10.736/115/H/2003

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

TUGAS AKHIR

RANCANG BANGUN PERANGKAT LUNAK UNTUK NORMALISASI SKEMA RELASI SECARA OTOMATIS

RSIf
005.1
Iwa
r-1
1996

PERPUSTAKAAN
ITS
Tgl. Terima 9-7-2003
Terima Dari H
No. Agenda Prp. 117693

Disusun oleh :

KARUNIA P.M. IWAN

NRP. 2688.100.039

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
1996

RANCANG BANGUN PERANGKAT LUNAK UNTUK NORMALISASI SKEMA RELASI SECARA OTOMATIS

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar
Sarjana Teknik Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui,

Dosen Pembimbing

Dr. Ir. ARIEF DJUNAIDY, MSc.
NIP. 131 633 403

SURABAYA
SEPTEMBER, 1996

ABSTRAK

Normalisasi data adalah proses dekomposisi skema relasi — yaitu proses memecah skema relasi besar menjadi skema relasi yang lebih kecil — yang memenuhi bentuk normal. Proses ini terdiri dari beberapa tahap yaitu normalisasi bentuk pertama, normalisasi bentuk kedua, normalisasi bentuk ketiga dan normalisasi Boyce Codd. Tugas akhir ini bertujuan menyediakan perangkat otomatis untuk melakukan proses normalisasi. Perangkat otomatis ini adalah perangkat lunak yang menerapkan algoritma dekomposisi skema relasi bentuk normal ketiga dan bentuk normal Boyce Codd. Algoritma ini memanfaatkan ketergantungan fungsional yang merupakan salah satu batasan yang dimiliki oleh skema relasi. Berdasarkan batasan ini maka bisa dicari closure suatu skema relasi, lalu menemukan ketergantungan fungsional yang berulang, terakhir mendapatkan himpunan ketergantungan fungsional minimal. Berdasarkan ketergantungan fungsional minimal inilah skema relasi didekomposisi.

KATA PENGANTAR

Tugas akhir ini dibuat guna memenuhi salah satu syarat untuk meraih gelar sarjana pada Jurusan Teknik Informatika Fakultas Teknologi Industri Institut Teknologi Sepuluh Nopember Surabaya. Judul Tugas akhir ini adalah RANCANG BANGUN PERANGKAT LUNAK UNTUK NORMALISASI SKEMA RELASI SECARA OTOMATIS.

Pada kesempatan ini penulis menghaturkan terima kasih yang sebesar-besarnya kepada :

1. Mama dan Bapak yang telah bersabar cukup lama menunggu kelulusan penulis.
2. Ir. Arif Djunaidy, M.Sc. Ph.D. selaku dosen pembimbing dan dosen wali, yang telah bertahun-tahun membimbing penulis dalam membuat Tugas akhir. Terima kasih atas dukungan dan pengertiannya.
3. Ir. Gilbbran Subaggio yang telah menghibur penulis ketika melewati saat-saat yang berat dan memberikan nasehat dan bimbingan yang sangat berharga dan selalu memompakan semangat untuk segera menyelesaikan Tugas akhir ini.
4. Ir. Eko Indrajit M.Sc. yang telah bersusah payah mencarikan bahan-bahan rujukan tambahan dan mempelajari kembali konsep-konsep Tugas Akhir ini dan berkenan menginap di rumah penulis untuk bertukar pikiran dalam memahami algoritma inti Tugas Akhir ini.
5. Ir. Achmad Hadi, Ir. Seno HP, Bachtiar HS, Ir. Mustofa yang banyak membantu memberikan saran-saran terbaik untuk teknik pemrogramannya.
6. Rekan-rekan InfoSolusindo (ISI), Ir. Nixon Glenn, Ir. Isnin, Ir. Haryawan, Ir. Bambang, Ir. Iwan Syarif. Atas semua dukungannya baik moral maupun material.
7. Bapak dan Ibu Hanafi yang rela meminjamkan rumahnya untuk mengerjakan Tugas Akhir ini.
8. Indra dan Hengky yang banyak membantu penulis disaat-saat akhir penyelesaian buku Tugas Akhir ini.
9. Tante Anneke yang penuh perhatian dan banyak membantu penulis disaat-saat kritis.
10. Semua pihak yang mendukung kelancaran penulisan buku tugas akhir ini yang tidak bisa penulis sebutkan satu per satu di sini.

Semoga Allah SWT berkenan membalas semua kebaikan mereka dan Tugas Akhir ini bisa memberikan manfaat bagi kemajuan teknogi komputer di Indonesia. Amin.

DAFTAR ISI

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH – NOPEMBER

DAFTAR GAMBAR

BAB I

PENDAHULUAN

BAB I
PENDAHULUAN

1.1. LATAR BELAKANG

Desain basis data memegang peranan yang penting dalam proses rekayasa perangkat lunak. Desain yang benar akan meningkatkan kinerja sistem, sebaliknya desain yang salah akan menimbulkan banyak masalah antara lain data yang berulang (data redundant), data yang tidak konsisten (data inconsistent) dan yang paling berbahaya menyebabkan kemacetan sistem.

Selama ini, sejauh pengamatan penulis, hampir semua perancang basis data masih mendesain basis data secara intuitif atau informal. maksudnya, perancangan dilakukan tidak menggunakan prosedur formal. Hal ini tidak menjadi masalah dengan syarat :

(1) Pemahaman perancang terhadap konsep entiti dan hubungan antar entiti mantap.
(2) Jumlah atribut yang terlibat belum banyak.
(3) Tingkat kerumitan antar hubungan rendah.

Resiko terjadi kesalahan semakin besar jika ada dari tiga syarat diatas tidak dipenuhi. Jika hal ini terjadi maka perancangan basis data sudah saatnya menggunakan prosedur formal. Prosedur formal yang dimaksud disini adalah batasan-batasan yang berlaku jika suatu skema relasi didekomposisi menjadi bentuk normal. Selanjutnya prosedur formal ini disebut normalisasi. Normalisasi ini menggunakan sifat-sifat tambahan yang dimiliki skema relasi yaitu ketergantungan fungsional dan kunci utama.

Ketergantungan fungsional, ditulis $X \rightarrow Y$, adalah batasan yang mengikat antara dua himpunan atribut X dan Y yang merupakan himpunan bagian skema relasi R. Dalam hal ini nilai Y tergantung oleh nilai X atau dengan kata lain nilai X menentukan nilai Y. Dari ketergantungan fungsional ini dikenal aksioma armstrong, hukum inferensi, closure dan lain-lain.

Dari normalisasi yang menggunakan ketergantungan fungsional ini kemudian dikembangkan algoritma-algoritma yang menghasilkan skema relasi yang memenuhi syarat bentuk normal. Algoritma ini dikenal sebagai algoritma dekomposisi skema relasi.

1.2. TUJUAN

Berdasarkan uraian latar belakang diatas maka tujuan tugas akhir ini adalah sebagai berikut, " **Menyediakan perangkat otomatis untuk membantu merancang skema relasi basis data** ". Perangkat otomatis tersebut adalah sebuah perangkat lunak yang merupakan implementasi dari algoritma dekomposisi skema relasi.

1.3. PERMASALAHAN

Permasalahan yang dihadapi untuk membuat program ini adalah :

1. Algoritma yang ada di literatur masih memungkinkan dihasilkan skema relasi yang tidak diinginkan. Skema relasi ini bisa terbentuk karena memiliki urutan ketergantungan fungsional yang tidak tepat atau terdapat ketergantungan fungsional bolak-balik — ditulis $X \rightarrow Y$ dan $Y \rightarrow X$. Singkatnya Algoritma yang ada di literatur belum bisa menangani ketergantungan fungsional bolak-balik dan belum bisa mengurutkan ketergantungan fungsional agar menghasilkan skema relasi yang benar.

2. Algoritma yang ada di literatur tidak mengecek apakah semua sisi kanan ketergantungan fungsional yang akan diproses adalah atribut nonprima. Hal ini penting sebab jika memproses ketergantungan fungsional yang sisi kanannya adalah atribut prima maka bisa dipastikan skema relasi yang dihasilkan tidak sesuai dengan aturan bentuk normal Boyce Codd.

1.4. PEMBATASAN MASALAH

Perangkat lunak yang dirancang dan dibuat pada tugas akhir ini dibatasi untuk menormalkan skema relasi yang memenuhi bentuk normal Boyce Codd atau bentuk normal ketiga. Input program berupa file teks terdiri dari atribut skema relasi semesta R dan ketergantungan fungsional. Sedangkan outputnya berupa file teks yang terdiri dari skema relasi dalam bentuk normal.

INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

1.5. SISTEMATIKA PEMBAHASAN

Pembahasan tugas akhir ini dibagi menjadi lima bab. Bab I merupakan pendahuluan yang membahas latar belakang permasalahan, tujuan pembuatan tugas akhir, permasalahan yang ditimbul dan batasan masalah. Bab II membahas tentang basis data relasional dengan operasinya. Bab III membicarakan normalisasi data dengan menggunakan ketergantungan fungsional. Bab IV membahas desain algoritma normalisasi. Bab V membahas tentang pembuatan program. Bab VI adalah kesimpulan dan saran.

BAB II

BASIS DATA RELASIONAL DAN OPERASINYA

BAB II
BASIS DATA RELASIONAL
DAN OPERASINYA

Pada bab ini akan dibahas konsep dasar basis data relasional dan operasi-operasinya. Konsep ini pertama kali dikenalkan oleh Codd tahun 1970. Konsep ini memiliki dasar teori yang utuh dan berdasarkan struktur data yang seragam. Konsep ini berkembang terus dan menjadi mapan sehingga banyak paket-paket perangkat lunak komersial yang dibuat berdasarkan konsep ini.

Bab ini dimulai dengan membahas konsep model relasional (subbab 2.1.). Kemudian pada subbab 2.2. akan dibahas batasan integritas yang dianggap sebagai bagian penting dalam model relasional. Pada subbab 2.3. akan dibahas operasi pembaruan (update), subbab 2.4. akan dibahas aljabar relasional dan terakhir pada subbab 2.5. akan dibahas pengertian union compatibility.

2.1. KONSEP MODEL RELASIONAL

Relasi bisa dianggap sebagai tabel nilai-nilai dengan tiap barisnya menyajikan nilai suatu kumpulan data yang saling berhubungan. Nilai-nilai ini bisa ditafsirkan sebagai fakta yang menggambarkan entiti atau hubungan (relationship) yang sebenarnya. Nama tabel dan nama kolom dipakai untuk membantu memahami makna dari nilai tiap baris dari tabel tersebut. Berikut ini diberikan contoh tabel pada gambar 2.1. yang disebut tabel MAHASISWA.

MAHASISWA	NAMA	NRP	KELAS	MAYOR
	Juli	9801	A	COSC
	Wanda	9802	A	COSC
	Ninda	9803	B	COSC
	Sophie	9804	B	COSC

Gambar 2.1. Contoh Relasi dengan nama MAHASISWA

Setiap baris pada tabel di atas menyajikan fakta tentang entiti mahasiswa tertentu. Nama nama kolom — Nama, NRP, Kelas, Mayor — menjelaskan bagaimana menafsirkan nilai data-data pada masing-masing kolom.

Dalam terminologi model relasional, tiap baris disebut *tupel*, judul kolom disebut *atribut* dan tabelnya disebut *relasi*. Tipe data pada masing-masing kolom disebut *domain*. Pada subsubbab berikut ini akan dibahas terminologi ini — domain, tupel, atribut, relasi — lebih detail.

2.1.1. Domain, Tupel, Atribut dan Relasi

Domain D adalah himpunan nilai terkecil dari suatu atribut yang nilainya tidak bisa dibagi lagi selama model relasionalnya berlaku. Cara yang paling mudah untuk mengetahui domain suatu data ialah melihat tipe data tiap kolom dimana nilai-nilai tersebut berada, contoh domain adalah sebagai berikut :

- NoTelepon : Himpunan nomor telepon 10 digit yang berlaku di Indonesia
- NoTeleponLokal : Himpunan nomor telepon 7 digit yang berlaku di suatu daerah tertentu.
- NoKartuPenduduk : Himpunan nomor KTP yang masih berlaku sebanyak 9 digit.
- Nama : Himpunan nama-nama orang
- UmurKaryawan : Umur karyawan yang diterima pada suatu perusahaan nilainya antara 16 sampai dengan 65 tahun.

Skema relasi R ditulis $R(A_1,A_2, \ldots ,A_n)$ terdiri dari nama relasi R dan sejumlah atribut A_i yang digunakan untuk menggambarkan suatu relasi. Tiap atribut A_i adalah nama yang diperankan oleh domain D pada skema relasi R. D disebut domain dari A_i dan ditulis $dom(A_i)$. Skema relasi dipakai untuk menggambarkan relasi. R adalah nama relasi. Tingkat relasi adalah jumlah atribut pada skema relasinya. Berikut ini adalah contoh skema relasi yang menggambarkan skema relasi mahasiswa.

MAHASISWA (Nama, NRP, Tlp_Rmh, Alamat, TlpKtr, Umur, IP)
Tingkat skema relasi di atas adalah : 7

Nama skema relasi adalah : MAHASISWA

Domain skema relasi adalah :

1. dom(Nama) = Nama-nama
2. dom(NRP) = Nomor Registrasi Pelajar
3. dom(Tlp_Rmh) = Nomor Telepon Lokal
4. dom(Alamat) = Alamat rumah orang tua
5. dom(Tlp_Ktr) = Nomor Telepon Lokal
6. dom(Umur) = Umur karyawan
7. dom(IP) = Indeks Prestasi Mahasiswa

Anggota relasi (relation instance) — ditulis r(R) — dari skema relasi R (A_1,A_2, . . . ,A_n) adalah himpunan n-tupel ditulis r = {t_1,t_2, . . . ,t_n}. n-tupel adalah himpunan tupel sebanyak n. Adapun tupel ditulis t = < v_i,v_{i+1}, . . . , v_n >, tiap nilai v_i, $1 \leq i \leq n$, adalah anggota dom(A_i) atau bernilai null. Skema R disebut juga relasi intension dan anggota relasi disebut juga relasi ekstension.

Gambar 2.2. memperlihatkan contoh atribut dan tupel MAHASISWA. Tiap tupel pada relasi mewakili entiti mahasiswa tertentu. Dalam konteks ini, relasi digambarkan dalam bentuk tabel.Tupel relasi digambarkan sebagai baris tabel dan atributnya digambarkan sebagai judul kolom. Judul kolom inilah yang membantu pengguna memahami makna yang dikandung sebuah nilai untuk semua nilai pada kolom tersebut. Nilai null pada atribut menjelaskan bahwa nilai atribut tersebut belum diketahui atau nilainya tidak ada.

MAHASISWA	NAMA	NRP	TLP_RMH	ALAMAT	TLP_KTR	UMUR	IP
	Juli	98001	5352488	RL IIA/2	Null	20	4
	Ninda	98002	7513070	RL IF/20	8700741	22	3,5
	Wanda	98003	7312222	RL IIB/17	8700884	22	3,0
	Sophie	98004	8411111	RL IIIA/3	5673574	21	2,5
	Yasmin	98005	Null	RL IC/4	Null	27	Null

Gambar 2.2. Atribut dan tupel relasi MAHASISWA

Anggota skema relasi r(R) bisa juga didefinisikan sebagai himpunan bagian dari perkalian cartesian yang berasal dari definisi domain R :

$$r(R) \subseteq (dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n))$$

Perkalian cartesian menggambarkan semua kombinasi nilai-nilai yang mungkin untuk domain yang ada. Jumlah nilai atau kardinal domain D adalah | D |, Jika semua domain adalah terhingga (finite), maka jumlah total tupel hasil perkalian cartesian adalah

$$|dom(A_1)| * |dom(A_2)| * \ldots * |dom(A_n)|$$

2.1.2. Karakteristik Relasi

Relasi adalah himpunan tupel-tupel. Secara matematis definisi ini mengandung konsekuensi :

1. Dalam suatu relasi tidak mengenal urutan.
2. Antara tupel yang satu dengan yang lain harus berbeda, tidak boleh ada tupel yang sama.

Beda dengan file dimana record-recordnya secara fisik disimpan dalam bentuk yang berurutan. Record ke satu menyatakan bahwa record tersebut disimpan pertama kali demikian pula record kedua dan seterusnya. Demikian pula dengan tabel dimana baris dan kolomnya menyatakan urutan.

Dalam relasi, setiap tupel harus bernilai tunggal (atom) artinya nilai tersebut tidak bisa dibagi lagi menjadi beberapa bagian. Oleh sebab itu atribut komposit dan bernilai banyak tidak diizinkan. Nilai pada tupel bisa belum diketahui atau memang tidak mempunyai nilai. Untuk menyatakan hal ini dikenal sebuah nilai yang disebut null. Secara umum nilai null pada atribut mempunyai beberapa penafsiran yaitu :

1. Nilai atribut tidak diketahui.
2. Atribut tidak berlaku pada tupel ini.
3. Tupel ini tidak mempunyai nilai pada atribut tersebut.

2.1.3. Notasi Model Relasional

Berikut ini akan dijelaskan notasi-notasi yang digunakan pada sub bab - sub bab selanjutnya.

- Skema relasi dengan tingkat n ditulis dengan R (A_1, A_2, . . . ,A_n)
- N tupel t pada anggota relasi r (R) ditulis dengan t = < v_i , v_{i+1} , . . . , v_n > dimana v_i adalah nilai dari atribut A_i. Notasi berikut ini berhubungan dengan nilai komponen tupel.
- t[A_i] adalah nilai v_i di t untuk atribut A_i
- t[A_u , A_w , . . . , A_z] dimana A_u , A_w , . . . , A_z adalah daftar atribut R, adalah nilai suatu sub tupel < v_u , v_w , . . . , v_z > di t, untuk atribut yang tertera di daftar tersebut.
- Huruf Q, R, S menyatakan nama relasi.
- Huruf q, r, s menyatakan anggota relasi.
- Huruf t,u, v menyatakan tupel.
- Secara umum nama relasi seperti MAHASISWA, menjelaskan himpunan tupel-tupel pada relasi tersebut. Adapun MAHASISWA (Nama, NRP, . . ., IP) menyatakan skema relasi.
- Nama atribut kadang-kadang ditulis dengan nama relasi dimana atribut tersebut berada — Contoh : MAHASISWA.Nama, MAHASISWA.IP

Perhatikan tupel berikut ini, t = < 'Iwan' , '2688.100.039' , '870-0561'. , 'RL IIA No.2', null, 26, 4.0 >, yang berasal dari relasi MAHASISWA; maka t[Nama] = <'Iwan'> dan t[NRP,IP,Umur] = < '2688.100.039', 4.0 , 26 >

2.2. BATASAN MODEL RELASIONAL

Pada bagian ini akan dibahas batasan-batasan yang harus dipenuhi oleh skema relasi basis data, maksudnya suatu skema basis data bisa disebut skema relasi jika memenuhi batasan-batasan tersebut. Batasan-batasan itu adalah batasan domain,

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH – NOPEMBER

batasan kunci, batasan integritas entiti dan batasan integritas referensi. Batasan lainnya adalah ketergantungan fungsional. Batasan inilah yang menjadi dasar dalam mendesain skema relasi terutama dalam proses normalisasi yang akan dibahas pada bab 3 dan 4.

2.2.1. Batasan Domain

Batasan domain (Domain constraints) menyatakan bahwa setiap nilai atribut A harus berupa nilai tunggal dan berasal dari domain dom(A). Tipe data yang secara khusus diasosiasikan dengan domain diantaranya adalah tipe data numerik untuk integer (yaitu short integer, integer, long integer) dan angka real (float dan double precision float). Tipe data lainnya yang bisa diasosiasikan dengan domain ialah tipe data karakter, fixed length string, dan variable length string, contohnya adalah tanggal, waktu, dan mata uang. Domain bisa juga digambarkan dengan rentang nilai suatu tipe data atau tipe enumerasi dimana semua nilai yang mungkin ditulis secara eksplisit.

2.2.2. Batasan Kunci

Batasan kunci (Key constraints) menyatakan bahwa pada suatu skema relasi R selalu ada himpunan bagian atribut yang mempunyai sifat unik dimana tidak ada dua tupel yang mempunyai kombinasi nilai yang sama. Himpunan bagian itu disebut kunci super (superkey).

Kunci super (Super key) adalah : Himpunan bagian dari himpunan atribut pada skema relasi R yang sifatnya bisa membedakan antar tupel-tupel pada anggota relasinya sehingga tidak ada tupel-tupel yang kombinasi atributnya mempunyai nilai yang sama. Kunci (Key) adalah : Kunci super minimal.

Supaya lebih jelas dengan batasan ini lihat kembali skema relasi MAHASISWA yang terdapat pada gambar 2.2. Himpunan atribut {NRP} adalah kunci skema relasi MAHASISWA karena tidak ada tupel mahasiswa yang NRP nya sama. Himpunan atribut {NRP,NAMA,UMUR} adalah kunci super. Kunci super {NRP,NAMA,UMUR} bukan sebuah kunci sebab jika NAMA atau UMUR atau

kedua-duanya dihapus dari himpunan tersebut maka himpunan tersebut masih kunci super.

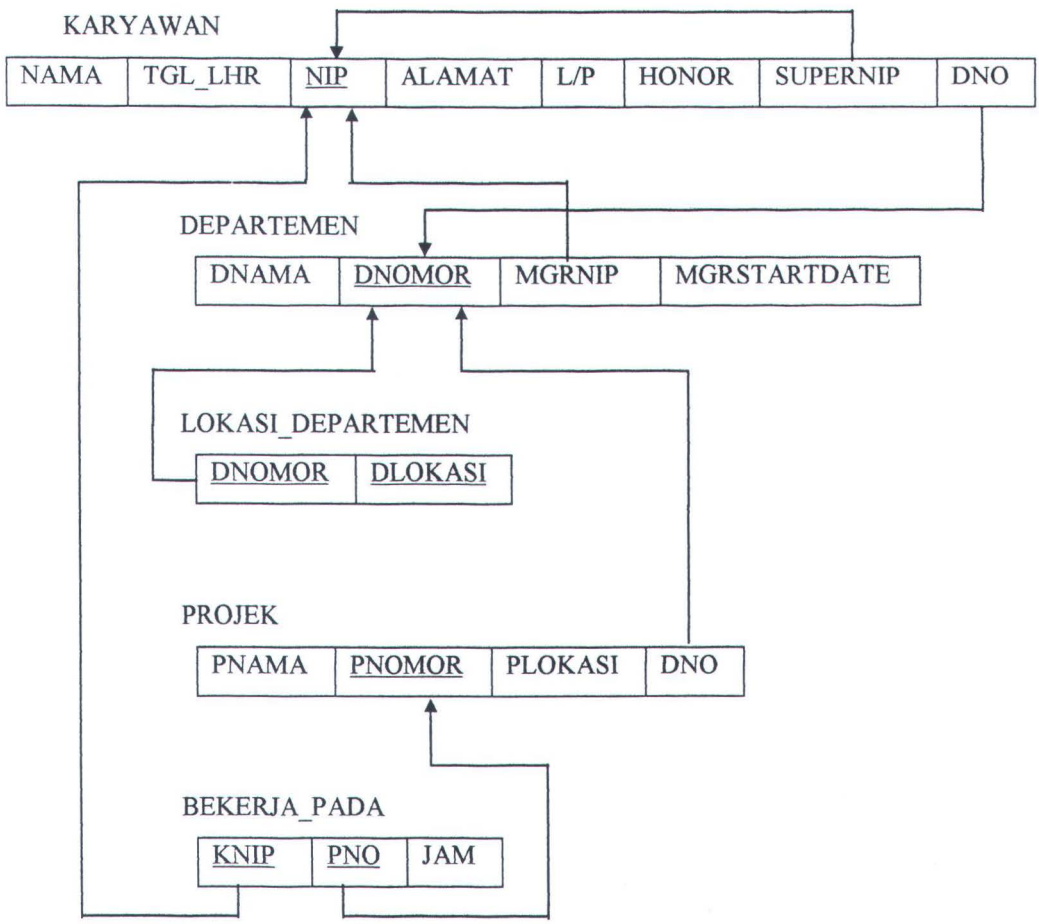
Dalam sebuah relasi biasanya terdiri dari beberapa kunci. Kunci-kunci itu disebut kunci calon (candidate key). Dari kunci calon tersebut dipilih satu kunci yang dipakai untuk identitas tupel-tupelnya. kunci ini disebut kunci utama (primary key). Biasanya kunci utama ditulis dengan garis bawah untuk membedakan dengan kunci calon. Sebaiknya kunci utama terdiri dari satu atribut atau dengan jumlah atribut sekecil-kecilnya.

2.2.3. Batasan Integritas Entiti

Batasan integritas entiti (Entity integrity constraints) menyatakan bahwa tidak ada kunci utama yang bernilai null sebab kunci utama digunakan untuk mengidentifikasi sebuah tupel. Jika ada kunci utama yang bernilai null maka tupel tersebut tidak dapat diidentifikasi. Selain itu tupel-tupel tersebut tidak dapat dibedakan satu dengan lainnya.

2.2.4. Batasan Integritas Rujukan

Batasan integritas rujukan (Referential integrity constraints) menyatakan bahwa tupel yang terdapat pada sebuah relasi dan merujuk pada tupel lain harus merujuk pada tupel yang ada. batasan ini membutuhkan kunci luar (foreign key). Untuk lebih jelasnya lihat gambar 2.3. berikut ini.



Gambar 2.3. Contoh batasan integritas rujukan

Atribut DNO pada relasi KARYAWAN menunjukan departemen tempat karyawan tersebut bekerja. Nilai DNO harus cocok dengan nilai DNOMOR pada relasi DEPARTEMEN. Atribut MGRNIP pada relasi DEPARTEMEN menunjuk pada atribut NIP pada relasi KARYAWAN artinya Manajer yang ditugaskan pada departemen yang bersangkutan harus menjadi karyawan perusahaan dan datanya terdapat pada relasi KARYAWAN.

Dalam sebuah relasi kadang terdapat suatu atribut yang merupakan kunci dari relasi lainnya. Atribut ini disebut **kunci luar (foreign key).** Secara formal suatu kunci disebut kunci luar jika,

1. Domain antara atribut R_1 dan R_2 sama

2. Nilai kunci luar yang ada di R_1 terdapat pada kunci utama di R_2 dengan asumsi kunci luar terdapat di relasi R_1 dan kunci utama terdapat di relasi R_2.

2.3. OPERASI UPDATE PADA RELASI

Operasi-operasi yang dapat dilakukan pada model relasional dapat dibedakan menjadi 2 jenis yaitu :

1. **Operasi retrieval** yaitu operasi untuk mendapatkan model relasional tertentu. Operasi ini juga dikenal sebagai operasi aljabar relasional. Operasi ini akan dibahas lebih mendalam pada sub bab 2.4.
2. **Operasi Update** yaitu operasi untuk mengubah anggota suatu relasi. Operasi ini terdiri dari tiga operasi dasar yaitu insert, delete, modify. Berikut ini akan dibahas operasi update lebih dalam.

2.3.1. Operasi Insert

Operasi ini menyisipkan tupel baru pada suatu relasi R. Operasi ini bisa melanggar empat batasan yang telah yang telah dibahas sebelumnya yaitu batasan domain, batasan kunci, batasan entiti, batasan rujukan. Operasi ini melanggar batasan domain jika nilai atribut yang diberikan tidak terdapat pada domain yang berhubungan. Operasi ini melanggar batasan kunci jika nilai kunci pada tupel baru t sudah ada pada anggota relasi r(R). Operasi ini melanggar batasan integritas entiti jika tupel baru t mempunyai kunci utama yang nilainya null. Operasi ini melanggar batasan integritas rujukan jika nilai kunci luar pada tupel baru t merujuk pada tupel yang tidak ada pada relasi yang dirujuki. Dibawah ini diberikan relasi KARYAWAN berikut contoh operasi insert yang benar dan yang melanggar batasan model relasional tersebut.

KARYAWAN

NAMA	NIP	TGL_LHR	ALAMAT	L/P	HONOR	DNO
John	199901	09-Jan-55	731 Fondren,Houston, TX	L	30000	5
Franklin	199902	08-Des-45	638 Voss,Houston, TX	L	40000	5
Alicia	199903	19-Jul-58	3321 Castle,Spring, TX	P	25000	4
Jennifer	199904	20-Jun-57	291Berry,Bellaire, TX	P	43000	4
Ramesh	199905	15-Sep-52	975 Fire Oak,Humble, TX	L	38000	5
Joyce	199906	31-Jul-62	5631 Rice,Houston, TX	P	25000	5
Ahmad	199907	29-Mar-59	980 Dallas,Houston, TX	L	25000	4
James	199908	10-Nop-27	450 Stone,Houston, TX	L	55000	1

DEPARTEMEN

DNAMA	DNOMOR	MGRNIP	MGRSTARTDATE
Marketing	5	199902	22-Mei-88
Produksi	4	199904	01-Jan-85
Administrasi	3	199908	19-Jun-81
Personalia	1	199905	22-Mei-80

Gambar 2.4. Relasi KARYAWAN dan DEPARTEMEN

1. Insert <'Cecilia','199909','05-Apr-50',6357 Windy Lane, Katy,TX,28000,4> ke dalam KARYAWAN.

- Operasi insert ini memenuhi semua batasan, jadi bisa dilakukan.

2. Insert <'Cecilia','199903','05-Apr-50',6357 Windy Lane, Katy,TX,28000,4> ke dalam KARYAWAN.

- Operasi insert ini melanggar batasan kunci karena tupel lain dengan NIP yang sama sudah ada pada relasi KARYAWAN.

3. Insert <'Cecilia',null,'05-Apr-50',6357 Windy Lane, Katy,TX,28000,4> ke dalam KARYAWAN.

- Operasi insert ini melanggar batasan integritas entiti karena karena kunci utamanya yaitu NIP bernilai null.

4. Insert <'Cecilia','199903','05-Apr-50',6357 Windy Lane, Katy,TX,28000,7> ke dalam KARYAWAN.

- Operasi insert ini melanggara batasan integritas rujukan karena nomor departemen yang ditunjukan oleh DNO = 7 tidak terdapat pada relasi DEPARTEMEN.

2.3.2. Operasi Delete

Operasi ini menghapus suatu tupel pada relasi. Operasi ini bisa melanggar batasan integritas rujukan jika tupel yang dihapus, dirujuk oleh kunci luar dari tupel lain pada basis data. Untuk menghapus suatu tupel, perlu ditetapkan suatu kondisi yang menentukan tupel mana yang dipilih. Dibawah ini diberikan relasi BEKERJA_PADA berikut contoh operasi delete yang benar dan yang melanggar batasan model relasional tersebut. Adapun relasi KARYAWAN dan DEPARTEMEN melihat pada gambar 2.4.

BEKERJA_PADA

NIP	PNO	JAM
199901	1	33
199901	2	8
199903	3	40
199904	1	20
199904	2	20
199906	2	10
199906	3	10
199906	10	10

Gambar 2.5. Relasi BEKERJA_PADA

1. Penghapusan tupel BEKERJA_PADA dengan NIP = '199906' dan NOPROJ = 10

- Operasi ini dapat dilakukan

2. Penghapusan tupel KARYAWAN dengan NIP = '199906'

- Operasi ini tidak dapat dilakukan, karena dua tupel di relasi BEKERJA_PADA merujuk ke tupel ini. Jika tupel ini dihapus, maka batasan integritas rujukan dilanggar

3. Penghapusan tupel KARYAWAN dengan NIP = '199904'

- Operasi ini akan mengakibatkan pelanggaran berat terhadap batasan integritas rujukan, karena tupel yang dihapus dirujuk oleh relasi KARYAWAN, DEPARTEMEN dan BEKERJA_PADA.

2.3.3. Operasi Modify

Operasi ini mengubah nilai tupel yang ada pada sebuah relasi. Operasi ini bisa melanggar batasan integritas rujukan jika tupel yang diubah, dirujuk oleh kunci luar dari tupel lain pada basis data. Operasi ini bisa melanggar batasan kunci jika kunci utama yang diubah ternyata identik dengan kunci utama yang sudah ada. Untuk lebih memahami operasi ini berikut diberikan contoh bagaimana melakukan operasi modify dan hal-hal apa yang perlu diperhatikan sehingga operasi ini bisa dilakukan. Contoh dibawah ini merujuk pada gambar 2.4.

1. Mengubah HONOR dari tupel KARYAWAN dengan NIP = '199901' menjadi 28000
   - Dapat dilakukan
2. Mengubah DNO dari tupel KARYAWAN dengan NIP = '199901' menjadi 1
   - Dapat dilakukan
3. Mengubah DNO dari tupel KARYAWAN dengan NIP = '199901' menjadi 7
   - Tidak dapat dilakukan, karena operasi ini melanggar batasan integritas rujukan .
4. Mengubah NIP dari tupel KARYAWAN dengan NIP = '199901' menjadi '199902'
   - Tidak dapat dilakukan, karena operasi ini melanggar batasan kunci dan batasan integritas rujukan .

2.4. ALJABAR RELASIONAL

Operasi aljabar relasional dibagi menjadi dua kelompok. Kelompok pertama adalah operasi himpunan yang berasal dari teori matematika tentang himpunan. Operasi ini berlaku karena tiap relasi yang didefinisikan adalah himpunan tupel-tupel. Operasi himpunan terdiri dari UNION, INTERSEKSI, DIFFERENCE dan PERKALIAN CARTESIAN. Kelompok lainnya terdiri dari operasi-operasi yang dikembangkan untuk basis data relasional. Operasi-operasi yang tergabung dalam kelompok ini adalah SELECT, PROJECT dan JOIN. Pada subbab berikut ini akan

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH – NOPEMBER

dijelaskan operasi-operasi yang dipakai untuk mendesain skema relasi. Operasi tersebut adalah operasi SELECT, PROJECT, PERKALIAN CARTESIAN dan JOIN.

2.4.1. Operasi Select

Operasi ini digunakan untuk memilih himpunan bagian dari tupel-tupel yang terdapat pada sebuah relasi yang memenuhi syarat. Contoh, untuk memilih himpunan bagian tupel-tupel KARYAWAN yang bekerja di departemen 4 dengan gaji lebih besar daripada Rp. 6.000.000,- yang ditulis dengan cara sendiri-sendiri adalah sebagai berikut:

$\sigma_{DNO=4}(KARYAWAN)$

$\sigma_{HONOR > 30000}(KARYAWAN)$

Secara umum operasi ini ditulis dengan cara sebagai berikut :

$\sigma_{<kondisi\ seleksi>}(<\text{nama relasi}>)$

Tanda sigma dipakai untuk menyatakan operator SELECT. Kondisi seleksi adalah ekspresi boolean yang berlaku untuk atribut relasi. Diketahui sebuah operasi SELECT yang ditulis sebagai berikut :

$\sigma_{(DNO=4\ AND\ HONOR > 25000)\ OR\ (DNO=5\ AND\ HONOR > 30000)}(KARYAWAN)$

Maka hasil dari operasi tersebut adalah sebagai berikut :

NAMA	NIP	TGL_LHR	ALAMAT	L/P	HONOR	DNO
Franklin	199902	08-Des-45	638 Voss,Houston, TX	L	40000	5
Jennifer	199904	20-Jun-57	291Berry,Bellaire, TX	P	43000	4
Ramesh	199905	15-Sep-52	975 Fire Oak,Humble, TX	L	38000	5

Gambar 2.6. Hasil Operasi Select

2.4.2. Operasi Project

Operasi SELECT memilih beberapa baris dari tabel tersebut sedangkan operasi PROJECT memilih kolom-kolom tertentu dari tabel tersebut. Contoh untuk mendapatkan sederetan NAMA,HONOR dan L/P bisa digunakan operasi PROJECT sebagai berikut :

$\pi_{NAMA,HONOR,L/P}(KARYAWAN)$

Hasilnya diperlihatkan pada gambar berikut ini,

NAMA	HONOR	L/P
John	30000	L
Franklin	40000	L
Alicia	25000	P
Jennifer	43000	P
Ramesh	38000	L
Joyce	25000	P
Ahmad	25000	L
James	55000	L

Gambar 2.7. Hasil Operasi Project

Secara umum relasi PROJECT ditulis sebagai berikut ,

$\pi_{<\text{daftar atribut}>}(<\text{nama relasi}>)$

Dimana π adalah simbol yang mewakili operasi PROJECT dan <daftar atribut> adalah sederetan atribut yang terdapat pada relasi.

2.4.3. Operasi Perkalian Cartesian

Operasi ini digunakan untuk mengkombinasikan dua tupel yang berbeda tipe. Secara umum hasil dari $R(A_1, A_2, \ldots, A_n) \times S(B_1, B_2, \ldots, B_n)$ adalah relasi Q dengan atribut yang jumlahnya n+m dan tupel-tupel berjumlah $n_R * n_s$. Untuk lebih jelasnya perhatikan contoh pemakaian perkalian cartesian berikut ini. Tujuan dari contoh berikut adalah untuk mendapatkan informasi tentang jiwa yang ditanggung oleh karyawan wanita.

$KARYAWAN\_WANITA \leftarrow \sigma_{L/P = \text{'P'}}(KARYAWAN)$

$NAMA\_KARYAWAN \leftarrow \pi_{NAMA,NIP}(KARYAWAN\_WANITA)$

$TANGGUNGAN\_KARYAWAN \leftarrow NAMA\_KARYAWAN \times TANGGUNGAN$

$TANGGUNGAN\_NYATA \leftarrow \sigma_{NIP = NIP}(TANGGUNGAN\_KARYAWAN)$

$HASIL \leftarrow \pi_{NAMA,NAMA\_TANGGUNGAN}(TANGGUNGAN\ NYATA)$

KARYAWAN_WANITA

NAMA	NIP	TGL_LHR	ALAMAT	L/P	HONOR	DNO
Alicia	199903	19-Jul-58	3321 Castle,Spring, TX	P	25000	4

NAMA	NIP	TGL_LHR	ALAMAT	L/P	HONOR	DNO
Jennifer	199904	20-Jun-57	291Berry,Bellaire, TX	P	43000	4
Joyce	199906	31-Jul-62	5631 Rice,Houston, TX	P	25000	5

NAMA_KARYAWAN

NAMA	NIP
Alicia	199903
Jennifer	199904
Joyce	199906

TANGGUNGAN

NIP	TANGGUNGAN	L/P	TGL_LHR	RELASI
199902	Alicia	P	05-Apr-76	Anak Perempuan
199902	Theodore	L	25-Okt-73	Anak Laki-laki
199902	Joy	P	03-Mei-48	Istri
199904	Abner	L	29-Peb-32	Suami
199901	Michel	L	01-Jan-78	Suami
199901	Alice	P	31-Des-78	Anak Perempuan
199901	Elizabeth	P	05-Mei-57	Istri

TANGGUNGAN_KARYAWAN

NAMA	NIP	NRP	TANGGUNGAN	L/P	TGL_LHR	...
Alicia	199903	199902	Alice	P	05-Apr-76	...
Alicia	199903	199902	Theodore	L	25-Okt-73	...
Alicia	199903	199902	Joy	P	03-Mei-48	...
Alicia	199903	199904	Abner	L	29-Peb-32	...
Alicia	199903	199901	Michael	L	01-Jan-78	...
Alicia	199903	199901	Alice	P	31-Des-78	...
Alicia	199903	199901	Elizabeth	P	05-Mei-57	...
Jennifer	199904	199902	Alice	P	05-Apr-76	...
Jennifer	199904	199902	Theodore	L	25-Okt-73	...
Jennifer	199904	199902	Joy	P	03-Mei-48	...
Jennifer	199904	199904	Abner	L	29-Peb-32	...
Jennifer	199904	199901	Michael	L	01-Jan-78	...
Jennifer	199904	199901	Alice	P	31-Des-78	...
Jennifer	199904	199901	Elizabeth	P	05-Mei-57	...
Joyce	199906	199902	Alice	P	05-Apr-76	...
Joyce	199906	199902	Theodore	L	25-Okt-73	...
Joyce	199906	199902	Joy	P	03-Mei-48	...
Joyce	199906	199904	Abner	L	29-Peb-32	...
Joyce	199906	199901	Michael	L	01-Jan-78	...
Joyce	199906	199901	Alice	P	31-Des-78	...
Joyce	199906	199901	Elizabeth	P	05-Mei-57	...

TANGGUNGAN_NYATA

NAMA	NIP	NRP	TANGGUNGAN	L/P	TGL_LHR
Jennifer	199904	987654321	Abner	L	29-Peb-32

HASIL

NAMA	TANGGUNGAN
Jennifer	Abner

Gambar 2.8. Hasil Operasi Perkalian Cartesian

2.4.4. Operasi Join

Operasi JOIN ditulis dengan ?, digunakan untuk menggabungkan dua tupel yang berelasi dari dua relasi yang berbeda menjadi satu tupel. Operasi ini sama dengan perkalian cartesian hanya operasi ini mempunyai kondisi yang harus dipenuhi sehingga tupel yang muncul harus sesuai dengan kondisi yang ditetapkan. Operasi ini belum terasa manfaatnya untuk relasi tunggal. Operasi ini baru berguna untuk sebuah basis data relasional. Supaya lebih memahami operasi ini perhatikan contoh berikut ini.

Untuk mendapatkan nama manager dari tiap departemen maka perlu menggabungkan tupel departemen dan tupel karyawan yang NRP-nya sama dengan NRP manager yang ada di tupel departemen. Untuk melakukan itu semua bisa menggunakan operasi JOIN. Kemudian melakukan operasi PROJECT untuk mendapatkan atribut-atribut yang diperlukan.

DEPT_MGR $\leftarrow$ DEPARTEMENT $_{\text{MGRNIP=NIP}}$ KARYAWAN

HASIL $\leftarrow \pi_{\text{DNAMA,NAMA}}$ (DEPT_MGR)

DEPT_MGR

DNAMA	DNO	MGRNIP	NAMA	NIP	...
Riset	5	199902	Franklin	199902	...
Administrasi	4	199904	Jennifer	199904	...
Kantor Cabang	1	199908	James	199908	...

Gambar 2.9. Hasil Operasi Join

2.5. UNION COMPATIBILITY

Dua relasi $R(A_1, A_2, \ldots, A_n) \times S(B_1, B_2, \ldots, B_n)$ disebut union compatible jika mereka mempunyai tingkat relasi n yang sama dan $dom(A_i) = dom(B_i)$ untuk $1 \leq i \leq n$. Artinya dua relasi tersebut mempunyai jumlah atribut yang sama dan domain yang sama untuk masing-masing pasangannya.

Jika operasi UNION, INTERSECTION dan DIFFERENCE akan digunakan pada relasi maka relasi-relasi yang terlibat harus union compatible sebab operasi tersebut bisa dilakukan pada pasangan tupel-tupel yang domainnya sama dan relasi tersebut mempunyai tingkat relasi n yang sama.

BAB III

NORMALISASI DATA DAN KETERGANTUNGAN FUNGSIONAL

BAB III
NORMALISASI DATA
DAN
KETERGANTUNGAN FUNGSIONAL

Pada bab ini akan dibahas teori untuk memilih skema relasi yang baik. Sebelum masuk pada pokok bahasan, kiranya pembaca perlu memahami dulu kriteria skema relasi yang baik secara informal atau intuitif. Tujuannya agar pembaca lebih memahami latar belakang teori-teori tersebut dikembangkan.

Pada bab 3.1. akan membahas desain skema relasi secara informal. Bab 3.2. akan membahas ketergantungan fungsional. Bab 3.3. akan membahas normalisasi data dengan menggunakan ketergantungan fungsional.

3.1. DESAIN SKEMA RELASI SECARA INFORMAL

Kualitas desain suatu skema relasi secara informal dapat diukur berdasarkan 4 (empat) hal berikut yaitu :

1. Makna dari atribut-atribut yang berelasi.
2. Isi yang berulang pada tupel.
3. Isi yang bernilai null pada tupel.
4. Tupel-tupel yang palsu (spurious) yang ada pada skema relasi tersebut.

Pada sub bab berikut ini akan dibahas satu per satu secara berurutan empat kriteria mengukur kualitas skema relasi tersebut.

3.1.1. Makna Atribut-Atribut yang Berelasi

Jika sejumlah atribut dikelompokkan untuk membentuk skema relasi, sedapat mungkin kelompok atribut itu mengandung suatu makna. Semakin mudah makna dari skema relasi dimengerti maka semakin baik desain skema relasinya. Untuk lebih jelasnya bisa disimak contoh skema basis data relasional PERUSAHAAN yang sudah disederhanakan sebagai berikut,

KARYAWAN

NAMA	NIP	TGL_LHR	ALAMAT	NO_DPT

DEPARTEMEN

NAMA_DPT	NO_DPT	MGRNIP

LOKASI_DEPARTEMEN

NO_DPT	LOK_DPT

PROJEK

NAMA_PRJ	NO_PRJ	LOK_PRJ	KODE_DPT

BEKERJA_DI

NIP	NO_PRJ	JAM_KERJA

Gambar 3.1. Skema basis data relasional PERUSAHAAN yang disederhanakan

Makna skema relasi KARYAWAN cukup sederhana, masing-masing tupel mewakili seorang pegawai dengan isi untuk masing-masing namanya (NAMA), nomor induknya (NIP), tanggal lahirnya (TGL_LHR), alamatnya (ALAMAT), nomor departemen dimana pegawai tersebut ditempatkan (NO_DPT). Atribut NO_DPT adalah kunci luar yang mewakili hubungan implisit antara KARYAWAN dan DEPARTEMEN. Makna dari skema DEPARTEMEN dan skema PROJEK juga sederhana. Masing-masing tupel DEPARTEMEN mewakili satu departemen dan masing-masing tupel PROJEK mewakili satu proyek. Atribut MGRNIP pada DEPARTEMEN mewakili seorang pegawai yang menjadi manager pada departemen itu. Sementara atribut KODE_DPT pada PROJEK mewakili departemen yang mengontrol proyek tersebut. kedua-duanya adalah kunci luar.

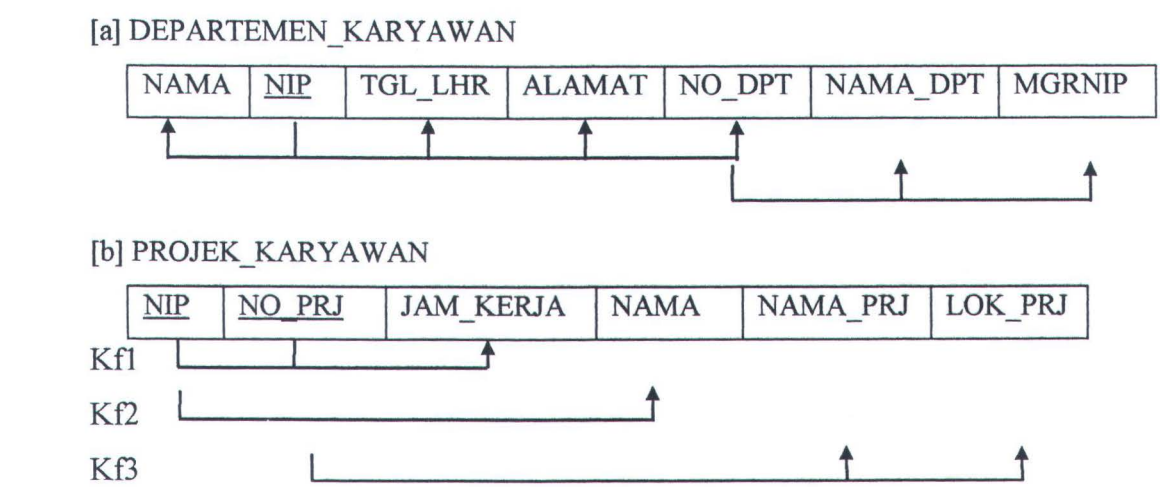
Makna dari dua skema relasi lainnya yang tampak pada gambar diatas lebih rumit. Masing-masing tupel pada LOKASI_DEPARTEMEN terdiri dari nomor departemen (NO_DPT) dan satu lokasi dari departemen tersebut (LOK_DPT). Masing-masing tupel yang terdapat pada BEKERJA_DI terdiri dari nomor induk pegawai (NIP), nomor proyek dimana pegawai tersebut bekerja (NO_PRJ) dan jumlah total jam yang dihabiskan per minggu oleh pegawai tersebut. Kedua skema tersebut terdefinisi dengan baik dan tidak bermakna ganda.

Skema LOKASI_DEPARTEMEN menyajikan atribut yang bernilai ganda (multivalue) dari DEPARTEMEN. Adapun BEKERJA_DI menyajikan hubungan M:N antara KARYAWAN dan PROJEK. Secara umum, semua skema relasi yang terdapat pada gambar 3.1. mempunyai desain yang baik jika berdasarkan kriteria mempunyai pengertian (semantik) yang jelas.

Berdasarkan uraian di atas dapat disimpulkan bahwa dalam merancang skema relasi harus sedemikian rupa sehingga mudah dalam menjelaskan artinya. Jangan menggabungkan atribut-atribut yang berasal dari berbagai macam entiti dan hubungannya dalam satu relasi. Secara intuitif jika sebuah skema diwakili oleh satu entiti dan satu hubungan maka artinya cenderung jelas.

3.1.2. Isi Yang Berulang Pada Tupel

Salah satu tujuan desain skema relasi adalah menghemat ruang yang digunakan oleh file. Pengelompokan atribut ke dalam skema relasi berpengaruh besar pada ruang penyimpanannya. sebagai contoh bandingkan besar ruang yang ditempati KARYAWAN dan DEPARTEMEN pada gambar 3.1. di atas dengan besar ruang yang ditempati oleh DEPARTEMEN_KARYAWAN pada gambar 3.2. di bawah yang mana merupakan hasil dari operasi NATURAL JOIN antara KARYAWAN dan DEPARTEMEN.



Gambar 3.2. Relasi DEPARTEMEN_KARYAWAN dan PROJEK_KARYAWAN

Pada DEPARTEMEN_KARYAWAN beberapa atribut yang berhubungan dengan departemen (NO_DPT, NAMA_DPT, MGRNIP) selalu diulang untuk setiap pegawai yang ditempatkan pada departemen. Lain halnya dengan relasi DEPARTEMEN pada gambar 3.1. dimana informasi departemen hanya muncul satu kali untuk setiap departemen. Adapun nomor departemen (NO_DPT) ditulis berulang kali pada relasi KARYAWAN untuk setiap pegawai yang ditempatkan pada departemen tersebut. Pengertian yang sama juga berlaku pada relasi PROJEK_KARYAWAN yang merupakan gabungan dari relasi BEKERJA_DI ditambah dengan atribut dari KARYAWAN dan PROJEK.

Persoalan serius lainnya yang terjadi pada desain relasi pada gambar 3.2. adalah munculnya penyimpangan ketika dilakukan pengubahan pada skema relasi (update anomalies). Pengubahan ini bisa terjadi dalam bentuk penyisipan (insertion anomalies), penghapusan (deletion anomalies), penyesuaian (modification anomalies)

.

Penyimpangan Penyisipan dibedakan menjadi dua macam, untuk menjelaskan hal ini kita memakai contoh relasi DEPARTEMEN_KARYAWAN :

1. Penyimpangan pertama terjadi ketika menyisipkan data pegawai baru ke dalam tupel DEPARTEMEN_KARYAWAN, konsekuensinya atribut departmen harus selalu diberi nilai. Selama pegawai tersebut sudah ditempatkan pada suatu departemen maka nilai atribut departemen diisi sesuai dengan departemen dimana pegawai tersebut ditempatkan, tetapi jika pegawai tersebut belum ditempatkan pada suatu departemen maka mau tidak mau atribut departemen diberi nilai null.

2. Penyimpangan kedua terjadi ketika akan menyisipkan data departemen baru yang belum mempunyai karyawan. Satu-satunya cara melakukannya adalah dengan mengisi nilai null pada atribut pegawai. Hal ini menimbulkan masalah sebab NIP adalah kunci utama (primary key) relasi DEPARTEMEN_ KARYAWAN dan tiap tupel mewakili satu pegawai. Ketika pegawai pertama ditempatkan pada departemen tersebut maka tupel yang atribut pegawainya bernilai null tidak dibutuhkan lagi.

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH – NOPEMBER

Penyimpangan Penghapusan. Penyimpangan ini berhubungan dengan penyimpangan penyisipan bentuk kedua dimana penghapusan tupel terakhir dari pegawai yang bekerja pada suatu departemen tertentu akan menghapus semua informasi yang berkaitan dengan departemen tersebut.

Penyimpangan Pengubahan. Penyimpangan ini terjadi jika seseorang mengubah nilai satu atribut pada suatu departemen – katakanlah manajer pada departemen 5 – maka semua tupel pegawai yang ditempatkan pada departemen 5 harus diubah. Jika tidak, data menjadi tidak konsisten, departemen yang sama akan dipimpin oleh manajer yang berbeda.

Berdasarkan uraian di atas dapat disimpulkan bahwa perlu merancang skema relasi yang tidak memungkinkan penyimpangan penyisipan, penyimpangan penghapusan dan penyimpangan pengubahan terjadi. Jika penyimpangan yang tersebut diatas terjadi maka perhatikanlah dalam pembuatan programnya sehingga program yang mengubah data tersebut bisa bekerja dengan benar.

3.1.3. Isi Yang Bernilai Null Pada Tupel

Untuk menjelaskan topik ini diandaikan ada suatu relasi gemuk (fat relation). Relasi gemuk (fat relation) adalah relasi yang terdiri dari banyak atribut yang dikelompokan dalam satu relasi. Jika suatu atribut pada relasi gemuk tidak digunakan maka atribut itu bernilai null. Hal ini akan menghabiskan tempat penyimpanan dan akan mempersulit pengertian yang dikandung pada atribut tersebut bila dilakukan operasi JOIN. Persoalan lain yang timbul dengan atribut bernilai null adalah operasi COUNT dan SUM tidak bisa dilakukan. Lebih jauh nilai null dapat mengandung banyak pengertian, yaitu :

- Atributnya tidak dipakai pada tupel tersebut.
- Nilai atributnya untuk tupel tersebut tidak diketahui.
- Nilai atributnya ada tetapi belum diisi.

Berdasarkan uraian di atas dapat disimpulkan bahwa penempatan atribut pada skema relasi yang bernilai null harus dihindari. Jika nilai null tidak dapat dihindari maka usahakan hal itu terjadi pada kasus-kasus tertentu dan jangan sampai terjadi pada sebagian besar tupel pada relasi tersebut.

Sebagai contoh : Jika hanya 10% pegawai yang ditempatkan di kantor maka penempatan atribut NO_KANTOR pada relasi KARYAWAN adalah kurang tepat. Lebih tepat jika dibuatkan relasi baru KANTOR_KARYAWAN (NIP_KARYAWAN, NO_KANTOR) yang tupel-tupelnya berisi semua pegawai yang ditempatkan dikantor.

3.1.4. Tupel -Tupel Yang Palsu

Perhatikan dua skema relasi LOKASI_KARYAWAN dan PROJEK1_ KARYAWAN pada gambar 3.3.(a), yang merupakan bentuk lain dari skema relasi PROJEK_KARYAWAN pada gambar 3.2.(b). Makna sebuah tupel pada LOKASI_ KARYAWAN adalah seorang pegawai yang namanya adalah NAMA bekerja pada beberapa proyek yang berlokasi di LOK_PRJ. Makna dari PROJEK1_KARYAWAN adalah seorang pegawai yang bernomor induk NIP bekerja selama JAM_KRJ per minggu pada suatu proyek yang nama, nomor, lokasi adalah NAMA_PRJ, NO_PRJ, LOK_PRJ. Gambar 3.3.(b) memperlihatkan isi dari LOKASI_KARYAWAN dan PROJEK1_KARYAWAN yang data-datanya berasal dari PROJEK_KARYAWAN.

[a]LOKASI_KARYAWAN

NAMA	LOK_PRJ

PROJEK1_KARYAWAN

NIP	NO_PRJ	JAM_KRJ	NAMA_PRJ	LOK_PRJ

[b] LOKASI_KARYAWAN

NAMA	LOK_PRJ
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
---	---
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Stafford
Borg, James E.	Houston

[c] PROJEK1_KARYAWAN

NIP	NO_PRJ	JAM_KRJ	NAMA_PRJ	LOK_PRJ
199901	1	32.5	Product X	Bellaire
199901	2	7.5	Product Y	Sugarland
199905	3	40.0	Product Z	Houston
199906	1	20.0	Product X	Bellaire
199906	2	20.0	Product Y	Sugarland
199902	2	10.0	Product Y	Sugarland
199902	3	10.0	Product Z	Houston
199902	10	10.0	Computerize	Stafford
199903	30	30.0	Newbenefits	Stafford
199903	10	10.0	Computerize	Stafford
199907	10	35.0	Computerize	Stafford
199907	30	5.0	Newbenefits	Stafford
199904	30	20.0	Newbenefits	Stafford
199904	20	15.0	Reorganization	Houston
199908	20	null	Reorganization	Houston

Gambar 3.3. Alternatif bentuk penyajian dari PROJEK_KARYAWAN

PROJEK1_KARYAWAN dan LOKASI_KARYAWAN adalah contoh desain skema relasi yang salah sebab jika pada skema tersebut dilakukan operasi NATURAL-JOIN maka tupel yang dihasilkan lebih banyak daripada asalnya yaitu skema PROJEK1_ KARYAWAN. Tupel-tupel tambahan ini disebut tupel palsu (spurious tupel) karena mereka menyajikan informasi yang salah.

Berdasarkan uraian diatas dapat disimpulkan bahwa rancanglah skema relasi yang bisa di JOIN dengan kondisi yang sama pada atribut-atributnya baik pada kunci utama maupun pada kunci luar sehingga menjamin tidak ada tupel palsu yang dihasilkan.

3.2. KETERGANTUNGAN FUNGSIONAL

Konsep paling penting dalam merancang skema relasi adalah ketergantungan fungsional. Bagian ini akan mendefinisikan secara formal konsep tersebut.

3.2.1. Definisi Ketergantungan Fungsional

Ketergantungan fungsional adalah batasan (constrain) antara dua himpunan atribut suatu basis data. Suatu skema relasi basis data mempunyai n atribut A_1 , A_2 , A_n maka seluruh basis data digambarkan dalam sebuah skema relasi universal R={A_1,

$A_2,..., A_n\}$. Konsep ini digunakan untuk mengembangkan teori formal tentang ketergantungan data.

Ketergantungan Fungsional (KF) ditulis $X \rightarrow Y$, antara dua himpunan atribut X dan Y yang merupakan subset dari R menjelaskan batasan pada tupel-tupel yang mungkin yang membentuk anggota relasi r pada R. Batasan ini menyatakan untuk dua tupel t_1 dan t_2 pada r dimana $t_1[X] = t_2[X]$ pasti juga mempunyai $t_1[Y] = t_2[Y]$. Berarti nilai Y ditentukan oleh nilai X. Dengan kata lain, nilai X sebagi satu-satunya nilai yang menentukan Y. Himpunan atribut X sisi kiri ketergantungan fungsional dan Y disebut sisi kanan ketergantungan fungsional. X secara fungsional menentukan Y pada suatu skema relasi R jika dan hanya jika dua tupel r(R) berlaku nilai X dan berlaku juga nilai Y. Dengan demikian :

- Jika batasan R menyatakan bahwa tidak bisa lebih dari satu tupel dengan nilai X pada anggota relasi r(R) maka X adalah kunci kandidat (candidate key) dari R. Implikasinya adalah $X \rightarrow Y$ adalah subset atribut Y pada R.
- Jika $X \rightarrow Y$ di R, belum tentu $Y \rightarrow X$ pada R

Ketergantungan adalah suatu sifat dari sebuah arti atau semantik dari atribut-atributnya. Pengertian tentang semantik atribut-atribut R digunakan untuk membuat ketergantungan fungsional berlaku untuk semua r pada R

3.2.2. Hukum Inferensi

Berikut ini adalah enam hukum yang dikenal dengan Hukum Inferensi untuk ketergantungan fungsional.

(IR 1) (Hukum Refleksif) jika $X \supseteq Y$, maka $X \rightarrow Y$
Artinya : Menetapkan suatu himpunan selalu menentukan dirinya sendiri

(IR 2) (Hukum Augmentasi) $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
Artinya : Menetapkan bahwa menambah suatu himpunan atribut yang sama pada kedua sisi (kiri dan kanan) ketergantungan fungsional akan menghasilkan ketergantungan fungsional baru yang valid.

(IR 3) (Hukum Transitif) $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$
Artinya : Pada ketergantungan fungsional berlaku sifat transitif.

(IR 4) (Hukum Dekomposisi) $\{X \rightarrow YZ\} \models X \rightarrow Y$

Artinya : Menetapkan bahwa atribut sisi kanan ketergantungan fungsional dapat dihapus. Dengan menggunakan hukum ini suatu KF $X \rightarrow \{A_1, A_2, \ldots, A_n\}$ dapat dipecah menjadi $\{ X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_n\}$

(IR 5) (Hukum Union) $\{X \rightarrow Y, X \rightarrow Z \} \models X \rightarrow YZ$

Artinya : Menetapkan bahwa suatu himpunan KF $\{ X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_n\}$ dapat disatukan menjadi KF $X \rightarrow \{A_1, A_2, \ldots, A_n\}$

(IR 6) (Hukum Pseudotransitif) $\{X \rightarrow Y, WY \rightarrow Z \} \models WX \rightarrow Z$

Masing-masing hukum inferensi dapat dibuktikan dari definisi ketergantungan fungsinal baik dengan bukti langsung atau dengan kontradiksi. Bukti dengan kontradiksi menganggap bahwa hukumnya tidak berlaku dan menunjukan bahwa hal ini tidak mungkin. Berikut ini akan dibuktikan bahwa (IR 1) sampai dengan (IR 3) adalah valid.

Bukti (IR 1)

Disebutkan bahwa $X \supseteq Y$ dan dua tupel t_1 dan t_2 ada di beberapa anggota relasi r pada R dimana $t_1[X] = t_2[X]$. Maka $t_1[Y] = t_2[Y]$ karena $X \supseteq Y$ maka $X \rightarrow Y$ pasti berlaku pada r.

Bukti (IR 2) (Dengan Kontradiksi)

Dianggap bahwa $X \rightarrow Y$ berlaku di setiap anggota relasi r pada R tapi $XZ \rightarrow YZ$ tidak berlaku. Maka pasti ada dua tupel t_1 dan t_2 di r dimana,

1. $t_1[X] = t_2[X]$
2. $t_1[Y] = t_2[Y]$
3. $t_1[XZ] = t_2[XZ]$
4. $t_1[YZ] \neq t_2[YZ]$

Hal ini tidak mungkin sebab dari (1) dan (3) dapat disimpulkan,

5. $t_1[Z] = t_2[Z]$

dari (2) dan (5) dapat disimpulkan,

6. $t_1[YZ] = t_2[YZ]$

yang kontradiksi dengan (4).

Bukti (IR 3)

Dianggap (1) $X \rightarrow Y$ dan (2) $Y \rightarrow Z$ berlaku di setiap anggota relasi r pada R. Maka untuk dua tupel t_1 dan t_2 di r dimana $t_1[X] = t_2[X]$, pasti ada (3) $t_1[Y] = t_2[Y]$ (dari asumsi (1)), dan juga mempunyai (4) $t_1[Z] = t_2[Z]$ (dari (3) dan asumsi (2)) ; maka $X \rightarrow Y$ pasti berlaku di r.

Hukum inferensi (IR 4) sampai (IR 6) dan beberapa hukum inferensi tambahan dapat dibuktikan dengan cara yang sama. Cara lebih mudah untuk membuktikan suatu hukum inferensi valid adalah dengan menggunakan hukum inferensi yang sudah dibuktikan. Sebagai contoh untuk membuktikan (IR 4) sampai (IR 6) menggunakn (IR1) sampai (IR 3) sebagai berikut,

Bukti (IR 4)

1. $X \rightarrow YZ$
2. $YZ \rightarrow Y$
3. $X \rightarrow Y$

Bukti (IR 5)

1. $X \rightarrow Y$
2. $X \rightarrow Z$
3. $X \rightarrow XY$
4. $XY \rightarrow YZ$
5. $X \rightarrow YZ$

Bukti (IR 6)

1. $X \rightarrow Y$
2. $WY \rightarrow Z$
3. $WX \rightarrow WY$
4. $WX \rightarrow WZ$

3.2.3. Closure Suatu Himpunan Atribut

X adalah himpunan atribut. Closure X adalah himpunan atribut yang secara fungsional ditentukan oleh X dalam suatu himpunan ketergantungan fungsional KF. Closure X pada KF ditulis X^+. Algoritma untuk mencari X^+ adalah sebagai berikut,

$X^+ = X$
repeat
    $OldX^+ = X^+$
    for setiap ketergantungan fungsional $Y \rightarrow Z$ pada KF do

if $Y \subseteq X^+$ then $X^+ = X^+ \cup Z$

until ($OldX^+ = X^+$)

Gambar 3.4. Algoritma Closure

Algoritma ini dimulai dengan memberi nilai X^+ sama dengan X. Dengan menggunakan (IR 1) dapat diketahui apakah suatu himpunan atribut secara fungsional tergantung pada X. Dengan menggunakan (IR 3) dan (IR 4) ditambahkan atribut ke X^+, untuk masing-masing ketergantungan fungsional di KF. Setiap ketergantungan fungsional yang ada diperiksa sampai tidak ada atribut lagi yang bisa ditambahkan sebagai contoh perhatikan himpunan ketergantungan fungsional berikut ini,

KF = { NIP → NAMA,

NO_PRJ → {NAMA_PRJ, LOK_PRJ},

{NIP, NO_PRJ} → JAM_KRJ }

dengan menggunakan algoritma di atas dapat dicari closure masing-masing X yang terdapat di KF.

$\{NIP\}^+$ = {NIP,NAMA}

$\{NO\_PRJ\}^+$ = {NO_PRJ, NAMA_PRJ, LOK_PRJ}

$\{NIP, NO\_PRJ\}^+$ = { NIP,NAMA,NO_PRJ, NAMA_PRJ,LOK_PRJ,JAM_KRJ}

3.2.4. Himpunan Ketergantungan Fungsional Yang Ekivalen

Dua himpunan ketergantungan fungsional E dan F disebut ekivalen jika $E^+ = F^+$. Dengan demikian yang dimaksud dengan ekivalen adalah jika setiap ketergantungan fungsional di E bisa diturunkan dari F dan setiap ketergantungan fungsional di F bisa diturunkan dari E. Jadi E ekivalen dengan F jika kondisi E cover F dan F cover E terpenuhi.

F cover E bisa ditentukan dengan cara mencari X^+ yang ada di F untuk setiap ketergantungan fungsional $X \rightarrow Y$ yang ada di E, kemudian memeriksa apakah X^+ mencakup semua atribut di Y. Jika semua Y yang ada di E tercakup maka F cover E. Hal yang sama juga dilakukan untuk menentukan E cover F

3.2.5. Himpunan Ketergantungan Fungsional Minimal

Suatu himpunan ketergantungan fungsional KF disebut minimal jika himpunan tersebut memenuhi kondisi sebagai berikut,

1. Setiap ketergantungan fungsional di KF mempunyai atribut tunggal di sisi kanannya.
2. Ketergantungan yang ada di KF tidak bisa dihapus dan masih mempunyai himpunan ketergantungan fungsional yang ekivalen.
3. Ketergantungan $X \rightarrow A$ di F tidak bisa diganti dengan ketergantungan $Y \rightarrow A$, dimana Y adalah proper subset X dan masih mempunyai himpunan ketergantungan fungsional yang ekivalen.

3.3.NORMALISASI DATA DENGAN MENGGUNAKAN KETERGANTUNGAN FUNGSIONAL

Pada bagian ini akan dibahas proses normalisasi dan 4 (empat) tipe normalisasi untuk skema relasi. Definisi bentuk normal kedua, ketiga dan Boyce Codd yang disajikan di sini dibuat berdasarkan ketergantungan fungsional dan kunci utama (primary key) skema relasinya. Bagian ini juga membahas bagaimana bentuk normal berkembang pada awal mulanya dan pemikiran yang ada di belakangnya.

3.3.1. Pengantar Normalisasi

Proses normalisasi diusulkan pertama kali oleh Codd (1972), dengan cara menguji suatu skema relasi dengan suatu rangkaian ujian untuk menyatakan apakah memenuhi bentuk normal tertentu. Mulanya Codd mengusulkan tiga bentuk normal yang disebut dengan bentuk normal pertama, kedua dan ketiga. Selanjutnya definisi yang lebih tegas daripada bentuk normal ketiga diusulkan oleh Boyce dan Codd dan disebut bentuk normal Boyce Codd. Semua bentuk normal ini berdasarkan ketergantungan fungsional antar atribut-atributnya. Selanjutnya bentuk normal ke empat (), dan bentuk normal ke lima diajukan berdasarkan konsep ketergantungan fungsional bernilai banyak dan sekutu (JOIN).

Normalisasi data bisa dilihat sebagai proses membagi skema relasi, selama skema relasi yang dibagi belum memuaskan maka skema relasi tersebut dibagi lagi

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH – NOPEMBER

menjadi skema relasi yang lebih kecil dengan syarat skema relasi tersebut masih mempunyai sifat-sifat yang dibutuhkan. Tujuan sebenarnya dari normalisasi adalah menjamin penyimpangan yang dibicarakan pada sub bab 3.1.2. tidak terjadi.

Bentuk normal membantu seorang perancang basis data dalam merancang basis data yang benar karena menyediakan,

- Kerangka kerja formal untuk menganalisa skema relasi berdasarkan kuncinya dan ketergantungan fungsional antara atribut-atributnya.
- Serangkaian pengujian yang bisa dilakukan pada suatu skema relasi sehingga suatu basis data relasional bisa dinormalkan hingga beberapa tingkat. Jika suatu ujian gagal maka relasi yang gagal tersebut harus dibagi menjadi relasi yang sesuai dengan test normalisasi.

Bentuk normal jika dibuat tanpa mempertimbangkan faktor lainnya tidak menjamin menjadi desain basis data yang baik. Pada umumnya amatlah tidak efisien jika menguji secara terpisah masing-masing skema relasi dalam suatu basis data. Sebab proses normalisasi melalui pembagian tersebut harus memenuhi sifat-sifat tambahan yang harus ada pada skema relasi yaitu,

- Kesatuan Sekutu (Lossless Join) yang menjamin masalah tupel-tupel palsu yang dibahas pada sub bab 3.1.4. tidak terjadi.
- Keutuhan Ketergantungan (Dependency Preservation) yang menjamin semua ketergantungan fungsional tetap ada pada semua skema relasi yang dihasilkan.

Perlu diingat bentuk normal yang dibahas di sini adalah bentuk normal pertama sampai dengan bentuk normal Boyce Codd yang prosesnya berdasarkan ketergantungan fungsional dan kuncinya.

Hal penting lainnya yang perlu diperhatikan adalah seorang perancang basis data tidak harus menormalkan suatu basis data sampai tingkat tertinggi karena bisa jadi relasi menjadi tidak normal seperti dibahas pada sub bab 3.1.2.

Atribut suatu skema relasi disebut **atribut prima** jika atribut itu menjadi anggota sembarang kunci R. Suatu atribut disebut **atribut non prima** jika atribut tersebut bukan atribut prima yaitu bukan anggota sembarang kunci. Pada contoh gambar 3.1. NIP dan NO_PRJ pada skema relasi pada skema relasi BEKERJA_DI

adalah atribut prima karena merupakan kunci R. sedangkan atribut lainnya pada skema BEKERJA_DI adalah non prima .

Berikut ini akan dibahas bentuk normal pertama (BN1), bentuk normal kedua (BN2), bentuk normal ketiga BN3 dan bentuk normal Boyce Codd (BNBC) yang diusulkan oleh Codd (1972) secara berurutan untuk menghasilkan bentuk normal yang baik.

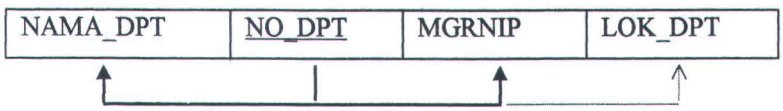
3.3.2. Bentuk Normal Pertama (BN1)

Bentuk normal pertama sekarang dianggap sebagai bagian definisi formal suatu relasi. Suatu skema relasi memenuhi bentuk normal pertama jika pada relasi tersebut tidak terdapat atribut bernilai banyak, atribut komposit dan kombinasinya. Domain atributnya bernilai tunggal, sederhana dan tidak bisa dibagi-bagi. Jadi bentuk normal pertama tidak membolehkan adanya himpunan nilai-nilai, suatu tupel nilai-nilai, atau kombinasi keduanya sebagai nilai atribut untuk suatu tupel dengan kata lain bentuk normal pertama tidak mengizinkan adanya relasi dalam relasi. Nilai atribut yang diizinkan adalah nilai tunggal atau nilai yang tidak dapat dibagi.

Sebagai contoh perhatikan skema relasi DEPARTEMEN pada gambar 3.1. dengan kunci utamanya adalah NO_DPT. Misalkan atribut LOK_DPT ditambahkan pada relasi tersebut dengan anggapan masing-masing departemen bisa mempunyai beberapa lokasi. Dengan demikian dapat dilihat bahwa skema ini tidak memenuhi bentuk pertama karena LOK_DPT bukan atribut tunggal seperti yang diperlihatkan pada gambar 3.5.(b) pada tupel pertama. Pada kasus ini atribut LOK_DPT dapat dilihat dalam dua cara,

1. Domain LOK_DPT berisi nilai tunggal tetapi beberapa tupel bisa mempunyai himpunan nilai atribut yang sama, dalam hal ini NO_DPT $\times\!\!\rightarrow$ LOK_DPT.
2. Domain LOK_DPT berisi himpunan nilai-nilai tidak tunggal, dalam hal ini NO_DPT $\rightarrow$ LOK_DPT tapi tiap himpunan dianggap sebagai anggota tunggal dari domain atribut.

[a] DEPARTEMEN



[b] DEPARTEMEN

NAMA_DPT	NO_DPT	MGRNIP	LOK_DPT
Research	5	199902	{Bellaire, Sugarland, Houston}
Administration	4	199904	{Stafford}
HeadQuarters	1	199908	{Houston}

[b] DEPARTEMEN

NAMA_DPT	NO_DPT	MGRNIP	LOK_DPT
Research	5	199902	Bellaire
Research	5	199902	Sugarland
Research	5	199902	Houston
Administration	4	199904	Stafford
HeadQuarters	1	199908	Houston

Gambar 3.5. Contoh normalisasi bentuk pertama
(a) skema relasi tidak memenuhi bentuk normal pertama
(b) Contoh anggota relasi
(c) Relasi bentuk normal pertama dengan pengulangan

Kedua contoh relasi DEPARTEMEN pada gambar 3.5. belum memenuhi bentuk normal pertama, sebenarnya keduanya belum memenuhi syarat sebagai relasi. Untuk menormalkan relasi DEPARTEMEN menjadi relasi bentuk pertama maka atributnya dibagi menjadi dua relasi yaitu relasi DEPARTEMEN dan LOKASI_DEPARTEMEN seperti yang diperlihatkan pada gambar 3.1. yaitu dengan mengeluarkan atribut LOK_DPT dan menempatkannya dalam relasi lain yaitu LOKASI_DEPARTEMEN bersama kunci utama DEPARTEMEN yaitu NO_DPT. Kunci utama relasi ini adalah gabungan antara {NO_DPT, LOK_DPT} yang diperlihatkan pada gambar 3.1.

Cara kedua untuk menormalkan menjadi bentuk pertama adalah dengan menjadikan setiap nilai yang terdapat pada LOK_DPT satu tupel baru pada relasi DEPARTEMEN. Syaratnya adalah kunci utamanya adalah gabungan {NO_DPT, LOK_DPT} dan perulangan (redudancy) terjadi. Solusi pertama lebih baik karena tidak mengakibatkan perulangan. Sebenarnya jika solusi kedua yang dipilih maka relasi akan dipecah lagi.

Bentuk normal pertama juga tidak membolehkan atribut-atribut komposit yang terdiri dari beberapa nilai. Relasi ini disebut relasi beranak (nested) karena tiap-tiap tupel bisa mempunyai relasi lagi didalamnya. Supaya lebih jelas lihatlah relasi PROJEK_ KARYAWAN pada gambar 3.5. Tiap-tiap tupel mewakili entiti KARYAWAN dan relasi PROJEK {NO_PRJ, JAM_KRJ}. Skema relasi dari PROJEK_KARYAWAN bisa ditulis sebagai berikut :

PROJEK_KARYAWAN(NIP,NAMA,{PROJEK(NO_PRJ, JAM_KRJ)})

Tanda kurung {} menyatakan atribut PROJEK bernilai banyak dengan atribut-atributnya ditulis diantara tanda kurung (). Perlu diketahui bahwa riset akhir-akhir ini sedang berusaha menjadikan relasi beranak bisa menjadi bagian model relasional secara formal.

[a] PROJEK_KARYAWAN

NIP	NAMA	PROJEK	
		NO_PRJ	JAM_KRJ

[b] PROJEK_KARYAWAN

NIP	NAMA	NO_PRJ	JAM_PRJ
199901	Smith, John B.	1	32.5
		2	7.5
199905	Narayan, Rhames K.	3	40.0
199906	English, Joyce A.	1	20.0
199902	Wong, Franklin T.	2	20.0
		2	10.0
		3	10.0
		10	10.0
		20	10.0
199903	Zelaya, Alicia J.	30	30.0
		10	10.0
199907	Jabbar, Ahmad V.	10	35.0
		30	5.0
199904	Wallace, Jennifer S.	30	20.0
		20	15.0
199908	Borg, James E.	20	Null

[c] PROJEK1_KARYAWAN

NIP	NAMA

PROJEK2_KARYAWAN

NIP	NO_PRJ	JAM_KRJ

Gambar 3.6. Contoh relasi beranak yang dinormalkan ke bentuk pertama
(a) Relasi beranak PROJEK_KARYAWAN
(b) Relasi PROJEK_KARYAWAN bentuk normal pertama dengan pengulangan
(c) Relasi bentuk normal pertama yang benar

NIP adalah kunci utama relasi PROJEK_KARYAWAN yang terdapat pada gambar 3.6.(a) dan 3.6.(b). Sementara NO_PRJ adalah kunci sebagian (partial key) pada tiap-tiap anak relasi sehingga masing-masing tupel mempunyai NO_PRJ yang unik. Untuk menormalkan relasi ini menjadi bentuk normal pertama maka perlu mengeluarkan atribut-atribut relasi anak menjadi relasi dan membuat kunci utama baru di situ. Kunci utama relasi baru itu adalah gabungan antara kunci sebagian dan kunci utama relasi semula. Pembagian dan pembuatan kunci utama baru menghasilkan skema yang diperlihatkan pada gambar 3.6. Prosedur seperti ini bisa dilakukan secara rekursif untuk relasi yang bertingkat banyak (multilevel nesting) untuk memecah menjadi relasi bentuk normal pertama.

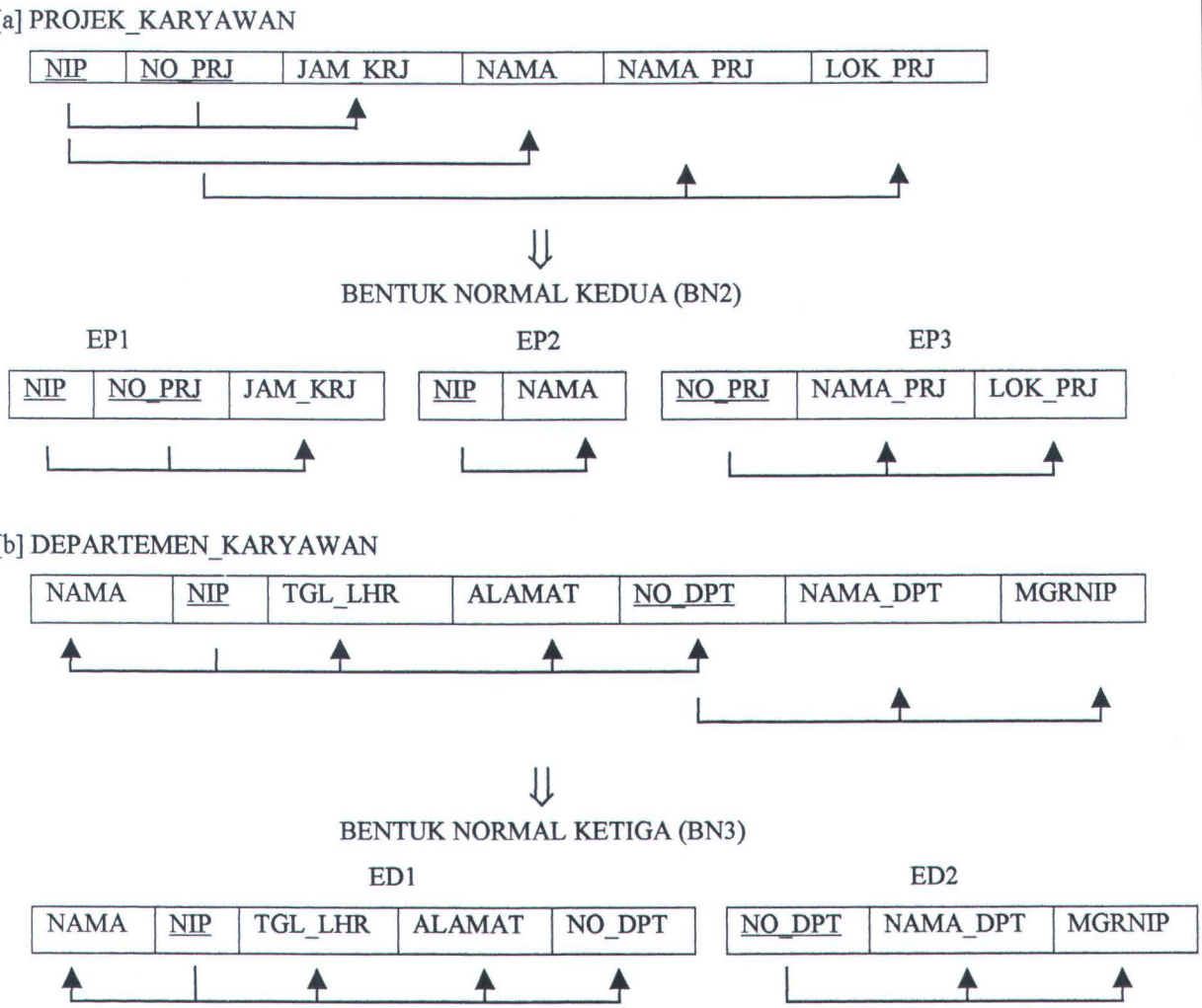
3.3.3. Bentuk Normal Kedua (BN2)

Bentuk normal kedua adalah berdasarkan konsep ketergantungan fungsional penuh (Full Functional Dependent). Sebuah ketergantungan fungsional $X \rightarrow Y$ disebut ketergantungan fungsional penuh jika penghapusan atribut sembarang A dari X, menyebabkan ketergantungan tidak berlaku lagi. Secara formal ditulis sebagai berikut, $A \in X$, $(X - \{A\}) \not\rightarrow Y$. Ketergantungan fungsional $X \rightarrow Y$ disebut ketergantungan fungsional sebagian sebagian (partial dependency) jika atribut sembarang $A \in X$ bisa di hapus dari X dan ketergantungan masih berlaku, secara formal ketergantungan sebagian ditulis $A \in X$, $(X - \{A\}) \rightarrow Y$. Pada gambar 3.2.(b) {NIP, NO_PRJ} → JAM_KRJ adalah ketergantungan penuh karena NIP → JAM_KRJ dan NO_PRJ → JAM_KRJ tidak berlaku. Sebaliknya {NIP, NO_PRJ}→NAMA adalah ketergantungan sebagian karena ketergantungan fungsional NIP→ NAMA berlaku.

Skema relasi R memenuhi bentuk normal kedua jika setiap atribut non prima A mempunyai ketergantungan fungsional penuh dengan kunci utama R. Relasi PROJEK_ KARYAWAN pada gambar 3.2. (b) memenuhi bentuk normal pertama tapi tidak memenuhi bentuk normal kedua. Atribut non prima NAMA melanggar ketentuan bentuk normal kedua karena terdapat ketergantungan fungsional kf2, demikian pula atribut non prima LOK_PRJ. NAMA_PRJ pada LOK_PRJ juga melanggar ketentuan bentuk normal kedua karena terdapat ketergantungan fungsional

kf3. Ketergantungan fungsional kf2, kf3 mengakibatkan NAMA, NAMA_PRJ pada LOK_PRJ mempunyai ketergantungan fungsional sebagian terhadap kunci utama {NIP, NO_PRJ}. Jadi relasi ini tidak memenuhi bentuk normal kedua.

Jika suatu skema relasi tidak memenuhi bentuk normal kedua maka bisa dinormalkan menjadi sejumlah relasi bentuk normal kedua dimana atribut non primanya hanya berhubungan dengan kunci utama yang mempunyai ketergantungan penuh. Ketergantungan fungsional kf1, kf2, kf3 menyebabkan PROJEK_KARYAWAN dibagi menjadi tiga skema relasi yaitu EP1, EP2, EP3 yang ditunjukkan oleh gambar 3.7.(a) yang mana masing-masing relasi memenuhi bentuk normal kedua.



Gambar 3.7. Proses Normalisasi

3.3.4. Bentuk Normal Ketiga (BN3)

Bentuk normal ketiga berdasarkan konsep ketergantungan transitif. Ketergantungan fungsional X → Y pada skema relasi R disebut ketergantungan transitif jika ada himpunan atribut Z yang bukan himpunan bagian sembarang kunci dan X → Z dan Z → Y berlaku. Contoh ketergantungan transitif terdapat pada gambar 3.2.(a) dalam relasi DEPARTEMEN_KARYAWAN dimana Ketergantungan NIP → MGRNIP adalah ketergantungan transitif. Hal ini terjadi karena ketergantungan NIP → NO_DPT dan NO_DPT → MGRNIP berlaku dan NO_DPT bukan merupakan himpunan bagian dari kunci DEPARTEMEN_KARYAWAN.

Berdasarkan definisi Codd, skema relasi R memenuhi bentuk normal ketiga, jika dia memenuhi bentuk normal kedua dan tidak ada atribut non prima yang transitif terhadap kunci utama. Skema relasi DEPARTEMEN_KARYAWAN pada gambar 3.2.(a) memenuhi bentuk normal kedua karena ada ketergantungan sebagian terhadap kunci yang ada, tapi relasi ini tidak berada pada bentuk normal ketiga karena ada ketergantungan transitif pada MGRNIP melalui NO_DPT. Relasi ini bisa dinormalkan dengan membaginya menjadi dua relasi bentuk normal ketiga yaitu ED1 dan ED2 yang diperlihatkan pada gambar 3.7.

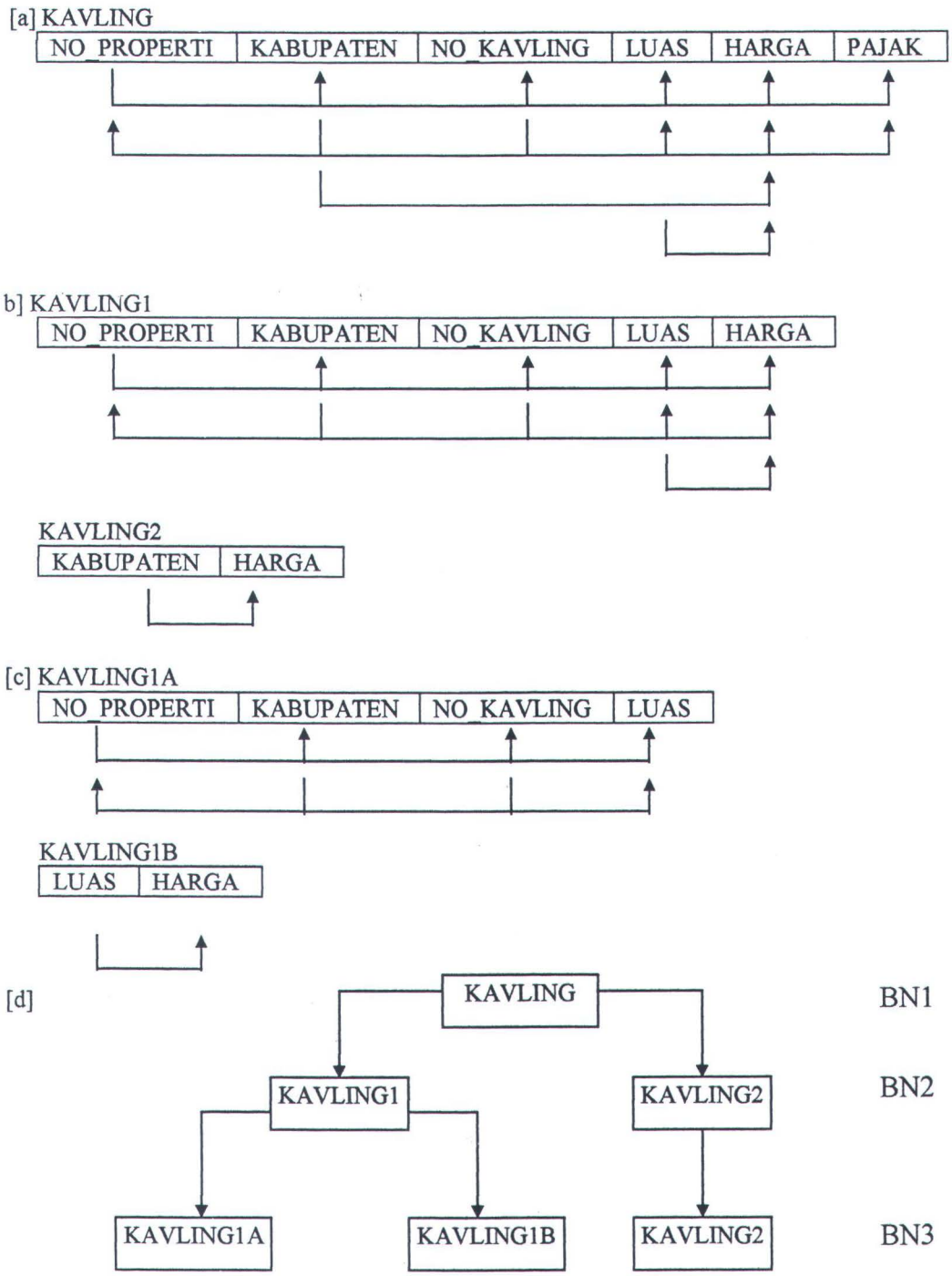
Secara intuitif ED1 dan ED2 mewakili fakta entiti sendiri-sendiri yaitu entiti KARYAWAN dan entiti DEPARTEMEN. Operasi natural join pada ED1 dan ED2 akan menghasilkan relasi DEPARTEMEN_KARYAWAN tanpa menghasilkan tupel-tupel palsu.

3.3.5. Definisi Umum Bentuk Normal Kedua

Skema relasi R berada dalam bentuk normal kedua jika setiap atribut non prima A tidak mempunyai ketergantungan sebagian terhadap sembarang kunci. Perhatikan skema relasi KAVLING yang diperlihatkan pada gambar 3.8. yang menggambarkan petak-petak tanah yang dijual di beberapa kabupaten.

Misalkan ada dua kunci kandidat yaitu NO_PROPERTI dan {KABUPATEN, NO_KAVLING} dimana nilai NO_KAVLING unik untuk masing-masing kabupaten tapi nilai NO_PROPERTI unik untuk lintas kabupaten. Berdasarkan dua kunci

kandidat ini dibuat dua fungsi ketergantungan yaitu kf1 dan kf2 seperti yang terdapat dalam gambar 3.8.(a).



Gambar 3.8. Normalisasi Ke Bentuk Normal Kedua dan Ketiga

MILIK PERPUSTAKA
INSTITUT TEKNOLOG
SEPULUH – NOPEMBER

Setelah itu dibuat dua ketergantungan fungsional lainnya yaitu,

kf3 : KABUPATEN → PAJAK

kf4 : LUAS → HARGA

Jika dituangkan dalam kata-kata maka ketergantungan fungsional kf3 menyatakan bahwa nilai pajak tergantung pada tiap-tiap kabupaten. Sementara ketergantungan fungsional kf4 menyatakan harga tanah ditentukan oleh luasnya bukan kabupaten dimana tanah itu berada.

Skema relasi KAVLING tidak memenuhi definisi umum bentuk normal kedua karena PAJAK mempunyai ketergantungan sebagian terhadap kunci kandidat {KABUPATEN, NO_KAVLING} melalui ketergantungan fungsional kf3. Untuk menormalkan skema relasi KAVLING menjadi bentuk normal kedua maka relasi tersebut harus dibagi menjadi KAVLING1 dan KAVLING2 seperti yang diperlihatkan pada gambar 3.8.(b). Relasi KAVLING2 disusun dengan mengeluarkan atribut PAJAK yang membuat relasi KAVLING menjadi tidak memenuhi bentuk normal kedua dan menempatkannya bersama KABUPATEN.

3.3.5. Definisi Umum Bentuk Normal Ketiga

Skema relasi R memenuhi bentuk normal ketiga jika ketergantungan fungsional X → A berlaku dimana,

(a) X adalah kunci super (superkey) R atau

(b) A adalah atribut prima.

Berdasarkan definisi ini maka KAVLING2 sudah memenuhi bentuk normal ketiga. Sebaliknya ketergantungan fungsional kf4 pada relasi KAVLING1 masih melanggar bentuk normal ketiga karena LUAS bukan kunci super KAVLING1 dan HARGA bukan atribut prima. Untuk menormalkan relasi ini dalam bentuk normal ketiga dilakukan dengan mengeluarkan atribut HARGA dan menempatkannya dengan LUAS pada relasi KAVLING1B. Sehingga KAVLING1A, KAVLING1B, KAVLING2 memenuhi syarat bentuk normal ketiga. Dua hal yang perlu diperhatikan pada definisi umum ini adalah :

- Definisi ini berlaku tanpa harus melalui bentuk normal kedua (BN2). Jika ketentuan bentuk normal ketiga diberlakukan pada skema relasi KAVLING

dengan ketergantungan fungsional kf1 sampai dengan kf4 maka dapat dilihat bahwa kf3 dan kf4 melanggar ketentuan bentuk normal ketiga sehingga skema relasi KAVLING bisa langsung dibagi menjadi KAVLING1A, KAVLING1B, KAVLING2 langsung.

- KAVLING1 melanggar ketentuan bentuk normal ketiga karena HARGA mempunyai ketergantungan transitif terhadap kunci kandidat KAVLING1 melalui atribut non prima LUAS

3.3.6. Bentuk Normal Boyce Codd

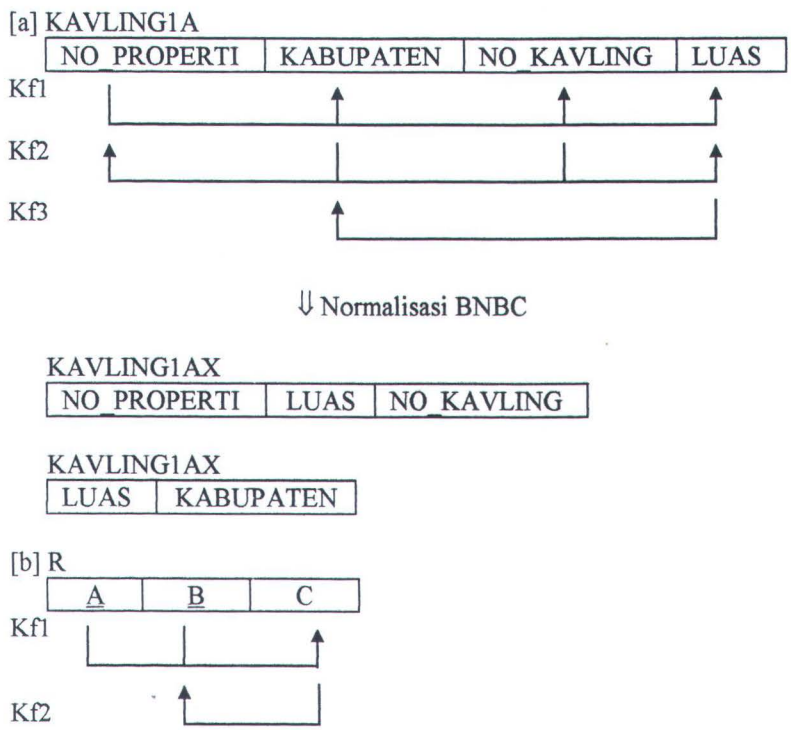
Bentuk normal Boyce Codd mempunyai ketentuan yang lebih ketat daripada bentuk normal ketiga. Oleh sebab itu setiap relasi yang memenuhi ketentuan bentuk normal Boyce Codd pasti juga memenuhi ketentuan bentuk normal ketiga. Secara intuitif bentuk normal yang lebih ketat daripada bentuk normal ketiga memang dibutuhkan. Hal ini untuk menyelesaikan permasalahan sebagai berikut, diketahui skema relasi KAVLING yang mempunyai empat ketergantungan fungsional kf1 sampai dengan kf4. Diketahui pula ternyata ada ribuan kavling yang berasal dari dua kabupaten berbeda yaitu kabuten Sidarjo dan kabupaten Malang. Besar kavling di kabupaten Sidoarjo adalah 0.5 , 0.6 , 0.7 , 0.8 , 0.9 , 1.0 are dan besar kavling di kabupaten Malang adalah 1.1 , 1.2 , 1.3 , ... , 1.9 , 2.0 are. Dengan situasi seperti ini maka dibutuhkan ketergantungan fungsional kf5 : LUAS $\rightarrow$ KABUPATEN Jika ketergantungan fungsional ini ditambahkan pada skema relasi maka skema relasi tetap memenuhi bentuk normal ketiga karena KABUPATEN adalah atribut prima.

Misalkan LUAS dan KABUPATEN yang terdapat pada kf5 terdiri dari 16 tupel pada relasi R (LUAS, KABUPATEN) maka hanya ada 16 kemungkinan nilai LUAS. Penyajian seperti ini akan mengurangi perulangan informasi yang sama di ribuan tupel pada relasi KAVLING1A. Bentuk Normal Boyce Codd (BNBC) adalah bentuk normal yang lebih kuat yang mengharuskan KAVLING1A dipecah lagi

Definisi Boyce Codd sedikit berbeda dengan definisi bentuk normal ketiga. Skema R memenuhi Bentuk Normal Boyce Codd (BNBC) jika ketergantungan fungsional X $\rightarrow$ A berlaku di R dan X adalah kunci supernya R. Sedangkan di bentuk normal ketiga mengizinkan A sebagai atribut prima jika X bukan kunci supernya.

Dalam contoh yang dijelaskan sebelumnya kf5 melanggar ketentuan bentuk normal Boyce Codd di relasi KAVLING1A karena LUAS bukan kunci supernya. Sebaliknya kf5 memenuhi bentuk normal ketiga karena KABUPATEN adalah atribut prima. Supaya relasi ini memenuhi bentuk normal Boyce Codd maka relasi ini perlu dipecah menjadi relasi KAVLING1AX dan KAVLING1AY yang diperlihatkan pada gambar 3.9.(a).

Pada prakteknya, hampir semua skema relasi yang memenuhi bentuk normal ketiga juga memenuhi bentuk normal Boyce Codd, hanya jika ketergantungan fungsional $X \rightarrow A$ yang berlaku di R dengan X bukan kunci supernya dan A adalah atribut prima maka R hanya memenuhi ketentuan bentuk normal ketiga. Sebaiknya desain skema relasi memenuhi bentuk normal Boyce Codd tetapi jika tidak memungkinkan bentuk normal ketiga sudah cukup. Tidak dianjurkan mendesain skema relasi yang hanya memenuhi ketentuan bentuk normal pertama dan kedua karena bentuk normal ini hanya batu loncatan ke bentuk normal ketiga dan Boyce Codd.



Gambar 3.9. Bentuk Normal Boyce Codd

(a) Normalisasi Bentuk Normal Boyce Codd Yang Menghapus Ketergantungan kf2
(b) Relasi R Yang Memenuhi Bentuk Normal Ketiga Tetapi Tidak Memenuhi Bentuk Normal Boyce Codd

BAB IV

PERANCANGAN ALGORITMA NORMALISASI

BAB IV
PERANCANGAN ALGORITMA NORMALISASI

Ada dua teknik dalam merancang skema relasi basis data. Teknik pertama adalah merancang konsep skema dengan menggunakan model data tingkat tinggi seperti model ER dan kemudian memetakan konsep skema tersebut kedalam himpunan relasi dengan menggunakan prosedur pemetaan. Cara ini disebut cara mendesain dari atas ke bawah (top down). Dengan cara ini prinsip-prinsip normalisasi secara informal yang dibicarakan pada bab 3 bisa diterapkan yaitu menghindari ketergantungan sebagian dan ketergantungan transitif pada skema relasinya. Prinsip tersebut bisa dilakukan setelah proses pemetaan dilakukan dan pada saat skema relasi telah dihasilkan.

Teknik kedua adalah teknik dengan pendekatan yang lebih murni. Teknik ini melihat desain skema relasi dalam lingkup ketergantungan fungsional yang berlaku pada atribut-atribut basis datanya. Teknik ini disebut sintesa relasional, karena skema relasi yang memenuhi bentuk normal ketiga dan bentuk normal Boyce Codd dihasilkan dengan cara menggabungkan bersama atribut-atribut yang berhubungan. Setiap skema relasi harus mewakili gabungan atribut-atribut yang logis dan memenuhi ukuran yang berlaku dalam menentukan layak tidaknya suatu relasi yang normal.

Pada bab 3 dibahas beberapa cara dalam mengukur kelayakan suatu relasi berdasarkan kunci dan ketergantungan fungsional – yang disebut Bentuk Normal Boyce Codd (BNBC) atau Bentuk Normal ke 3 (BN3). Selama proses normalisasi, pembagian relasi yang tidak memenuhi bentuk tertentu dilakukan terus sampai relasi tersebut memenuhi bentuk normal yang diinginkan. Kasus ekstrem pada proses desain seperti ini disebut pembagian ketat (strict decomposition). Dengan pendekatan seperti ini sintesa skema relasi besar yang disebut relasi semesta (universal relation) dimana semua atribut dikumpulkan dilakukan. Pembagian atribut dilakukan terus sampai tidak ada yang bisa dibagi lagi.

Pada bagian ini akan dipelajari beberapa algoritma yang menggunakan ketergantungan fungsional untuk mendesain basis data relasional. Di sini akan

dibahas pula keutuhan ketergantungan (dependency preservation) dan kesatuan sekutu (lossless joins) yang digunakan pada desain algoritma untuk mendapatkan pengelompokkan atribut yang baik. Di sini akan dibuktikan bahwa bentuk normal menjadi salah jika dilihat per skema.

4.1. DEKOMPOSISI ATRIBUT DAN BENTUK NORMAL YANG SALAH

Pembahasan algoritma untuk mendesain basis data relasional dimulai dari skema relasi semesta $R = \{A_1, A_2, \ldots, A_n\}$ yang memuat semua atribut basis data. Secara implisit semua atribut dianggap mempunyai nama yang unik. Ketergantungan fungsional yang berlaku pada atribut-atribut tersebut dibuat oleh perancang basis data dan merupakan masukan bagi algoritma. Berdasarkan ketergantungan fungsional yang ada, algoritma mengelompokan atribut-atribut menjadi skema relasi $D = \{R_1, R_2, \ldots, R_n\}$ yang akhirnya merupakan skema relasi basis data. D disebut dekomposisi skema relasi R.

Kondisi yang harus terpenuhi dalam mengelompokan R menjadi D adalah semua atribut yang ada di R paling sedikit muncul disatu skema relasi R_i sehingga tidak ada atribut yang hilang. Secara formal hal ini ditulis sebagai berikut,

$$\bigcup_{i=1}^{m} R_i = R$$

Kondisi ini disebut keutuhan atribut (attribute preservation) pada dekomposisi. Kondisi lain yang harus terpenuhi adalah setiap relasi R_i yang ada di dekomposisi D memenuhi bentuk normal Boyce Codd atau bentuk normal ketiga. Kondisi ini tidak akan tercapai jika dalam proses pembagiannya hanya mempertimbangkan atribut kelompoknya sendiri. Supaya lebih jelas perhatikan relasi LOKASI_KARYAWAN pada gambar 3.3. yang mana relasi tersebut memenuhi bentuk normal ketiga dan bentuk normal Boyce Codd. Perlu diketahui setiap relasi yang terdiri dari dua atribut maka relasi tersebut memenuhi bentuk normal Boyce Codd. Meskipun relasi LOKASI_KARYAWAN memenuhi bentuk normal Boyce Codd, relasi ini masih menghasilkan tupel-tupel palsu jika dilakukan operasi join dengan relasi PROJEK1_KARYAWAN (yang tidak memenuhi bentuk normal Boyce Codd).

Kejadian ini bisa dilihat pada gambar 4.1. Dengan demikian relasi LOKASI_KARYAWAN adalah relasi yang salah karena konvolusi semantik antara atribut LOK_PRJ yang mencantumkan lokasi suatu proyek dengan karyawan yang bekerja di suatu proyek selalu menghasilkan satu tupel. Melakukan operasi join antara relasi LOKASI_KARYAWAN dengan relasi PROJEK (yang memenuhi bentuk normal Boyce Codd) juga menghasilkan tupel-tupel palsu. Oleh sebab itu untuk mencegah hal ini dibutuhkan persyaratan tambahan untuk melakukan dekomposisi di D.

NIP	NO_PRJ	JAM_KRJ	NAMA_PRJ	LOK_PRJ	NAMA
199901	1	32.5	Produk X	Bellaire	Smith, John B.
*199901	1	32.5	Produk X	Bellaire	English, Joyce A.
199901	2	7.5	Produk Y	Sugarland	Smith, John B.
*199901	2	7.5	Produk Y	Sugarland	English, Joyce A.
*199901	2	7.5	Produk Y	Sugarland	Wong, Franklin T.
199905	3	40.0	Produk Z	Houston	Narayan, Ramesh K.
*199905	3	40.0	Produk Z	Houston	Wong, Franklin T.
*199906	1	20.0	Produk X	Bellaire	Smith, John B.
199906	1	20.0	Produk X	Bellaire	English, Joyce A.
*199906	2	20.0	Produk Y	Sugarland	Smith, John B.
199906	2	20.0	Produk Y	Sugarland	English, Joyce A.
*199906	2	20.0	Produk Y	Sugarland	Wong, Franklin T.
*199906	2	10.0	Produk Y	Sugarland	Smith, John B.
*199902	2	10.0	Produk Y	Sugarland	English, Joyce A.
199902	2	10.0	Produk Y	Sugarland	Wong, Franklin T.
*199902	3	10.0	Produk Z	Houston	Narayan, Ramesh K.
199902	3	10.0	Produk Z	Houston	Wong, Franklin T.
199902	10	10.0	Komputerisasi	Stafford	Wong, Franklin T.
*199902	20	10.0	Reorganisasi	Houston	Narayan, Ramesh K.
199902	20	10.0	Reorganisasi	Houston	Wong, Franklin T.

Gambar 4.1. Contoh Operasi Normal Join ke PROJEK_KARYAWAN Dan LOKASI_KARYAWAN

4.2. DEKOMPOSISI DAN KEUTUHAN KETERGANTUNGAN

Secara informal suatu kondisi dikatakan memenuhi kriteria keutuhan ketergantungan jika setiap ketergantungan fungsional $X \rightarrow Y$ di F minimal ditemukan pada satu skema R_i dimana R_i merupakan anggota D yaitu himpunan skema relasi yang sudah dipecah-pecah. Setiap ketergantungan yang ada harus dipertahankan karena setiap ketergantungan di F mewakili batasan pada basis data. Jika ada satu ketergantungan tidak terdapat pada suatu relasi R_i maka tidak perlu memaksa untuk mencari ketergantungan tersebut sampai ketemu. Karena bisa jadi ketergantungan

tersebut baru terlihat jika operasi join pada dua atau lebih skema relasi dilakukan. Tapi cara seperti ini tidak efisien dan pada prakteknya tidak dilakukan.

Sebaiknya suatu skema relasi tidak diwakili oleh satu ketergantungan fungsional saja, akan lebih efisien jika skema relasi tersebut diwakili oleh gabungan beberapa ketergantungan fungsional. Untuk lebih memperjelas pengertian ketergantungan fungsional berikut ini dijelaskan konsep tersebut secara formal.Diketahui himpunan ketergantungan fungsional F pada R, proyeksi F terhadap R_i yang ditulis dengan $\pi_F(R_i)$ dimana R_i merupakan himpunan bagian R, adalah himpunan ketergantungan $X \rightarrow Y$ pada F^+ yang atribut $X \cup Y$ terdapat R_i. Proyeksi F pada masing-masing skema relasi R_i pada D yaitu himpunan relasi yang telah dipecah adalah himpunan ketergantungan fungsional pada F^+dimana atribut-atribut yang terletak disisi kanan dan kiri ada di R_i. Dengan demikian dekomposisi D= R_1, R_2, . . . , R_n } dari R mempunyai ketergantungan yang utuh terhadap F jika gabungan proyeksi F untuk setiap R_i di D setara dengan F. Secara matematik ditulis sebagai berikut $((\pi_F(R_1)) \cup \ldots \cup (\pi_F(R_m)))^+ = F^+$.

Jika pengelompokan atribut tidak memenuhi ketentuan keutuhan ketergantungan berarti ada ketergantungan yang hilang. Contoh pengelompokan atribut yang tidak memenuhi ketentuan keutuhan ketergantungan diperlihatkan pada gambar 3.9.(a). yang mana ketergantungan fungsional kf2 hilang ketika KAVLING1A dipecah menjadi {KAVLING1AX, KAVLING1AY}. Adapun dekomposisi yang terdapat pada gambar 3.8. memenuhi hukum keutuhan ketergantungan. Supaya setiap dekomposisi selalu memenuhi ketentuan keutuhan ketergantungan maka pengelompokannya harus mengikuti aturan yang ditentukan oleh suatu algoritma. Algoritma ini disebut algoritma dekomposisi skema relasi menjadi bentuk ketiga seperti yang ditunjukan oleh gambar 4.2. berikut ini,

1. *Cari cover minimal G di F*
2. *Untuk setiap X di ketergantungan fungsional G*
   buatlah skema relasi (X UNION A_1 UNION A_2 . . . . UNION A_n) dengan nama D dimana $X \rightarrow A_1$, $X \rightarrow A_2$, ... , $X \rightarrow A_n$ adalah ketergantungan fungsional di G dengan X merupakan sisi kirinya.

3. *Satukan atribut-atribut yang tersisa dalam sebuah skema relasi untuk menjamin kondisi keutuhan atribut.*

Gambar 4.2. Algoritma Dekomposisi Skema Relasi Menjadi Bentuk Ketiga

1. $G := F$
2. *Ubahlah setiap ketergantungan fungsional di G yang berbentuk* $X \rightarrow A_1, A_2, \ldots,$ A_n *menjadi n ketergantungan fungsional* $X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_n$.
3. *Untuk setiap ketergantungan fungsional* $X \rightarrow Y$ *di G*

   Untuk setiap atribut B yang merupakan elemen X

   Hitunglah X^+ *berdasarkan himpunan ketergantungan fungsional*

   $\{ (G - (X \rightarrow A)$ *UNION* $((X-B) \rightarrow A) \}$

   Jika X^+ *mengandung A, maka gantilah* $X \rightarrow A$ *dengan* $(X-B) \rightarrow A$ *di G*
4. *Untuk Setiap ketergantungan fungsional yang tersisa* $X \rightarrow A$ *di G*

   Hitunglah X^+ *berdasarkan himpunan ketergantungan fungsional* $G - (X \rightarrow A)$

   Jika X^+ *mengandung A, maka hapuslah* $X \rightarrow A$ *di G*

Gambar 4.2.(a) Algoritma Mencari Cover Minimal G yang berasal dari F

Algoritma ini pasti menghasilkan ketergantungan fungsional yang memenuhi ketentuan keutuhan ketergantungan tapi belum tentu memenuhi ketentuan kesatuan sekutu. Langkah pertama dari gambar 4.1. adalah mencari cover minimal G dari F. Algoritma untuk mencari cover minimal dijabarkan pada gambar 4.2.(a) dibawahnya. Minimal cover G dari F yang dimaksud di sini adalah himpunan ketergantungan fungsional minimal yang ekivalen dengan F. Algoritma mencari cover minimal dibuat berdasarkan tiga kondisi yang harus dipenuhi oleh himpunan ketergantungan fungsional (lihat sub bab 3.2.5.). Langkah kedua dari algoritma ini adalah memastikan bahwa setiap ketergantungan fungsional di G mempunyai satu atribut di sisi kanannya. Langkah ketiga adalah menghapus atribut-atribut redundant di sisi kiri ketergantungan fungsional. Langkah keempat adalah memastikan bahwa tidak ada ketergantungan fungsional redundant.

Kebenaran algoritma ini tidak dibuktikan secara formal. Tapi bisa diamati bahwa setiap skema relasi yang dihasilkan algoritma ini selalu memenuhi bentuk normal ketiga. Hal ini bisa terjadi karena G adalah cover minimal yang memenuhi

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH – NOPEMBER

syarat himpunan ketergantungan fungsional minimal. Oleh sebab itu semua ketergantungan fungsional di G tidak ada yang terhapus oleh algoritma tersebut. Karena G adalah cover minimal F maka G setara dengan F. Algoritma ini disebut sintesa relasional karena setiap skema relasi R_i di D dibuat dari himpunan ketergantungan fungsional G dari sisi kirinya sama

4.3. DEKOMPOSISI DAN KESATUAN SEKUTU

Ketentuan lainnya yang harus dipenuhi ketika seorang perancang basis data mengelompokan atribut-atribut dalam suatu relasi D adalah ketentuan Kesatuan Sekutu. Ketentuan ini mensyaratkan bahwa pengelompokan atribut-atribut tersebut harus menghasilkan relasi D yang tidak menghasilkan tupel palsu jika dilakukan operasi join satu sama lain. Untuk mengingat kembali topik ini bisa dibaca lagi sub bab 3.1.4. dengan contoh-contob gambar 3.3. Karena ketentuan ini berlaku untuk setiap skema relasi maka ketentuan ini juga berlaku untuk setiap anggota relasinya – yaitu setiap anggota relasi yang skema relasinya memenuhi ketergantungan fungsional yang berlaku. Jadi ketentuan kesatuan sekutu selalu mempunyai keterkaitan yang erat dengan ketergantungan fungsional yang berlaku. Secara formal dekomposisi $D = \{R_1, R_2, \ldots, R_m\}$ yang berasal dari ketergantungan fungsional F di R dikatakan memenuhi ketentuan kesatuan sekutu jika setiap anggota relasi r di R memenuhi persamaan berikut,

$$*(\pi_{<R1>}(r), \pi_{<R2>}(r), \ldots, \pi_{<Rm>}(r)) = r$$

Pembagian skema relasi dari PROJEK_KARYAWAN menjadi LOKASI_ KARYAWAN dan PROJEK1_KARYAWAN pada gambar 3.3. tidak memenuhi ketentuan kesatuan sekutu. Untuk mengetahui apakah suatu skema relasi memenuhi ketentuan kesatuan sekutu dibuat suatu algoritma yang mampu memeriksa persyaratan tersebut. Algoritma ini disebut algoritma memeriksa syarat kesatuan sekutu. Algoritmanya adalah sebagai berikut,

1. *Buatlah sebuah matrik S dengan baris i mewakili relasi R_i sudah didekomposisi di D dan kolom j mewakili atribut A_j pada R.*

2. *Berilah S (i , j) := b_{ij} untuk semua elemen matrik*

   (tiap-tiap b_{ij} adalah simbol yang berbeda pada indek (i , j))

3. *Untuk setiap baris i yang mewakili skema relasi R_i*

   Untuk setiap kolom j yang mewakili atribut A_j

   Jika R_i mengandung atribut A_j maka

   Ubahlah S (i , j) := a_j

   (tiap-tiap a_{ij} adalah simbol yang berbeda pada indek (i , j))

4. *Ulangi perintah berikut ini sampai tidak ada perubahan lagi di S*

   Untuk setiap ketergantungan fungsional X → Y di F

   Untuk semua baris S yang simbolnya sama dengan kolom

   yang mewakili atribut di X

   Jika suatu baris mempunyai simbol "a" dan

   pada kolom yang sama juga terdapat simbol "a" maka

   Ubahlah baris lainnya yang berhubungan dengan kolom tersebut

   menjadi simbol "a"

5. *Jika ada satu baris yang semua elemennya "a" maka himpunan relasi tersebut memenuhi ketentuan kesatuan sekutu. Jika tidak maka tidak memenuhi.*

Gambar 4.3. Algoritma Untuk Menguji Apakah Suatu Himpunan Relasi Memenuhi Ketentuan Kesatuan Sekutu

Langkah pertama yang dilakukan algoritma Memeriksa kesatuan sekutu adalah membuat matrik S. Baris matrik S menggambarkan relasi-relasi yang telah dipecah sedangkan kolom matrik S terdiri dari seluruh atribut R. Matrik S bisa dianggap sebagai relasi semesta R dengan anggota relasinya r adalah relasi-relasi yang telah dipecah. Langkah kedua adalah memberi nilai "b" pada semua tupel yang ada. Langkah ketiga adalah memberi nilai "a" pada masing-masing tupel berdasarkan atribut yang dipunyai tupel tersebut. Langkah keempat adalah memberi nilai "a" pada tupel yang mempunyai nilai yang sama berdasarkan ketergantungan fungsionalnya. Langkah kelima adalah memeriksa apakah ada baris yang nilai semau kolomnya adalah "a". Jika ada maka relasi-relasi tersebut memenuhi syarat kesatuan sekutu, jika tidak maka relasi-relasi tersebut tidak memenuhi syarat kesatuan sekutu.

[a] R = {NIP, NAMA, NO_PRJ, NAMA_PRJ, LOK_PRJ, JAM } D = { R_1 , R_2 }
R_1 = LOKASI_KARYAWAN = { NAMA, LOK_PRJ }
R_2 = PROJEK1_KARYAWAN = { NIP, NO_PRJ, JAM, NAMA_PRJ, LOK_PRJ }

KF = { NIP → NAMA; NO_PRJ → {NAMA_PRJ, LOK_PRJ}; {NIP,NO_PRJ}→ JAM }

	NIP	NAMA	NO_PRJ	NAMA_PRJ	LOK_PRJ	JAM
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

tidak ada perubahan pada matrik setelah dijalankan algoritma

[b] KARYAWAN PROJEK BEKERJA_PADA

NIP	NAMA

NO_PRJ	NAMA_PRJ	LOK_PRJ

NIP	NO_PRJ	JAM

[c] R = {NIP, NAMA, NO_PRJ, NAMA_PRJ, LOK_PRJ, JAM } D = { R_1 , R_2 , R_3 }
R_1 = KARYAWAN = { NIP, NAMA}
R_2 = PROJEK = { NO_PRJ, NAMA_PRJ, LOK_PRJ }
R_3 = BEKERJA_PADA = { NIP, NO_PRJ, JAM }

KF = { NIP → NAMA; NO_PRJ → {NAMA_PRJ, LOK_PRJ}; {NIP,NO_PRJ}→ JAM }

	NIP	NAMA	NO_PRJ	NAMA_PRJ	LOK_PRJ	JAM
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

matrik S sebelum algoritma dijalankan

	NIP	NAMA	NO_PRJ	NAMA_PRJ	LOK_PRJ	JAM
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	~~b_{32}~~ a_2	a_3	~~b_{34}~~ a_4	~~b_{35}~~ a_5	a_6

matrik S setelah algoritma dijalankan

Gambar 4.4. Contoh Penggunaan Algoritma Memeriksa Kesatuan Sekutu

Gambar 4.4.(a) menunjukan bagaimana cara menggunakan algoritma memeriksa kesatuan sekutu terhadap skema relasi PROJEK_KARYAWAN yang dipecah menjadi skema relasi PROJEK1_KARYAWAN dan LOKASI_ KARYAWAN. Algoritma memeriksa kesatuan sekutu tidak berhasil mengubah nilai "b" sembarang menjadi "a" sehingga matrik S tidak menghasilkan baris yang semua kolomnya bernilai "a". Jadi relasi hasil pemecahan tidak memenuhi hukum kesatuan sekutu.

Gambar 4.4.(b) menunjukan pecahan skema relasi PROJEK_KARYAWAN lainnya yang memenuhi syarat kesatuan sekutu. Gambar 4.4.(c) memperlihatkan bagaimana menggunakan algoritma memeriksa kesatuan sekutu terhadap skema relasi PROJEK_KARYAWAN.

Algoritma diatas hanya berfungsi memeriksa apakah suatu skema relasi yang dipecah memenuhi hukum kesatuan sekutu. Pertanyaannya, apakah ada algoritma yang memecah skema relasi R = { A_1, A_2, . . . , A_n } menjadi skema relasi yang terpecah-pecah dalam sebuah himpunan D = {R_1, R_2, . . . , R_m}dimana setiap R_i memenuhi bentuk normal Boyce Codd dan D memenuhi syarat kesatuan sekutu. Jawabannya adalah ada. Algoritma untuk keperluan itu ada. Sebelum membahas algoritma itu, akan dibahas beberapa syarat lainnya untuk memecah skema relasi yang memenuhi aturan kesatuan sekutu.

SYARAT 1

Suatu himpunan pecahan relasi D = {R_1, R_2 } memenuhi ketentuan kesatuan sekutu berdasarkan suatu himpunan ketergantungan fungsional di R jika dan hanya jika

- Ketergantungan Fungsional KF (($R_1 \cap R_2$) $\rightarrow$ ($R_1 - R_2$)) ada di F^+, atau
- Ketergantungan Fungsional KF (($R_1 \cap R_2$) $\rightarrow$ ($R_2 - R_1$)) ada di F^+

Perhatikan bahwa syarat ini merupakan cara termudah untuk memeriksa apakah suatu suatu relasi memenuhi ketentuan kesatuan sekutu. Ketentuan ini hanya berlaku untuk skema relasi yang terdiri dari dua pecahan. Berikutnya akan dibahas apakah ketentuan ini berlaku untuk proses normalisasi yang dibahas pada sub bab 3.3 dan 3.4.

SYARAT 2

Jika himpunan skema relasi pecahan D = {R_1, R_2, . . . , R_m} memenuhi ketentuan kesatuan sekutu dan skema relasi pecahan D_1 = {Q_1, Q_2, . . . , Q_m} yang berasal dari R_i memenuhi ketentuan kesatuan sekutu maka skema relasi pecahan D_2 = {R_1, R_2, . . . , R_{i-1}, Q_1, Q_2, . . . , Q_k, Ri_{+1}, ... , R_m } di R juga memenuhi ketentuan kesatuan skema relasi.

Syarat 2 mengatakan bahwa jika skema relasi pecahan D sudah memenuhi ketentuan kesatuan sekutu lalu ada satu skema relasi yaitu R_i yang merupakan

anggota D dipecah lagi menjadi skema relasi pecahan D_1 yang juga memenuhi ketentuan kesatuan sekutu maka jika skema relasi R_i diganti dengan D_1 maka skema relasi tersebut juga memenuhi kesatuan sekutu.

Dengan menggunakan syarat 1 dan syarat 2 maka dikembangkan algoritma dekomposisi relasi bentuk normal Boyce Codd yang memenuhi ketentuan kesatuan sekutu untuk membuat skema relasi pecahan D di R yang memenuhi ketentuan sekutu dan setiap skema relasi pecahan R_i di D memenuhi bentuk normal Boyce Codd.

1. *D:= {R}*
2. *Lakukan selama masih ada skema relasi di D yang tidak memenuhi bentuk normal Boyce Codd*
   {
   Pilihlah skema relasi Q di D yang tidak memenuhi bentuk normal Boyce Codd.
   Carilah ketergantungan fungsional $X \rightarrow Y$ di Q yang melanggar bentuk normal Boyce Codd.
   Gantilah Q di D dengan dua skema $(Q - Y)$ dan $(X \cup Y)$
   }

Gambar 4.5. Algoritma Dekomposisi Yang Memenuhi BNBC dan Ketentuan Kesatuan Sekutu.

Perlu penulis sampaikan disini bahwa suatu skema relasi hasil pecahan yang memenuhi ketentuan kesatuan sekutu dan keutuhan ketergantungan hanya memenuhi bentuk normal ketiga. Algoritma berikut ini akan membahas bagaimana cara menghasilkan skema relasi pecahan D yang memenuhi bentuk normal ketiga sekaligus memenuhi ketentuan keutuhan ketergantungan dan ketentuan kesatuan sekutu. Algoritma ini merupakan pengembangan dari algoritma dekomposisi skema relasi menjadi bentuk ketiga.

1. *Cari cover minimal G di F*
2. *Untuk setiap X dari ketergantungan fungsional di G*
   Buatlah skema relasi (X UNION A_1 UNION $A_2 \ldots$ UNION A_m) di D dimana $X \rightarrow A, X \rightarrow A, \ldots, X \rightarrow A$ adalah ketergantungan di G dan X adalah sisi kirinya.
3. *Tempatkan atribut-atribut yang tersisa pada sebuah relasi untuk menjamin kondisi keutuhan atribut.*
4. *Jika skema relasi yang dihasilkan tidak mempunyai kunci R maka buatlah satu skema relasi lagi yang salah satu atributnya adalah kunci R*

Gambar 4.6. Algoritma Dekomposisi Yang Memenuhi BN3

BAB V

PEMBUATAN PROGRAM

BAB V
PEMBUATAN PROGRAM

Pembuatan perangkat lunak tugas akhir ini menggunakan pendekatan pemrograman berorientasi obyek (Object Oriented Programming / OOP). Dengan demikian diharapkan pembuatan program bisa lebih terstruktur. Keuntungan pemrograman berorientasi obyek adalah,

1. Fungsi yang dibuat lebih terlokalisasi. Jadi jika programer membuat fungsi dengan nama yang sama tapi berlainan class tidak menjadi masalah.
2. Hubungan antara fungsi dan variabel yang dipakai semakin jelas.
3. Keterkaitan antar fungsi lebih jelas.

Class bisa juga dilihat sebagai perluasan struktur data ditambah fungsi-fungsi yang menggunakan struktur data tersebut. Karena pembuatan program menggunakan pendekatan obyek maka bab ini membahas class-class apa saja yang dibuat, bagaimana hubungan antara class yang satu dengan yang lain, bagaimana struktur masing-masing class dan sebagainya.

5.1. PEMBUATAN CLASS

Class yang dibuat pada program ini adalah,

1. Class TokenHandle

   Class ini bertanggung jawab membaca data dari file, mengubah dari format teks menjadi format data yang sesuai dengan konsep ketergantungan fungsional dan selanjutnya menyimpan hasil proses ke dalam bentuk teks ke file.

2. Class Atribut
   Class ini bertanggung jawab memproses data atribut.

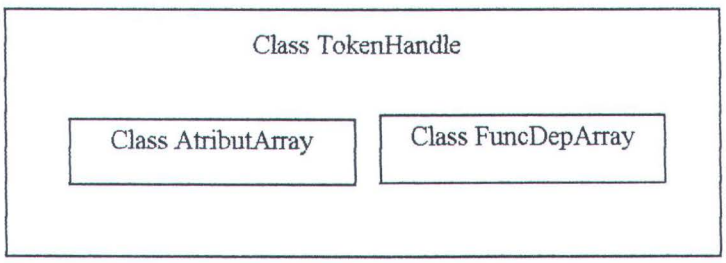
3. Class AtributArray
   Class ini bertanggung jawab memproses data himpunan atribut.

4. Class FuncDep
   Class ini bertanggung jawab memproses data ketergantungan fungsional

5. Class FuncDepArray
   Class ini bertanggungjawab memproses data himpunan ketergantungan fungsional.

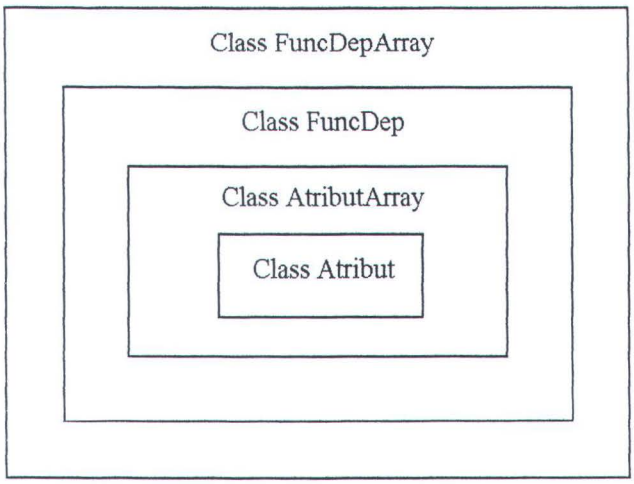
5.2. CLASS CONTAINMENT

Class TokenHandle adalah class yang bertanggung jawab membaca data dari file, mengubah dari format teks menjadi format data yang sesuai dengan konsep ketergantungan fungsional dan selanjutnya menyimpan hasil proses ke dalam bentuk teks ke file.memproses data dan menyimpan data ke file teks. Data-data yang dibaca dari file merupakan himpunan atribut dan himpunan ketergantungan fungsional. Oleh sebab itu di class TokenHandle terdapat objek yang berasal dari AtributArray dan FuncDepArray. Jadi class TokenHandle dibangun dari class AtributArray dan FuncDepArray. Hal ini bisa dijelaskan dengan gambar sebagai berikut,



Gambar 5.1. Class Containment TokenHandle

Class Atribut adalah class yang paling dasar dalam rancangan class-class pada program ini. Ibarat sebuah rumah, class ini merupakan pondasi rumah tersebut. Tugas class Atribut adalah mengubah data karakter dari file menjadi data atribut yang siap dipakai oleh class AtributArray. Class AtributArray adalah class yang mengolah data-data atribut untuk keperluan tertentu. Class AtributArray bisa digunakan jika data dalam bentuk atribut sudah disediakan.oleh class Atribut. Class FuncDep dibangun dari class AtributArray. Tugas dari class FuncDep adalah menyediakan data ketergantungan fungsional yang akan digunakan oleh class FuncDepArray. Class FuncDepArray bertugas mengolah data-data ketergantungan fungsional untuk keperluan tertentu. Class FuncDepArray bisa digunakan jika data dalam bentuk ketergantungan fungsional sudah disediakan oleh class FuncDep. Untuk lebih jelasnya hubungan antar class digambarkan oleh diagram berikut ini.



Gambar 5.2. Class Containment Fungsi Dasar

5.3. STRUKTUR CLASS

Pada sub bab ini akan diterangkan lebih rinci variabel anggota dan fungsi anggota masing-masing class. Pembahasan dimulai dari class TokenHandle, Atribut, AtributArray, FuncDep dan yang terakhir adalah class FuncDepArray. Program ini dibuat dengan menggunakan Borland C++ 3.1. Penulis berasumsi siapapun yang membaca sub bab ini mengerti desain program berorientasi objek dan memahami pembuatan program dengan menggunakan Borland C++ 3.1.

5.3.1. Struktur Class TokenHandle

Header class TokenHandle adalah sebagai berikut,

```
TokenHandle
{
  char Token[MAXTOKEN+1];
  char Fname[MAXPATH];
  AtributArray atrSet;
  AtributArray PrimeAtribut;
  FuncDepArray fdSet;
  FuncDepArray fdSetPrime;
  int Line_no;

public :
  FILE *FIN;
  TokenHandle ();

  void Read (char *fname);
```

```
    void Save (char *fname);
    void NormalForn3 ();
    void BoyceCoddNF ();

  private :
    void ReadAtribut ();
    void ReadFuncDep ();
    void ReadToken ();
    void ReadLHS(FuncDep &fd);
    void ReadRHS(FuncDep &fd);
};
```

Gambar 5.3. Header Class TokenHandle

Variabel anggota Class TokenHandle adalah sebagai berikut,

Token
Tugasnya : Menyimpan token dari file

Fname
Tugasnya : Menyimpan nama file yang menyimpan data.

AtrSet
Tugasnya : Menyimpan himpunan atribut skema relasi.

PrimeAtribut
Tugasnya : Menyimpan himpunan atribut prima

FdSet
Tugasnya : Menyimpan himpunan ketergantungan fungsional

FdSetPrime
Tugasnya : Menyimpan himpunan ketergantungan fungsional yang sisi kanannya atribut prima

Line_no
Tugasnya : Menyimpan nomor baris data yang dibaca pada file.

FIN
Tugasnya : Menyimpan pointer file yang digunakan untuk membaca data dari file dan menulis data ke file.

Fungsi anggota Class TokenHandle adalah sebagai berikut,

void Read (char *fname)
proses : Membaca data dari file dan menyimpan datanya ke variabel atrSet dan variabel fdSet. Nama file yang dibaca berasal dari argumen fname.

void Save (char *fname)
proses : Menyimpan hasil proses normalisasi ke file. Nama file tempat data disimpan berasal dari argumen fname.

void NormalForm3 ()
proses : Menormalkan skema relasi berdasarkan ketergantungan fungsional menjadi bentuk normal ketiga.

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH – NOPEMBER

void BoyceCoddNF ()
proses : Menormalkan skema relasi berdasarkan ketergantungan fungsional menjadi bentuk normal Boyce Codd.

void ReadAtribut ()
proses : Membaca token atribut dan menyimpannya di variabel atrSet.

void ReadFuncDep ()
proses : Membaca token ketergantungan fungsional dan menyimpannya di variabel fdSet.

void ReadToken ()
proses : Membaca token dari file teks. Jika token tersebut atribut maka fungsi ReadAtribut () yang dijalankan. Jika token tersebut ketergantungan fungsional maka fungsi ReadFuncDep () yang dijalankan.

void ReadLHS (FuncDep &fd)
proses : Membaca token ketergantungan fungsional yang ada di sisi kanan.

Void ReadRHS (FuncDep &fd)
Proses : Membaca token ketergantungan fungsional yang ada disisi kiri.

5.3.2. Struktur Class Atribut

Header class Atribut adalah sebagai berikut,

```
class Atribut : public SimpleObject
{
    char _token[MAXTOKEN + 1];
    int _prime;
    int _use;

public:
    Atribut ();
    Atribut (Atribut &t);
    Atribut (char *token,int p=0,int u=0);

    int operator == (Atribut &t) const;
    Atribut &operator = (Atribut &t);
    void Init(char *token, int p=0, int u=0);
    void SetPrime ();
    void SetUse ();
    char *GetToken() { return _token };
    int GetPrime () { return _prime };
    int GetUset() { return _use};
```

```
private :
    int isEqual ( RCObject obj) const
        { return operator == ((Atribut &) obj ); }
};
```

Gambar 5.4. Header Class Atribut

Variabel anggota class Atribut adalah sebagai berikut,

_token
Tugasnya : Menyimpan token atribut.

_prime
Tugasnya : Menyimpan nilai 0 dan 1. Jika 0 maka atribut tersebut adalah atribut non prima, jika 1 maka atribut tersebut adalah atribut prima.

_use
Tugasnya : Menyimpan nilai 0 dan 1. Jika 0 maka atribut tersebut belum digunakan, jika 1 maka atribut tersebut sudah digunakan.

Fungsi anggota class atribut adalah sebagai berikut,

void SetPrime ()
proses : Menjadikan variabel _prime bernilai satu

void SetUse ()
proses : Menjadikan variabel _use bernilai satu

char *GetToken ()
proses : Mengambil nilai variabel _token dan mengirim ke fungsi yang memanggilnya.

int GetPrime ()
proses : Mengambil nilai variabel _prime dan mengirim ke fungsi yang memanggilnya.

int GetUse ()
proses : Mengambil nilai variabel _use dan mengirim ke fungsi yang memanggilnya.

5.3.3. Struktur Class AtributArray

Header class AtributArray adalah sebagai berikut,

```
Class AtributArray : public Array
{

public :
AtributArray ();
AtributArray (char *token);
~AtributArray ();
```

```
    Atribut *operator [] (int idx) const;
    int *operator ==  (AtributArray  &AA) const;
    AtributArray  &operator = (AtributArray &AA);
    int SetUse (char *a);
    int AddUniqe (Atribut &t);
    int SetPrime (AtributArray &AA);
    int IsSubSet (AtributArray &AA);
    void PrintAll ();
    void Print ();
    void SaveRemainingAtribut(FILE *fout);
    void Destroy ();
    void Save (FILE *fout);
};
```

Gambar 5.5. Header Class AtributArray

Fungsi anggota AtributArray adalah sebagai berikut,

int SetUse (char *a)
Proses : Menge-set variabel _use pada atribut a

int AddUniqe (Atribut &t)
Proses : Menambah atribut di array atribut

void SetPrime (AtributArray &AA)
Proses : Menge-set variabel _prime pada atribut yang terdapat di AtributArray AA.

Int IsSubSet (AtributArray &BB)
Proses : Memeriksa apakah himpunan atribut yang ada di (*this) adalah subset AtributArray BB.

void SaveRemainingAtribut (FILE *fout)
Proses : Menyimpan atribut yang tidak termasuk dalam relasi yang sudah didekomposisi.

5.3.4. Struktur Class FuncDep

Header class FuncDep adalah sebagai berikut ,

```
class FuncDep : public SimpleSort
{
    public :
        AtributArray        *LHS,
                    *RHS;
        int _key;
        int _subset;
        FuncDep();
        FuncDep(FuncDep &fd);
        ~FuncDep();
        int operator == (FuncDep &fd) const ;
```

```
    int operator < (FuncDep &fd) const ;
    int operator > (FuncDep &fd) const ;
    FuncDep &operator = (FuncDep &t);
    void SetKey () {_key=1;}
    void addLHS (char *token);
    void addRHS (char *token);
    void CopyLHSFrom (AtributArray &AA);
    void CopyRHSFrom (AtributArray &AA);
    void FlushArray ();
    void IncSubSet () {_subset++;}
    int GetSubSet () {return _subset;}
    int GetKey () {return _key;}
    AtributArray *GetLHS () {return LHS;}
    AtributArray *GetRHS () {return RHS;}
    void GetAllAtribut (AtributArray &AA);
    void Save (FILE *fout);
    void SubtractLHS (FuncDep &fd, int index);
    void DeleteElementRHS (Atribut &A);
    void Destroy();
    void Print();
  private :
    int isEqual(RCObject obj) const
     {return operator == ((FuncDep &) obj);}
    int isLessThan (RCObject obj) const
     {return operator > ((FuncDep &) obj);}
};
```

Gambar 5.6. Header Class FuncDep

Variabel anggota class FuncDep adalah sebagai berikut,

AtributArray *LHS
Tugasnya : Menyimpan data sisi kiri ketergantungan fungsional.

AtributArray *RHS
Tugasnya : Menyimpan data sisi kanan ketergantungan fungsional.

int _key
Tugasnya : Menyimpan nilai 0 atau 1. Jika 1 maka sisi kanan ketergantungan fungsional adalah kunci kandidat. Jika 0 maka sebaliknya.

int _subset
Tugasnya : Menyimpan nilai 0 atau 1. Jika 1 maka ketergantungan fungsional adalah subset dari yang lain. Jika 0 maka sebaliknya.

Fungsi anggota class FuncDep adalah sebagai berikut,

void SetKey ()
Proses : Menge-set variabel _key menjadi 1.

void addLHS (char *token)
proses : Menambah atribut sisi kiri ketergantungan fungsional.

void addRHS (char *token)
proses : Menambah atribut sisi kanan ketergantungan fungsional.

void SubtractLHS (FuncDep &fd, int index)
Proses : Menghapus atribut sisi kiri ketergantungan fungsional.

void DeleteElementRHS (Atribut &A)
Proses : Menghapus atribut sisi kanan ketergantungan fungsional.

5.3.5. Struktur Class FuncDepArray

Header class FuncDepArray adalah sebagai berikut,

```
Class FuncDepArray : public SortedArray
{
  public :
      FuncDepArray () ;
      ~FuncDepArray () ;

      FuncDep *operator [] (int idx);

      void add (Object _FAR &);
      void Closure (AtributArray &AA, AtributArray &AAp);
      void FindKey (AtributArray &atrSet, AtributArray &PrimeAtribut);
      void Extract ();
      void Subtract (FuncDepArray &fdSet, int index);
      int Find (FuncDep &fd);
      void ReduceLHS ();
      void RemoveRedundant ();
      void Union ()
      void SubSetOn ()
      void MovePrimeAtributTo (FuncDepArray &fdPrime, AtributArray &
      atrPrime);
      void ReplaceViolateBCNF(FuncDepArray &fdPrime);
      void Print ();
      void Save (FILE *fout);
      void SaveKey (FILE *fout);
      void Destroy ();
};
```

Gambar 5.7. Header Class FuncDepArray

Fungsi anggota class FuncDepArray adalah sebagai berikut,

void Closure (AtributArray &AA, AtributArray &AAp)

Proses : Mencari closure ketergantungan fungsional yang ditunjuk oleh pointer (*this) berdasarkan himpunan ketergantungan fungsional AA dan hasilnya disimpan di himpunan atribut Aap.

void FindKey (AtributArray &atrSet, AtributArray &PrimeAtribut)

Proses : Mencari kunci dari himpunan ketergantungan fungsional yang ditunjuk oleh pointer (*this) hasilnya disimpan di variabel atrSet.

void Extract ()

Proses : Memecah sisi kanan ketergantungan fungsional sehingga hanya terdiri dari satu atribut saja.

void Subtract (FuncDepArray &fdSet, int index)

Proses : Menghapus satu ketergantungan fungsional berdasarkan indeknya dan hasilnya disimpan di variabel fdSet.

void ReduceLHS ()

Proses : Menghapus atribut sisi kiri dari ketergantungan fungsional yang tidak signifikan.

void RemoveRedundant ()

Proses : Menghapus ketergantungan fungsional yang redundant.

void MovePrimeAtributTo (FuncDepArray &fdPrime, AtributArray & atrPrime)

Proses : Memindahkan ketergantungan fungsional yang atribut sisi kanannya prima ke variabel fdPrime

void ReplaceViolateBCNF(FuncDepArray &fdPrime)

Proses : Memecah ketergantungan fungsional yang tidak memenuhi bentuk normal Boyce Codd.

BAB VI

KESIMPULAN DAN SARAN

BAB VI
KESIMPULAN DAN SARAN

6.1. KESIMPULAN

1. Untuk menjalankan perangkat otomatis ini, perancang basis data harus membuat semua ketergantungan fungsional dari semua atribut-atributnya. Hal ini bukanlah pekerjaan yang mudah untuk skema relasi yang terdiri dari ratusan atribut. Jika gagal dalam merumuskan satu atau dua ketergantungan yang penting maka perangkat otomatis ini akan menghasilkan skema relasi yang tidak sesuai dengan yang diharapkan.

2. Algoritma program ini adalah algoritma non deterministik artinya keluaran program ini tidak selalu sama. Misalnya proses untuk mendapatkan ketergantungan fungsional minimal, untuk sebuah himpunan ketergantungan fungsional bisa terdiri dari beberapa kombinasi ketergantungan fungsional minimal. Sehingga jika sebuah himpunan ketergantungan fungsional diolah berulang kali untuk dicari ketergantungan fungsional minimalnya maka ketergantungan fungsional minimal yang muncul bisa tidak sama, karena hasil ketergantungan fungsional minimalnya tergantung dari urutan masuknya.

6.2. SARAN

Berdasarkan kesimpulan di atas maka penulis menyarankan ,

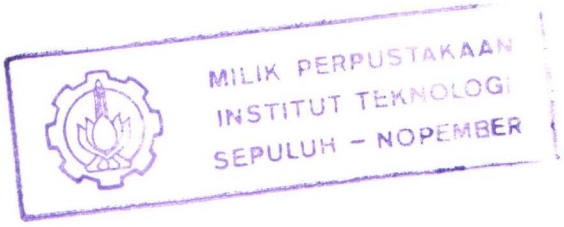
1. Memperkecil ruang lingkup atau jumlah atribut yang akan diproses sehingga akan meminimalkan kemungkinan kesalahan yang terjadi ketika memecah skema relasi.
2. Hal ini bisa dilakukan jika dalam mendesain skema relasi selain menggunakan perangkat otomatis ini juga menggunakan diagram ER.

Jadi pertama kali yang dilakukan oleh perancang basis data adalah membuat konsep skema degan menggunakan model ER dan memetakan relasinya. Selanjutnya konsep skema tersebut dinormalkan dengan menggunakan perangkat otomatis ini. Dengan demikian skema relasi yang dihasilkan mendekati sempurna

DAFTAR PUSTAKA

DAFTAR PUSTAKA

1. C.L. Liu, "Element of Discrete Mathematics", Mc Graw – Hill, 1985
2. Donald F. Stanat and David F. Mc Allister, "Discrete Mathematics in Computer Science", Prentice Hall, 1977
3. Henry F. Korth and Abraham Silberchatz, "Database System Concepts", McGraw – Hill, 1991
4. Ramez Elmasri and Shamkant B. Navanthe, "Fundamental of Database System", The Benjamin / Cumming, 1989



LAMPIRAN

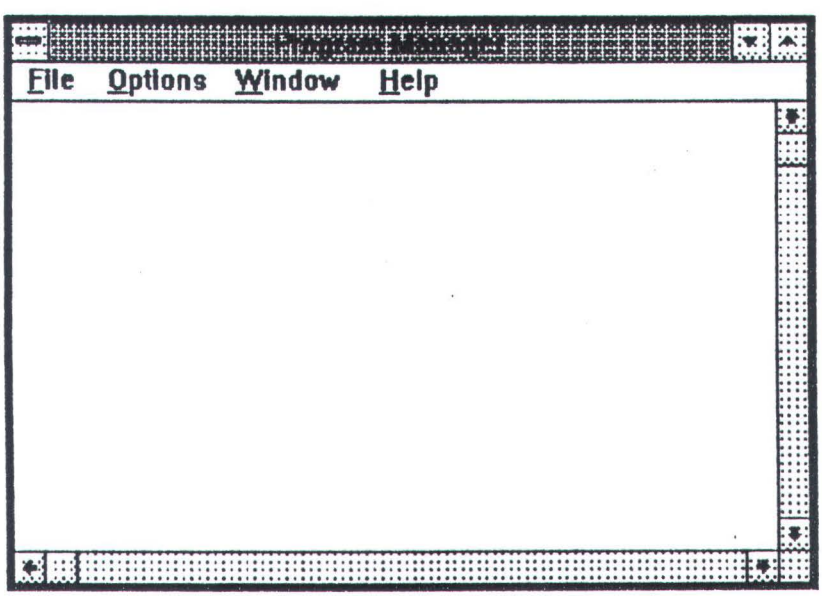
PETUNJUK PENGGUNAAN PROGRAM

Pada bagian ini penulis akan menjelaskan bagaimana menggunakan program yang penulis buat. Penulis membuat program ini dalam sistem operasi Windows 3.1. Sebelum menjalankan program ini sistem operasi Windows harus dijalankan terlebih dahulu. Setelah itu program ini dipanggil. Untuk memanggil program ini ikuti petunjuk petunjuk sebagai berikut,

1. Jalankan sistem operasi Windows dengan mengetikkan win pada C prompt lalu tekan enter.

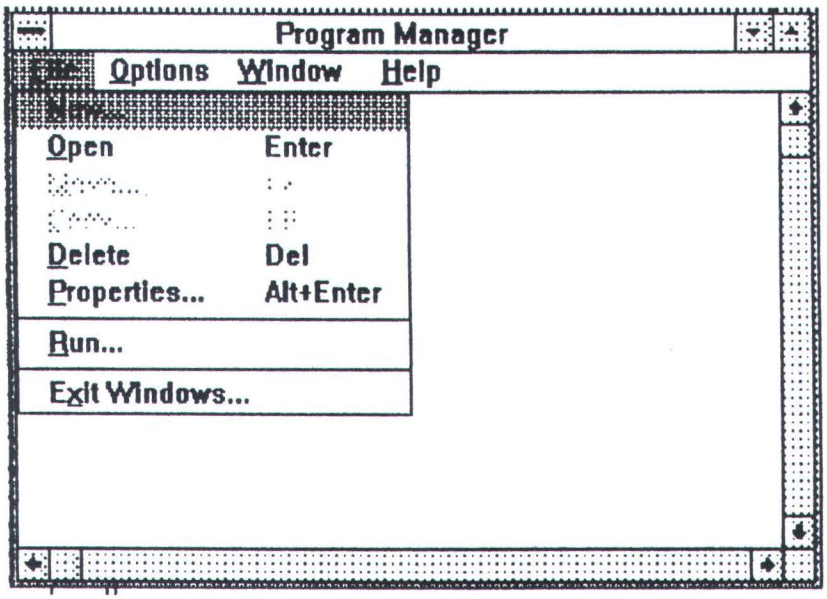
C > win ↵

Tunggu beberapa saat, dilayar akan muncul Window Program Manager seperti yang terlihat pada gambar berikut ini,



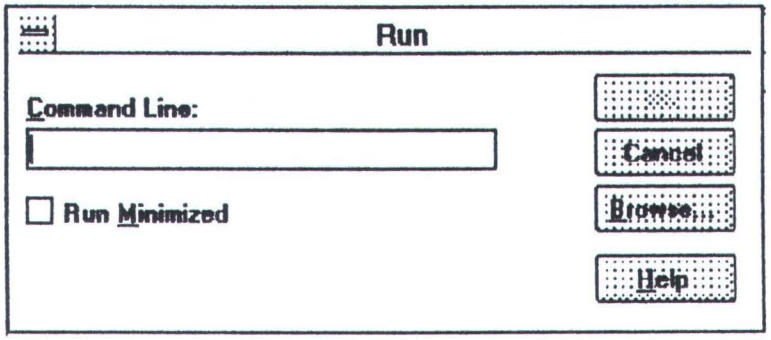
Gambar A.1. Program Manager.

2. Click kata File yang terletak di pojok kiri atas dengan mouse atau tekan tombol Alt F pada keyboard secara bersamaan. Pada layar akan tampak gambar berikut ini,



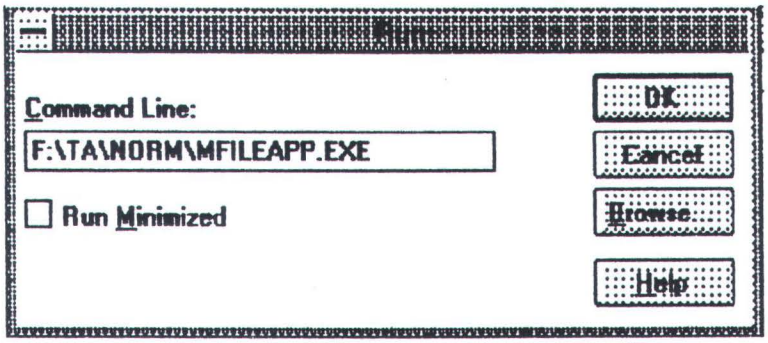
Gambar A.2. Window File

3. Click kata Run yang terdapat pada Window File atau tekan tombol Alt R pada keyboard secara bersamaan . Kemudian akan muncul window yang tampak seperti gambar berikut,



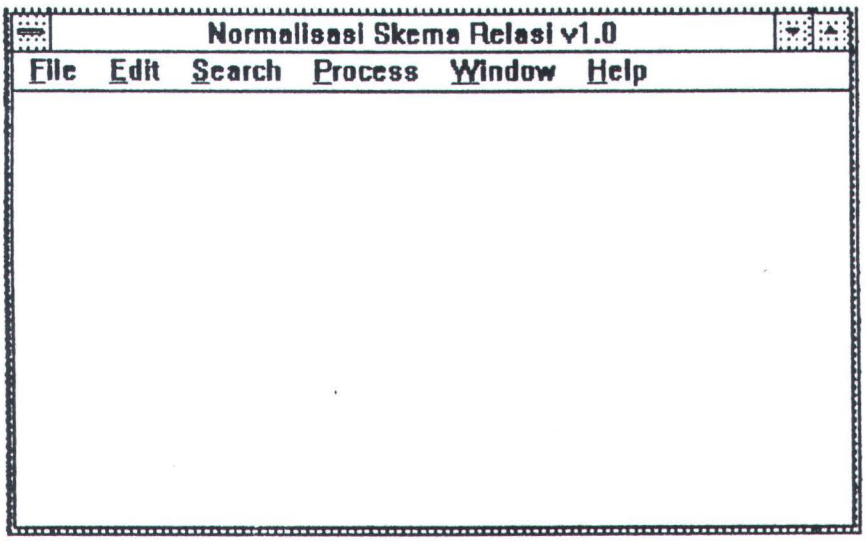
Gambar A.3. Window Run

4. Ketik nama program pada Window Run seperti yang terlihat pada gambar berikut ini kemudian tekan button OK.



Gambar A.4. Ketik nama program.

Tunggu beberapa saat kemudian akan muncul Window program yang tampak seperti gambar berikut,



Gambar A.5. Window Program Normalisasi.

Sebelum menjalankan program normalisasi, pemakai harus membuat file input terlebih dahulu. File input berekstensi txt. Progam baru bisa dijalankan jika file input telah tersedia. Hasil proses program disimpan dalam bentuk file output dan dapat dilihat setelah program di eksekusi. File output berekstensi out. Berikut

ini akan dijelaskan secara berurutan bagaimana membuat file input, menjalankan programnya dan yang terakhir file output.

1. FILE INPUT

Input program terdiri dari atribut dan ketergantungan fungsional. Bentuk input berupa file teks yang terdiri dari dua bagian. Bagian pertama adalah bagian atribut dan bagian kedua adalah bagian ketergantungan fungsional. Bagian atribut ditandai dengan sebuah perintah titik yaitu .at dan bagian ketergantungan fungsional ditandai dengan sebuah perintah titik .fd. Berikut ini diberikan sebuah contoh file input,

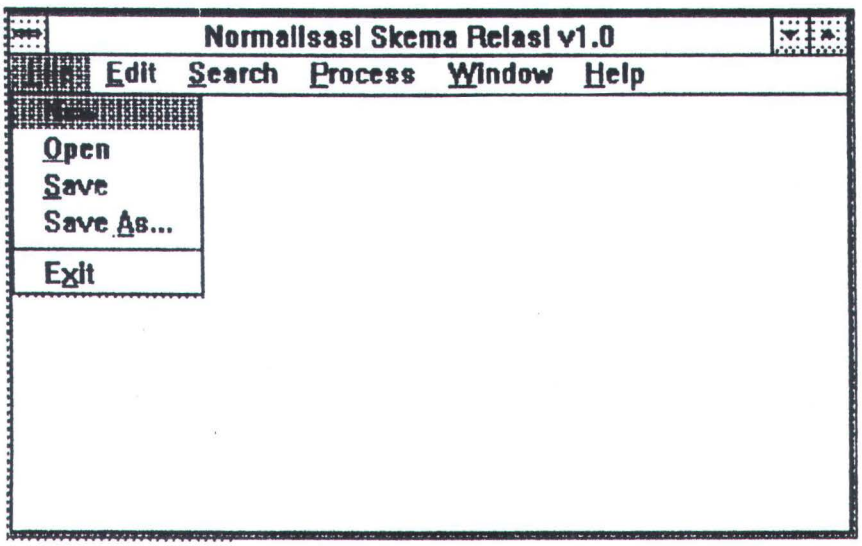
```
.at
NOBUKTI,NOACC,NAMA_ACC,TGL_TRANS,KETERANGAN,JUMLAH,
DK, POST;
.fd
NOBUKTI,NOACC  →  NAMA_ACC,TGL_TRANS, KETERANGAN,
                  JUMLAH, DK,POST
NOBUKTI        →  TGL_TRANS,KETERANGAN
NOACC          →  NAMA_ACC;
```

Gambar A.6 Contoh file input.

Setelah pemakai selesai menulis bagian atribut atau bagian ketergantungan fungsional, bagian tersebut harus diakhiri dengan tanda semicolon (;). Hal ini bertujuan untuk membantu computer bahwa dia telah selesai membaca suatu bagian. Pemakai membutuhkan sebuah editor teks untuk membuat sebuah file

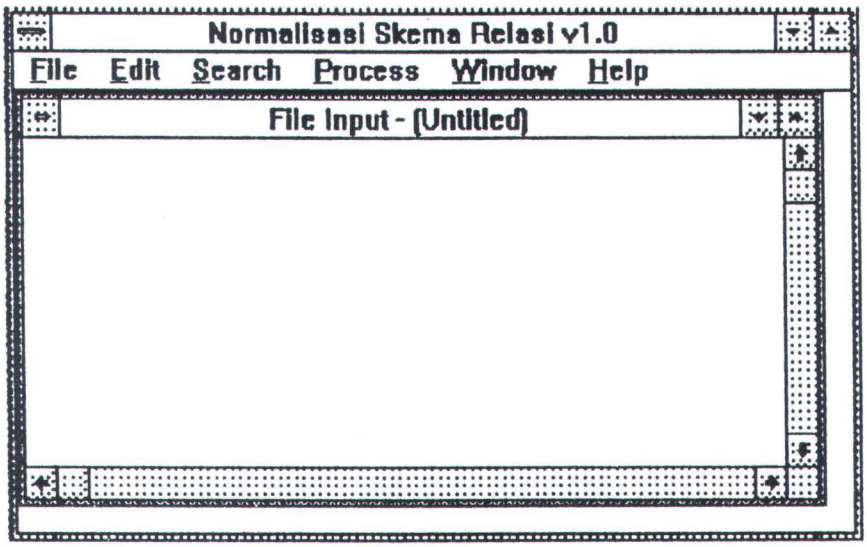
input. Program ini juga menyediakan editor teks untuk membuat file input. Cara menggunakan editor teks pada program ini dapat dilihat pada petunjuk berikut ini,

1. Click kata File yang terletak di pojok kiri atas dengan mouse atau tekan tombol Alt F pada keyboard secara bersamaan. Pada layar akan tampak gambar berikut ini,



Gambar A.7.Window file

2. Click kata New yang terdapat pada Window File atau tekan tombol Alt N pada keyboard secara bersamaan . Kemudian akan muncul window yang tampak seperti gambar berikut,

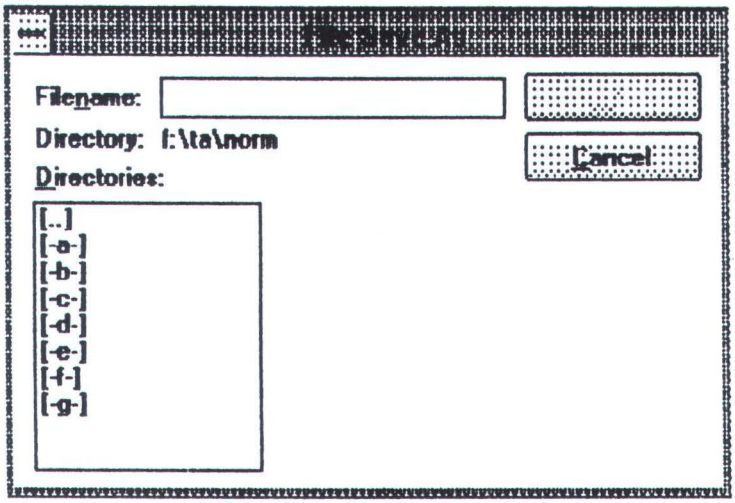


Gambar A.8. Window editor

Window ini adalah window editor, pemakai menggunakan window ini untuk menulis file input. Penulis menganjurkan pemakai untuk menulis kembali contoh file input yang terdapat pada gambar A.6. supaya lebih memahami cara penulisan file input sekaligus cara menggunakan editor ini. Setelah selesai menulis file input maka file tersebut harus disimpan. Untuk menyimpan file input tersebut ikuti petunjuk sebagai berikut,

1. Click kata File yang terletak di pojok kiri atas dengan mouse atau tekan tombol Alt F pada keyboard secara bersamaan. Pada layar akan tampak gambar A.7.

2. Click kata Save yang terdapat pada Window File atau tekan tombol Alt S pada keyboard secara bersamaan . Kemudian akan muncul window yang tampak seperti gambar berikut,

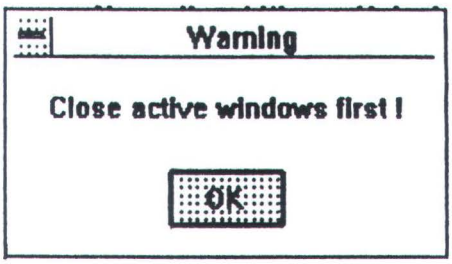


Gambar A.9. Window Save As

3. Ketik nama file input yang akan disimpan dan diakhiri dengan ekstensi txt. Setelah itu tekan button OK atau tekan tombol Enter ↵ dengan demikian file input telah tersimpan.

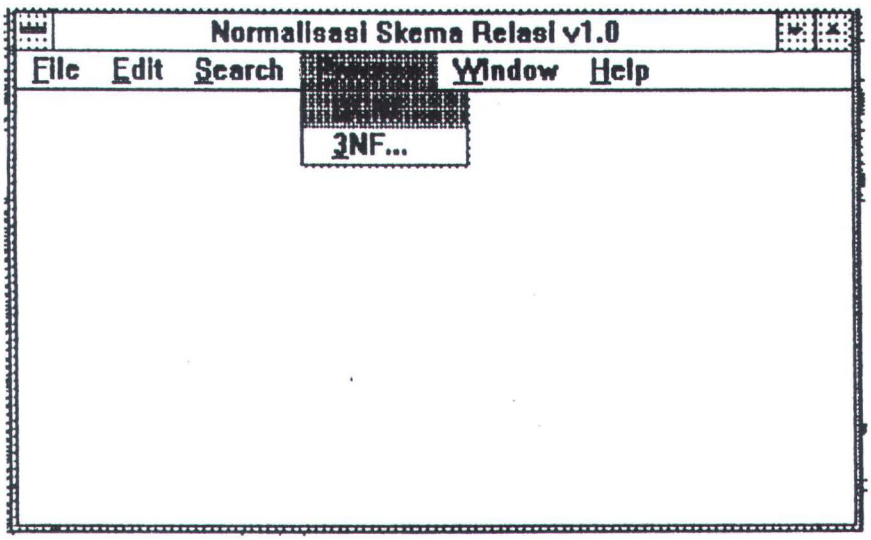
2. EKSEKUSI PROGRAM

Hal penting yang harus diperhatikan sebelum menjalankan program ini adalah menutup semua Window yang aktif pada program tersebut. Karena program tidak dapat mengakses file input yang sedang dipakai oleh suatu Window. Jika pemakai mengeksekusi program tanpa menutup Window yang aktif terlebih dahulu maka program akan mengeluarkan peringatan untuk menutup Window yang aktif terlebih dahulu seperti yang diperlihatkan pada gambar A.11. Untuk menjalankan program ini ikuti petunjuk berikut ini,



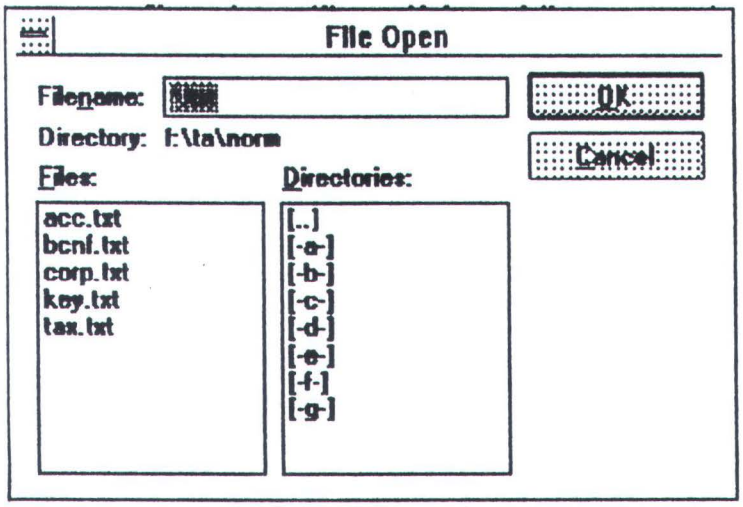
Gambar A.10. Window peringatan

1. Click kata Process yang terletak nomor 4 dari kiri dengan mouse atau tekan tombol Alt P pada keyboard secara bersamaan. Pada layar akan tampak gambar berikut ini,



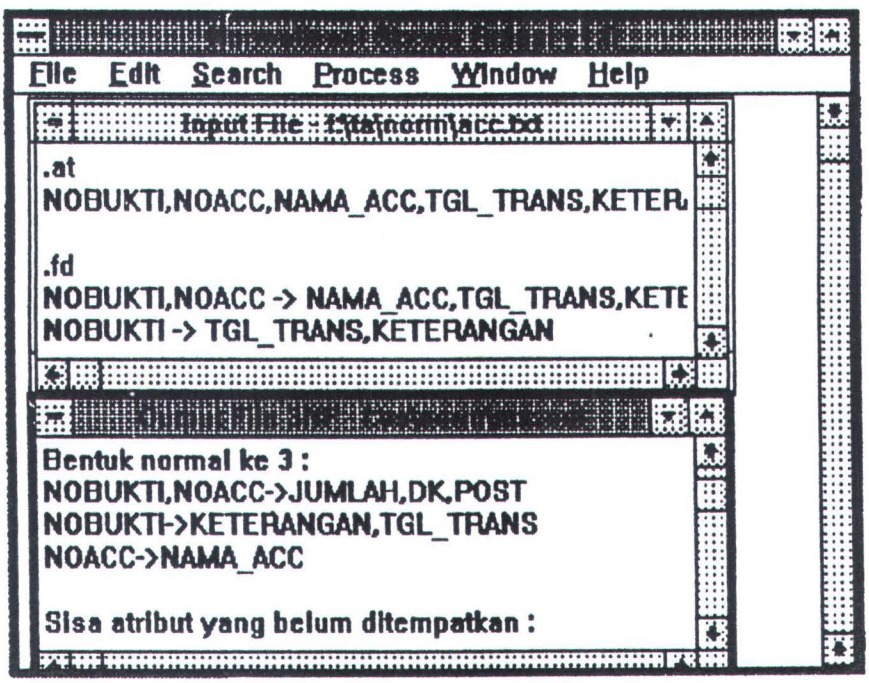
Gambar A.11. Window Process.

2. Click kata 3NF yang terdapat pada Window Process atau tekan tombol Alt 3 pada keyboard secara bersamaan . Kemudian akan muncul window yang tampak seperti gambar berikut,



Gambar A.12. Window File Open

3. Ketik nama file input atau click dengan mouse file input yang akan diproses yang terdapat pada daftar file. Kemudian tekan button OK atau tekan tombol Enter ↵ maka akan muncul dua window yang terdiri dari file input dan file output yang terlihat pada gambar berikut ini,



Gambar A.13. Window Input dan Output

3. FILE OUTPUT

Hasil eksekusi program disimpan dalam sebuah file. File ini terdiri dari tiga bagian yaitu,

1. Bentuk Normal Ketiga

   Bagian ini memuat skema relasi yang sudah memenuhi bentuk normal ketiga.

2. Sisa Atribut

   Bagian ini memuat atribut-atribut yang belum ditempatkan pada skema relasi.

3. Kunci (key) skema relasi.

   Untuk lebih jelas bagaimana bentuk file output bisa dilihat pada gambar A.13 diatas.

USULAN ULANG TUGAS AKHIR

JURUSAN TEKNIK KOMPUTER

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

PENGUSUL

a. Nama : Karunia P.M. Iwan
b. NRP : 288 2600 157
c. Dosen Wali : Ir. Arif Djunaidy M.Sc. Ph.D.

JUDUL PENELITIAN : **RANCANG BANGUN PERANGKAT LUNAK UNTUK NORMALISASI SKEMA RELASI SECARA OTOMATIS**

IKHTISAR PENELITIAN :

Normalisasi data bisa dilihat sebagai proses membagi (dekomposisi) skema relasi - yang belum memenuhi kriteria basis data yang baik - dengan cara memecah atribut-atributnya menjadi skema relasi yang lebih kecil dimana skema tersebut masih memiliki sifat-sifat yang dibutuhkan (desirable properties). Kompleksitas proses ini cenderung meningkat jika melibatkan jumlah atribut yang banyak sehingga kemungkinan kesalahan yang terjadi juga meningkat jika dilakukan secara manual.

Tugas akhir ini akan meneliti perancangan dan pembuatan perangkat lunak untuk meng-implementasikan prosedur formal normalisasi data sehingga proses normalisasi menjadi lebih cepat dan mudah.

Surabaya, 25 September 1995
Pembimbing

Ir. Arif Djunaidy M.Sc. Ph.D.
NIP : 131 633 403

Mengetahui
Ketua Jurusan

DR. Ir. Handayani Tjandrasa, M.Sc.
NIP : 130 532 048

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH – NOPEMBER

I. PENDAHULUAN

Perancangan basis data memegang peranan yang penting dalam proses rekayasa perangkat lunak. Perancangan yang benar akan meningkatkan kinerja sistem, sebaliknya perancangan yang ceroboh akan menimbulkan banyak masalah antara lain data yang berulang (data redundant) dan data yang tidak sama (data inconsistent) dll.

Pada proyek pengembangan software sering ditemui praktek- praktek perancangan basis data berdasarkan selera perancangnya hal ini bisa dimengerti sebab dalam perancangannya melibatkan atribut-atribut yang banyak sehingga jika menggunakan prosedur formal yang ada membutuhkan waktu yang lama, melelahkan dan kecenderungan kesalahan yang terjadi juga besar. Oleh sebab itu untuk mempercepat proses perancangan dibutuhkan alat yang dapat membantu perancangan basis data sehingga dapat menghasilkan desain basis data yang baik.

II. PERUMUSAN MASALAH

Setelah proses pemetaan ER diagram dihasilkan Skema relasi. Skema relasi ini harus dicek untuk mengetahui apakah sudah memenuhi syarat normalisasi. Jika belum memenuhi syarat normalisasi maka skema relasi tersebut harus dinormalisasi. Proses normalisasi terdiri dari beberapa tahap yaitu normalisasi bentuk pertama sampai dengan normalisasi bentuk ke tiga dimana pada tahap-tahap ini dibutuhkan parameter tambahan yaitu "functional dependencies". Tugas akhir ini berusaha meneliti perancangan dan pembuatan perangkat lunak untuk beberapa algoritma normalisasi dengan memasukkan skema relasi dan "functional dependencies" dalam bentuk file teks.

III. TINJAUAN PUSTAKA

Proses normalisasi diusulkan pertama kali oleh Codd tahun 1972. Awalnya dia mengusulkan tiga bentuk normal yang dia sebut sebagai bentuk normal pertama (1NF), bentuk

normal kedua (2NF), dan bentuk normal ketiga (3NF). Belakangan dia mengusulkan bentuk normal ketiga yang disempurnakan yang dikenal dengan nama Boyce Codd Normal Form (BCNF). Semua bentuk normal ini dibuat berdasarkan "functional dependencies" dari atribut-atribut yang berelasi.

"Functional dependencies", yang ditulis dalam bentuk $X \rightarrow Y$ (X dan Y adalah himpunan atribut yang merupakan himpunan bagian dari R dan R adalah himpunan atribut yang berelasi), adalah fungsi yang mampu menggambarkan hubungan (constraint) antara dua himpunan atribut-atribut basis data. hubungan itu menyatakan bahwa untuk dua tuple t1 dan t2 pada r (r adalah himpunan tuple pada R) dimana $t1[X] = t2[X]$, maka harus ada $t1[Y] = t2[Y]$. Hal ini berarti nilai dari Y yang merupakan komponen tuple pada r tergantung atau ditentukan oleh nilai komponen X.

Pada Functional depedencies berlaku beberapa aturan yang bisa disebutkan beberapa disini sebagai contoh yaitu, aturan refleksi, aturan augmentasi, aturan transitif. Aturan-aturan ini disebut juga aksioma Armstrong. Berdasarkan Functional dependencies berikut aturan-aturan yang berlaku maka dapat diturunkan algoritma untuk proses normalisasi.

IV. TUJUAN PENELITIAN

Membuat perangkat lunak untuk membantu merancang skema basis data relasional dengan menormalisasi sejumlah skema relasi berdasarkan sejumlah "functional dependencies" yang didefinisikan

V. MANFAAT PENELITIAN

Perangkat lunak ini akan membantu proses perancangan basis data yang diharapkan dapat menghasilkan skema basis data relasional yang baik. Skema ini bisa menjadi acuan dalam proses desain basis data sehingga proses perancangan basis data bisa lebih cepat, mudah dan optimal.

VI. METODE PENELITIAN

- Studi literatur
- Perancangan representasi input dan output program
- Perancangan perangkat lunak
- Pembuatan perangkat lunak

VII. DAFTAR PUSTAKA

- Henry F.Korth dan Abraham Silberschatz, "Database System Concepts", McGraw-Hill, 1991
- Ramez Elmasri dan Shamkant B.Navanthe, "Fundamental of Database System", The Benjamin/ Cumming, 1989
- C. L. Liu, "Elements of Discrete Mathematics", McGraw-Hill, 1985
- Donald F.Stanat dan David F.McAllister, "Discrete Mathematics in Computer Science", Prentice Hall, 1977

VIII. JADWAL KEGIATAN

No.	Kegiatan	Bulan ke					
		1	2	3	4	5	6
1.	Studi literatur	■	■				
2.	Mendesain input/output program		■				
3.	Mendesain perangkat lunak		■	■			
4.	Membuat perangkat lunak			■	■	■	
5.	Revisi						■
6	Membuat buku TA					■	■