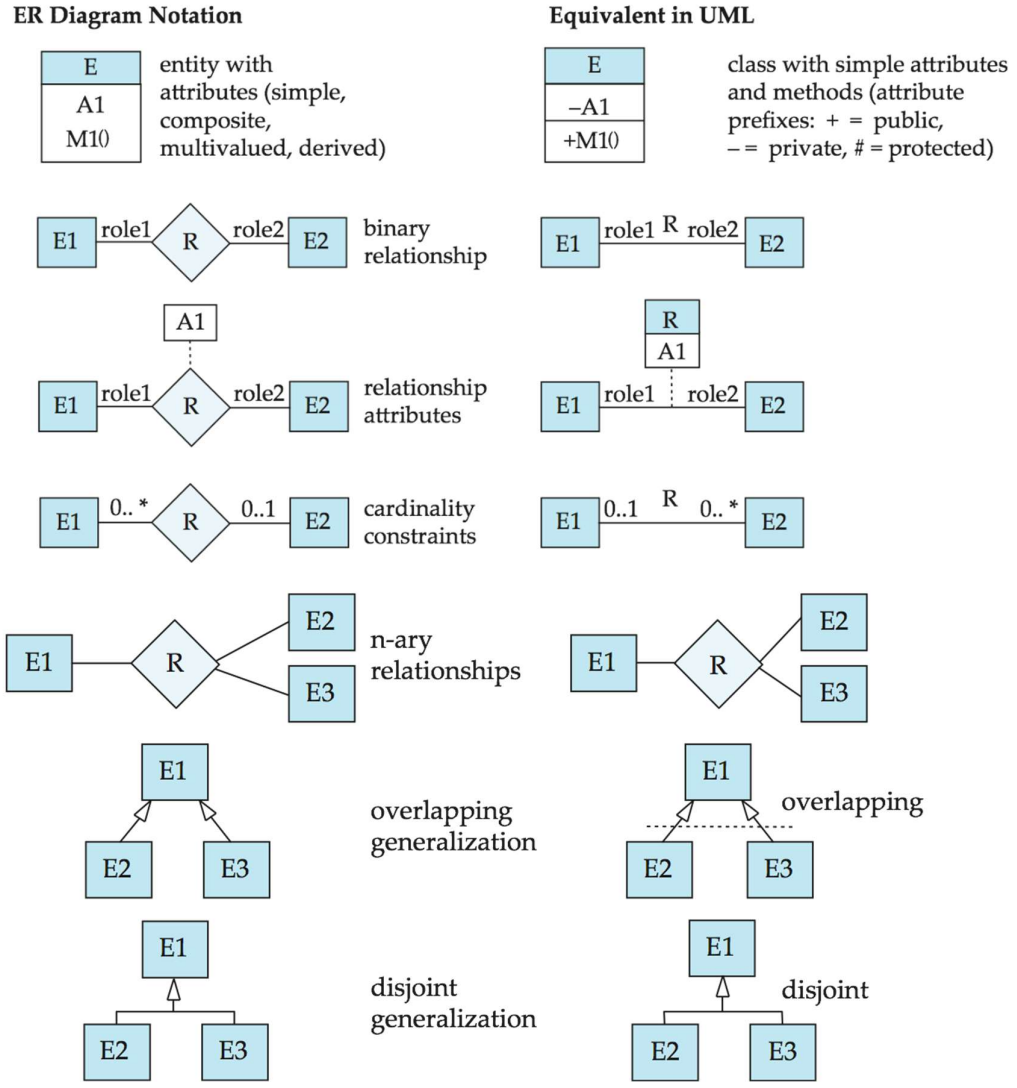
4.6. UML

a) UML : Bahasa Pemodelan Terpadu

b) UML memiliki banyak komponen untuk memodelkan aspek-aspek berbeda dari keseluruhan sistem perangkat lunak

c) Diagram Kelas UML sesuai dengan Diagram E-R, tetapi beberapa perbedaan.

4.6.1. ER vs. UML Class Diagrams



a) Set hubungan biner direpresentasikan dalam UML hanya dengan menggambar garis yang menghubungkan kumpulan entitas. Hubungan mengatur nama ditulis berdekatan dengan garis.

b) Peran yang dimainkan oleh entitas yang ditetapkan dalam himpunan relasi juga dapat ditentukan dengan menulis nama peran pada baris, berdekatan dengan kumpulan entitas.

c) Hubungan nama set dapat secara bergantian ditulis dalam kotak, bersama dengan atribut dari himpunan relasi, dan kotak terhubung, menggunakan garis putus-putus, ke garis yang menggambarkan himpunan relasi.

BAB 5. DIAGRAM ER LANJUTAN

5.1. Tujuan Instruksional

A. Tujuan Instruksional Umum

Mahasiswa memahami konsep ER lanjutan

B. Tujuan Instruksional Khusus

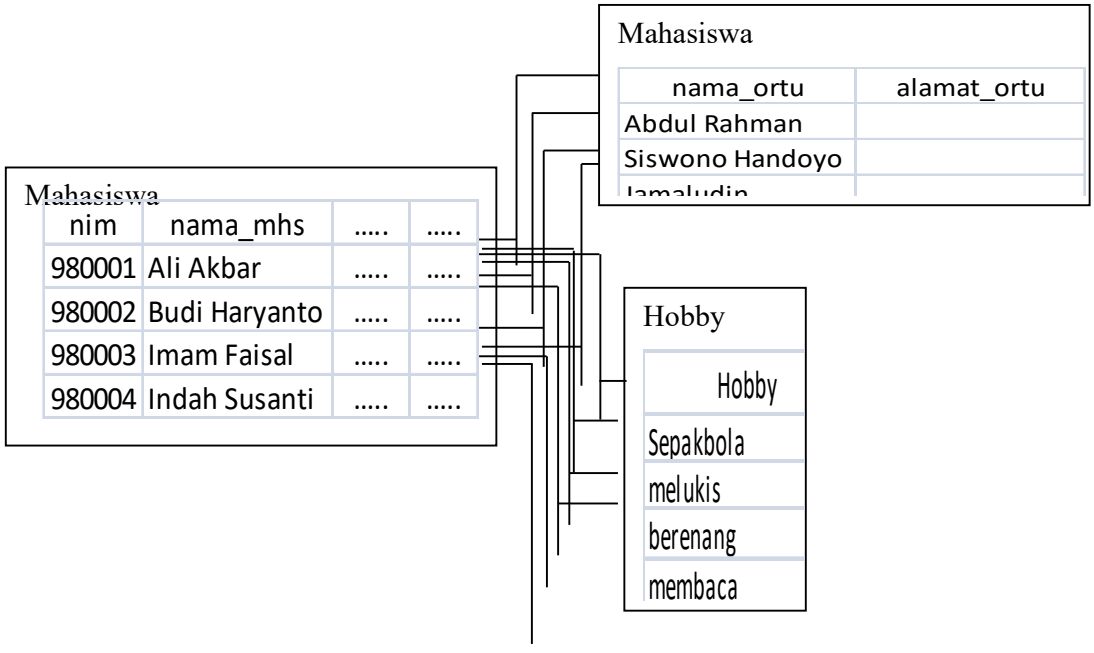
Mahasiswa dapat menerapkan ER lanjutan

5.2. Varian Entitas

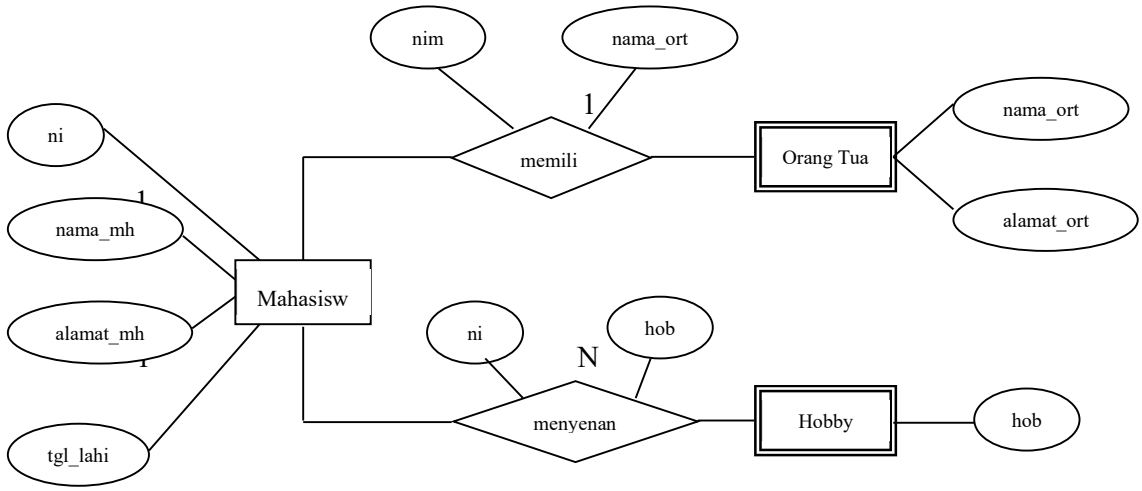
Idealnya, himpunan entitas yang kita libatkan dalam sebuah Diagram E-R adalah himpunan entitas kuat/bebas (Strong Entity Sets). Himpunan entitas demikian tidak memiliki ketergantungan dengan himpunan entitas lainnya. Himpunan entitas Mahasiswa, Dosen dan Kuliah sebagaimana yang ditunjukkan di bab sebelumnya dapat digolongkan sebagai himpunan entitas kuat, yang masing-masing dapat berdiri sendiri. Sebagai himpunan entitas yang kuat/bebas, kemunculan entitas-entitas di dalamnya tidak tergantung pada keberadaan entitas di himpunan entitas yang lain. Ketiga himpunan entitas tersebut juga bukan merupakan bagian (sub) dari himpunan entitas yang lain.

- Himpunan Entitas Lemah (*Weak Entity Sets*)

  Himpunan Entitas Lemah berisi entitas-entitas yang kemunculannya tergantung pada eksistensinya dalam sebuah relasi terhadap entitas lain (*Strong Entity*). Sebagai contoh, untuk melengkapi data mahasiswa kita juga ingin mengelola data hobbi dan orang tua. Berikut adalah contoh fakta yang dapat kita gunakan beserta relasi yang terjadi:



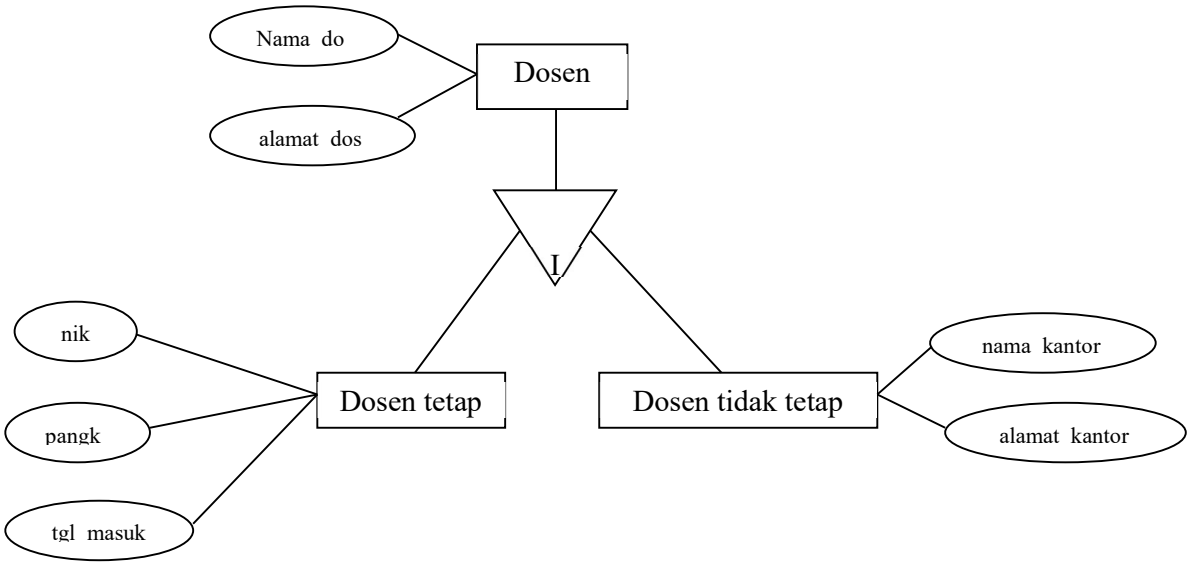
Gambar 5.1. Tabel Mahasiswa, Orang Tua dan Hobbi



Gambar 5.2. Orang Tua dan Hobby sebagai Entitas Lemah (*Weak Entity Set*)

- Sub Entitas (*Subtype Entities*)

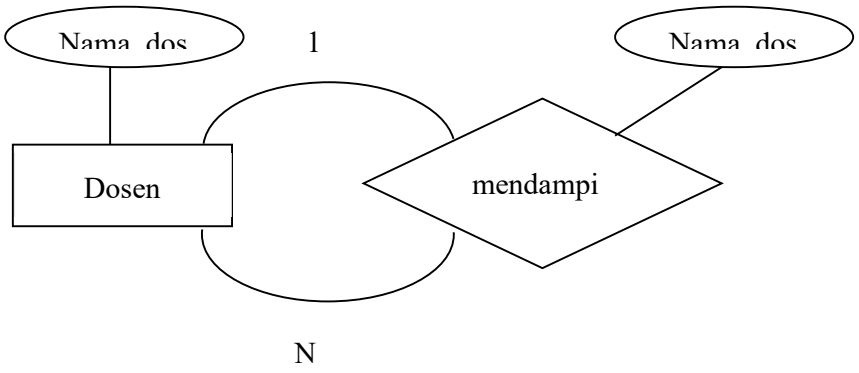
  Himpunan entitas yang beranggotakan entitas yang merupakan bagian dari himpunan entitas yang lebih superior/utama. Sub entitas ini merupakan hasil dekomposisi (spesialisasi) himpunan entitas berdasarkan pengelompokan tertentu. Contoh Sub Entitas :



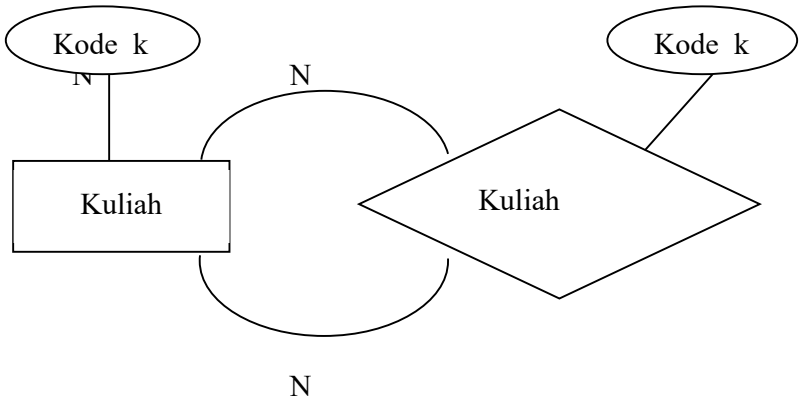
Gambar 5.3. Dua buah Sub Entitas sebagai Hasil Spesialisasi

5.3. Varian Relasi

- Relasi Tunggal (*Unary Relation*)
  Relasi yang terjadi dari sebuah himpunan entitas ke himpunan entitas yang sama.

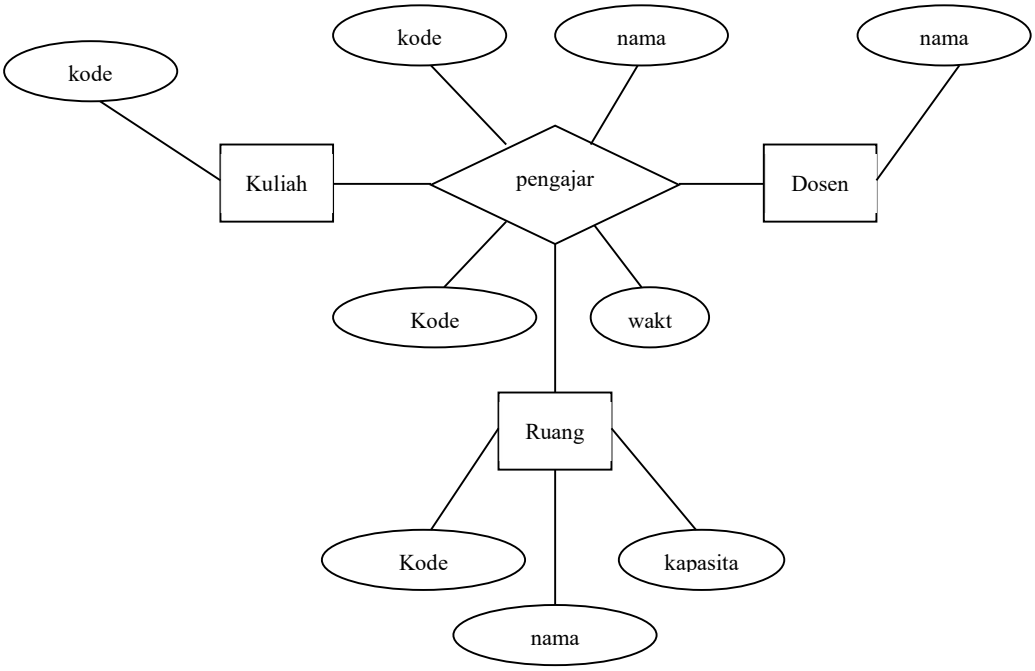


Gambar 5.4. Relasi Tunggal Dosen – Mendampingi



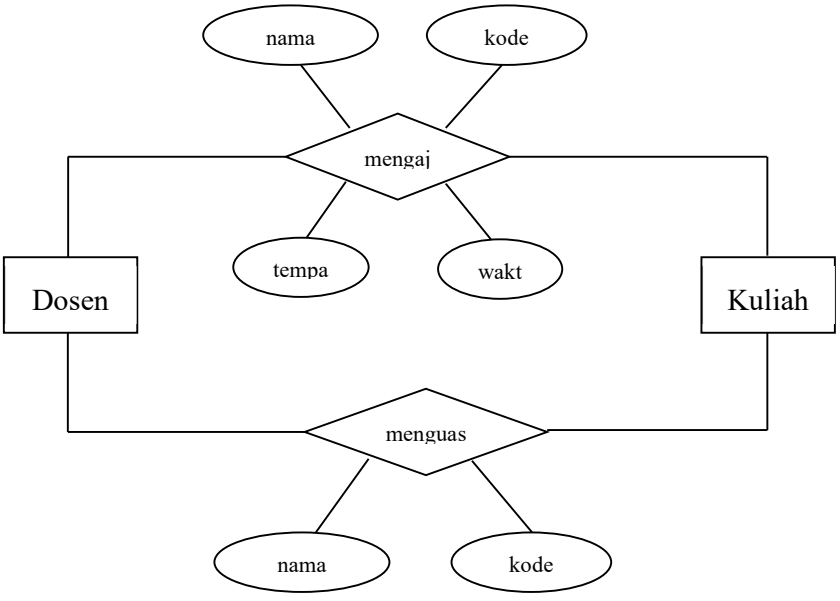
Gambar 5.5. Relasi Tunggal Kuliah - Prasyarat

- Relasi Multi Entitas (N-ary Relation)
  Relasi dari 3 himpunan entitas atau lebih.



Gambar 5.6. Relasi Multi Entitas

- Relasi Ganda (*Redundant Relation*)
  Relasi yang muncul antara dua himpunan entitas tidak hanya satu relasi.



Gambar 5.7. Relasi Ganda

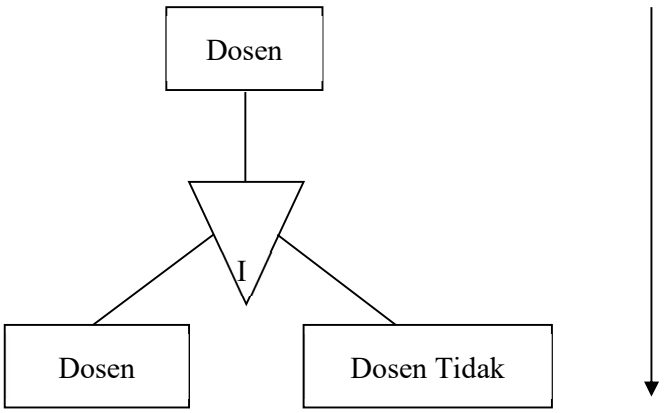
5.4. Spesialisasi dan Generalisasi

Pada sebuah himpunan entitas dimungkinkan adanya pengelompokan entitas-entitas yang menjadi anggotanya. Entitas yang ada pada himpunan entitas Dosen dapat dibagi dalam 2 kelompok, yaitu :

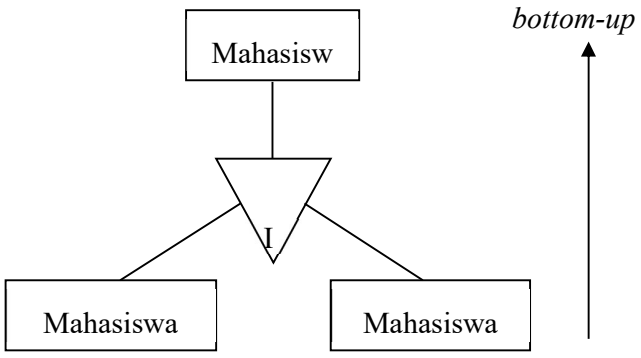
a. Dosen Tetap
b. Dosen Tidak Tetap

Adanya pembedaan atribut menyebabkan entitas dosen tidak mungkin disatukan dalam sebuah himpunan entitas saja. Karena itu pemisahan (spesialisasi) dilakukan.

- Spesialisasi
  Pengelompokan yang melahirkan himpunan entitas baru (*top-down*)
- Generalisasi
  Pengelompokan beberapa entitas melahirkan satu himpunan entitas dengan atribut sama (*bottom up*)



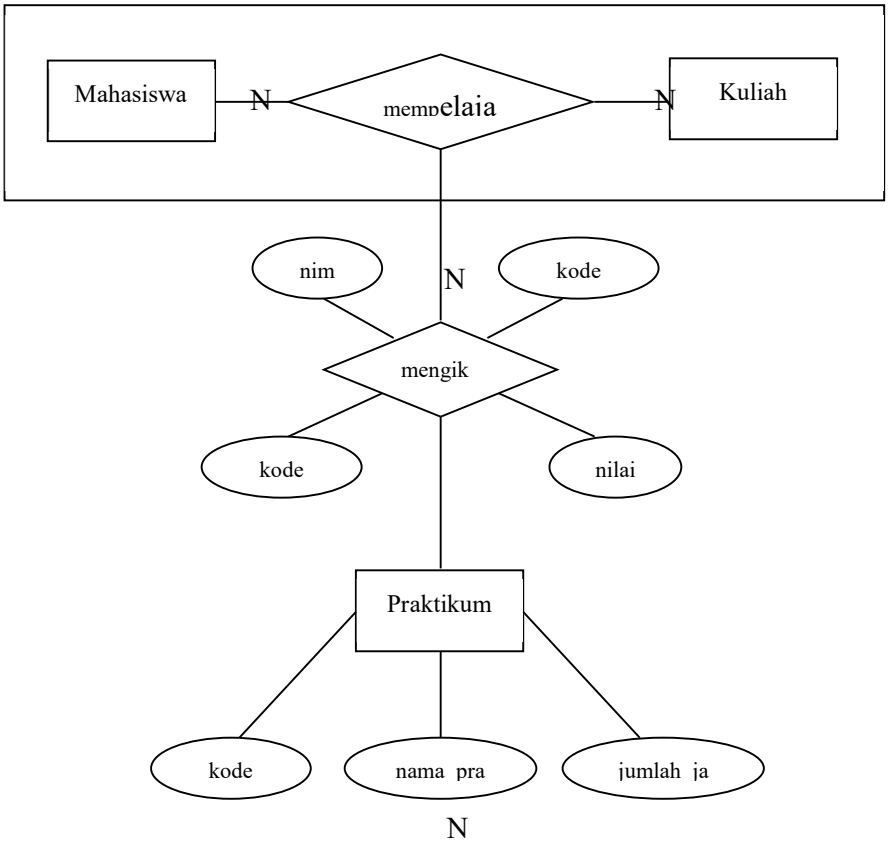
Gambar 5.8. Spesialisasi



Gambar 5.9. Generalisasi

5.5. Agregasi

Relasi yang secara kronologis mensyaratkan telah adanya relasi lain. Sebuah relasi terbentuk tidak hanya dari entitas tapi juga mengandung unsur dari relasi lain.



Gambar 5.10. Agregasi

5.6. Latihan dan Evaluasi

Tugas Pribadi

1. Suatu hasil ujian Negara cicilan adalah sebagai berikut :

NO_MHS	NAMA	MATA UJIAN	NIP	DOSEN	NILAI
11896	Ali	Basis Data	320001123	Ir. Rian	A
11897	Sita	Basis Data	320001123	Ir. Rian	B
11897	Sita	Fisika I	320022211	Johan, M.Sc.	A
11899	Edi	Basis Data	320001123	Ir. Rian	B

11901	Rika	SIM	320001123	Ir. Rian	A

a. Tabel apa saja yang diperlukan dan sebutkan pula nama-nama atributnya
b. Sebutkan kunci primer untuk masing-masing tabel!

2. Perusahaan computer Lucindo bermaksud mencatat transaksi pembelian barang dengan menggunakan computer. Data yang perlu dicatat :

NO_FAKTUR	Nomor faktur pembelian yang berasal dati pemasok
TGL_FAKTUR	Tanggal faktur
TGL_DITERIMA	Tanggal penerima barang
NAMA_PASOK	Nama Pemasok
ALAMAT_PASOK	Alamat pemasok
KOTA_PASOK	Kota pemasok
NAMA_BARANG	Nama barang yang dibeli
JUMLAH	Jumlah per barang

Perlu diketahui :

- Semua barang yang tercantum pada faktur diterima secara serentak.
- Sebuah faktur bisa mengandung bermacam-macam barang, sebagaimana terlihat pada gambar berikut.

PT ADIJAYA COMPUTER
Jl Cikini Raya 11 Telp. (0274) 589155
Yogyakarta

FAKTUR PENJUALAN
No : 1998000104
Tanggal : 09/03/1998

Kepada : PT Lucindo
Jl. Raya Janti
Yogyakarta

No	Barang	Jumlah	Harga Satuan	Total
1.	CD TVGA	2 box	1.000.000,-	2.000.000,-
2.	Monitor	1 buah	650.000,-	650.000,-
			Total	2.650.000,-

Terbilang : Dua juta enam ratus lima puluh ribu rupiah

- Ada kemungkinan bahwa faktur dari dua pemasok bernomor sama, sehingga tidaklah mungkin nomor faktur dipakai sebagai kunci primer.

Dengan berdasarkan teori perancangan basis data yang telah dibahas pada bab ini, tabel apa saja yang perlu dibentuk dan disebutkan pula atributnya? Cantumkan pula kunci primer untuk setiap tabel!

BAB 6. IMPLEMENTASI BASIS DATA

6.1. Tujuan Instruksional

C. Tujuan Instruksional Umum

D. Tujuan Instruksional Khusus

6.2. Pengantar

- Tahap implementasi basis data merupakan upaya untuk membangun basis data fisik yang ditempatkan dalam media penyimpan (disk) dengan bantuan DBMS
- Tahap ini diawali dengan melakukan transformasi dari model data yang telah selesai dibuat struktur basis data sesuai DBMS yang dipilih
- Secara umum, sebuah ERD akan diwujudkan menjadi sebuah **basis data** secara fisik. Sedangkan komponen ER yang berupa himpunan entitas dan himpunan relasi akan diwujudkan menjadi **tabel-tabel**. Atribut-atribut yang melekat pada masing-masing himpunan entitas dan himpunan relasi akan dinyatakan sebagai **field** dari tabel yang sesuai
- Performansi basis data ditentukan oleh :
  - Kualitas dan bentuk perancangan basis data
  - Kualitas mesin / komputer
  - Platform yang dipilih
  - Sistem operasi
  - DBMS yang digunakan

6.3. Pengkodean / Abstraksi data

- Data yang dilihat oleh pemakai awam (end-user) bisa berbeda dengan bagaimana data / informasi itu disimpan. Apa yang dilihat oleh end-user bisa jadi merupakan hasil pengolahan yang tidak disimpan sama sekali dalam basis data, atau bisa dinyatkan dalam bentuk lain
- Alasan untuk membuat suatu pengkodean adalah untuk efisiensi ruang penyimpanan
- Dari pemakaiannya, ada dua bentuk pengkodean :
  - **Eksternal (*user-defined coding*)**
    - Mewakili pengkodean yang telah digunakan secara terbuka dan dikenal dengan baik oleh pemakai awam

- Contoh : Nomor mahasiswa dan Kode matakuliah→sudah dikenal baik oleh pemakai awam

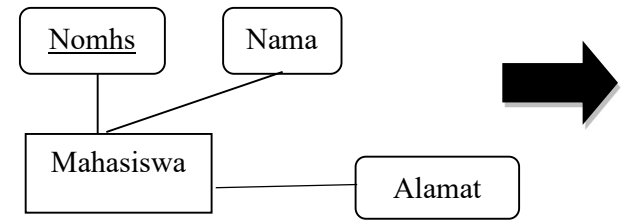
- **Internal (sistem coding)**
  - Menggambarkan bagaimana data disimpan dalam kondisi sebenarnya, sehingga lebih berorientasi pada mesin

- Ada tiga bentuk pengkodean :
  - Sekuensial
    - Pengkodean dilakukan dengan mengasosiasikan data dengan kode yang urut
    - Contoh : predikat kelulusan "Sangat Memuaskan", "Cukup Memuaskan", "Memuaskan" →dikodekan dengan huruf "A", "B", "C"
  - Mnemonic
    - Pengkodean dilakukan dengan membentuk suatu singkatan dari data yang hendak dikodekan.
    - Contoh : "Laki-laki"→dikodekan 'L"; "Perempuan"→dikodekan "P"
  - Blok
    - Pengkodean dinyatakan dalam format tertentu
    - Contoh : Nomor mahasiswa dengan format XX.YY.ZZZZ →terdiri atas XX = 2 digit tahun masuk, YY = 2 digit kode jurusan, ZZZZ = 4 digit nomor urut

6.4. Transformasi Model data ke Basis data fisik

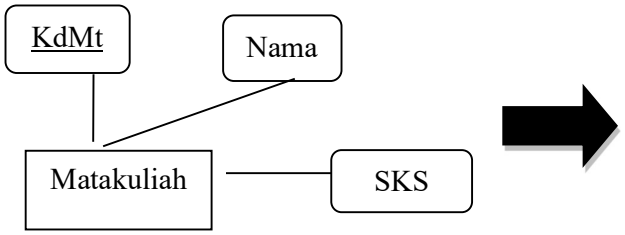
- Aturan umum dalam pemetaan model data yang digambarkan dalam ERD (levelkonseptual) menjadi Basis data fisik (level fisik) adalah :

  a. Setiap himpunan entitas diimplementasikan sebagai sebuah tabel (file data)



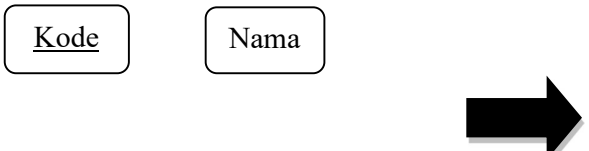
Tabel Mahasiswa

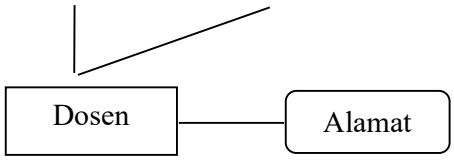
Nomhs	Nama	Alamat



Tabel Matakuliah

KdMtk	Nama	SKS



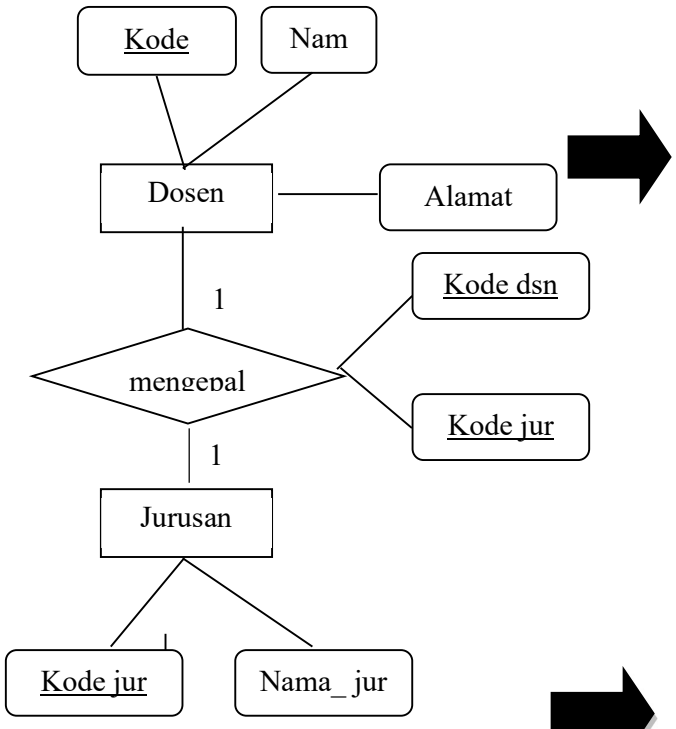


Tabel Dosen

Kode	Nama	Alamat

Gambar 6.1. Implementasi Entitas - Tabel

b. Relasi dengan derajat relasi satu-ke-satu, yang menghubungkan 2 buah entitas akan dipresentasikan dalam bentuk penambahan atribut relasi ke table yang mewakili salah satu dari kedua himpunan entitas.



Tabel Dosen

Kode_dsn	Nama	Alamat

Tabel Jurusan

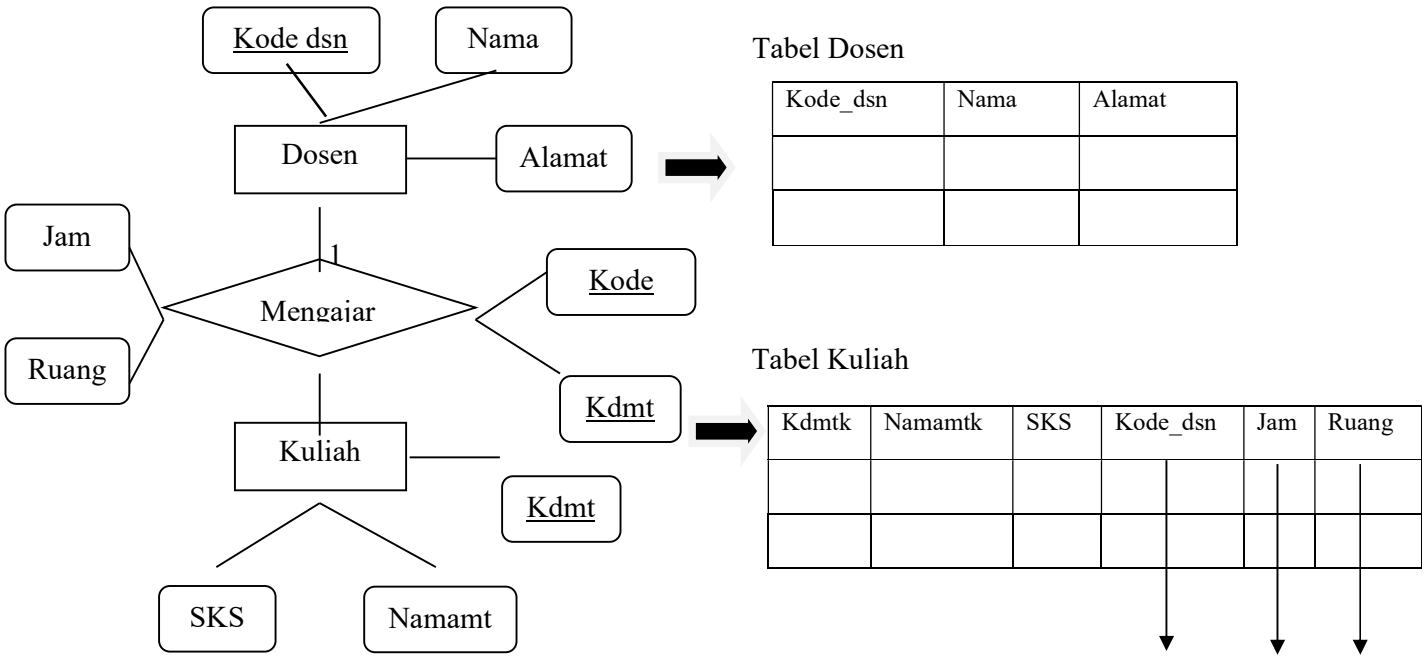
Kode_jur	Nama_jur	Kode_dsn

Field yang

ditambahkan dari relasi
Mengepalai

Gambar 6.2. Implementasi Derajat Relasi Satu ke Satu

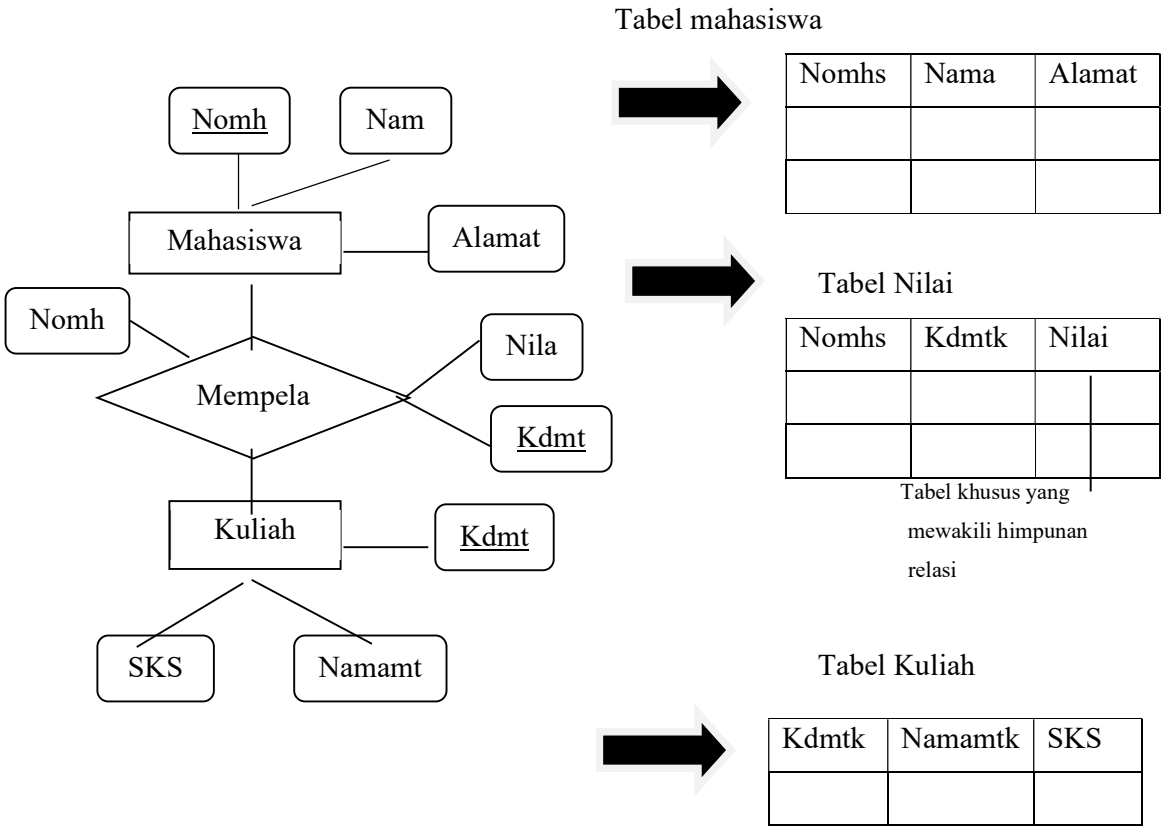
c. Relasi dengan derajat relasi satu-ke-banyak, yang menghubungkan 2 buah himpunan entitas, juga akan dipresentasikan dalam bentuk pemberian/pencantuman atribut kunci drai himpunan entitas pertama (yang berderajat 1) ke tabel yang mewakili himpunan entitas kedua (yang berderajat M)



Field dari relasi mengajar

Gambar 43. Implementasi Derajat Relasi Satu ke Banyak

d. Relasi dengan derajat banyak-ke-banyak, menghubungkan 2 entitas diwujudkan dalam bentuk tabel khusus, yang memiliki field (atau foreign key) yang berasal dari kunci dari himpunan entitas yang dihubungkannya



Gambar 5.4. Implementasi Derajat Relasi Satu ke Banyak

6.5. DBMS dan Struktur Tabel

- Dalam menentukan struktur dari tabel, paling tidak setiap struktur tabel berisikan nama field, tipe field, dan ukurannya
- Tatacara penamaan field, pilihan tipe field serta fasilitas tambahan lainnya untuk struktur tabel sangat tergantung pada DBMS yang digunakan
- Tipe data yang bersifat umum adalah :
  - **Data Alphanumerik**, isinya berupa angka tapi tidak menunjukkan jumlah, sehingga dianggap sebagai teks. Misalnya : Nomhs, NIP
  - **Data Numerik**, isinya berupa angka yang menunjukkan jumlah. Misalnya : SKS, gaji pokok.
  - **Data Bilangan Bulat (integer)**, Byte (1 byte), Small-integer (2 byte), Long integer (4 byte)
  - **Data bilangan nyata**, single (4 byte), Double (8 byte). Tipe data single dapat menampung hingga 7 digit pecahan, sedangkan double hingga 15 digit pecahan.

- Dalam komputasi, data integer akan membutuhkan waktu lebih cepat dalam pengolahan data dibandingkan real. Begitu juga, karena ruang penyimpanan yang dibutuhkan lebih kecil, maka data single akan lebih cepat dalam pengolahan dibandingkan double

- **Data uang (currency)**, pemakaian tipe ini sangat membantu dalam mengatur tampilan data yang berkaitan dengan nilai uang, misalnya dengan adanya pemisahan ribuan/jutaaan dan adanya tanda mata uang
- **Data teks**, ada dua jenis yaitu ukuran tetap (fixed character) dan ukuran dinamis (variable character). Misalnya fieldnomhs lebih tepat bertipe fixed character karena ukurannya pasti dan pendek. Sedangkan nama mahasiswa sebaiknya bertipe variable character karena panjang dan bervariasi.

- Pertimbangan dalam menentukan tipe data bagi setiap field adalah :
  - Kecukupan domain
    - Harus dapat menjamin bahwa tipe data yang dipilih pada tiap fieldakan dapat menampung semua nilai yang akan diisikan ke dalam field tersebut.
  - Efisiensi ruang penyimpanan
    - Apabila pemilihan tipe data tidak tepat (berlebihan), akibatnya akan memperbesar ukuran tabel secara keseluruhan
  - Kecepatan pengolahan data
    - Pada akhiranya, pemilihan tipe data yang tidak tepat juga mengakibatkan pengaksesan data menjadi lebih lambat.

6.6. Indeks dan Struktur penyimpanan

- Pada tahap implementasi, atribut-atribut entitas / relasi yang ditetapkan sebagai kunci (key) akan diwujudkan sebagai Indeks Primer (Primary index). Dan dapat juga ditambahkan Secondary index
- Ada 2 indeks

a. Indeks Primer (primary index)
   - IP pada setiap tabel hanya ada satu dan hampir selalu berasal (ditentukan) dari kunci primer yang telah ditetapkan dalam sebuah entitas / relasi
   - IP yang baik terdiri atas field-field dengan kriteria sbb :
     - Field yang menjadi komponen IP harus bersifat mandatory (datanya tidak boleh kosong atau berisi nilai null)
     - Keseluruhan nilai IP bersifat unik

- Nilai-nilainya lebih permanen (idealnya tidak pernah berubah)
- Berukuran kecil (pendek)dengan jumlah field minimal (sedikit)

b. Indeks sekunder (secondary index)

- Digunakan untuk mendukung keberadaan IP yang dibuat untuk suatu tabel dengan alasan untuk mempermudah berbagai cara pengaksesan ke suatu tabel
- Misalnya : field Nama_Mahasiswa→untuk memudahkan pencarian data berdasar nama mahasiswa; disamping pencarian berdasar NOMHS
- Catatan :
  - Jumlah IS dalam sebuah tabel boleh lebih dari Satur
  - Nilai-nilai field yang menjadi pembentuk IS tidak harus bersifat unik

6.7. Struktur penyimpanan

- Ada 7 pilihan struktur penyimpanan dasar yang dapat diterapkan pada suatu Tabel (bergantung pada DBMS yang dipakai) yaitu : Pile, Heap, hash, Sekuensial, Berindeks, File berindeks, Multiring

a. Heap

- Merupakan struktur penyimpanan yang paling sederhana dan paling hemat dalam kebutuhan ruang penyimpanan
- Setiap baris data disusun berdasar kronologis penyimpanannya. Record yang pertama disimpan akan ditempatkan di posisi awal ruang penyimpanan, dan begitu seterusnya.
- Pengubahan data tidak akan mengubah urutan record tersebut. Jika terjadi pengahpusan, maka record-record dibawahnya akan dimampatkan untuk mengisi tempat yang kosong akibat penghapusan.
- Pencarian data berjalan dengan lambat, karena dilakukan secara sekuensial baris demi baris
- Struktur ini cocok untuk tabel berukuran kecil dan jarang berubah

b. Hash

- Baris-baris data ditempatkan berdasar nilai alamat fisik yang diperoleh dari hasil perhitungan (fungsi hashing) terhadap nilai key-nya. Karena itu penempatan record dalam tabel tidak tersusun berdasarkan kedatangannya. Bisa jadi record yang terakhir dimasukkan justru menempati urutan pertama

- Memiliki performansi yang paling baik dalam hal pencarian data tunggal berdasar kunci indeks
- Struktur ini cocok untuk tabel-tabel yang sering menjadi acuan bagi tabel lain
- Kelemahannya membutuhkan ruang penyimpanan awal yang besar, untuk menjamin agar record-record yang disimpan tidak menempati alamat yang sama → dibutuhkan alokasi ruang penyimpanan

c. Sekuensial berindeks

- Menempatkan data engan urutan tertentu berdasarnilai indeks primernya Record yang memiliki nilai IP paling kecil dibandingkan record yang lain akan ditempatkan di awal ruang penyimpanan tabel meskipun dimasukkan belakangan
- Performansi turun pada saat terjadi penambahan atau perubahan data yang menyangkut nilai indeks primernya, karena perlu dilakukan penataan ulang
- Struktur ini cocok untuk tabel yang sifatnya statis, dan untuk pencarian data kelompok dalam suatu tabel (lebih baik daripada hash)

d. File berindeks

- Dikembangkan dari struktur heap. Record-record disusun berdasar kronologis penyimpanannya (seperti heap). Namun disediakan pula file indeks yang disusun berdasar nilai key setiap record yang berguna untuk membantu proses pencarian data ke suatu tabel
- Terdapat 2 komponen yaitu komponen data dan komponen indeks. Komponen data disusun dengan struktur heap, dan komponen indeks disusun dengan struktur sekuensial berindeks
- Struktur ini cocok untuk tabel yang dinamis dan berukuran besar

BAB 7. NORMALISASI

7.1. Tujuan Instruksional

E. Tujuan Instruksional Umum

Mahasiswa memahami konsep normalisasi

F. Tujuan Instruksional Khusus

Mahasiswa dapat melakukan normalisasi

7.2. Pengertian

a) Normalisasi adalah suatu teknik yang menstrukturkan data dalam cara-cara tertentu untuk membantu mengurangi atau mencegah timbulnya masalah yang berhubungan dengan pengolahan data dalam basis data

b) Kriteria yang mendefinisikan level-level pada normalisasi adalah bentuk normal (norm form)

c) Normalisasi adalah sebuah metode yang urut dalam penerapan aturan-aturan untuk mendesain database dengan tujuan agar (table menjadi normal):
   - Meminimalkan redundancy dan pengulangan data
   - Mempertahankan integritas data
   - Menambah konsistensi dan stabilitas
   - Menghilangkan potensi anomaly ketika mengolah data

d) Anomali yang dimaksud:
   - insertion anomalies, deletion anomalies, update anomalies.
   - Level-level dalam normalisasi disebut Normal Form (NF)
   - Pengagas pertama: Edgar F. Codd

7.3. Tujuan Normalisasi

a) Normalisasi perlu dilakukan agar kerelasian dalam basis data menjadi mudah dimengerti, mudah dipelihara, mudah memprosesnya, dan mudah untuk dikembangkan sesuai kebutuhan baru.

b) Tahapan Tahapan Normalisasi Normalisasi
   - Bentuk Normal tahap Pertama (1st Normal Form/1NF)
   - Bentuk Normal tahap Kedua (2nd Normal Form/2NF)

- Bentuk Normal tahap Ketiga (3rd Normal Form/3NF)
- Bentuk Normal tahap Keempat (4th Normal Form/4NF)
- Bentuk Normal tahap Kelima (5th Normal Form/5NF)

7.4. Penyimpangan dalam modifikasi

- Penyimpangan dalam proses modifikasi data disebut anomalies
- Ada 3 bentuk penyimpangan :

a. Delete anomalies

- Adalah proses penghapusan suatu entity logik yang mengakibatkan hilangnya informasi tentang entity yang tidak direlasikan secara logik

Tabel 7.1. Tabel Kuliah

No. Mhs	**Nama**	**Kode Mtk**	**SKS**
123456	Ali Baba	INA 101	3
123457	Pipiyot	TFD 234	2
123467	Nirmala	INA 201	3
123445	Lala	INA 101	3

Apabila “Ali baba” membatalkan mengambil matakuliah “INA 101”, maka apabila record tersebut dihapus akan menyebabkan seluruh informasi tentang ‘Ali baba” akan ikut terhapus

b. Insert anomalies

- Adalah proses penyisipan entity logic yang memerlukan penyisipan entity logik yang lain

c. Update anomalies

- Adalah proses mengupdate data pada suatu entity logik yang mengakibatkan perubahan pada lebih dari satu tempat dalam suatu relasi
- Contoh : Perubahan SKS pada “INA 101” tidak hanya dilakukan pada satu record saja, tetapi pada record dan relasi lain yang memuat data tersebut.

7.5. Keharusan menghilangkan masalah-masalah akibat ketergantungan

- Yang harus dilakukan adalah jika struktur data dalam relasi dirancang sedemikian rupa sehingga atribut-atribut bukan kunci hanya tergantung pada atribut kunci dan tidak pada atribut lain

- Ada 3 ketergantungan :

a. Functional Dependence (FD)

- FD akan muncul diantara dua rinci data dalam suatu struktur data jika nilai salah satu rinci data mengimplikasikan nilai pada rinci data kedua
- Atau rinci data pertama menentukan (determines) rinci data kedua
- Contoh :
  Matakuliah (Kode, Nama, SKS, Semester)
  FD= Matakuliah.Kode→(Matakuliah.Nama,Matakuliah.Semester)
  Matakuliah.nama→(Matakuliah.Kode, Matakuliah.Semester)

b. Full Functional Dependence (FFD)

- Suatu rinci data dikatakan FFD pada suatu kombinasi rinci data jika FD pada kombinasi rinci data dan tidak FD pada bagian lain dari kombinasi rinci data
- Contoh : SKS pada tabel matakuliah hanya bergantung pada kode matakuliah, dan tidak ditentukan oleh siapa yang mengambil matakuliah tersebut.

c. Transitive Dependence (TD)

- Muncul jika suatu nilai pada rinci data pertama menentukan nilai pada rinci data kedua yang bukan CK, dan nilai pada rinci data kedua menentukan nilai pada rinci data ketiga
- Jadi TD terjadi jika suatu nilai rinci data mempunyai ketergantungan dengan pada dua nilai rinci data

7.6. Efek-efek Normalisasi

- Akibat yang muncul dalam proses normalisasi :

a. Masalah kekangan dalam basis data
   - Duplikasi rinci data
   - Adanya Integritas referensial yang harus terjaga dan nilai-nilai pada AK tidak boleh null maka proses dekomposisi akan menghasilkan suatu set yang yang inheren pada batasan integritas referensial

b. Ketidakefisienan dalam menampilkan kembali data tersebut

7.7. Atribut Tabel

- Atribut adalah karakteristik atau sifat yang melekat pada sebuah tabel, atau disebut juga kolom data
- Pengelompokan atribut :

a. Atribut Key

- Adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris data dalam tabel secara unik (tidak boleh ada dua atau lebih baris data dengan nilai yang sama untuk atribut tertentu)
- Ada 3 key
  - Superkey
    - Merupakan satu atau kumpulan atribut yang dapat membedakan setiap baris data dalam sebuah tabel secara unik
    - Contoh : superkey di tabel mahasiswa
      - (no mhs, nama, alamat, tgl lahir)
      - (nomhs, nama, tgl lahir)
      - (nomhs, nama)
      - (nomhs)
  - Candidate key
    - Merupakan kumpulan atribut minimal yang dapat membedakan setiap baris data dalam sebuah tabel secara unik
    - Candidate key adalah satu atau lebih kolom yang nilainya dapat digunakan untuk memberi identitas unik sebuah baris dalam table.
    - Sebuah CK pasti superkey, tapi belum tentu sebaliknya
    - Contoh : pada tabel mahasiswa
      - (nomhs)
      - (nama)
  - Primary key
    - Sebuah Primary Key adalah satu atau lebih kolom yang nilainya untuk memberi identitas unik sebuah baris dalam tabel. Primary key dipilih diantara candidate key yang ada.
    - Dari beberapa CK dapat dipilih satu untuk dijadikan PK, yang memiliki keunikan paling baik
    - Contoh : dari tabel mahasiswa, yang layak dijadikan PK adalah no.mhs

b. Atribut deskriptif
   - Merupakan atribut yang bukan merupakan anggota dari PK

c. Atribut sederhana
   - Adalah atribut atomik yang tidak dapat dipilah lagi
   - Contoh : Nomhs, Nama

d. Atribut Komposit
   - Adalah atribut yang masih bisa diuraikan lagi menjadi sub-atribut yang masing-masing memiliki makna
   - Contoh : Alamat→Alamat, Kota, Propinsi, Kode Pos

e. Atribut bernilai tunggal
   - Ditujukan pada atribut-atribut yang memiliki paling banyak satu nilai untuk setiap baris data
   - Contoh : Nomhs, Nama, Tanggal lahir →hanya dapat berisi satu nilai untuk seorang mahasiswa

f. Atibut bernilai banyak
   - Ditujukan pada atribut-atribut yang dapat diisi dengan lebih dari satu nilai, tapi jenisnya sama
   - Contoh : pada tabel mahasiswa dapat ditambah atribut HOBBY, karena seorang mahasiswa dapat memiliki beberapa hobby

g. Atribut harus bernilai (mandatory)
   - Adalah atribut yang tidak boleh kosong, atau harus ada nilainya. Misalnya data Nomhs dan Nama mahasiswa
   - Nilai NULL digunakan untuk mengisi atribut yang demikian yang nilainya belum siap atau tidak ada
   - NULL (karakter ke 0) tidaksama dengan SPASI (karakter ke 32).

7.8. Domain dan tipe data

- Domain, memiliki pengertian yang hampir sama dengan tipe data, namun domain lebih ditekankan pada batas-batas nilai yang diperbolehkan pada suatu atribut
- Contoh : data SKS bertipe integer. Namun dalam kenyataan tidak ada sks yang bernilai negatif. Berarti domain nilai sks adalah integer > 0
- Tipe data merujuk pada kemampuan penimpanan data yang mungkin bagi suatuatribut secara fisik a kenyataan pemakaiann

7.9. Bentuk-bentuk normal

- Normalisasi merupakan sebuah teknik dalam logical desain sebuah basis data, teknik uk struktur relasi yang baik (tanpa redundansi)
- Istilah yang disepakati sebelum Normalisasi:

Formal Name	*Common Name*	*Also Known As*
Relation	Table	Entity, File
Tuple	Row	Record
Attribute	Column	Field

- Kasus Normalisasi

PT. AMIKOM Invoice Order barang
Jl. Ring Road Utara

Nama Cus : **Andi Sun**
Alamat : **Jl. Kaliurang No.90 Yk**

No Order : N001 Tanggal 12-Jul-07

Kode	Nama Barang	Qty	Hrg Sat	Sub Total
B001	RINSO	4	5,000	20,000
B002	POLO	5	400	2,000
			Total	22,000

PT. AMIKOM Invoice Order barang
Jl. Ring Road Utara

Nama Cus : **Sutikno**
Alamat **Jl. Kali Ola No. 89 Solo**

No Order : N002 Tanggal 14-Jul-07

Kode	Nama Barang	Qty	Hrg Sat	Sub Total
B003	DANCOW	5	7,000	35,000
B001	RINSO	3	5,000	15,000
B004	MIE GORENG	5	1,000	5,000
			Total	55,000

- Bentuk Tidak Normal

Bentuk Tidak Normal

NoOrder	NmCus	AlmtCus	TglOrder	KdBrg	NmBrg	Qty	HrgSat	SubTotal	Total
N001	Andi Sun	Jl. Kaliurang No.90 Yk	12-Jul-06	B001	RINSO	4	5,000	20,000	22,000
				B002	POLO	5	400	2,000	
N002	Sutikno	Jl. Kali Ola No. 89 Solo	14-Jul-06	B003	DANCOW	5	7,000	35,000	55,000
				B001	RINSO	3	5,000	15,000	
				B004	MIE GORENG	5	1,000	5,000	

Bentuk Flat File

NoOrder	NmCus	AlmtCus	TglOrder	KdBrg	NmBrg	Qty	HrgSat	SubTotal	Total
N001	Andi Sun	Jl. Kaliurang No.90 Yk	12-Jul-06	B001	RINSO	4	5,000	20,000	22,000
N001	Andi Sun	Jl. Kaliurang No.90 Yk	12-Jul-06	B002	POLO	5	400	2,000	22,000
N002	Sutikno	Jl. Kali Ola No. 89 Solo	14-Jul-06	B003	DANCOW	5	7,000	35,000	55,000
N002	Sutikno	Jl. Kali Ola No. 89 Solo	14-Jul-06	B001	RINSO	3	5,000	15,000	55,000
N002	Sutikno	Jl. Kali Ola No. 89 Solo	14-Jul-06	B004	MIE GORENG	5	1,000	5,000	55,000

- Bentuk-bentuk normal

  a. Normal pertama (1st normal form)

  - Syarat:
    - Tidak ada atribut yang duplikat dalam sebuah tabel.
    - Tidak ada baris yang duplikat dalam sebuah tabel.
    - Nilai cell dalam harus atomic value (single value).
    - Tidak ada pengulangan group data (pengulangan item dikolom).
  - Langkah:
    - Hilangkan atribut yang duplikat.
    - Buatlah menjadi table terpisah untuk masih-masing group data dan buat atribut relasinya (jika ada).
    - Identifikasi setiap set relasi data dengan satu atau beberapa kolom unik (primary key).

  - Aturan :
    - Mendefinisikan atribut kunci
    - Tidak adanya grup berulang
    - Semua atribut bukan kunci tergantung pada atribut kunci

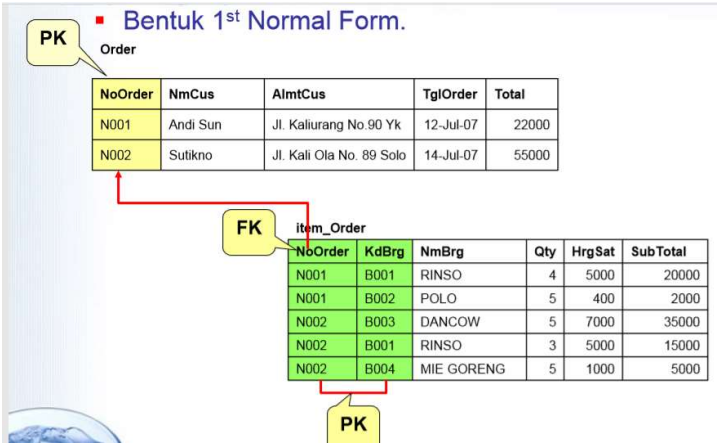
  - Yang dilarang di dalam 1st normal form

NoNota	NmCus	Barang	Qty	HrgBrg	Sub Total
N001	Andi Sun	RINSO;POLO	4;5	5000;500	20000

NoNota	NmCus	Brg1	JmlBrg1	Hrgbrg1	Brg2	JmlBrg2	Hrgbrg2	SubTotal
N001	Andi Sun	RINSO	4	5000	POLO	5	500	20,000

NoNota	NmCus	NmBrg	Qty	HrgSat	SubTotal
N001	Andi Sun	RINSO	4	5,000	20,000
		POLO	5	400	2,000
N002	Sutikno	DANCOW	5	7,000	35,000
		RINSO	3	5,000	15,000
		MIE GORENG	5	1,000	5,000

  - Bentuk 1st normal form

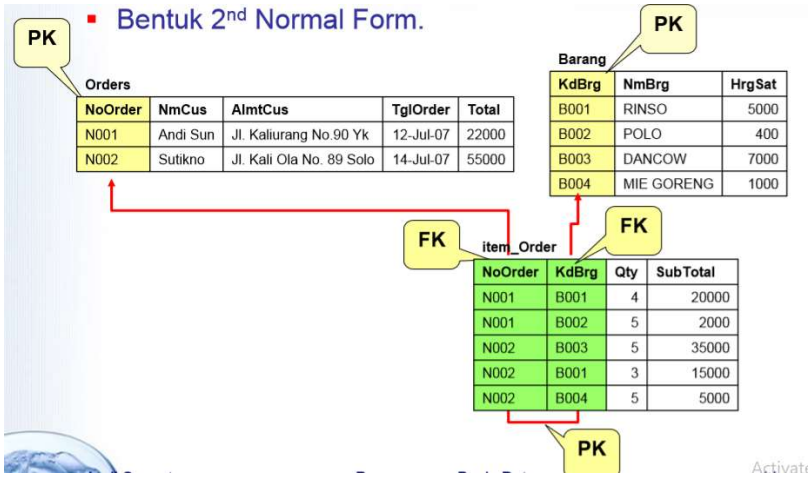


b. Normal kedua (2nd normal form)

- Syarat:
  - Sudah memenuhi 1NF
  - Atribut non-key secara fungsi tergantung penuh pada primary key
  - Tidak ada partial dependencies: tidak ada atribut yang tergantung pada sebagian dari primary key (untuk kasus composite primary key)

- Langkah:
  - Jika ada atribut yang tergantung pada sebagian primary key, pecah menjadi table sendiri atau cari data yang terulang kemudian pecah menjadi table sendiri.
  - Kemudianbuatrelasidiantaraset data yang dipisahkan.

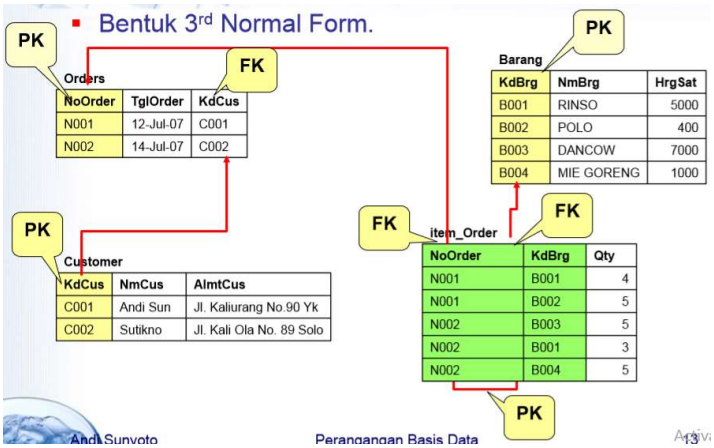
- Aturan :
  - Sudah memenuhi bentuk normal pertama
  - Tidak ada ketergantungan parsial (dimana seluruh field hanya tergantung pada sebagian field kunci)
- Bentuk 2nd normal form



c. Normal ketiga (3rd normal form)

- Syarat:
  - Memenuhi 2NF
  - Tidak ada atribut yang tergantung secara transitif pada non-key lainnya
- Langkah:
  - Hapus atau pisahkan menjadi table sendiri atribut yang tergantung pada kolom non-key (biasanya pada atribut turunan).
  - Pastikan semua atribut non-key tergantung pada primary key.

- Aturan :
  - Sudah berada dalam bentuk normal kedua
  - Tidak ada ketergantungan transitif (dimana field bukan kunci tergantung pada field bukan kunci lainnya)
- Bentuk 3rd normal form

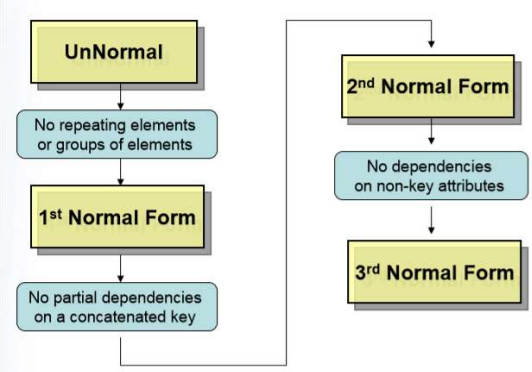


d. Normal Boyce-Codd (Boyce Codd Norm Form)

- Sebuah table dikatakan memenuhi BCNF jika untuk semua dengan notasi X ->Y, maka X harus merupakan super key pada table tersebut.
- Jika belum demikian maka table tersebut harus di dekomposisikan ulang berdasar KF yang ada, sedemikian hingga X menjadi super key dari table hasil dekomposisi.
- Sebuah table dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi X -> A, dimana A mewakili semua atribut tunggal didalam table yang tidak ada di dalam X, maka:
  - X haruslah super key pada table tersebut
  - Atau A merupakan bagian dari primary key pada table tersebut.

- Aturan :
  - Sudah berada dalam bentuk normal ketiga
  - Semua determinannya merupakan candidate key

e. Proses Normalisasi



- Catatan :
  - Bentuk normal seharusnya berada dalam bentuk normal tertinggi dan bergerak dari bentuk normal pertama dan seterusnya untuk setiap kali membatasi hanya satu jenis redundansi
  - Keseluruhan ada 5 bentuk normal. Tiga bentuk normal pertama menekankan redundansi muncul dari *Functional Dependencies*sedangkan bentuk keempat dan kelima menekankan redundansi yang muncul dari kasus *Multi Valued Dependencies*

7.10. Latihan dan Evaluasi

Tugas Kelompok

1. Terdapat relasi yang belum dinormalisasi sebagai berikut.

NO_TRAN	BARANG	JUMLAH	HARGA_UNIT	PELANGGAN	TELPON
A0001	Gula Pasir Minyak	1 2	300.000 280.000	Adil	589125
A0002	Beras	3	225.000	Formula	123456
A0003	Gula Pasir	1	310.000	Adil	589125
A0004	Kacang	1	160.000	Setia	144567

   a) Bagaimana bentuk normal pertamanya?
   b) Bagaimana komposisi relasi-relasi setelah bentuk normal kedua diterapkan?

2. Sebuah relasi beserta isinya adalah sebagai berikut :
   MAHASISWA(NIM, NAMA, DOSEN_WALI)

   NIM = nomor induk mahasiswa

   NAMA = nama mahasiswa

   DOSEN_WALI = nama dosen wali (pembimbing akademis)

   Seorang mahasiswa hanya memiliki seorang dosen wali.

   a) Apakah DOSEN_WALI mempunyai dependensi fungsional terhadap NIM?
   b) Apakah NIM mempunyai dependensi fungsional terhadap DOSEN_WALI?

3. Relasi PROYEK berupa :
   PROYEK (KODE_PROYEK, NAMA_PEGAWAI, BAGIAN)

   Sampel data :

KODE_PROYEK	NAMA_PEGAWAI	BAGIAN
A203	Amir	EDP
A203	Udin	HRD
A203	Wawan	HRD
A204	Amir	EDP
A204	Fika	Akunting
A205	Amir	EDP
A205	Wawan	HRD
A205	Fika	Akunting

   Menurut Anda, pertanyaan-pertanyaan manakah yang benar?

   a) KODE_PROYEK → NAMA_PEGAWAI
   b) KODE_PROYEK → BAGIAN

c) (KODE_PROYEK, NAMA_PEGAWAI) → BAGIAN
d) NAMA_PEGAWAI → BAGIAN
e) BAGIAN → KODE_PROYEK
f) BAGIAN → NAMA_PEGAWAI
g) BAGIAN → (KODE_PROYEK, NAMA_PEGAWAI)

4. Berdasarkan relasi PROYEK pada soal di atas :
   a) Apakah PROYEK memenuhi bentuk normal pertama? Jelaskan!
   b) Apakah PROYEK memenuhi bentuk normal kedua? Jelaskan!
   c) Apakah PROYEK memenuhi bentuk normal ketiga? Jelaskan!
   d) Anomali apa saja yang mungkin terjadi pada relasi tersebut (dengan memberikan contoh)?
   e) Apa kunci primer bagi relasi PROYEK?
   f) Apakah pada relasi tersebut terdapat dependensi transitif ? jika ada, sebutkan!
   g) Bagaimana bentuk relasinya agar persoalan-persoalan anomali dapat dihilangkan?

BAB 8. SQL

8.1. Tujuan Instruksional

G. Tujuan Instruksional Umum

Mahasiswa memahami fungsi SQL

H. Tujuan Instruksional Khusus

Mahasiswa dapat melakukan fungsi SQL

8.2. Pengantar SQL

DBMS umumnya menyediakan program khusus (utilitas/*Utility*) yang dapat digunakan secara interaktif untuk melakukan berbagai operasi terhadap basis data, seperti pembuatan table, penghapusan table, penambahan data, pengubahan data, pencarian data, penghapusan data dan lain-lain. Namun di samping adanya program khusus itu, DBMS juga umumnya menyediakan sekumpulan perintah (dalam bentuk *commandline*, yakni perintah yang dituliskan pemakai) untuk maksud yang sama. Perintah-perintah ini dapat diberikan (dan dikerjakan oleh DBMS) melalui utilitas lain yang juga disediakan DBMS atau melalui program/aplikasi yang dibuat sendiri oleh pemakai. Kumpulan perintah ini dapat disebut sebagai Bahasa Basis Data (*Database Language*).

Ada banyak sekali bahasa basis data yang pernah dibuat untuk masing-masing DBMS. Namun akhirnya yang menjadi standar adalah SQL. SQL merupakan kependekan dari *Structured Query Languange* (Bahasa Query yang Terstruktur). Istilah *Query Languange* memang tidak tepat sama dengan istilah Bahasa Basis Data (*Database Languange*). Bahasa Basis Data terdiri atas *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML).

8.3. Struktur Dasar

Sebuah ekspresi SQL dasar sebenarnya hanya terdiri atas 3 klausa, yaitu : **select, from** dan **where :**

- Klausa **select** digunakan untuk menetapkan daftar atribut (*field*) yang diinginkan sebagai hasil *query.*
- Klausa **from** digunakan untuk menetapkan table (atau gabungan table) yang akan ditelusuri selama *query* data dilakukan.

- Klausa **where**, yang sifatnya opsional, digunakan sebagai predikat (criteria) yang harus dipenuhi dalam memperoleh hasil *query*.

Sintaks (cara penulisan) dari ekspresi SQL dasar dengan 3 klausa tersebut adalah :

Select *A1 [, A2, … , An]*
From *t1 [, t2, … , tm]*
[**where** P]

Dimana :

- A1, A2, …, An merupakan daftar atribut.
- t1, t2, …, tm merupakan daftar table.
- P merupakan predikat *query*.
- [] merupakan tanda optional (boleh digunakan, boleh tidak digunakan).

8.4. Struktur Select

Bentuk lain struktur select :

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
* | expression [ AS output_name ] [, ...]
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY expression [, ...] ]
[ HAVING condition [, ...] ]
[ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]
[ ORDER BY expression [ ASC | DESC | USING operator ] [, ...] ]
[ FOR UPDATE [ OF tablename [, ...] ] ]
[ LIMIT { count | ALL } ]
[ OFFSET start ]
```

8.5. Klausa Select

- Untuk melihat semua kolom dari suatu tabel :
  SELECT * FROM nasabah;
- Untuk melihat kolom-kolom tertentu:
  SELECT nama_nasabah FROM nasabah;

  SELECT id_nasabah, nama_nasabah FROM nasabah;

- Secara umum:
  SELECT <nama kolom,...> FROM <nama tabel>;

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

```
SELECT nama_nasabah AS "Nama Nasabah"
FROM nasabah;

SELECT nama_nasabah AS "Nasabah", alamat_nasabah AS "Alamat
Nasabah"
FROM nasabah;
```

8.6. Klausa From

Klausa ini digunakan untuk menetapkan tabel yang kita jadikan sebagai sumber (lokasi) pencarian data. Sebagaimana kita ketahui, basis data terdiri atas sejumlah tabel yang saling berhubungan. Karena itu, akan seringkali ada kebutuhan untuk melakukan *query* tidak hanya dari satu tabel, tapi dengan merelasikan beberapa tabel sekaligus. Upaya ini dilakukan karena atribut-atribut yang kita harapkan sebagai hasil *query* tidak hanya tersedia di sebuah tabel, tapi berada di sejumlah tabel.

Contoh-contoh sebelumnya hanya menunjukkan *query* terhadap sebuah tabel. Sebagai hasil implementasi, tabel Kuliah terdiri atas atribtu-atribut (*field*) *kode_kul, nama_kul, sks, semester* dan *kode_dos*. Jika kita ingin menampilkan data kuliah beserta dosen-dosen yang mengajarkannya, maka kita tidak hanya dapat melakukan *query* dari tabel Kuliah saja, karena data seperti nama dosen tidak tersimpan di tabel ini, tetapi berada di tabel Dosen. Untuk memenuhi keinginan itu, kita dapat menggunakan ekspresi SQL berikut :

```
select *
    from kuliah, dosen
    where kuliah.kode_dos = dosen.kode_dos
```

Perlu diperhatikan, melakukan *query* terhadap 2 tabel atau lebih tidak bisa dilakukan sembarangan. Tabel-tabel yang menjadi sumber *query* harus memiliki keterhuungan (relasi). Pada perintah di atas, keterhubungan itu diwakili oleh kesamaan nilai pada atribut *kode_dos* dan kita tahu bahwa atribut ini dimiliki oleh kedua tabel. Ekspresi ‘kuliah.kode_dos’ menunjukkan nilai *kode_dos* yang berasal dari tabel Kuliah dan ‘dosen.kode_dos’ menunjukkan nilai *kode_dos* yang berasal dari tabel Dosen.

Kita dapat menggunakan nama alias untuk tabel-tabel pada klausa *from* untuk menyederhanakan penulisan. Ekspresi *query* di atas, dengan hasil yang sama, dapat pula dinyatakan dengan ekspresi sebagai berikut :

```
select *
    from kuliah k, dosen d
    where k.kode_dos = d.kode_dos
```

Terhadap sumber data (tabel *query*) yang banyak, tanda * (asterisk) pada klausa select akan mengacu pada semua atribut yang ada di semua tabel yang disebutkan pada klausa **from**. Jika kita hanya ingin menampilkan atribut-atribut tertentu saja, maka nama tabel atau aliasnya dapat kita gunakan untuk memperjelas asal atribut yang kita tampilkan tersebut, misalnya :

```
select k.kode_kul, k.nama_kul, d.nama_dos
    from kuliah k, dosen d
    where k.kode_dos = d.kode_dos
```

8.7. Klausa Where

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

```
SELECT nama_nasabah FROM nasabah
WHERE nama_nasabah = 'Ali Topan';


SELECT nama_nasabah, alamat_nasabah
FROM nasabah
WHERE id_nasabah = 2;
```

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

```
    SELECT * FROM nasabah
WHERE nama_nasabah = 'Rina Marsudi'
AND alamat_nasabah = 'Jl. Kusumanegara 30';


    SELECT * FROM nasabah
WHERE nama_nasabah = 'Ali Topan'
OR id_nasabah = 2;
```

- Pencarian NULL

1. Gunakan IS NULL untuk mencari NULL:

```
SELECT * FROM rekening
WHERE kode_cabang IS NULL;
```

2. Gunakan IS NOT NULL untuk mencari yang tidak NULL:

```
SELECT * FROM rekening
WHERE kode_cabang IS NOT NULL;
```

- Pencarian String

1) Gunakan LIKE untuk mencari string tertentu:

```
SELECT * FROM nasabah
WHERE nama_nasabah LIKE 'Ali Topan';
```

2) Bisa menggunakan %, berarti cocok untuk semua *substring*

```
SELECT * FROM nasabah
WHERE alamat_nasabah LIKE '%negara%';
```

3) Bisa menggunakan _ untuk 1 huruf. Tanda '_' berarti cocok untuk semua karakter pada posisi yang sesuai

```
SELECT * FROM nasabah
WHERE nama_nasabah LIKE 'Ali T_p_n';
```

1. Menggunakan tambahan klausa **between** dengan criteria yang berbentuk *range* nilai tertentu dengan, misalnya untuk menampilkan *record-record* kuliah yang diselenggarakan antara semester 3 hingga semester 5 :

```
select *
from kuliah
where semester between 3 and  5
```

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

```
SELECT * FROM nasabah
WHERE nama_nasabah ILIKE '% marsudi';
```

8.8. ORDER BY

Digunakan untuk mengurutkan hasil SELECT.

1) Untuk mengurutkan dari kecil ke besar:

```
SELECT * FROM nasabah
ORDER BY nama_nasabah;
```

2) Untuk mengurutkan dari besar ke kecil:

```
SELECT * FROM nasabah
ORDER BY nama_nasabah DESC;
```

Jika ada WHERE, maka ORDER BY ditaruh sesudah WHERE.

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

```
SELECT * FROM nasabah_has_rekening
ORDER BY no_rekening, id_nasabah;
```

4) Menentukan DESC untuk kolom(-kolom) tertentu, misalnya:

```
SELECT * FROM nasabah_has_rekening
ORDER BY no_rekening, id_nasabah DESC;
```

```
SELECT * FROM nasabah_has_rekening
ORDER BY no_rekening DESC, id_nasabah;
```

8.9. LIMIT & OFFSET

Digunakan untuk membatasi jumlah baris yang ditampilkan dalam SELECT.

a. Hanya menampilkan 3 baris pertama:

```
SELECT * FROM nasabah
ORDER BY id_nasabah
LIMIT 3;
```

b. Menampilkan 2 baris setelah melewati 2 baris pertama:

```
SELECT * FROM nasabah
ORDER BY id_nasabah
LIMIT 2 OFFSET 2;
```

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

8.10. Fungsi Agregasi

Di samping enampilkan nilai-nilai atribut yang ada di dalam tabel, sering pula ada kebutuhan untuk menampilkan data-data agregasi, seperti banyaknya *record*, total nilai atribut, rata-rata nilai atribut, nilai atribut terbesar ataupun nilai atribut terkecil. Data agregasi semacam itu dapat diperoleh dengan menggunakan fungsi-fungsi berikut ini :

- **count** untuk mendapatkan nilai banyaknya *record* hasil *query,*
- **sum** untuk mendapatkan nilai total suatu atribut numeric hasil *query*,
- **avg** untuk mendapatkan nilai rat-rata suatu atribut numeric hasil *query*,
- **max** untuk mendapatkan nilai terbesar dari suatu atribut hasil *query*,
- **min** untuk mendapatkan nilai terkecil dari suatu atribut hasil *query*.

Berikut adalah contoh-contoh penggunaan fungsi agregasi tersebut :

1. Menampilkan banyaknya *record* mahasiswa :

   select count (*)

   from mahasiswa

2. Menampilkan banyaknya mahasiswa angkatan 98 :

   select count (*)

   from mahasiswa

   where nim **like** '98%'

3. Menampilkan total sks untuk kuliah di semester 2 :

   select sum(sks)

   from kuliah

   where semester = 2

4. Menampilkan rata-rata sks untuk semua mata kuliah :

   select avg(sks)

   from kuliah

5. Menampilkan indeks nilai terbesar yang diperoleh mahasiswa untuk mata kliah dengan kode kuliah 'IF-110' :

   select max(indeks_nilai)

   from nilai

   where kode_kul = 'IF-110'

6. Menampilkan tanggal lahir paling tua yang ada di tabel mahasiswa :

   select min(tgl_lahir)

   from mahasiswa

8.11. Manipulasi Data

1. Penambahan Record

   Sintaks SQL untuk penambahan *record* baru ke sebuah tabel adalah :

   Insert into t [(A1, A2, …, An)]

   Values (v1, v2, …, vn)

   Di mana :

   - t adalah nama tabel yang akan mengalami penambahan *record*
   - A1, A2, …, An adalah nama-nama atribut yang akan diisi nilai
   - v1, v2, …, vn adalah nilai-nilai yang akan mengisi atribut-atribut tersebut

   Berikut ini adalah contoh perintah SQL untuk melakukan penambahan *record* baru ke tabel Mahasiswa :

   Insert into Mahasiswa (nim, nama_mhs, alamat_mhs, tgl_lahir)

   Values (‘980011’, ‘Siti Zubaedah’, ‘Jl. Guntur Kulon 12, Bogor’, ‘02/03/1973’)

   Atau :

   Insert into Mahasiswa

   Values (‘980011’, ‘Siti Zubaedah’, ‘Jl. Guntur Kulon 12, Bogor’, ‘02/03/1973’)

   Jika kita melakukan penambahan *record* baru dengan perintah berikut :

   Insert into Mahasiswa (nim, nama_mhs, alamat_mhs)

   Values (‘980011’, ‘Siti Zubaedah’, ‘Jl. Guntur Kulon 12, Bogor’)

   Maka atribut *tgl_lahir* yang tidak disebutkan dalam perintah **insert** tersebut akan diisi dengan nilai *null*.

2. Pengubahan Record

   Sintaks SQL untuk pengubahan nilai atribut pada suatu *record* dari sebuah tabel adalah :

   update t

   set *assignment*

   [**where** P]

   Di mana :

   - t adalah nama tabel yang akan mengalami pengubahan *record*
   - *assignment* adalah ekspresi pemberian nilai baru untuk suatu atribut yang akan kita ubah

- P merupakan predikat atau criteria untuk pemilihan *record* yang akan dikenai perubahan, jika klausa **where** ini tidak digunakan, maka perubahan akan dilakukan pada semua *record* di dalam tabel t

Berikut ini adalah perintah SQL untuk mengubah nilai atribut *sks* untuk mata kuliah tertentu :

update kuliah

set sks = 4

where kode_kul = 'IF-310'

3. Penghapusan Record

Sintaks SQL untuk penghapusan *record* dari sebuah tabel adalah :

delete from t

[**where** P]

Di mana :

- t adalah nama tabel yang akan mengalami penghapusan *record*
- P merupakan predikat atau criteria untuk menentukan *record* mana saja yang akan dikenai penghapusan, jika klausa **where** ini tidak digunakan, maka penghapusan akan dilakukan pada semua *record* di dalam tabel t

Berikut ini adalah perintah SQL untuk menghapus *record* kuliah tertentu :

delete from kuliah

where kode_kul = 'IF-310'

Berikut ini adalah perintah SQL untuk menghapus beberapa *record* di tabel kuliah :

d**elete from** kuliah

where kode_kul **like** 'MA%'

sedang perintah SQL berikut ini adalah untuk menghapus semua *record* dari tabel Kuliah (sehingga tabel Kuliah menjadi kosong) :

d**elete from** kuliah

4. Kontrol Transaksi

Operasi-operasi manipulasi data merupakan bagian dari suatu transaksi. Transaksi sendiri bisa terdiri atas satu atau beberapa operasi menipulasi data. Perintah-perintah menipulasi data tidak akan benar-benar disimpan (ditulis di dalam *disk*) jika kita belum memberikan perintah control transaksi. Perintah control transaksi itu adalah :

commit [work]

dan

rollback [work]

Jika perintah **commit** digunakan, maka semua operasi manipulasi basis data akan ke dalam *disk* dan transaksi dinyatakan selesei. Sedang jika perintah **rollback** yang kita gunakan, maka semua operasi manipulasi basis data yang belum di-**commit**, akan dibatalkan (tidak jadi disimpan ke dalam *disk*). Klausa **work** sifatnya opsional dan tidak berpengaruh terhadap pengertian kedua perintah tersebut.

Data Definition Language

Data Definition Language (DDL) berkaitan dengan perintah-perintah untuk pendefinisian objek-objek basis data. Salah satu objek terpenting adalah tabel. Berikut ini adalah Sintaks SQL untuk melakukan pembuatan tabel baru di dalam basis data :

create table t (A1 D1, A2 D2, … , An Dn)

di mana :

- t adalah nama tabel yang akan dibuat
- A1, A2, …, An adalah nama-nama atribut yang akan terdapat di dalam tabel t
- D1, D2, …, Dn adalah domain nilai masing-masing atribut tersebut yang ditentukan berdasarkan tipe datanya

Berikut ini adalah contoh perintah SQL untuk membuat tabel Mahasiswa :

create table mahasiswa
(nim char(6),
nama_mhs varchar(30),
alamat_mhs varchar(60),
tgl_lahir date)

tabel yang telah dibuat juga dapat dibatalkan keberadaannya (penghapusan tabel) dengan menggunakan perintah SQL dengan sintaks berikut ini :

drop table t

contoh penghapusan terhadap tabel Mahasiswa :

drop table mahasiswa

Jika terhadap tabel yang kita buat, ingin kita sertakan pula adanya Indeks Primer berdasarkan atribut tertentu di dalam tabel, maka klausa **primary key** dapat kita gunakan. Berikut ini adalah contoh ekspresi SQL untuk pembuatan tabel Mahasiswa sekaligus dengan pendefinisian Indeks Primer berdasarkan *nim* :

```
create table mahasiswa
        (nim char(6),
        nama_mhs varchar(30),
        alamat_mhs varchar(60),
        tgl_lahir date,
        primary key (nim))
```

Jika jumlah atribut yang membentuk Indeks Primer ada lebih dari satu, seperti yang ada di tabel Nilai, contoh ekspresi SQL-nya adalah :

```
create table nilai
        (nim char(6),
        Kode_kul char(6),
        Indeks_kul char(1),
        tgl_lahir date,
        primary key (nim, kode_kul))
```

Struktur sebuah tabel juga dapat kita ubah, tanpa harus menghapus dan kemudian membangunnya kembali denagn definisi struktur yang baru. Perintah pengubahan struktur selain lebih praktis, juga tidak mengakibatkan hilangnya data yang sudah ada di dalam tabel (jika memang sudah terisi data). Perubahan struktur ini dapat berupa penambahan atribut atau pengurangan/penghapusan atribut tertentu. Sintaks SQL untuk perubahan struktur tabel yang berbentuk penambahan atribut baru ke tabel t adalah :

```
alter table t add A D
```

di mana t mewakili tabel, A mewakili nama atribut dan D mewakil tipe data untuk atribut A tersebut.

Sedang untuk penghapusan atribut dari tabel t, sintaks SQL-nya adalah :

```
alter table t drop A
```

Berikut ini adalah contoh ekspresi SQL untuk penambahan atribut baru bernama *ip* di tabel Mahasiswa :

alter table mahasiswa **add** ip numeric (5, 2)

Jika atribut *ip* ingin dihapus dari tabel Mahasiswa, ekspresi SQL-nya :

alter table mahasiswa **drop** ip

8.15. Latihan dan Evaluasi

Tugas Pribadi

Tabel : tblpengarang				
Kd_peng	**Nama**	**Alamat**	**Kota**	**Kelamin**
1	Asadi	Jl. Beo 34	Yogya	P
2	Rian H.	Jl. Solo 123	Yogya	P
3	Suadi Marwan	Jl. Semangka II/1	Bandung	P
4	Siti Halimah	Jl. Sukaria 5	Solo	W
5	Amir Hamzah	Jl. Gajah Mada 18A	Kudus	P
6	Suparman	Jl. Setia 1	Jakarta	P
7	Jaja M	Jl. Hangtuah 3	Bandung	P
8	Saman	Jl. Gedong Kuning	Yogya	P
9	Anwar Junaidi	Jl. Tidar 6A	Magelang	P
10	Fatmawati	Jl. Renjana 4	Bogor	W

Tabel : tblbuku		
Kd_buku	**Judul**	**Kd_peng**
1	Pemrograman C++	1
2	Pengantar Basis Data	1
3	Panduan Microsoft Office	2
4	Pemrograman Visual dBASE	1
5	System Pakar	4
6	Pemrograman C++	3
7	Visual C++	6
8	QBASIC	5
9	Pemrograman Pascal	8

10	Pemrograman Delphi	8

1. Berdasarkan tabel tblpengarang, tuliskan pernyataan SQL untuk menampilkan :
   a) Nama pengarang yang kode pengarangnya adalah 5.
   b) Semua nama pengarang yang tinggal di Yogya.
   c) Semua nama pengarang yang tidak tinggal di Yogya.
   d) Semua nama pengarang berkelamin wanita.
   e) Nama, alamat, dan kota pengarang yang berkelamin pria.
2. Berdasarkan tabel tblbuku, tuliskan pernyataan SQL untuk menampilkan:
   a) Semua judul buku yang ditulis oleh pengarang berkode 1.
   b) Semua data (semua kolom dan semua baris).
3. Tuliskan pernyataan untuk menciptakan tabel bernama tblpegawai, dengan komposisi sebagai berikut :
   NIM bertipe karakter, panjang 6

   Nama bertipe karakter, panjang 25

   Gaji bertipe numeric 8 digit
4. Tuliskan pernyataan untuk membuat indeks dengan nama idx_nip yang dikenakan terhadap kolom NIP pada tabel tblpegawai.
5. Bagaimana cara menghapus indeks idx_nip yang baru saja Anda ciptakan?
6. Bagaimana perintahnya agar panjang Nama (pada tabel tblpegawai) tidak lagi erupa 25 karakter, melainkan menjadi 35 karakter?
7. Bagaimana caranya agar semua kota Yogya yang ada pada tabel tblpengarang diubah menjadi Yogyakarta?
8. Apa perintah untuk menghapus data pegawai pada tabel tblpegawai yang NIP-nya adalah 123456?
   Diinginkan untuk menghapus tabel tblpegawai. Apa perintahnya?

BAB 9. SUB QUERY

9.1. Tujuan Instruksional

C. Tujuan Instruksional Umum

Mahasiswa memahami fungsi sub query

D. Tujuan Instruksional Khusus

Mahasiswa dapat menerapkan fungsi sub query

9.2. Sintak SubQuery

Suatu *subquery* adalah suatu pernyataan SELECT yang dilekatkan di dalam suatu klausa pada pernyataan SELECT lain. *Subquery* bisa sangat bermanfaat ketika diperlukan untuk memilih baris baris dari suatu table dengan suatu kondisi yang tergantung pada data di dalam tabel itu sendiri.

Kita dapat menempatkan *subquery* di dalam sejumlah klausa klausa SQL, termasuk berikut :

1. Klausa WHERE
2. Klausa HAVING
3. Klausa FROM

Di dalam Syntax : *Operator* termasuk suatu kondisi pembanding seperti >, =, atau IN

Kondisi kondisi pembanding dibagi dalam dua kelas : *singlerow Operator* (>,=,>=,<,<>,<=) dan m*ultiplerow operator* (IN, ANY, ALL).

Subquery lebih dikenal sebagai suatu SELECT bersarang (*nested*), *sub*SELECT, atau pernyataan *inner* SELECT. Secara umum s*ubquery* dieksekusi pertama kali, dan hasilnya digunakan untuk melengkapi kondisi query pada query utama (atau *outer*).

```
SELECT select_list
FROM table
            WHERE expr_operator
            (SELECT  select_list
              FROM table);
```

Subquery (*inner query*) dieksekusi sekali sebelum query utama (*outer query*). Hasil dari *subquery* digunakan oleh query utama.

Contoh Kasus :

Siapakah yang memiliki penghasilan lebih dari Abel?

Jawabannya :

Query Utama

Pegawai mana yang memiliki penghasilan lebih dari Abel?

Sub Query

Berapakah penghasilan Abel?

Tipe Tipe dari *Subquery*

1. *Singlerow subquery*

Query yang mengembalikan hanya satu baris dari pernyataan *inner* SELECT (SELECT terdalam).

2. *Multiplerow* s*ubquery*

Query yang mengembalikan lebih dari satu baris dari pernyataan *inner* SELECT.

9.3. Menggunakan suatu *Subquery* untuk Memecahkan suatu Persoalan

Misalkan ingin menulis suatu query untuk mencari tahu penghasilan siapa yang lebih besar daripada penghasilan Abel.

Untuk memecahkan masalah ini, diperlukan *dua* query: satu query untuk mencari berapa banyak penghasilan Abel, dan query kedua untuk mencari penghasilan siapa yang lebih besar dari jumlah itu.

Kita dapat memecahkan persoalan ini dengan menggabungkan dua query, menempatkan satu query *di dalam* query lain.

Inner query (atau *subquery*) mengembalikan suatu nilai yang digunakan *outer query* (atau query utama). Penggunaan suatu *subquery* sama dengan penggunaan dua query berturut turut dan menggunakan hasil dari query pertama sebagai nilai pencari dalam query yang kedua.

```
SELECT last-name
FROM  employees
WHERE salary >
            (SELECT salary
            FROM employees
            WHERE last_name = ‘Abel’);
```

9.4. Pedoman Pedoman untuk Menggunakan Subquery

1. *Subquery* diapit tanda kurung.

2. Tempatkan *subquery* di sebelah kanan dari kondisi pembanding.

3. Klausa ORDER BY dalam *subquery* tidak diperlukan kecuali dilakukan pemeringkatan (*TopNanalysis*).
4. Gunakan *singlerow operator* pada *singlerow subquery*, dan gunakan *multiplerow operator* pada *multiplerow subquery*.

4. Single Row Subqueries

Suatu *singlerow subquery* adalah mengembalikan satu baris dari pernyataan *inner* SELECT. Tipe
dari *subquery* ini menggunakan suatu *singlerow operator*.
Beberapa operator pembanding single row :

operator
=
>
>=
<
<=
<>

Tampilkan pegawai pegawai yang job Idnya sama dengan pegawai 141 :

```
SELECT last_name, job_id
FROM employees
WHERE job_id = (SELECT job_id
                FROM employees
                WHERE employee_id = 141 );
```

5. Multiple Row Subqueries

Subquery subquery yang mengembalikan lebih dari satu baris disebut *multiplerow subqueries*. Kita menggunakan suatu *multiplerow operator*, disamping suatu *singlerowoperator*, pada suatu *miltiplerowsubquery*. *Multiplerowoperator* memperkirakan satu atau lebih nilai-nilai :

Contoh

Cari pegawaipegawai yang mendapat penghasilan yang sama dengan penghasilan minimum untuk setiap departemen. *Inner query* dieksekusi pertama kali, menghasilkan suatu hasil query.

Blok query utama kemudian memproses dan menggunakan nilai nilai yang dikembalikan oleh *inner query* untuk melengkapi kondisi pencariannya.

```
SELECT last_name, salary, department_id
FROM employess
WHERE salary IN (2500, 4200, 6000, 7000, 800, 8600, 17000);
```

9.5. Operator ANY

(dan sinonimnya, operator SOME) membandingkan suatu nilai pada *setiap* nilai yang dikembalikan oleh suatu *subquery*. Contoh di bawah menampilkan para pegawai yang bukan IT programmers dan penghasilan siapa yang kurang dari beberapa IT programmer. Penghasilan maksimum yang didapat seorang programmer adalah $ 9,000.

<ANY maksudnya kurang dari maksimum. >ANY maksudnya lebih dari minimum. =ANY adalah sama dengan IN.

```
SELECT employee_id, last_name, job_id, salary
FROM employee
WHERE salary < ANY (SELECT salary
                    FROM employee
                    WHERE job_id = 'IT PROG') AND job_id <>
'IT_PROG'
```

9.6. Operator ALL

Membandingkan suatu nilai *untuk setiap* nilai yang dikembalikan oleh suatu *subquery*. Contoh menampilkan para pegawai yang penghasilannya kurang dari penghasilan dari semua pegawai dengan suatu job ID IT_PROG dan siapa yang bukan IT_PROG.

>ALL maskudnya lebih dari maksimum, dan <ALL maksudnya kurang dari minimum.

Operator NOT dapat digunakan dengan operator operator IN, ANY, dan ALL.

```
SELECT employee_id, last_name, job_id, salary
FROM employee
WHERE salary < ALL (SELECT salary
                    FROM employee
                    WHERE job_id = 'IT PROG')
AND job_id <> 'IT_PROG'
```

BAB 10. INTEGRITAS BASIS DATA

10.1. Tujuan Instruksional

E. Tujuan Instruksional Umum

Mahasiswa memahami konsep integritas

F. Tujuan Instruksional Khusus

Mahasiswa dapat menerapkan konsep integritas

10.2. Tentang Integritas

- Sebagai sarana untuk meyakinkan bahwa nilai-nilai data dalam sistem basis data selalu benar, konsisten, selalu tersedia.
- Integrity constraints (batasan integritas) menjaga dari kejadian yg merusak database, dg menjamin perubahan database tidak menghasilkan kehilangan konsistensi data.
- Domain constraints adalah bentuk paling umum dari integrity constraint.
- Dapat dilakukan dengan cara :
  - Pastikan bahwa nilai-nilai data adalah benar sejak dimasukkan pertama kali
  - Membuat program untuk mengecek keabsahan data pada saat dimasukkan ke computer
    - Penolakan / pembatalan aksi (cancelation)
    - Pengisian nilai kosong pada field tertentu (nullify)
    - Penjalaran perubahan (cascade)

10.3. Integritas keunikan data

Dilakukan melalui :

a. Pendefinisian struktur tabel dengan membuat indeks primer yang bersifat unik
b. Pengkodean di dalam aplikasi pada saat pemasukan / penambahan data→lebih *user-friendly*
c. Kedua cara diterapkan bersama-sama

10.4. Integritas domain data

Dilakukan melalui :

a. Penetapan tipe data pada setiap field di dalam tabel
b. Pengisian *validation rule* dari DBMS **Integritas Referensial/Referential Integrity (realasi antar table)**

a. Memastikan sebuah nilai yg muncul pada suatu tabel dari atribut yg diberikan juga muncul pada atibut tabel lain..
b. Harus selalu dijaga karena kesalahan referensial dapat menimbulkan kesalahan baru dalam basis data
c. Dilakukan pengecekan pada proses penambahan, pengubahan, dan penghapusan data.

- Contoh: jika "Perryridge" adalah nama cabang pada satu baris di tabel *account*, maka ada sebuah baris di tabel *branch* yg berisi "Perryridge".
- Contoh Referential Integrity pada SQL

create table *customer*
(customer-name char(20)**,**
customer-street char(30),
customer-city char(30),
primary key (*customer-name))*
create table *branch*
(branch-name char(15)**,**
branch-city char(30),
assets integer,
primary key *(branch-name))*

- Contoh Referential Integrity pada SQL

create table *account*
(account-number char(10)**,**
branch-name char(15),
balance integer,
primary key (*account-number),*
foreign key (*branch-name)* **references** *branch)*
create table *depositor*
(customer-name char(20)**,**
account-number char(10)**,**
primary key *(customer-name, account-number),*
foreign key *(account-number)* **references** *account,*
foreign key *(customer-name)* **references** *customer)*

d. Aksi Cascading pada SQL

create table *account*

. . .

foreign key*(branch-name)* **references** *branch*

on delete cascade

on update cascade

. . . **)**

- **on delete cascade** jika ada langgaran batasan referensial berupa penghapusan pd tabel *branch*,maka otomatis ada penghapusan baris pd tabel *account* yg merefer tabel *branch*.
- Alternatif cascading:
  - **on delete set null**
  - **on delete set default**
- Nilai Null foreign key membuat komplikasi pada referential integrity SQL, dan cara mencegah terbaik dg **not null**

e. Assersi

- Sebuah *assersi* adalah sebuah ekspresi predikat dari suatu kondisi yang harus dipenuhi database.
- Assersi pada SQL bentuknya **create assertion** <nama-assersi> **check** <predikat>
- Sewaktu assersi dibuat, sistem menguji nya untuk validitas, dan menguji lagi untuk setiap perubahan yg mungkin melanggar assersi
- Contoh Assersi : Jumlah semua pinjaman paling banyak sama dengan jumlah tabungan pada setiap cabang

create assertion *sum-constraint* **check**
(not exists (select * from *branch*
where (select sum*(amount)* **from** *loan*
where *loan.branch-name =*
branch.branch-name)
>= **(select sum***(amount)* **from** *account*
where *loan.branch-name =*
branch.branch-name)))

10.5. Integritas aturan nyata

i. Sifatnya sangat kasuistis, tidak berlaku umum. Pada kasus yang berbeda, aturannya bisa berbeda pula.

ii. Untuk mengakomodasi adanya *business* role ini, dengan menyiapkan table khusus yang menampung nilai-nilai konstanta yang dibutuhkan aplikasi pada saat dijalankan yang mudah diubah tanpa mengakibatkan perubahan aplikasi maupun struktur basis data.

BAB 11. SEKURITAS BASIS DATA

11.1. Tujuan Instruksional

G. Tujuan Instruksional Umum

Mahasiswa memahami konsep sekuritas basis data

H. Tujuan Instruksional Khusus

Mahasiswa dapat menerapkan sekuritas basis data

11.2. Tentang Sekuritas

proteksi dari berbagai upaya untuk mencuri atau mengubah data. Dapat dibedakan atas :

a. Level sistem Database

   Mekanisme Otentifikasi dan Otorisasi membolehkan user tertentu yang hanya mengakses data

b. Level Operating system

   Super-users dapat melakukan apapun yg diinginkan ke database! level securiti yang baik dari sistem operasi diperlukan

c. Level Network : menggunakan enkripsi untuk mencegah Eavesdropping (orang yg tidak berhak membaca pesan) dan Masquerading (orang yg mengirim pesan diduga pengguna yg berwenang)

d. Level Fisik

   Akses fisik ke komputer yg dapat merusak data oleh pengacau; securiti tradisional dengan kunci diperlukan

   Komputer harus dijaga dari banjir, api, dsb.

e. Level Manusia

   Pengguna harus discreening untuk memastikan pengguna yg berhak tidak memberi akses ke pengacau

   Pengguna dilatih pemilihan password dan keamanan

11.3. Otorisasi

a. Read authorization – boleh membaca, tetapi tidak bisa memodifikasi data.
b. Insert authorization – boleh menambah data baru, tetapi tidak bis memodifikasi data yg telah ada.

c. Update authorization – boleh memodifikasi, tetapi tidak bisa menghapus data.
d. Delete authorization – boleh menghapus data

11.3.1. Bentuk otorisasi untuk memodifikasi skema database :

a. Index authorization – boleh membuat dan menghapus indeks.
b. Resources authorization – boleh membuat tabel baru.
c. Alteration authorization – boleh menambah dan menghapus atribut tabel.
d. Drop authorization – boleh menghapus tabel.

11.4. Otorisasi dan View

- User dapat diberi otorisasi pada view (tampilan), tanpa ada otorisasi diberikan pada tabel yang digunakan dalam definisi view
- Kemampuan view menyembunyikan data dlm melayani baik untuk mempermudah penggunaan sistem atau meningkatkan keamanan dg memungkinkan user mengakses data yang dibutuhkan

a) Contoh View

Misalnya seorang pegawai bank perlu mengetahui nama pelanggan dari masing-masing cabang, tetapi tidak diizinkan untuk melihat informasi spesifik pinjamannya.
Pendekatan: dilarang mengakses langsung ke tabel *loan*, tetapi memberikan akses ke view *cus-loan*, yang hanya terdiri dari nama-nama pelanggan dan cabang di mana mereka memiliki pinjaman. View dari *cust-loan* dlm SQL sbb:

create view *cust-loan* as
select *branchname*, *customer-name*
from *borrower, loan*
where *borrower.loan-number = loan.loan-number*

b) Otorisasi pada Views

- Pembuatan view tidak memerlukan otorisasi sumber daya sewaktu tidak punya tabel yg telah dibuat.
- Pembuat view hanya mendapatkan hak istimewa yang tidak diberikan otorisasi tambahan di luar yang sudah ia punyai.
- Contoh. jika pembuat view *cust-loan* hanya punya otorisasi *membaca* pada tabel *borrower* dan *loan*, dia hanya punya otorisasi membaca pada *cust-loan.*

o **Pemberian Privileges (hak istimewa)**

- Otoritas user bisa diberikan ke user lain yg dapat dinyatakan dengan graf

- Simpul menyatakan user.
- Akar adalah database administrator.
- Pandang graph untuk otoritas update loan.
- Busur $U_i \rightarrow U_j$ menyatakan user U_i memberikan otoritas update loan ke U_j.

- **Spesifikasi Securiti pada SQL**
  - Statemen grant digunakan untuk memberikan otoritas :

    grant <daftar privilege >
    on <nama tabel atau nama view > to <daftar user >
  - <daftar user > adalah:
    - suatu user-id
    - *public*, semua user valid yg diberikan privilege
    - sebuah role (peran)
  - **Privileges (Hak Istimewa) pada SQL**
  - select: memungkinkan membaca tabel,atau mampu query dengan view
    - Contoh: grant users U_1, U_2, dan U_3 select otoritas pada tabel *branch* :

      grant select on *branch* to U_1, U_2, U_3
  - insert: kemampuan menambahkan baris
  - update: kemampuan mengupdate menggunakan statement SQL
  - delete: kemampuan menghapus baris.
  - references: kemampuan mendeklarasikan foreign key sewaktu membuat tabel.
  - usage: pada SQL-92; otoritas user menggunakan domain spesifik
  - all privileges: menggunakan bentuk pendek untuk memungkinkan semua privileges

- **Roles**
  - Role mengijinkan hak umum suatu kelas user dengan membuat "role"
  - Hak dapat diberikan atau dicabut dari peran, sepertihalnya user Roles dapat diberikan kepada pengguna, dan bahkan untuk peran lainnya SQL: 1999 mendukung roles

    create role *teller*
    create role *manager*
    grant select on *branch* to *teller*
    grant update (*balance*) on *account* to *teller*
    grant all privileges on *account* to *manager*

    grant *teller* to *manager*

grant *teller* to *alice, bob*
grant *manager* to *avi*

- **Pencabutan otoritas pada SQL**
  - Statemen revoke digunakan untuk mencabut otoritas.

    revoke<privilege list>

    on <nama tabel atau nama view > from <user list> [restrict|cascade]
  - Contoh: revoke select on *branch* from U_1, U_2, U_3 cascade
  - Pencabutan hak dari user dapat menyebabkan user lain juga kehilangan hak; disebut Cascading dari mencabut.
  - Kita dapat mencegah Cascading dengan membuat batasan (**restrict)**:

    revoke select on *branch* **from** U_1, U_2, U_3 **restrict**

    dengan **restrict**, perintah **revoke gagal** jika pencabutan diperlukan.
  - <privilege-list> mungkin semua dicabut
  - Jika <revokee-list> publik meliputi semua user maka akan kehilangan hak kecuali ia diberikan secara eksplisit.
  - Jika hak yang sama diberikan dua kali untuk user yang sama, user dapat mempertahankan hak setelah pencabutan.
  - Semua hak yang bergantung pada hak yang dicabut maka akan dicabut juga
- **Batasan Otoritas SQL**
  - SQL tidak mendukung otoritas pada level baris. Misal : kita tidak bisa membatasi mhs untuk hanya melihat (pd baris yg disimpan) nilai mereka
  - Dengan berkembangnya Web mengakses ke databases, pengaksesan database menjadi utama untuk server aplikasi. End users tidak mempunyai database user ids, namun mereka semu dipetakan ke database sama user id
  - Semua end-user aplikasi (spt aplikasi web) mungkin dipetakan ke database user tunggal
  - Tugas otoritas di atas pada program aplikasi tidak didukung dari SQL

DAFTAR PUSTAKA

[1] Adi Nugroho, Konsep Pengembangan Sistem Basis Data, Informatika, Bandung, 2004.

[2] Abdul Kadir, Konsep & Tuntunan Praktis Basis Data, Andi, Yogyakarta,1999.

[2] C.J. Date, Pengenalan Sistem Basis Data, Addison Wesley, 2000.

[2] Fathansyah, Basis Data, Informatika, Bandung, 2001.

[3] Bambang Hariyanto, Sistem Basis Data, Informatika


LABORATORIUM
TEKNIK INFORMATIKA
UNIVERSITAS AHMAD DAHLAN


UNIVERSITAS AHMAD DAHLAN