Chapter 2

RELASI

2.1 PENGENALAN

Model relasional diperkenalkan oleh Codd pada tahun 1971. Semenjak itu relasi memainkan bagian yang penting dalam kebanyakan metodologi perancangan database. Relasional penting dalam perancangan database karena tiga alasan. Pertama, relasional membuat komunikasi yang bagus antara user dan disainer. Relasi mewakili struktur data dalam suatu cara yang siap dipahami oleh user dan para professional komputer. Relasi melakukan ini dengan menampilkan struktur data yang sederhana yang dapat memasukan nilai untuk membantu penjelasan.

Alasan penting ke dua untuk penggunaan model relasional berkenaan dengan tercapainya kriteria rancangan database yang esensial. Model relasional mendefinisikan kriteria yang demikian dalam hal relasi bentuk normal. Definisi bentuk normal akan dibicarakan pada Chapter 4 setelah beberapa diskusi pendahuluan dalam Chapter ini dan Chapter berikutnya. Defnisi bentuk normal kemudian digunakan untuk mengevaluasi rancangan yang diusulkan.

Terakhir, model relasional mempunyai satu keuntungan tambahan. Struktur data yang diwakili oleh relasi dapat dikonversi dengan mudah ke DBMS relasional dan diimplementasikan langsung pada komputer yang mendukung RDBMS. Sehingga ketika anda mempunyai sebuah rancangan relasional, anda dapat mendefinisikannya dengan bahasa-bahasa yang disediakan oleh sistem.

Chapter ini akan menggambarkan seperti apa bentuk relasi dan mendefisikan beberapa isitilah yang berkaitan dengan relasi. Dan kemudian dilanjutkan dengan perbedaan antara cara penyimpanan data yang baik dan buruk menggunakan relasi.

2.2 APAKAH RELASI

Relasi pertama kali didefinisikan menggunakan teori relasi dalam matematika. Teori ini dapa digunakan secara tepat untuk mendefinisikan kriteria desain formal. Presentasi teori ini telah disederhanakan kedalam bentuk yang mudah diterima untuk komunitas yang lebih luas. cara paling mudah untuk **mendefinisikan relasi sebagai tabel dengan nilai-nilai data yang disimpan dalam baris tabel**. Contoh dari relasi atau tabel diilustrasikan pada Gambar 2.1. Gambar ini menunjukkan menunjukan sebuah relasi yang disebut WORK terbuat dari tiga kolom, yaitu PERSON-ID, PROJ-NO dan TOTAL-TIME-SPENT-BY-PERSON-ON-PROJECT. Ini menyimpan waktu yang dibutuhkan oleh orang dalam berbagai proyek. Gambar 2.1 juga memasukan relasi lain, yaitu PERSONS. Relasi ini

menyimpan detail tentang orang yang bekerja pada proyek. Database relasional adalah koleksi dari banyak relasi.

Gambar 2.1 Relasi WORK dan PERSONS

PERSON-ID	PROJ-NO	TOTAL-TIME-SPENT-BY-PERSON-ON-PROJECT
P1	PROJ1	20
P3	PROJ1	16
P2	PROJ2	35
P2	PROJ3	42
P3	PROJ2	17
P3	PROJ1	83
P4	PROJ3	41

PERSONS

PERSON-ID	DATE-OF-BIRTH	NAME
P1	Jan 89	Jae
P4	Feb 88	Mary
P3	Aug 90	Adrea
P2	Jun 68	Jef

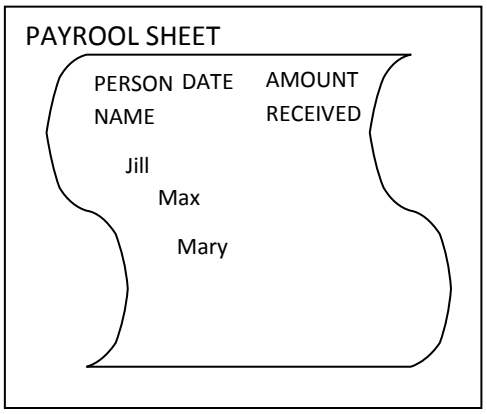
Representasi-representasi relasional berguna karena sangat mirip dengan banyak struktur data yang ditemukan dalam praktek. Sebagai contoh, banyak data dalam praktek telah ditampilkan sebagai daftar. Struktur data umum tersebut sangat serupa dengan relasi yang dilustrasikan dalam Gambar 2.2. Gambar ini menunjukan daftar payroll dari pembayaran yang telah dilakukan ole seseorang, dan sebuah kartu index yang membuat data detail setiap orang. Gambar 2.2 juga menunjukan bagaimana berkas parool dengan mudah direpresentasikan sebagai relasi PAYROLL dan bagaimana file index personal dengan mudah direpresentasikan oleh relasi PERSONNEL. Representasi yang gambling dari struktur user umum membuat relasi terlihat seperti representasi alami dari data user dan dengan demikian cukup mudah untuk diterima sebagai alat komunikasi.

Terminology

Deskripsi relasi dan sifat-sifatnya membutuhkan banyak penjelasan dan definisi resmi yang telah diberikan pada bagian sebelumnya. Definisi ini dilustrasikan pada Gambar 2.3.

Definisi paling penting adalah adalah definisi tentang relasi atau tabel. Kata relasi lebih cendrung digunakan dalam pekerjaan matematika sementara kata tabel lebih digunakan dalam praktek sebagai alternatif istilah relasi. Istilah atribut relasi didefinisikan sebagai kolom tabel dan istilah *tuple* mendefinisikan sebagai baris tabel.

Gambar 2.2 Relasional dan struktur yang serupa dalam praktek



PAYROLL

PERSON-NAME	DATE	AMOUNT-RECEIVED
Jill	9 Mar 87	66
Max	11 Apr 87	564
Mary	6 May 87	688

Gambar 2.3 Teriminology

DOMAIN	PART-ID	DESCRIPTION	COLORS	WEIGHTS	DIMENSIONS
	Type:	Type:	Type:	Type:	Type:
	Alpha	string	string	numeric	numeric

PARTS (nama relasi)

PART-NO	PART-NAME	COLOR	WEIGHT	MAX-DIM
P1	saw	black	12	25
P2	hammer	brown	20	19
P3	drill	gray	86	45

Tuple atau Baris

Atribut atau kolom

Kunci relasi

Terdapat dua istilah penting lainnya, domain dan kunci relasi. Domain didefinisikan sebagai nilai nilai yang atribut dapat ambil, atau alternatifnya, nilai yang dapat muncul dalam kolom. Domain tidfak hanya tipe. Domain adalah nama yang mengidentifikasi sebuah himpunan nilai, dan nilai-nilai tersebut dari beberap tipe. Dengan demikian memungkinkan untuk mempunyai dua domain yang mempunyai tipe sama tapi berbeda domain. Sebagai contoh, dalam Gambar 2.3 terdapat 5 domain, PART-ID, DESCRIPTIONS, COLORS, WEIGHTS and DIMENSIONS. Beberapa dari domain ini mempunyai tipe yang sama. Sebagai contoh, WEIGHTS dan DIMENSIONS kedua-duanya domain numeris. Atribut dapat mengambil nilai dari domainnya. Dengan demikian nilai dalam atribut COLOR berasal dari domain, COLORS, dan dapat mengampil sebagai nilai dalam daftar warna.

Ide dibalik domain adalah bahwa domain mengidentifikasi arti atribut dan dengan demikian membuatnya kurang berarti dibandingkan dengan nilai atribut yang berasal dari domain yang berbeda bahkan jika domain tersebut mempunyai tipe yang sama. Dengan demikian WEIGHT dan MAX-DIM tidak dapat dibandingkan walaupun mereka mempunyai tipe yang sama.

Kunci relasi adalah sebuah himpunan atribut dimana nilai-nilai yang dapat digunakan untuk memilih baris dari relasi secara individu. Topik ini akan didiskusikan lebih dalam pada Chapter 3 karena topik ini sangat penting dalam penentuan relasi bentuk normal.

Struktur Logikal dan Physical

Sangat penting untuk diingat bahwa relasi adala sebuah representasi logical dan bukan representasi physical data. Relasi menggambarkan struktur data tanpa mengkuatirkan bagaimana untuk mengakses data dan bagaimana data disimpan. Representasi logical berarti bahwa dalam relasi:

- Tidak terdapat duplikasi baris
- Urutan dari baris tidak penting; dan
- Setiap kolom dalam relasi mempunyai nama unik dalam relasi tersebut

Pada awalnya analist tidak perlu kuatir tentang organisasi physical dari data. Mereka hanya kuatir tentang bagaimana menghadirkan data menggunakan relasi. Struktur physical akan menjadi penting kemudian selama perancangan physical ketika desainer harus mempertimbangkan layout physical data dan index digunakan untuk mengakses data dalam relasi.

Terdapat hal penting lainnya untuk dipertimbangkan ketika penggunaan relasi untuk data model. Ingat bahwa relasi haruslah representasi yang tidak ambigu (membingungkan). Seseorang harus dapat melihat suatu himpunan relasi dan menerangkan dari mereka tentang data user. Salah satu kebutuhan penting adalah menggunakan nama yang berarti.

Sebuah ilustrasi dari nama yang tidak bermakna ditunjukan pada Gambar 2.4(a). Struktur yang sama ditungjukan pada Gambar 2.4(b). Perbedaannya adalah bahwa Gambar 2.4(b) cendrung memberikan penjelasan diri, sementara itu gambar 2.4(a) tidak menyertakan makna yang menyeluruh tentang datanya.

Gambar 2.4

RELASI-A

X	Y	Z
P7	SALES	1 JUNE 83
P3	PRODUCTION	8 MAR 91
P5	SALES	1 FEB 81

WORK-IN

PERSON-ID	DEPARTMENT	DATE-STARTED
P7	SALES	1 JUNE 83
P3	PRODUCTION	8 MAR 91
P5	SALES	1 FEB 81

2.3 METODA YANG BAGI DALAM PENYIMPANAN DATA

Sejauh ini belum ada disebutkan bagaimana atribut dikombinasikan dalam suatu relasi. Sangat mungkin untuk mengambil sebarang atribut dan membuat relasi yang melibatkan atribut. Sebenarnya, memungkinkan untuk mengambil semua item data dalam sistem dan meletakkan mereka dalam satu relasi. Dengan demikian suatu relasi dapat mendeskripsikan keseluruhan sistem. Relasi yang demikian disebut relasi universal dalam pekerjaan secara teoritis.

Akan tetapi, kebanyakan database relasional memuat sejumlah relasi. Penting untuk menyimpan data dalam sejumlah relasi dari pada dalam satu relasi universal untuk menghilangkan redundansi. Mudah untuk memelihara kekonsistenan dalam database yang tidak mengandung redundansi. Konsistenansi biasanya berarti bahwa setiap baris dalam relasi adalah independent dari baris manapun. Kemudian memungkinkan untuk merubah sebarang baris data dalam sebarang relasi secara independent dari sebarang baris lain dalam sebarang relasi.

Untuk penjelasan lebih jauh tentang redundansi dan konsistensi, diperlukan penguraian tentang kedua istilah tersebut. Secara informal suatu fakta sering berasosiasi dengan beberapa nilai dengan sebuah objek atau berasosiasi dengan dua objek. Dalam sebuah sistem yang didiskripsikan dengan atribut, sebuah fakta ada kalau nilai dari sebuah atribut menentukan paling banyak satu nilai dari atribut yang lain. Dengan demikian DATE-OF-BIRTH adalah sebuah fakta tentang orang. Karena seseorang diidentifikasikan oleh sebuah nilai PERSON-ID, maka PERSON-ID menentukan nilai dari DATE-OF-BIRTH.

Adalah hal umum untuk membedakan secara informal nilai-tunggal fakta dengan nilai-ganda fakta. Sebuah nilai-tunggal fakta mempunyai hanya satu nilai tunggal. Sebuah nilai-ganda dapat mengambil lebih dari satu nilai. Dengan demikian DATE-OF-BIRTH adalah sebuah fakta nilai-tunggal dari PERSON-ID sebab setiap orang hanya mempunyai satu DATE-OF-BIRTH. Pada sisi lain SKILL mempunyai fakta nilai ganda karena seseorang dapat mempuyai lebih dari satu SKILL.

Sangatlah penting untuk membendakan dengan fakta turunan dan fakta dasar.

Fakta Turunan dan Fakta Dasar

Seringkali satu fakta diketahui tentang sebuah objek yang menyebabkan beberapa fakta lain tentang sebuah objek. Sebagai contoh, kita mengetahui bahwa DATE-OF-BIRTH adalah fakta tentang PERSON-ID. Kita juga mengetahui DAY-OF-WEEK untuk setiap tanggal (sebagai contoh, 5 Maret 1985 adalah hari Selasa). Dari kedua fakta

ini kita dapat menurunkan fakta bahwa seseorang yang lahir pada 5 Maret 1985 lahir pada hari Selasa. Ini adalah fakta turunan. Dabase logical seharusnya tidak menyimpan fakta turunan.

2.4 REDUNDANSI

Salah satu tujuan dari perancangan dabase logical untuk menghindari adanya redundansi dalam database. Redundansi terjadi ketika fakta yang sama disimpan lebih dari satu kali atau ketika fakta turunan disimpan. Redundansi, terpisah dari penggunaan penyimpanan yang tidak perlu, cendrung menyebabkan kesusahan dalam operasi memperbaharui data dan menyebabkan ketidakkonsistenan dalam database. Akan tetapi, harus ditunjukan bahwa kemudian dalam redundansi boleh dikenalkan dalam hal pengontrolan untuk alasan performan. Metoda ini akan didiskusikan dalam Chapter 9.

Terdapat dua bentuk redundansi: penyimpanan ganda untuk fakta yang sama atau penyimpanan fakta turunan.

Rududansi Penyimpan Ganda Fakta yang Sama

Gambar 2.5 mengilustrasikan relasi PROJECT-DATA, dimana menyipan beberapa fakta yang sama lebih dari satu kali. Dalam relasi PROJECT-DATA, PROJECT-BUDGET dari proyek dismpan lebih dari satu kali. Sebenarnya ini disimpan sebanyak terdapat orang berkerja dalam proyek tersebut. Redundant penyimpan fakta yang demikian mempunyai banyak kelemahan. Sebagai contoh:

- Jika PROJ-BUDGET untuk sebuah PROJ-NO berupah maka kita harus merubah lebih dari satu baris dalam relasi.
- Setiap kali sebuah baris baru untuk seseorang yang bekerja pada sebuah proyek ditambahkan pada sebuah proyek juga butuh untuk melihat budget proyek dan memasukannya dalam baris baris baru.
- Sebuah proyek tanpa orang ditugaskan didalamnya, seperti PROJ4, hanya akan mempunyai sebuah nilai PROJECT-BUDGET tapi tidak mempunyai nilai untuk atribut lainnya. Hal ini berarti bahwa beberapa operasi sederhana harus dilakukan secara berbeda, tergantung dari pernyataan database. Dengan demikian penambahan orang baru pada PROJ2 membutuhkan penambahan sebuah baris baru. Penambahan orang baru pada PROJ4 membutuhkan dua nilai nol.

PROJECT-DATA

PERSON-ID	PROJ-NO	PROJECT-BUDGET	TOTOL-TIME-SPENT-BY-PERSON-ON-PROJECT
P1	PROJ1	20	20
P3	PROJ1	20	16

P2	PROJ2	17	35
P2	PROJ3	84	42
P3	PROJ2	17	17
P2	PROJ1	20	83
P4	PROJ3	84	41
-	PROJ4	90	-

Redundansi dan Duplikasi

Anda seharusnya tidak bingung dengan istilah redundansi dengan duplikasi nilai. Dublikasi nilai kadang-kadang dibutuhkan dalam database sementara redundansi harus dihindari.

Gambar 2.6(a) mengilustrasikan relasi USE. Relasi ini mempunyai lebih dari satu baris dengan nilai sama di PROJECT-ID, sebagai contoh PROJ1. Ralsi ini juga mempunyai lebih satu baris dengan nilai sama dari PROJ2. Dibutuhkan untuk menyimpan nilai-nilai ini lebih dari satu kali karena setiap proyek can menggunakan lebih dari satu bagian. Dengan demikian setiap penyimpanan dari satu nilai dari PROJ1 mendeskripsikan suatu fakta berbeda. Ketika PROJ1 disimpan dalam baris sama dengan P1 maka ini menjadi bagian dari fakta yang menyatakan bagaimana banyak P1 bagian PROJ1 digunakan. Ketika PROJ1 muncul dalam baris yang sama dengan P3 maka ini menjadi bagian dari fakta yang menyatakan berbapa banyak P3 bagian PROJ1 digunakan. Dengan demikian relasi USE mempunyai beberapa duplikasi nilai tapi tidak ada redundansi penyimpanan fakta.

Gambar 2.6 Duplikasi dan Redundansi

USE

PROJ-ID	PART-NO	QTY-USED
PROJ1	P1	17
PROJ2	P2	85
PROJ1	P3	73
PROJ2	P2	80

(a) Non-redundan data

ASSIGNMENTS

PERSON-ID	DEPT	DATE-OF-BIRTH	DATE-STARTED	DATE-FINISHED
P1	SALES	1 JUN 53	2 JUN 80	5 AUG 83
P2	SALES	3 JUL 51	5 AUG 81	9 DEC 82
P3	ACCOUNTING	8 AUG 60	3 FEB 79	17 JUL 82
P1	PRODUCTION	1 JUN 53	1 JUL 82	3 FEB 85

(b) Redundant data

Dalam relasi ASSIGNMENTS, sebuah nilai dari DATE-OF-BIRTH, katakanlah 1 JUNE 53, muncul lebih dari satu kali. Setiap kemunculan seperti itu dalam kasus ini, bagaimanapun, dihubungkan dengan orang yang sama dan dengan demikian

mendeskripsikan fakta yang sama. Dengan demikian relasi ASSIGNMENTS memuat data yang redundant.

Redundansi melalui Penyimpanan Fakta Turunan

Gambar 2.7 mengilustrasikan penyimanan data turunan. Disini sesorang bekerja pada satu department, setiap departemen mempunyai satu orang manejer, dan manejer tinggal pada satu MANAGER-ADDRESS. Ketiga himpunan fakta disimpan di dalam relasi ASSIGNMENT, MANAGEMENT and LOCATION dalam Gambar 2.7.

Gambar 2.7 juga memasukan relasi ADDRESSES. Relasi ini menyimpan alamat setiap manajer. Relasi ADDRESSS menyimpan PERSON-MANAGER-ADDRESS, diman fakta nilai tunggal tentang orang – adalah alamat dimana manejer tinggal.

Akan tetapi, anda harus mencatat bahwa semua informasi dalam relasi ADDRESSES dapat diturunkan dari tiga relasi yang lain. Sebagai contoh, Joe bekerja di DEP1 dan Mr. Beck adalah manejer DEP1. Mr. Beck tinggal di Glebe. Dengan demikian manejer Joe tinggal di Glebe, yang berada pada baris 1 dalam relasi ADDRESSES. Dengan demikian relasi ADDRESSES menyimpan fakta turunan dan hal tersebut adalah redundant.

Salah satu problem dengan penyeimpanan relasi yang diturunkan adalah perobahan dalam satu relasi berakibat pada relasi lain. Sebagai contoh, konten ADDRESSES dipengaruhi oleh perobahan relasi yang lain. Dengan demikian:

- Jika seseorang pindah dari satu department ke departemen lain maka baris dalam ASSIGNMENT akan berubah. Perubahan ini juga berarti bahwa PERSON-MANAGER-ADDRESS dalam ADDRESSES untuk setiap orang akan berubah.
- Jika seorang manejer dari departemen berubah, maka a sebuah baris dalam MANAGEMENT dirubah. Perubahan ini juga berarti bahwa PERSON-MANAGER-ADDRESS untuk semua orang dalam department akan berubah.
- Jika seorang manejer baru pindah ke alamat baru maka sebuah baris dalam LOCATION akan berubah. Perubahan ini juga berarti bahwa PERSON-MANAGER-ADDRESS untuk semua orang dalam department yang dimanajeri oleh manejer tersebut juga mesti berubah.

Gambar 2.7 Relasi Turunan

WORK

PERSON	DEPT
Joe	DEP1
Jill	DEP2
Mary	DEP2
Jim	DEP3

PERSON

DEPT	MANAGER
DEP1	Mr. Beck
DEP2	Ms. Amis
DEP3	Ms. Tang

LOCATION

MANAGER	MANAGER-ADDRESSES
Ms. Amis	New town
Mr. Beck	Glebe
Ms. Tang	Glebe

ADDRESSES

PERSON	PERSON-MANAGER-ADDRESSES
Joe	Glebe
Jill	Newtown
Mary	Newtown
Jim	Glebe

2.5 MENGHILANGKAN REDUNDANSI

Redundansi dihilangkan dengan dua cara. Relasi yang menyimpan data turunan harus dibuang dalam database. Redundansi disebabkan oleh penyeimpanan fakta ganda biasanya dihilangkan dengan decomposisi.

Dekomposisi

Sebuah relasi yang menyimpan fakta sama lebih dari satu kali dapat di dekomposisi kedalam relasi yang menyimpan fakta satu kali.

Sebagai contoh, relasi PROJECT-DATA dalam gambar 2.5 di dekomposisi kedalam relasi-relasi PROJECT dan WORK seperti yang ditunjukan dalam Gambar 2.8.

Perhatikan problem yang terjadi dalam relasi PROJECT-DATA dalam GAMBAR 2.5 tidak terjadi lagi dalam relasi yang baru. Dengan demikian dalam relasi yang baru:

- Jika sebuah PROJECT-BUDGET dirobah maka hanya satu baris dalam relasi PROJECTS yang mesti dirubah.
- Ketika sebuah baris baru untuk orang yang berkeja pada sebuah proyek ditambahkan pada sebuah proyek kita tidak butuh melihat ke budget proyek.
- Sebuah proyek yang tidak mempunyai orang tidak membutuhkan perlakuan khusus.

PROJECTS

PROJ-NO	PROJECT-BUDGET
PROJ1	20
PROJ2	17
PROJ3	84
PROJ4	90

WORK

PERSON-ID	PROJ-NO	TOTAL-TIME-SPENT-BY-PERSON-ON-PROJECT
P1	PROJ1	20
P3	PROJ1	16
P2	PROJ2	35
P2	PROJ3	42
P3	PROJ2	17
P3	PROJ1	83
P4	PROJ3	41