MODUL VIII

PERANCANGAN BASIS DATA

Dalam melakukan perancangan basis data dikenal dua prosedur utama yaitu prosedur desain tingkat pertama dan tingkat kedua. Prosedur desain tingkat-pertama meliputi analisis fungsional, yang pada akhirnya merujuk kepada analisis yang harus didukung oleh model data. Walaupun terdapat perbedaan antara analisis data dan analisis fungsional, tetapi ada ketergantungan diantara keduanya. Data tidak akan dimasukkan dalam suatu model kecuali terdapat kemungkinan adanya transaksi yang akan menggunakan data dan suatu transaksi tidak akan dapat dijalankan tanpa merujuk kepada data yang akan dioperasikan. Desain basis data tingkat ke dua diperlukan untuk meningkatkan unjuk kerja model konseptual sedemikian rupa sehingga sesuai dengan batasan-batasan dari analisis fungsi yang telah dilakukan pada desain tingkat pertama. Efisiensi yang dimaksud misalnya untuk mempercepat waktu dalam menanggapi suatu transaksi.

Kompetensi dasar yang diharapkan dari materi ini adalah mahasiswa mampu melakukan perancangan basis data. Kemampuan itu dapat diukur jika mahasiswa dapat melakukan perancangan koseptual, dapat melakukan pemodelan logikal, dapat melakukan perancangan fisikal dan dapat mengimplementasikan sistem basis data.

Materi modul ini meliputi perancangan konseptual basis data. pemodelan logikal, perancangan fisikal dan implementasi sistem basis data.

A. DESAIN TINGKAT PERTAMA

Deskripsi tingkat pertama (*first-level)* digunakan karena diperlukan pekerjaan selanjutnya untuk mentransformasikan data ke dalam suatu implementasi akhir.

Prosedur Desain Tingkat Pertama

a. Buat suatu rumusan masalah dengan menggunakan ER diagram. Diagram yang digambarkan pada langkah ini akan terlihat lebih kecil dari sebenarnya dan ini gunanya untuk membantu dalam menentukan langkah awal dalam memikirkan tentang data.
b. Siapkan list untuk transaksi yang didukung model data.
c. Siapkan list untuk atribut.
d. Tuliskan dibawah daftar (list) jenis entity bersifat identify, kemudian pilih identifier untuk setiap jenis data. Untuk setiap entity tulis dibawahnya suatu tabel yang hanya memuat identifiernya.
e. Gambar suatu ER diagram yang memperlihatkan hubungan antara jenis-jenis entity, termasuk tingkat hubungannya dan kelas keanggotan secara detail.
f. Lakukan pemeriksaan apakah ER diagram benar-benar dapat membantu transaksi, ubah bila perlu. Ingat : Temukan dan tentukan connection-trap potential agar itu tidak menjadi hambatan karyawanan selanjutnya.
g. Dengan menggunakan aturan yang ada, teruskan langkah di atas menjadi tabel skeleton dan sesuaikan dengan ER diagramnya. Hapus semua atribut yang digunakan dalam identifier skeleton tabel dari daftar atribut.
h. Tambahkan atribut yang belum dimasukkan dalam daftar atribut ke dalam tabel, dan pastikan bahwa penempatan atribut sudah mengikuti aturan seperti yang ada. Jika semua atribut telah dimasukkan dalam tabel, hapus dari daftar atribut. Jika penempatan suatu atribut pada tabel akan dibuat sebagai suatu determinan yang bukan kandidat identifier dari tabel, kembalikan keduanya (atribut dan determinan) ke daftar atribut.
i. Jika setiap atribut dalam daftar atribut tidak dapat ditempatkan pada tabel, tentukan entity/hubungan selanjutnya sesuai keperluan . Periksa lagi apakah penempatan atribut sebelumnya terdapat kesalahan. Jika ada ulangi langkah 5.
j. Tentukan setiap atribut/transaksi yang harus dimasukkan dalam model ini atau dalam model pengembangan selanjutnya. Jika demikian, tambahkan pada daftar atribut dan daftar transaksi, kembali ke langkah 6 untuk menentukan transaksi yang baru dan langkah 8 untuk menentukan yang baru.
k. Periksa apakah pemilihan entity, hubungan dan atribut sudah tepat. Periksa juga apakah tabel sudah normal penuh. Demikian juga apakah semua tarnsaksi

sudah didukung pada tingkat atribut. Jika perlu diubah, ulangi seluruh prosedur.

l. Buang semua entity yang tidak perlu.

Penjelasan prosedur desain :

Langkah 3 : transaksi yang terdaftar pada langkah 2 akan diindikasikan sebagai beberapa atribut yang dianggap perlu. Jika terdapat suatu yang membahayakan dengan terlalu banyaknya atribut, hilangkan atribut pertama yang jelas memiliki kesamaan dengan atribut lain di bawahnya.

Jika suatu atribut merupakan gabungan, hal ini demi keamanan dalam unsur pokok. Misalnya kode pelanggan terdiri dari kode penjaja area dengan suatu nomor-seri-pelanggan. Tabel pelanggan (kode-pelanggan, nama-pelanggan, kode-area penjaja) terlihat sebagai normalisasi penuh, tapi kenyataannya masih berisi duplikasi redundant yaitu pada kode-area-penjaja yang merupakan salah satu manifestasi untuk menjelaskan/menegaskan dan yang lainny merupakan sesuatu yang yang tersembunyi dalam kode-pelanggan. Demikian juga tabel Pelanggan (kode-pelanggan, nama-pelanggan, nama-area-penjaja) juga terlihat normal penuh, tetapi kode-area-penjaja tersembunyi dalam nama-area-penjaja dan ini bukan merupakan kandidat-identifier dari pelanggan. Oleh karena itu beberapa atribut pada dasarnya merupakan gabungan yang potensial dan kelebihan itu dapat menyebabkan bertambahnya atribut yang sebenarnya tidak diperlukan. Suatu harga dapat disusun dalam bentuk (pounds-price, pence-price) atau tanggal dalam bentuk (tahun, bulan, hari).

Langkah 4 : Menghilangkan entity yang meragukan. Keperluan untuk entity selanjutnya akan diklakukan dengan langkah 9. Jika ditemukan kessulitan dalam memilih suatu identifier, maka dapat dimulai dengan mengiventarisasi suatu identifier# untuk penggunaan sementara. Sebagai contoh : Seorang karyawan mungkin saja diidentifikasikan dengan {nama-karyawan, alamat, tanggal-lahir}, tetapi itu akan lebih disederhanakan dengan menginventarisasi karyawan#, jika itu bukan merupakan keadaan sebenarnya. Pada beberapa kasus, nilai atribut gabungan akan berubah jika

nama-karyawan atau alamatnya berubah sehingga membuatnya tidak sesuai untuk dijadikan identifier.

Langkah 5 : Jika tingkat hubungan meragukan, apakah banyak : banyak atau 1 : banyak dan I : banyak atau 1:1, maka itu merupakan halangan yang tidak diinginkan. Perlu diingat bahwa hubungan 1:banyak lebih sederhana dibanding banyak : banyak demikian juga 1 : 1 lebih sederhana ari 1: banyak. Selain itu terdapat kemungkinan bahwa hubungan 1 : banyak pada saat tertentu dapat berubah menjadi banyak : banyak untuk jangka waktu panjang. Demikian halnya dengan 1 : banyak dapat berubah menjadi banyak : banyak .

Langkah 7 & 8 : Penghapusan atribut dapat dilakukan setelah dipastikan bahwa semua atribut sudah didistribusikan dan tidak terdapat atribut yang terlihat lebih sering tampil dari pada yang lain dalam suatu tabel. Pada langkah 8 akibat adanya suatu atribut yang ditempatkan sebelumnya pada daftar atribut akan menyebabkan suatu entity yang baru akan diperlukan.

Langkah 11 : Jika ternyata kandidat untuk sub-entity dapat dipisah, tapi akan menyebabkan setiap keputusan yang membutuhkan analisis secara detail akan tertunda.

Langkah 12 : Sampai sejauh ini, kita dapat menentukan pilihan atribut, entity dan hubungan dengan dugaan saat ini sebagai representasi enterprise yang akurat.

Contoh Desain Tingkat Pertama

Pada sebuah perpustakaan. Setiap peminjam (*barrower*) diidentifikasikan dengan peminjam#, dan setiap copy buku accesion# (perpustakaan mungkin memiliki lebih dari satu copy buku). Alamat dan nama setiap peminjam merupakan alat komunikasi, misalnya seperti keterlambatan pengembalian buku. Informasi mengenai buku seperi judul, pengarang, penerbit, tanggal publikasi, nomor buku standar internasional (ISBN), harga pembelian dan harga sekarang, dimana harga pembelian adalah harga yang dibayar diperpusakaan saat membeli dan harga sekarang adalah harga yang berlaku saat ini. Seorang peminjam boleh meminjam buku dengan jumlah terbatas, didasarkan atas klasifikasi status dewasa atau junior. Buku-buku yang dipinjam keluar

dapat dipesan oleh peminjam yang lain sesudah dkembalikan terlebih dahulu. jika edisi yag baru dari suatu buku diterbitkan, maka semua copy dari edisi sebelumnya akan dikeluarkan dari stock.

Langkah-langkah desain :

Langkah 1 : Gambarkan menurut versi yang kita miliki,

Langkah 2 : Buat daftar dari transaksi berikutnya

1. Agenda secara detail peminjam baru
2. Agenda secara detail buku-buku tambahan baru
3. Buat daftar pinjaman
4. Catatan pinjaman yang dikembalikan
5. Hapus peminjam
6. Hapus buku-buku tambahan baru
7. Buku cadangan
8. Hapus cadangan
9. Perbaiki harga yang berlaku
10. Pengirim/pemberitahuan kepada peminjam yang terlambat mengembalikan.

Langkah 3 : Buat daftar atribut, yaitu :

peminjam#, buku tambahan#, nama peminjam, alamat peminjam, judul, nama pengarang, nama penerbit, tanggal publikasi, ISBN, harga pembelian, harga sekarang, batas waktu peminjam, status peminjam, batas pinjaman, waktu tersisa.

Beberapa atribut mungkin saja dapat dijadikan sebagai composit. nama peminjam dapat digunakan nama keluarga/marga dilengkapi degan inisialnya, dan alamat peminjam dapat dimasukan ke dalam beberapa baris alamat akan tetapi secara keseluruhan tidak akan menimbulkan perubahan yang berarti pada struktur model data. Atribut yang paling dibutuhkan adalah ISBN, suatu ISBN terdiri dari 4 bagian yang menunjukan. bahasa penerbitan, penerbit, nomor seri dan angka pengenal. Contohnya ISBN 0-7131-2815-1, yang berarti dipublikasikan dalam bahas Inggris (0), diterbitkan oleh Edward Arnold (7131), nomer seri 2815, dan angka pengenal 1. Diasumsikan bahwa stock

perpustakaan hanya terdiri dari buku-buku bahasa Inggris, berarti bagian pertama dapat dihilangkan. Angka pengenal yang merupakan tanda untuk verifikasi harus lebih baik karena merupakan ciri khusus dari struktur data, akan tetapi ini dapat dihilangkan dari desain tingkat pertama. Dengan dihilangkannya bagian di atas berarti masih terdapat 2 bagian yang tersisa. Nama atribut diganti dengan ISBNX untuk mengindikasikan bahwa itu bukan merupakan full ISBN. Untuk menghindari kesulitan yang mungkin terjadi, ISBNX yang terdiri dari (kode penerbit, serial#) akan digunakan dalam prosedur desain. Tetapi akan lebih aman jika digunakan atribut individual untuk menyusun tabel full normalisasi, lalu kemudian menambah ISBNX untuk (kode penerbit,serial#) jika bentuk ini menjadi lebih baik.

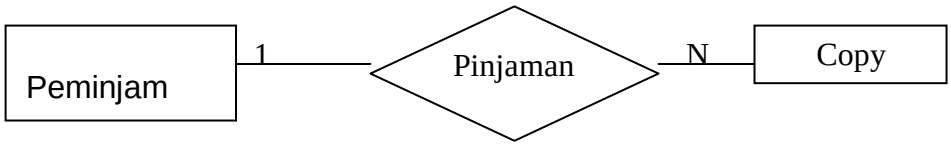
Langkah 4 : Identifier peminjam# dan buku tambahan# pada dasarnya menunjukan bahwa peminjam dan copy adalah kemungkinan akan menjadi pilihan yang sesuai untuk jenis entity. Pada tingkat ini akan menjadi pilihan yang sesuai untuk jenis entity. Pada tingkat ini memunginkan memilih entity lainnya. Tetapi untuk sementara waktu kita bekerja hanya dengan dua, yaitu entity peminjam (peminjam#,.....) dan copy (buku tambahan#, ...).

Langkah 5 : Seorang peminjam boleh meminjam beberapa copy sekaligus, tetapi setiap copy dapat dipinjam oleh lebih dari satu peminjam. Suatu copy tidak dipinjam, atau seorang peminjam tidak meminjam satu copy pun. ER diagram seperti pada gambar 54.

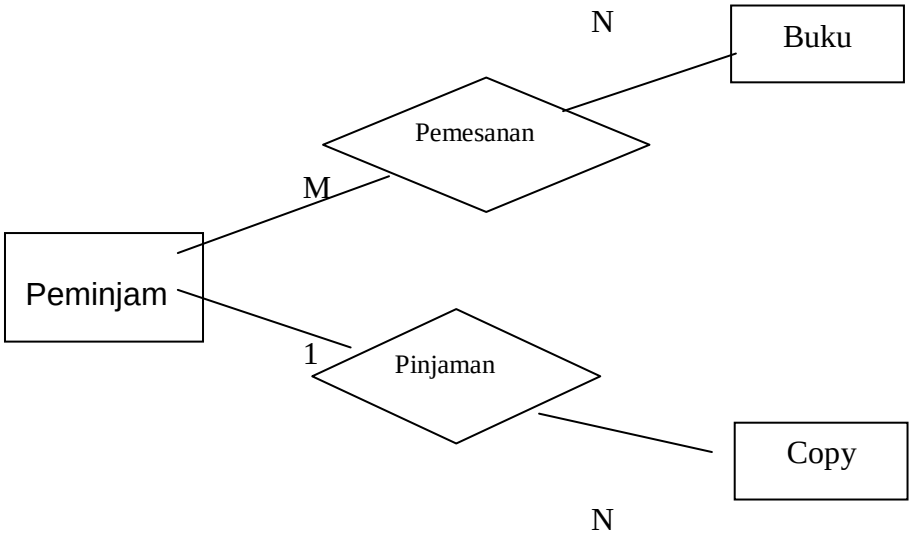
Langkah 6: Transaksi 1, 2, 5 dan 6 kelihatannya seperti disediakan untuk membuat dan menghilangkan tampilan peminjam dan copy. Peminjam transaksi (3 dan 4) dapat diproses melalui relationship pinjaman. Pada kenyataannnya, prosesnya tidak sesederhana itu, karena akan lebih banyak analisis detil seperti pada langkah 11. Harga sekarang mungkin dapat berfungsi sebagai atribut dari copy, sehingga transaksi 9 dapat dilakukan. informasi yang diperlukan untuk pinjaman yang terlambat kembali (transaksi 10) kemungkinan dapat ditemukan dengan menggunakan copy. pinjaman dan peminjam. hanya pada problem yang merupakan cadangan (transaksi 7 dan 8), misalnya mungkin seorang peminjam memesan beberapa buku cadangan, dan diisi lain buku yang sama sudah dicadangkan untuk beberapa pemesan sehingga model seperti pada gambar

54.a akan terdiri dari pengulanagan (repeating groups). Langkah penyelesaian pertama adalah menambahkan relationship cadangan di antara peminjam dan copy, akan tetapi sebenarnya ini tidak menguntungkan karena peminjam tidak dicadangkan suatu copy khusus tetapi buku khusus. Jika terdapat beberapa copy dari buku yang sama, maka itu tidak bisa diterima oleh peminjam. kelemahan dari diagram ER seperti ini untuk mendukung transaksi pemesanan secara jelas dan memisahkan antar buku dan copy. Bukan bukunya yang dipinjamkan, melainkan copynya. Diagram dapat diperbaiki dengan memasukan entity buku dan relationship pemesanan (gambar 54.b), tetapi ini akan menimbulkan conection trap antara copy dengan buku. Jika copy merupakan pengembalian dari peminjam, maka conection trap yang terjadi harus dihilangkan. Hal ini dapat dilakukan dengan menghubungkan buku ke copy melalui relationship Stock (gambar 54.c). diasumsikan bahwa buku diidentifikasikan dengan ISBNX, akan tetapi persoalan pada langkah 7 dan 8 dapat menimbulkan kesulitan.

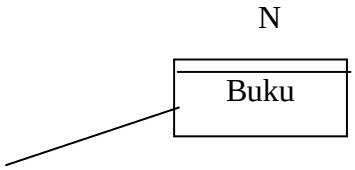
(a)

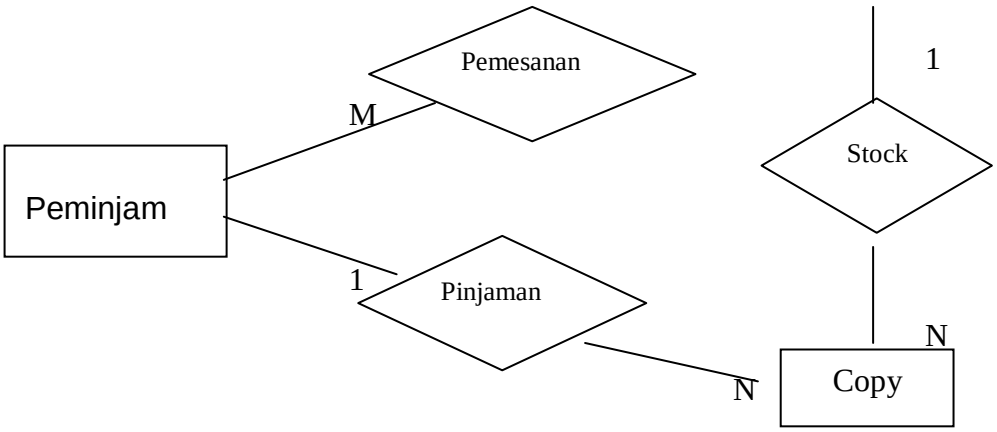


(b)



(c)





Gambar 54. Pengembangan ER diagram (langkah 5& 6)

Langkah 7 : Tabel skeletonnya adalah :

Tabel entity *Tabel relationship*

Peminjam (peminjam#, ...) Pinjaman (buku-tambahan#, peminjam#, ...)

Copy (buku-tambahan#, ISBNX, ..) Pemesanan (peminjam#, ISBNX, ...)

Buku (ISBNX, ...)

Langkah 8 : Penempatan atribut :

Peminjam (peminjam#, nama-peminjam, alamat-peminjam)

Copy (buku tambahan#, ISBNX, harga-pembelian)

Buku (ISBNX, judul, tanggal publikasi, harga-sekarang)

Pinjaman (buku tambahan#, peminjam#, tanggal-pinjaman)

Pemesanan (peminjam#, ISBNX, tanggal –pemesanan)

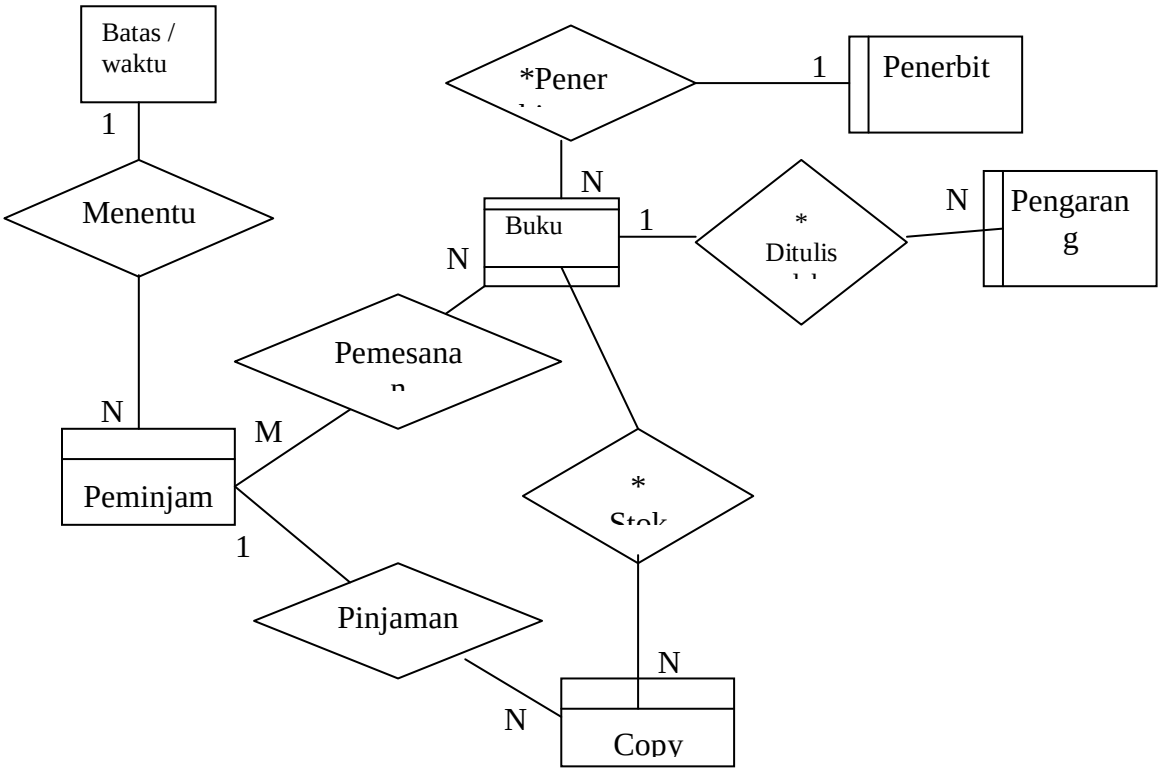
Berlawanan dengan asumsi pada langkah 6, harga sekarang tidak merupakan atribut dari copy. Perlu diingat bahwa batas waktu peminjaman ditempatkan sebelum status peminjam, karena yang terakhir akan menjadi suatu determinan dari batas suatu peminjaman tetapi bukan kandidat identifier dari peminjam. dengan alasan yang sama terdapat dua atribut yang dikembalikan ke daftar atribut jika status peminjam telah berakhir sebelum batas waktu peminjaman. Dengan asumsi bahwa hanya ada satu nama penerbit untuk setiap buku, akan memudahkan untuk menempatkan nama penerbit pada tabel buku. Sehingga code penerbit yang bisa dimasukan dalam ISBNX akan menjadi determinan dari nama penerbit, karena nilai dari nama penerbit akan bersifat duplikasi redundan jika terdapat beberapa copy buku yang sama. Nama penerbit harus ditempatkan disebelah kiri pada daftar atribut untuk sementara waktu.

Langkah 9: Atribut yang belum ditempatkan adalah nama pengarang, status peminjam, batas waktu pinjaman dan penerbit. Nama pengarang tidak harus ditempatkan dalam

buku, karena akan menimbulkan pengulangan (repeating group). Jika atribut dari pengarang lain tidak perlu dibukukan, maka dengan menerangkan authorship sebagai sub entity dari buku sudah dianggap cukup.

Pada contoh yang diberikan pada langkah 8, atribut status peminjam dan batas pinjaman tidak bisa ditempatkan dalam peminjam. Untuk ini dibutuhkan entiti baru, yaitu batas waktu. Jika status peminjam sudah tidak memenuhi syarat sebagai atribut, batas waktu peminjaman akan dimasukan pada peminjam pada status yang sama dana implikasi yang ditimbulkannya adalah bahwa status peminjam akan termasuk dalam model. Batas peminjaman kemudian harus dihapus dari gambar peminjam dan dikembalikan ke dafatar atribut.

Nama penerbit dapat dihapus dengan membuat entity penerbit, yang diidentifikasikan sebagai code penerbit. relationship ini dapat direperesentasikan pada tabel dengan menempatkan code penerbit dalam ISBNX. Model termasuk authorship, batas waktu dan penerbit dapat dilihat pada gambar 55. Kelas keanggotaan dari penerbit dalam penerbitan merupakan obligatory, dengan asumsi bahwa hanya penerbit yang terdaftar yang akan dibukukan sebagai penerbit dari buku yang terdapat diperpustakaan.



Peminjam (peminjam#, nama peminjam, alamat peminjam, status peminjam)

Copy (buku tambahan#, ISBNX, harga-pembelian)
Buku (ISBNX,judul, tanggal publikasi, harga-sekarang)
Batas/waktu (status-peminjam, batas/waktu peminjaman)
Pengarang (ISBNX, nama-pengarang)
Penerbit (code penerbit, nama penerbit)
Pemesanan (peminjam#, ISBNX, tanggal pemesanan)

Gambar 55. ER model lengkap tingkat pertama

Langkah 10 : Transaksi lain yang akan dimasukkan :

1) Simpan data lengkap dari buku baru
2) Pengahapusan buku
3) Pemberitahuan kepada peminjam bahwa cadangan saat ini ada stock
4) Rubah ststus peminjam
5) Temukan buku-buku yang ditulis oleh pengarang yang menyumbang.

Model nampaknya mampu untuk mendukung transaksi-transaksi tersebut.

Langkah 11 : setiap transaksi harus dianalisa secara detil sehingga kebenaran dari model dapat dicek. Contohnya peminjam mungkin saja dicatat secara sederhana dengan penyimpan pinjaman buku, alternatifnya adalah tabel peminjam dan copy harus diperiksa pertama kali untuk memastikan bahwa buku tambahan# dan peminjam# sudah benar. Penghapusan copy terakhir dari buku sumbangan memperlihatkan bahwa penghapusan tidak hanya terhadap tampilan copy (dalam kontek ER modeling, acuannya adalah tampilan baris, bukan tampilan tabel lengkap), tetapi juga tampilan buku yang berasosiasi dan mungkin saja termasuk penerbit dan pengarangnya, ditambah lagi tabel untuk memeriksa pinjaman dan pemesanan yang mengacu kepada buku tambahan# atau ISBNX dari copy.

Salah satu cara yang sangat berguna untuk memeriksa/memastikan bahwa semua transaksi dapat didukung pada tingkat atribut adalah dengan menggambar suatu grid transaksi/atribut (gambar 56). code yang digunakan pada grid adalah :

D-hapus R-panggil S-simpan U-perbaiki

		Transaksi					
Entity/ Relationship	Atribut	Pinjaman		Kembali		Status	
Peminjam	peminjam# Nama-peminjam alamat-peminjam status peminjam	R1 R		R2 R R		R4 R R	 U2
Copy	buku-tambahan# ISBNX harga-pembelian	R2		R7 R			
Buku	ISBNX judul tanggal publikasi harga sekarang			R5 R			
Batas/waktu	status peminjam batas-pinjaman	R3 R					
Pengarang	ISBNX nama-pengarang			R6* R			
Penerbit	code-penerbit nama-penerbit						
Pinjaman	buku-tambahan# peminjam# tanggal-pinjaman	 R4*	S5 S5 S5	D1 D D			
Pemesanan	peminjam# ISBNX tanggal-pemesanan			R R3 R3	D7 D7 D		

Gambar 56. Grid transaksi /atribut

Tanda (*) memberi indikasi bahwa akses kepada beberapa baris dalam satu tabel mungkin diperlukan pada sebagian dari transaksi. Angka indeks memperlihatkan bahwa perintah yang ada ditabel segera diakses. Untuk memanggil kembali dan menghapus indeks yaitu dengan mengaplikasikan ke atribut, digunakan sebagai masukan titik ke tabel. Untuk menyimpan dan memperbaiki indeks yaitu dengan mengaplikasikannya pada semua atribut yang terlibat dalam pelaksanaan. Suatu transaksi yang diakses dari tabel yang sama pada transaksi yang berbeda akan memerlukan kolom lebih dari satu. Mekanisme secara fisik yang sebenarnya untuk memproses transaksi pada dasarnya tidak layak dilakukan, tetapi hanya bersifat sementara untuk menanggulangi agar model menjadi layak untuk suatu transaksi. Dengan demikian, jika suatu transaksi adalah untuk memeriksa status peminjam, misalnya peminjam#, pencatuman grid hanya untuk tujuan membuat peminjam# dan status peminjam, walaupun pada akhirnya ternyata bahwa suatu catatan (record) akan memuat semua detil tentang peminjam yang harus dipanggil kembali secara fisik.

Contoh berikut didasarkan pada gambar 56. Komentar atas transaksi Pinjaman lebih detil dari komentar untuk pengembalian dan status.

Pinjaman :

Menggunakan peminjam# jalan masuk ke peminjam untuk memeriksa validitas peminjam# dan mendapatkan status peminjam. Penggunaan buku tambahan# sebagai jalan masuk ke copy untuk memeriksa validitas buku-tambahan#. Penggunaan status peminjam sebagai jalan masuk ke batas/waktu untuk menemukan batas pinjaman. Penggunaan peminjam# secara berulang sebagai jalan masuk ke peminjam adalah menemukan jumlah copy peminjam yang mendapatkan pinjaman. Jika peminjam tidak melebihi batas/waktu yang telah ditentukan, maka peminjam baru tidak perlu ditampilkan.

Pengembalian:

(Jika diasumsikan bahwa suatu transaksi akan masuk didalamnya, maka jika sewaktu-waktu dapat diapikasikan, catat itu sebagai pemesanan yang siap pakai). Hapus tampilan pinjaman. temukan ISBNX untuk buku tambahan#. temukan peminjam, jika ada maka

buat tanggal pemesanan pertama untuk buku. Temukan nama peminjam dan alamatnya. Temukan judul dan pengarang, setelah itu hapus pemesanan.

Status :
Penggunaan peminjam# untuk menemukan nama peminjam (untuk memeriksa peminjam secara teliti) dan untuk memeriksa status peminjam sebelumnya. jika perlu, perbaiki status peminjam.

Langkah 12 : setiap tabel entity terdiri dari minimal satu atribut sebagai tambahan identifiernya, sehingga tidak akan ditemukan tabel yang berlebih (superfluous). Jika pengarang diidentifikasikan dengan nama pengarang, maka itu berarti bahwa itu merupakan suatu entity. Tabel pengarang mungkin akan dianggap sebagai kelebihan.

Sebagai tambahan, jika pekerjaan dilakukan berdasarkan skenario maka hal itu merupakan suatu gagasan yang bagus dengan menggaris bawahi pemilihan entity relationship dan atribut dengan menggunakan warna yang berbeda untuk setiap katagori.

B. DESAIN BASIS DATA TINGKAT KEDUA

Desain basis data tingkat ke dua diperlukan untuk meningkatkan unjuk kerja model konseptual sedemikian rupa sehingga sesuai dengan batasan-batasan dari analisis fungsi yang telah dilakukan pada desain tingkat pertama. Efisiensi yang dimaksud misalnya untuk mempercepat waktu dalam menanggapi suatu transaksi.

Sebenarnya desain tingkat kedua, secara fakta dapat dikatakan sebagai dasar untuk desain konseptual. yaitu pemetaan skema eksternal melalui skema konseptual ke skema internal. Jadi setiap skema memiliki pandangan (view) yang berbeda tentang data.

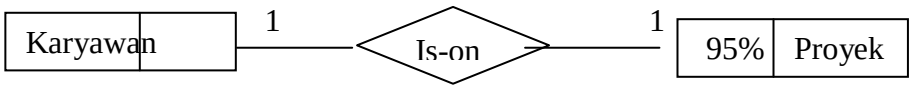
Untuk mewujudkan semua itu diperlukan FLEXING, yaitu : (a) dengan menghilangkan tabel yang tidak perlu, (b) dengan memisahkan (spliting) atribut bernilai null menjadi sub entitas.

1. Flexing dengan Elimainasi Tabel

Flexing dengan menghilangkan tabel yang tidak perlu,harus mempertimbangkan derajat keanggotaan entitas. Hubungan ini meliputi hubungan 1 : 1, 1 : banyak dan banyak : banyak.

a. Hubungan 1 :1,

Jika keduanya non obligatory, diperlukan 3 tabel, sehingga mengurangi unjuk kerja. Jumlah tabel dapat dikurangi dengan membuat anggapan tertentu tentang klas keanggotaan suatu entitas.



Karyawan (karyawan#, nama-karyawan)

Proyek (proyek#, nama proyek, waktu mulai proyek)

Is-on (karyawan#, proyek#)

Gambar 57. Model ER tingkat pertama

Dengan menggangap satu dari dua entitas tersebut sebagai obligatory, cukup 2 tabel untuk menghubungkan entitas.

Karyawan (karyawan#, nama-karyawan)

Proyek (proyek#, nama proyek, waktu mulai proyek, karyawan#,)

Untuk menghemat ukuran penyimpanan, serta akses lebih cepat perlu diperhatikan ketentuan seperti di bawah ini :

1) Bila ke dua entitas dianggap sebagai obligatory, hubungan antar entitas cukup satu tabel saja.

   Karyawan- Proyek (karyawan#, nama-karyawan, proyek#, nama proyek, waktu mulai proyek)

   Ruang simpan lebih besar, akses lebih lambat untuk transaksi (a) karena tabelnya lebih besar, (b) banyak transfer atau data superflous, (c) dua entitas itu tak mungkin disimpan dalam kedua bagian terpisah.

2) Dengan melakukan post satu identifier dan beberapa atribut dari satu entitas ke entitas lain.
   Karyawan (karyawan#, nama-karyawan, proyek, nama-proyek)

   Proyek (proyek#, waktu mulai proyek)

   90% dari nilai proyek# dan nama proyek akan bernilai null.

b. Hubungan 1 : banyak

Teknik flexing untuk hubungan 1 : banyak tetap menggunakan anggapan tertentu terhadap klas keanggotaan suatu entitas.

- Pada desain tingkat pertama, diperlukan tabel penghubung untuk menjelaskan hubungan ke dua entitas karen entitas “banyak” merupakan non obligatory.
- Dengan flexing entitas “banyak” dianggap sebagai obligatory, perlu 2 tabel.
- Tabel diperoleh dengan melakukan post identifier dari entitas non obligatory ke entitas yang dianggap sebagai obligatory, tabel relasi tidak diperlukan.
- Bila kejadian pada relasi 1: banyak, sebenarnya 1:1 dan kedua entitas adalah obligatory, maka relasi diperlukan seperti relasi 1:1 yang obligatory penuh.

- Kalau relasi 1 : banyak sebenarnya adalah 1 : few, maka entitas few di posted.

keuntungan : akses lebih cepat,

kerugian : repeating group pada entitas yang diposted.

Contoh :

Karyawan (karyawan#, nama-karyawan)

Proyek (proyek#, nama proyek, waktu mulai proyek, karyawan-senior#, karyawan-yunior#,)

Cara lain adalah dengan membuat sub entitas guna menghindarkan terjadinya repeating grup. Entitas yang dapat dipecah menjadi beberapa sub entitas adalah yang memiliki klas keanggotaan non obligatory, yaitu entitas "many". Sub entitas yang dibuat dapat dihapus bila ternyata atribut yang menjadi penyusunnya lebih diperlukan untuk menjelaskan entitas.

Contoh : kualifikasi karyawan dipecah menjadi sub entitas

Kualifikasi-Karyawan (karyawan#, nama-kualifikasi, waktu diperlukan)

Detail kualifikasi ini bila dimasukan ke entitas karyawan akan menjadi repeating group.

Karyawan (karyawan#, nama-karyawan),

Kualifikasi (nama-kualifikasi, waktu diperlukan)

c. Hubungan banyak : banyak

Bila hubungan banyak:banyak dapat dianggap sebagai few : banyak, tabel hubungan dapat dibentuk dengan membuat post identifier dari entitas few ke entitas banyak sebagai repeting groups.

Cara yang serupa dapat diterapkan untuk hubungan few : few, 1 : few, few : banyak.

Cara flexing yang lain adalah dengan menempatkan atribut dari salah satu tabel entitas ke tabel penghubung, sebagai fasilitator akses data.

Contoh :

Karyawan (karyawan#, nama-karyawann)

Is-on (karyawan#, proyek#, nama-proyek, waktu mulai proyek)

2. Flexing dengan Splitting

a. Flexing dengan splitting maksudnya adalah memisahkan atribut bernilai null menjadi sub-entitas.
b. Pemisahan sub entitas telah dilakukan pada desain tingkat pertama, namun pada desain tingkat ke dua ini harus dilakukan pemeriksaan ulang terhadap atribut dan kelompok atibut yang mungkin memiliki nilai null.
c. Atribut-atribut itu kemudian dipisahkan menjadi sub entitas.
d. Pemisahan harus tetap mengacu kepada tabel entitasnya, apakah masih dapat dipecah menjadi beberapa bagian atau tidak.

Contoh :

Mesin (mesin#, jenis, waktu diperlukan, pusat kerja, kapasitas, waktu perbaikan, waktu terakhir rusak).

Bila ada 2 transaksi :

Mesin#, jenis, pusat kerja, kapasitas

Mesin#, waktu diperlukan, waktu perbaikan, waktu terakhir rusak.

Tabel dapat dipecah menjadi :

Mesin –loading (mesin#, jenis, pusat kerja, kapasitas)

Mesin –detil (mesin#, waktu diperlukan, waktu perbaikan, waktu terakhir rusak)

Ukuran simpan lebih kecil, sedikit tranfer data saat akses baris, baris yang ditangani dimemori lebih banyak.

3. Atribut Turunan

Untuk meningkatkan unjuk kerja (*performance*), mempercepat akses ke jalur data. Untuk itu perlu diberikan atribut tambahan pada tabel. Fungsi atribut tambahan adalah sebagai tanda untuk mengingatkan apakah akses ke tabel yang lain diperlukan, atau untuk mencegah terjadinya karyawanan yang tidak perlu.

Contoh :

Nilai yang diturunkan dari order total mungkin dipengaruhi perubahan harga, sehingga perlu disimpan nilai order total sebelum perubahan harga.

Bila terjadi redundan atribut, seperti tabel-tabel berikut :

Karyawan (karyawan#, nama karyawan, proyek, nama proyek)

Proyek (proyek#, nama-proyek, waktu mulai proyek).

4. Contoh desain Tingkat Kedua

Dari kasus 56 dan 57 :

Tinjauan tabel Pinjaman. relasi 1 : banyak , tebel pinjaman dapat dieleminasi dengan mem-posted-kan peminjam#, dan waktu pinjaman ke copy :

Copy (buku tambahan#, ISBNX, harga pembelian, peminjam, waktu pinjaman)

Kalau copy an yang sedang dipinjam prosentasenya tinggi, *posting* akan mengurangi ukuran penyimpanan, karena buku tambahan# tidak duplikasi pada copy dan pinjaman.

Sebaliknya ukuran penyimpanan menjadi tingg karena ada banyal null untuk peminjam# dan tanggal-pinjam.

Pengarang, kalau diasumsikan nilai nama-pengarang per buku sampai tiga saja, entitas pengarang dapat dikonsolidasikan ke buku, atau pengarang pertama saja yang ke buku, sisanya tetap di pengarang.

Atribut tanda, pengarang tunggal pada buku dapat dijadikan indkasi untuk perlunya tabel pengarang.

Batas, akses ke batas pinjaman ke peminjam dapat disederhanakan dengan menempatkannya batas pinjaman ke peminjam. tabel batas (limit) dapat dihilangkan, tetapi karena tabel limit tidak terlalu menghabiskan memori dan justru menghilangkan keluwesan sistem, maka tabel limit tetap dipertahankan.

Buku dan copy, untuk kebanyakan buku hanya satu yang distok, jadi stok memiliki relasi 1 : 1 dengan demikian tabel buku dan copy menjadi :

Copy (buku tambahan#, ISBNX, judul, tanggal publikasi, harga pembelian, harga-sekarang).

Judul, tanggal publikasi dan harga sekarang dapat menjadi redundan kalau banyak copyan buku stok. Ukuran copy dapat direduksi dengan spliting atribut-atribut tersebut.

Copy (buku tambahan#, ISBNX, judul)

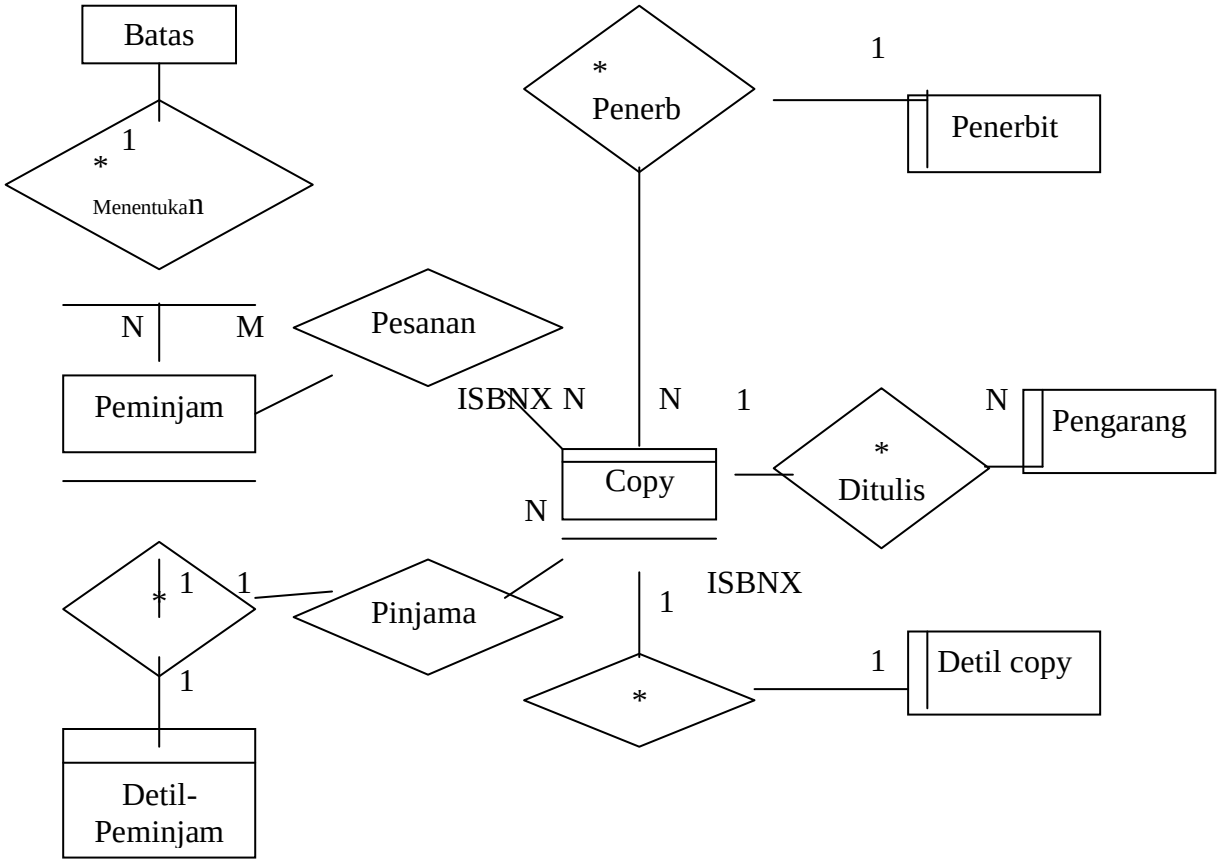
Detail copyan (buku tambahan#, tanggal publikasi, harga pembelian, harga sekarang)

Peminjam, nama peminjam dan alamat peminjam tidak selalu diperlukan, jadi dapat juga displit untuk mempercepat transaksi Pinjaman.

Atribut tambahan, dalam proses Pinjaman diperlukan banyak waktu untuk menghitung jumlah copy yang sudah dipinjam. Ini dapat dipersingkat dengan menambah tribut

hitungan pinjaman pada peminjam. Atribut tanda-pemesanan pada pinjaman juga dapat digunakan untuk mempercepat transaksi ini.

Konsolidasi model tingkat ke dua, dapat disajikan dalam gambar 57 berikut.



Gambar 58. Model lengkap ER tingkat ke dua

Peminjam (peminjam#, status peminjam, jumlah pinjaman)

Detil-Peminjam (peminjam#, nama peminjam, alamat peminjam)

Copy (buku tambahan#, ISBNX, judul)

Detail Copy (buku tambahan#, harga pembelian, waktu publikasi, harga sekarang)

Batas/wkatu (staus-peminjam, batas/waktu peminjaman)

Pengarang (ISBNX, nama-pengarang)

Penerbit (code penerbit, nama penerbit)

Pinjaman (buku tambahan#, peminjam, waktu pionjam, tanda-pemesanan)

Pemesanan (peminjam#, ISBNX, tanggal pemesanan)

Latihan 7

1. Jelaskan apa yang dimaksud dengan perancangan basis data tingkat pertama dan kedua!
2. Uraian dengan singkat prosedur perancangan basis data tingkat pertama!
3. Apa yang Anda ketahui tentang flexing dalam desain tingkat kedua?
4. Buatkan sebuah model lengkap ER tingkat ke dua!
5. Berdasar no 4 buatkan tabel skletonnya!

Rangkuman

Perancangan basis data tingkat pertama merupakan desain awal dalam suatu kasus perancangan basis data yangbisa dieskripsikan menurut versi pembuat atau pemakai. Sedangkan desain tingkat kedua, secara fakta dapat dikatakan sebagai dasar untuk desain konseptual. yaitu pemetaan skema eksternal melalui skema konseptual ke skema internal. Jadi setiap skema memiliki pandangan (view) yang berbeda tentang data. Untuk mewujudkan semua itu diperlukan FLEXING, yaitu : (a) dengan menghilangkan tabel yang tidak perlu, (b) dengan memisahkan (spliting) atribut bernilai null menjadi sub entitas.

Tes Formatif

Pilihlah B apabila jawaban Benar, dan S apabila jawaban Salah

1. Pada perancangan basis data, desain tingkat-pertama digunakan karena diperlukan pekerjaan selanjutnya untuk mentransformasikan data ke dalam suatu implementasi akhir. (B/S)

2. Untuk analisis data, prosedur desain tingkat-pertama meliputi analisis fungsional, yang pada akhirnya merujuk kepada analisis yang harus didukung oleh model data. (B/S) .B.
3. Terdapat perbedaan antara analisis data dan analisis fungsional, sehingga masing-masing akan berdiri sendiri-sendiri. (B/S)
4. Suatu rumusan masalah dengan menggunakan E-R diagram, diagram yang digambarkan pada langkah ini akan terlihat lebih kecil dari sebenarnya. (B/S)
5. Desain basis data tingkat kedua diperlukan untuk meningkatkan unjuk kerja model konseptual. (B/S)
6. Sebenarnya desain tingkat kedua, secara fakta dapat dikatakan sebagai dasar untuk desain operasional. (B/S)
7. Yang dimaksud dengan *flexing*, yaitu menghilangkan tabel yang tidak perlu dan memisahkan atribut bernilai nol menjadi sub entitas. (B/S)
8. *Flexing* dengan menghilangkan tabel yang tidak perlu, harus mempertimbangkan derajat keanggotaan entitas. (B/S)
9. Sub entitas yang dibuat dapat dihapus bila ternyata atribut yang menjadi penyusunnya lebih diperlukan untuk menjelaskan entitas. (B/S)
10. Fungsi atribut tambahan adalah sebagai tanda untuk mengingatkan apakah akses ke tabel yang lain diperlukan. (B/S)

Cocokkan jawaban saudara dengan kunci jawaban Tes Formatif yang terdapat pada bagian akhir modul ini. Hitunglah jawaban saudara yang benar. Kemudian gunakan rumus di bawah ini untuk mengetahui tingkat penguasaan saudara terhadap materi kegiatan belajar ini.

Rumus :

$$\text{Tingkat Penguasaan} = \frac{\text{Jumlah jawaban saudara yang benar}}{10} \times 100\ \%$$

Arti tingkat penguasaan yang saudara peroleh adalah :

90 - 100 % = Baik Sekali;

80 - 90 % = Baik;

70 - 80 % = Cukup;

< 70 % = Kurang.

Bila saudara memperoleh tingkat penguasaan 80 % atau lebih, saudara dapat meneruskan dengan kegiatan belajar (modul) selanjutnya. Sedangkan jika tingkat penguasaan saudara masih berada di bawah 80 %, saudara diwajibkan mengulangi kegiatan belajar (modul) ni, terutama bagian yang belum saudara kuasai secara baik.

DAFTAR PUSTAKA

LREPP – II/B. 1995. *Pedoman Digitasi Untuk Pemetaan Tata Guna Tanah Berdasarkan Perangkat Lunak Arc/Info*, BPN & Consultant Association GTZ ITC-LAPI MES, Jakarta.

Muryono, Slamet. 2003. *Cartographic Modelling and Its Application To Allocate Suitable Land For Rural Settlement*, Majalah Ilmiah Widya Bhumi Sekolah Tinggi Pertanahan Nasional Yogyakarta, Nomor 9 Tahun 4, Februari 2003

Prahasta Eddy. 2001. *Konsep-konsep Dasar Sistem Informasi Geografis*, Informatika, Bandung.

Suyudi, Bambang. Tanpa Tahun. *Penyusunan Basis Data Topografi Digital Indonesia*, Makalah, Tidak Dipublikasikan.