BAB 9

Normalisasi

Ketika kita merancang suatu basis data untuk suatu sistem relational, prioritas utama dalam mengembangkan model data logical adalah dengan merancang suatu epresentasi data yang tepat bagi relationship dan constrainnya (batasannya). Kita harus mengidentifikasi suatu set relasi yang cocok, demi mencapai tujuan di atas. Tehnik yang dapat kita gunakan untuk membantu mengidetifikasi relasi-relasi tersebut dianamakan Normalisasi.

Proses normalisasi pertama kali diperkenalkan oleh E.F.Codd pada tahun 1972. normalisasi sering dilakukan sebagai suatu uji coba pada suatu relasi secara berkelanjutan untuk menentukan apakah relasi tersebut sudah baik atau masih melanggar aturan-aturan standar yang diperlakukan pada suatu relasi yang normal (sudah dapat dilakukan proses insert, update, delete, dan modify pada satu atau beberapa atribut tanpa mempengaruhi integritas data dalam relasi tersebut).

Proses normalisasi merupakan metode yang formal/standar dalam mengidentifikasi dasar relasi bagi primary keynya (atau candidate key dalam kasus BCNF), dan dependensi fungsional diantara atribut-atribut dari relasi tersebut. Normalisasi akan membantu perancang basis data dengan menyediakan suatu uji coba yang berurut yang dapat diimplementasikan pada hubungnan individualshingga skema relasi dapat di normalisasi ke dalam bentuk yang lebih spesifik untuk menghindari terjadinya error atau inkonsistansi data, bila dilakuan update tehadap relasi tersebut dengan Anomaly.

9.1 BEBERAPA DEFINISI NORMALISASI

- Normalisasi adalah suatu proses memperbaiki / membangun dengan model data relasional, dan secara umum lebih tepat dikoneksikan dengan model data logika.
- Normalisasi adalah proses pengelompokan data ke dalam bentuk tabel atau relasi atau file untuk menyatakan entitas dan hubungan mereka sehingga terwujud satu bentuk database yang mudah untuk dimodifikasi.
- Normalisasi dapat berguna dalam menjawab 2 pertanyaan mendasar yaitu: “apa yang dimaksud dengan desain database logical?” dan “apa yang dimaksud dengan desain database fisikal yang baik? What is phisical good logical database design?”.
- Normalisasi adalah suatu proses untuk mengidentifikasi “tabel” kelompok atribut yang memiliki ketergantungan yang sangat tinggi antara satu atribut dengan atrubut lainnya.

- Normalisasi bisa disebut jga sebagai proses pengelompokan atribut-atribut dari suatu relasi sehingga membentuk WELL STRUCTURED RELATION.

  WELL STRUCTURED RELATION adalah sebuah relasi yang jumlah kerangkapan datanya sedikit (Minimum Amount Of Redundancy), serta memberikan kemungkinan bagi used untuk melakukan INSERT, DELETE, MODIFY, terhadap baris-baris data pada relasi tersebut, yang tidak berakibat terjadinya ERROR atau INKONSISTENSI DATA, yang disebabkan oleh operasi-operasi tersebut.

  Contoh:

  Terdapat sebuah relasi Mahasiswa, dengan ketentuan sebagai berikut.
  - Setiap Mahasiswa hanya boleh mengambil satu mata kuliah saja.
  - Setiap matakuliah mempunyai uang kuliah yang standar (tidak tergantung pada mahasiswa yang mengambil matakuliah tersebut).

Tabel 9.1 *Relasi Kuliah*

NIM	KODE-MTK	BIAYA
92130	CS-200	75
92200	CS-300	100
92250	CS-200	75
92425	CS-400	150
92500	CS-300	100
92575	CD-500	50

Relasi Kuliah di atas merupakan sebuah relasi yang sederhana dan terdiri dari 3 kolom / atribut. Bila diteliti secara seksama, maka akan ditemukan redundancy pada datanya, dimana biaya kuliah selalu berulang pada setiap mahasiswa. Akibatnya besar kemungkinan terjadi error atau inkonsistensi data, bila dilakukan update terhadap relasi tersebut dengan Anomaly.

Anomaly merupakan penyimpangan-penyimpangan atau error atau inkonsistensi data yang terjadi pada saat dilakukan proses delete, insert ataupun modify dalam suatu basis data.

9.2 MACAM-MACAM PENYIMPANGAN (ANOMALY)

9.2.1 INSERTION ANOMALY

Insertion Anomaly, merupakan error atau kesalahan yang terjadi sebagai akibat dari operasi menyisipkan (insert) tuple / record pada sebuah relasi.

Contoh: ada matakuliah baru (CS-600) yang akan diajarkan, maka matakuliah\ tersebut tidak bisa di insert / disisipkan ke dalam Relasi Kuliah di atas sampai ada mahasiswa yang mengambil matakuliah tersebut.

9.2.2 DELETE ANOMALY

Delete Anomaly merupakan error atau kesalahan yang terjadi sebagai akibat operasi penghapusan (delete) terhadap tupe / record dari sebuah relasi.

Contoh: mahasiswa dengan NIM ➔92425 (pada Relasi Kuliah di atas), memutuskan untuk batal ikut kuliah CS-400, karena ia merupakan satu-satunya peserta matakuliah tersebut, maka bila record / tuple tersebut dihapus / delete akan berakibat hilangnya informasi bahwa mata kuliah CS400, biayanya 150.

9.2.3 UPDATE ANOMALY

Update Anomaly merupakan error atau kesalahan yang terjadi sebagai akibat oerasi perubahan (update) tuple / record dari sebuah relasi.

Contoh: biaya kuliah untuk matakuliah CS-200 (pada relasi kuliah di atas) akan dinaikkan dari 75 menjadi 100, maka harus dilakukan beberapa kali modifikasi terhadap record-record atau tuple-tuple mahasiswa yang mengambil matakuliah CS-200 tersebut, agar data teap konsisten.

Berdasarkan teori normalisasi (yang akan dibahas kemudian),maka relasi kuliah di atas harus dipecah menjadi 2 relasi terpisah sebagai berikut.

Tabel 9.2 Relasi Kuliah dipecah menjadi (a) Kuliah_Mahasiswa dan (b) Kuliah_Biaya

NIM	KODE-MTK
92130	CS-200
92200	CS-300
92250	CS-200
92425	CS-400
92500	CS-300
92575	CD-500

KODE-MTK	BIAYA
CS-200	75
CS-300	100
CS-400	150
CD-500	50

9.3 PROBLEM-PROBLEM PADA RELASI YANG SUDAH DINORMALISASI

9.3.1 PERFORMANCE PROBLEM

Contoh: sebelum normalisasi untuk menghasilkan listing data seperti pada relasi Kuliah, dapat digunakan instruksi sintaks SQL sebagai berikut.

SELECTION *
FROM KULIAH

Setelah dinormalisasi untuk menghilangkan listing data-data yang sama, harus terlebih dahulu menggabungkan tuple-tuple dari relasi Kuliah_Mahasiswa dengan tuple-tuple Kuliah_Biaya, dengan menggunakan perintah SQL sebagai berikut.

SELECTION NIM KULIAH_BIAYA.KODE-MTK, BIAYA

FORM KULIAH_MAHASISWA, KULIAH_BIAYA
WHERE KULIAH_MAHASISWA.KODE-MTK =
KULIAH_BIAYA.KODE-MTK

Dimana hasil yang diperoleh sama, tetapi waktu eksekusi akan lebih lama.

9.3.2 REFENTIAL INTEGRITY PROBLEM

Berupa Maintenance Consistency Of Reference antara dua buah elemen yang terkait. Contoh: mengacu kepada relasi Kuliah_Mahasiswa dan Kuliah_Biaya

1. Jangan menambah record baru pada relasi Kuliah_Mahasiswa, kecuali Kode_Mtk untuk record baru tersebut sudah terdapat di relasi Kuliah_Biaya.
2. Jangan menghapus (delete) record pada relasi Kuliah_Biaya bila masih terdapat record-record yang berada pada relasi Kuliah_Mahasiswa, memilikil Kode_Mtk yang value Kode_Mtk yang akan dihapus.

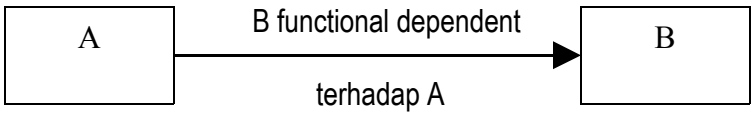
9.4 FUNCTIONAL DEPENDENCIES (KETERGANTUNGAN KETERGANTUNGAN FUNGSIONAL)

Salah satu konsep utama yang berhubungan dengan normalisasi adalah functional dependency (ketergantungan fungsional). Suatu ketergantungan fungsional menggambarkan relationship/hubungan di antara atribut-atribut.

9.4.1 FUNCTIONAL DEPENDENCY (KETERGANTUNGAN FUNGSIONAL)

Functional dependency (ketergantungan fungsional) menggambarkan relationship/hubungan antara atribut-atribut dengan relasi. Sebagai contoh: Jika A dan B adalah atribut-atribut dari relasi R. B dikatakan functionally dependent (bergantungan fungsional) terhadap A (dinotasikan dengan A ➔ B), jika masing-masing nilai dari A dalam relasi R berpasang-an secara tepat dengan satu nilai dari B dalam relasi R.

Ketergantungan antara atribut-atribut A dengan B dapat dilihat pada gambar 9.1 di bawah ini.



Gambar 9.1 *Diagram functional dependency (ketergantungan fungsional)*

9.4.2 DETERMINANT

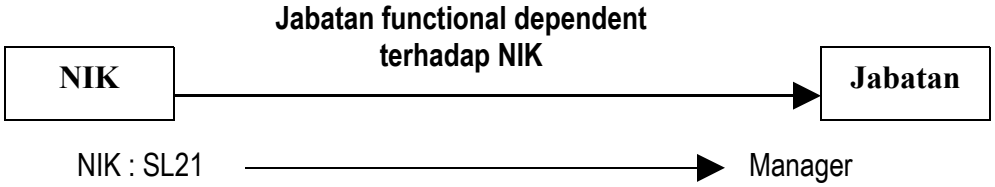
Determinan dari suatu functional dependency (ketergantungan fungsional) menunjuk/ mengarahkan ke atribut atau kelompok atribut-atribut yang berbeda pada sebelah kiri anak panah gambar 9.1 di atas.

Ketika terdapat suatu functional dependency (ketergantungan fungsional) atribut atau group atribut-atribut di sebelah kiri anak panah dinamakan determinan. Pada gambar 9.1 ditunjukkan bahwa A adalah determinan dari B. Sebagai contoh kasus dapat dilihat pada tabel (tabel 9.3 Relasi/Tabel Staff_Cabang) di bawah ini.

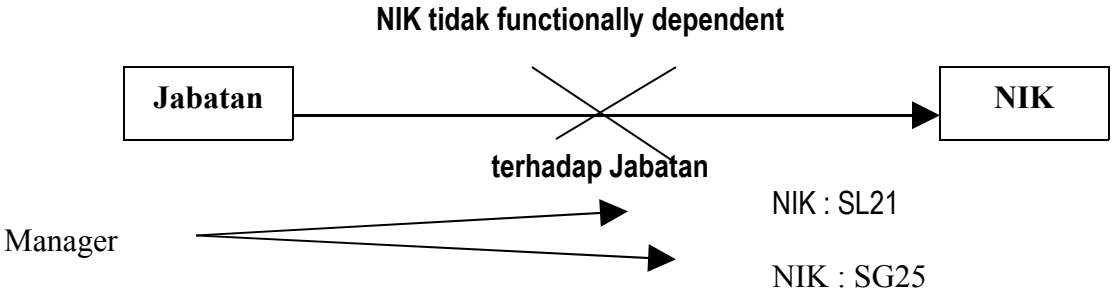
Tabel 9.3 *Relasi/Tabel Staff_Cabang*

NIK	Nama	Alamat	Jabatan	Gaji	Kd_Cabang	Alamat_Cabang	No_Telpon
SL21	John White	Jl.Nanas VIII/85, Jakarta	Manager	30000	B5	Jl.Zeta/34, Tangerang	0171-886-1212
SG37	Ann Beech	Jl.Sawo/185, Jakarta	Analist	12000	B3	Jl.Beta/3, Tangerang	0141-339-2178
SG14	David Ford	Jl.Kecap/10, Jakarta	Deputy	18000	B3	Jl.Beta/3, Tangerang	0141-339-2178
SA9	Mary Howe	Jl.salak/95, Jakarta	Assistant	9000	B7	Jl.Alpha/12, Tangerang	01224-67111
SG5	Susan Brand	Jl.Melon/5, Jakarta	Manager	24000	B3	Jl.Beta/3, Tangerang	0141-339-2178
SL41	Julie Lee	Jl.Jeruk/24, Jakarta	Assistant	9000	B5	Jl.Zeta/34, Tangerang	0171-886-1212

Berdasarkan tabel 9.3 di atas kita dapat membuat beberapa diagram yang menunjukkan functional dependency (gambar 9.2 (a)), dan diagram yang bukan functional dependency (gambar 9.2 (b)).



(a). Jabatan memiliki ketergantungan fungsional terhadap NIK



(b). NIK tidak memiliki ketergantungan fungsional terhadap Jabatan

Gambar 9.2 *Diagram functional dependency (ketergantungan fungsional)*

Kardinalitas relasi antara NIK dan Jabatan adalah 1:1 untuk pasangan masing-masing staff number hanya memiliki satu pasangan. Kardinalitas relasi antara Jabatan dan NIK adalah 1: M, dimana terdapat beberapa staff number yang berasosiasi/berelasi dengan Jabatan. Pada kasus ini NIK adalah determinan dari ketergantungan fungsional dari relasi Staff_Cabang sebagai berikut.

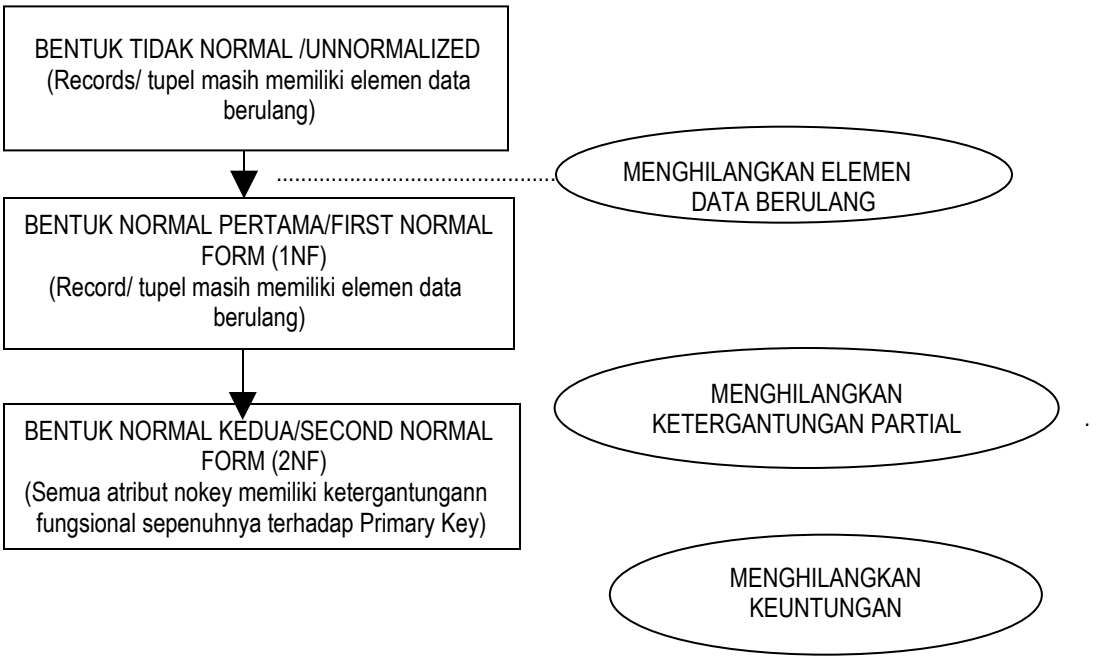
- NIK ➔ Nama
- NIK ➔ Alamat
- NIK ➔ Jabatan
- NIK ➔ Gaji

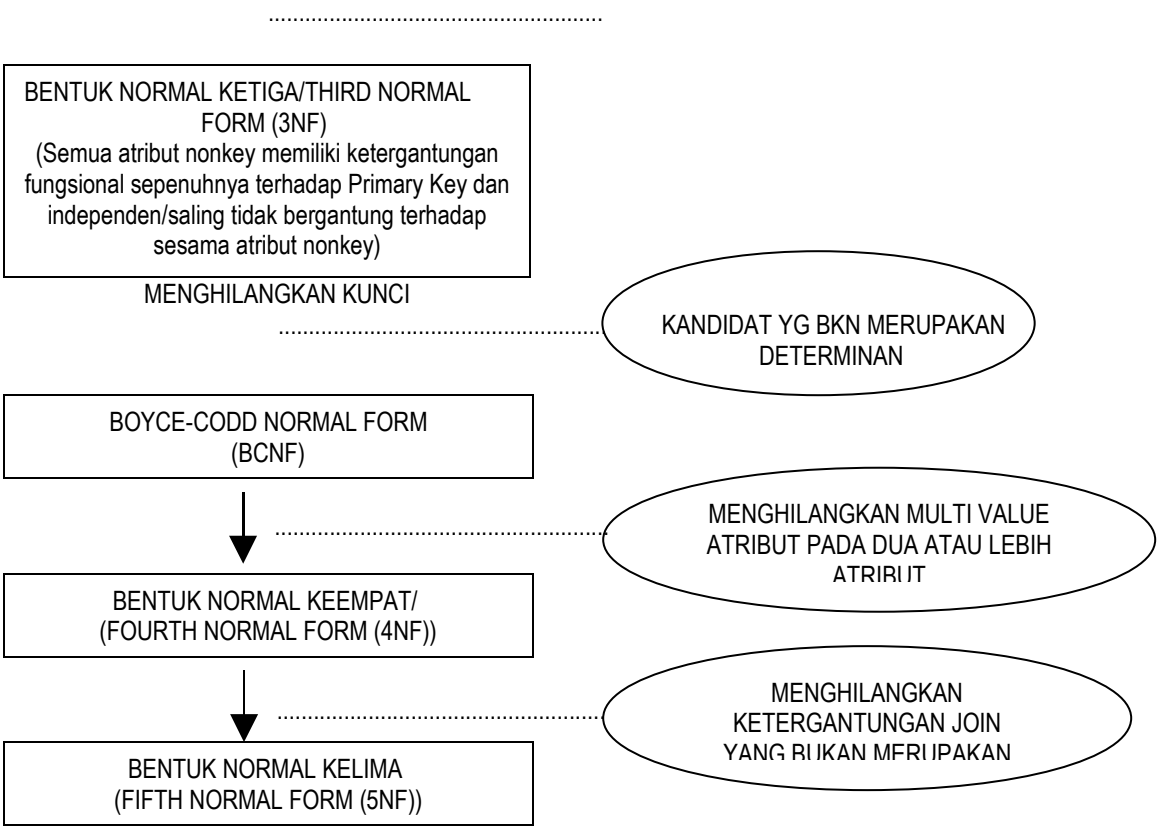
- NIK ➔ Kd_Cabang
- NIK ➔ Alamat_Cabang
- NIK ➔ No_Telpon
- Kd-Cabang ➔ Alamat_Cabang
- Kd_Cabang ➔ No_Telpon
- Alamat_Cabang ➔ Kd_Cabang
- No_Telpon ➔ Kd_Cabang

Terdapat 11 ketergantungan fungsional dalam relasi Staff_Brach dengan NIK, Brach_No, Alamat_Cabang, dan No_Telpon sebagai determinan. Bentuk alternatif/lain yang dapat ditunjukkan/dibuatkan untuk ketergantungan fungsional seperti di atas sebagai berikut.

- NIK ➔ Nama, Alamat, Jabatan, Gaji, Kd_Cabang, Alamat_Cabang, No_Telpon
- Kd_Cabang ➔ Alamat_Cabang, No_Telpon
- Alamat_Cabang ➔ Kd_Cabang
- No_Telpon ➔ Kd_Cabang

Candidate Key untuk relasi Staff_Cabang dapat diidentifikasi dengan mengetahui/mengenal atribut (kelompok atribut-atribut) yang lebih uniq pada masing-masing baris dalam relasi tersebut. Jika relasi tersebut memiliki lebih dari satu candidate key, maka kita lakukan identifikasi candidate key yang dapat ditentukan sebagai primary key dari relasi tersebut yang lebih uniq, lebih sederhana, dan lebih sering digunakan. Seluruh atribut yang bukan primary (tidak terpilih sebagai primary key) harus bergantung fungsional (functionally dependent) pada primary key tersebut. Konsep dari functional dependency (ketergantungan fungsional) merupakan dasar untuk dapat melakukan proses normalisasi yang akan dibahas pada subbab di bawah ini.





9.5 LANGKAH-LANGKAH PEMBENTUKAN NORMALISASI

9.5.1 BENTUK TIDAK NORMAL (UNNORMALIZED FORM)

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikukti format tertentu, dapat saja data tidak lengkap atau terduplikasi. Data dikumpulkan apa adanya sesuai dengan saat menginput.

Hal 1 — ***Rumah_Impian*** — Tanggal : 7-Oct-95

Perincian Pelanggan

Nama Pelanggan : John Key — Nomor Pelanggan CR 76

No_Property	Alamat Property	Tgl_Pinjam	Tgl_Selesai	Biaya	No_Pemilik	Nama_Pemilik
PG4	Jl. Aai / 07, Jakarta	1-Jul-93	31-Aug-95	350	CO40	Ewin
PG16	Jl. Huzai /12, Jakarta	1-Sep-95	1-Sep-96	450	CO93	Durki

Gambar 9.3 Form/Faktur Rumah_Impian Perincian Pelanggan

Tabel 9.4 Tabel Pelanggan_Biaya yang belum normal (unnormal)

No_Pelanggan	Nama	Nomor Property	Alamat Property	Tgl_Pinjam	Tgl_Selesai	Biaya	No_Pemilik	Nama_Pemilik
CR76	Badi	PG4	Jl. Aai / 07, Jakarta	1-Jul-93	31-Aug-95	350	CO40	Ewin
		PG16	Jl. Huzai /12, Jakarta	1-Sep-95	1-Sep-96	450	CO93	Durki

CR56	Sirajuddin	PG4	Jl. Aai / 07, Jakarta	1-Sep-92	10-Jun-93	350	CO40	Ewin
		PG36	Jl. Azhar / 49, Jakarta	10-Oct-93	1-Dec-94	375	CO93	Durki
		PG16	Jl. Huzai /12, Jakarta	1-jan-95	10-Aug-95	450	CO93	Durki

Dari tabel di Pelanggan Biaya (Tabel 9.3) di atas terdapat multiple value pada beberapa atributnya. Misalkan terdapat dua (2) nilai untuk No_Property yaitu PG4 dan PG16 untuk Nama Pelanggan (Nama) Badi.

Untuk mentransformasikan tabel yang belum ternomalisasi di atas menjadi tabel yang memenuhi kriteria 1NF adalah kita harus merubah seluruh atribut yang multivalue menjadi atribut single value, dengan cara menghilangkan repeating group pada tabel di atas.

Repeating Group (elemen data berulang) adalah (No_Property, Alamat_Property, Tgl_Pinjam, Tgl_Selesai, Biaya, No_Pemilik, Nama_Pemilik)

9.5.2 BENTUK NORMAL KE SATU (FIRST NORMAL FORM / 1 NF)

Pada tahap ini dilakukan penghilangan beberapa group elemen yang berulang agar menjadi satu harga tunggal yang berinteraksi di antara setiap baris pada suatu tabel, dan setiap atribut harus mempunyai nilai data yang atomic (bersifat atomic value). Atom adalah zat terkecil yang masih memiliki sifat induknya, bila terpecah lagi maka ia tidak memiliki sifat induknya.

Syarat normal ke satu (1-NF) antara lain:

1. setiap data dibentuk dalam flat file, data dibentuk dalam satu record demi satu record nilai dari field berupa “atomic value”.
2. tidak ada set atribute yang berulang atau bernilai ganda.
3. telah ditentukannya primary key untuk tabel / relasi tersebut.
4. tiapatribut hanya memiliki satu pengertian.

Langkah pertama yang dilakukan pada Tabel Pelanggan Biaya (pada Tabel 9.3) tersebut adalah menghilangkan elemen data yang berulang dengan data-data Pelanggan yang sesuai pada setiap baris. Hasil dari tabel yang telah memenuhi bentuk normal pertama dapat dilihat pada Tabel 9.4. kita dapat mengidentifikasi primary key untuk relasi Pelanggan_Biaya yang masih memiliki composite key (No_Pelanggan, No_Property). Pada kasus ini kita akan memperoleh primary key yang bersifat composite key.

Relasi Pelanggan_Biaya dapat didefinisikan sebagai berikut. Pelanggan_Biaya = (<u>No_Pelanggan</u>, <u>No_Property</u>, Nama, Alamat_Property, Tgl_Pinjam, Tgl_Selesai, Biaya, No_Pemilik, Nama_Pemilik)

Tabel 9.5 *Tabel Pelanggan Biaya dalam bentuk normal kesatu (1-NF)*

No_Pelanggan	Nama	Nomor Property	Alamat Property	Tgl_Pinjam	Tgl_Selesai	Biaya	No_Pemilik	Nama_Pemilik
CR76	Badi	PG4	Jl. Aai / 07, Jakarta	1-Jul-93	31-Aug-95	350	CO40	Ewin
CR76	Badi	PG16	Jl. Huzai /12,	1-Sep-95	1-Sep-96	450	CO93	Durki

			Jakarta					
CR56	Sirajuddin	PG416	Jl. Aai / 07, Jakarta	1-jan-95	10-Aug-95	450	CO93	Durki
CR56	Sirajuddin	PG36	Jl. Azhar / 49, Jakarta	10-Oct-93	1-Dec-94	375	CO93	Durki
CR56	Sirajuddin	PG4	Jl. Huzai /12, Jakarta	1-Sep-95	10-Jun-93	350	CO40	Ewin

9.5.3 BENTUK NORMAL KE DUA (SECOND NORMAL FORM / 2 NF)

Bentuk normal kedua didasari atas konsep full functional dependency (ketergantungan fungsional sepenuhnya) yang dapat didefinisikan sebagai berikut.

Jika A adalah atribut-atribut dari suatu relasi, B dikatakan full functional dependency (memiliki ketergantungan fungsional terhadap A, tetapi tidak secara tepat memiliki ketergantungan fungsional dari subset (himpunan bagian) dari A.

FULL FUNCTIONAL DEPENDENCY (Ketergantungan Fungsional Sepenuhnya)

Suatu ketergantungan fungsional A → B adalah ketergantungan fungsional sepenuhnya, jika perpindahan beberapa atribut dari A menghasilkan tepat satu pasangan pada atribut B. Suatu ketergantungan fungsional A → B adalah ketergantungan fungsional sebagian, jika ada beberapa atribut yang dapat dihilangkan dari A sementara ketergantungan tersebut tetap berlaku /berfungsi.

Sebagai contoh dapat dilihat pada suatu ketergantungan fungsional di bawah ini.

NIK, Nama → Kd_Cabang.

Dari kasus di atas dapat dikatakan bahwa masing-masing nilai dari NK, Nama berasosiasi/berelasi dengan nilai tunggal dari Kd_Cabang. Dengan demikian, relasi di atas tidak memiliki ketergantungan fungsional sepenuhnya (full funcional dependency), karena Kd_Cabang juga masih memiliki ketergantungan fungsional pada himpunan bagian dari (NIK, Nama). Dengan kata lain, Kd_Cabang adalah Full functional dependecy hanya pada NIK.

Bentuk normal kedua memungkinkan suatu relasi memiliki composite key, yaitu relasi dengan primary key yang terdiri dari dua atau lebih atribyut. Suatu relasi yang memiliki sigle atribut untuk primary keynya secara otomatis pada akhirnya menjadi 2-NF.

Syarat normal kedua (2-NF) sebagai berikut.

1. Bentuk data telah memenuhi kriteria bentuk normal kesatu.
2. Atribute bukan kunci (non-key) haruslah memiliki ketergantungan fungsionla sepenuhnya (fully functional dependency) pada kunci utama / primary key.

Dengan demikian untuk membentuk normal kedua haruslah sudah ditentukan primary keynya. Primary key tersebut haruslah lebih sederhana, lebih unik, dapat

mewakili atribute lain yang menjadi anggotanya, dan lebih sering digunakan pada tabel / relasi tersebut.

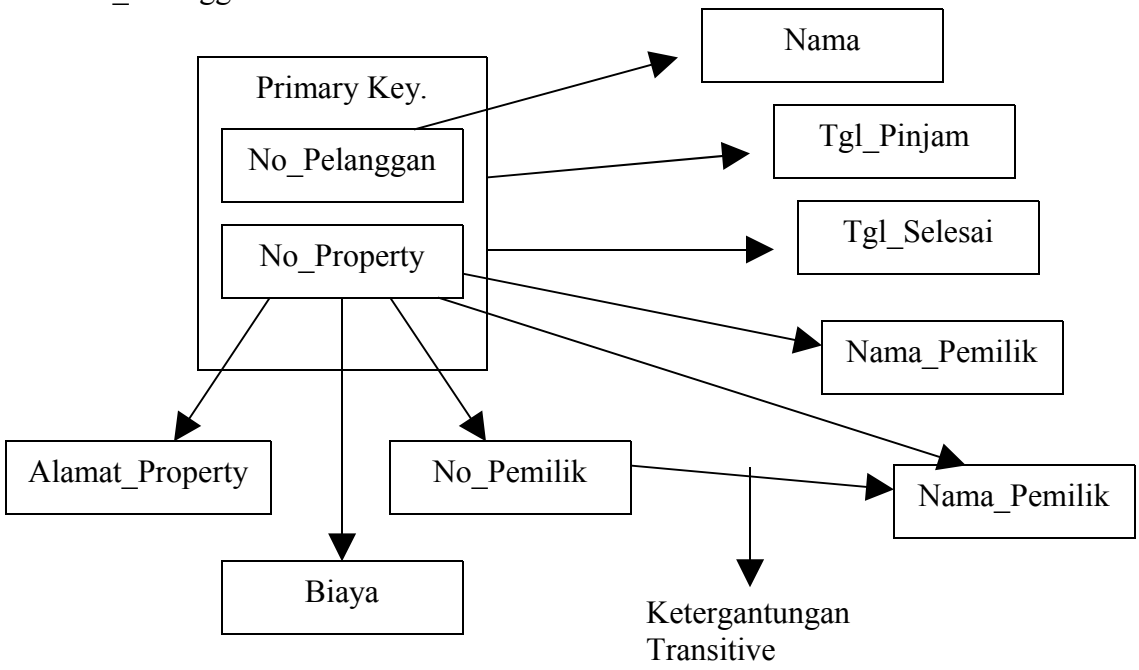
Langkah pertama kita harus mengidentifikasi ketergantungan fungsional dalam relasi Pelanggan_Biaya, sebagai berikut.

No_Pelanggan, No_Property ➔Tgl_Pinjam, Tgl_Selesai
No_Pelanggan ➔Nama
No_Property ➔Alamat_Property, Biaya, No_Pemilik, Nama_Pemilik
No_Pemilik ➔ Nama_Pemilik

Gambar 9.2 di bawah ini akan mengilustrasikan ketergantungan fungsional yang berada dalam relasi Pelanggan_Biaya.

Berdasarkan Functional Dependency (Ketergantungan Fungsional) relasi Pelanggan_Biaya pada gambar 9.2 di atas, maka dapat dilihat beberapa kondisi sebagai berikut.

1. Primary key pada Pelanggan_Biaya di atas adalah No_Pelanggan, dan No_Property.
2. Atribut Pelanggan (Nama) hanya memiliki (partially dependency) ketergantungan pada sebagian / pada salah satu primary key di relasi Pelanggan_Biaya yaitu hanya atribut No_Pelanggan.



Gambar 9.4 *Functional Dependency (Ketergantungan Fungsional) pada relasi Pelanggan_Biaya*

3. Sementara atribut-atribut property (Alamat_property, Biaya, No_Pemilik, Nama_Pemilik), juga hanya memiliki (partially dependency) ketergantungan pada sebagian / pada salah satu primary key di relasi Pelanggan_Biaya yaitu hanya atribut No_Property.
4. Atribut-atribut Biaya Property (Tgl_Pinjam, Tgl_Selesai), memiliki (fully dependent) tergantung sepenuhnya pada semua primary key di relasi Pelanggan_Biaya yaitu atribut No_Pelanggan, dan No_Property.

5. Pada gambar 9.2 di atas juga menunjukkan sebuah (transitive dependency) ketergantungan transitif terhadap primary key, namun hal itu tidak akan melanggar ketentuan pada normalisasi ke dua (2-NF). Ketergantungan transitive akan dihilangkan pada normalisasi ketiga (3-NF).

Seluruh identifikasi yang dilakukan pada point 1 sampai 4 di atas menunjukkan bahwa relasi Pelanggan_Biaya di atas belum memenuhi kriteria bentuk normal ke dua (2-NF). Kita perlu membuat beberapa relasi Pelanggan_Biaya di atas ke dalam bentuk normal ke dua (2-NF), agar seluruh atribut non-key (yang bukan primary key) dapat memiliki ketergantungan sepenuhnya (full functionally dependent) terhadap primary key. Ketiga buah relasi tersebut dapat ditulis dalam bentuk ini.

1. Relasi / Tabel Pelanggan dengan atribut-atribut:
   No_Pelanggan, Nama. {No_Pelanggan berfungsi sebagai primary key pada tabel / relasi tersebut}.
2. Relasi / Tabel Biaya dengan atribut-atribut: No_Pelanggan, No_Property, Tgl_Pinjam, Tgl_Selesai. {No_Pelanggan dan No_Property berfungsi sebagai primary key pada tabel / relasi tersebut}.
3. Relasi / Tabel Property_Pemilik dengan atribut-atribut: No_Property, Alamat_Property, Biaya, No_Pemilik, Nama_Pemilik. { No_Property berfungsi sebagai priamry key pada tabel / relasi tersebut}.

Tabel / Relasi yang telah memenuhi kriteria normal ke dua (2-NF) adalah sebagai berikut.

Tabel 9.6 *Tabel Pelanggan Biaya dalam bentuk normal kedua (2-NF)*

No_Pelanggan	Nama
CR76	Badi
CR56	Sirajuddin

(a) Relasi Pelanggan

No_Pelanggan	No_Property	Tgl_Pinjam	Tgl_Selesai
CR76	PG4	1-Jul-93	31-Aug-95
CR76	PG16	1-Sep-95	1-Sep-96
CR56	PG4	1-Sep-95	10-Jun-93
CR56	PG36	10-Oct-93	1-Dec-94
CR56	PG16	1-Jan-95	10-Aug-95

(b) Relasi Biaya

No_Property	Alamat_Property	Biaya	No_Pemilik	Nama_Pemilik
PG4	Jl. Aai / 07, Jakarta	3530	CO40	Ewin
PG16	Jl.Huzai /12,Jakarta	450	CO93	Durki
PG36	Jl.Suciana / 68, Bogor	375	CO93	Durki

(c) Relasi Property_Pemilik

9.5.4 BENTUK NORMAL KE TIGA (THIRD NORMAL FORM / 3 NF)

Walaupun relasi 2-NF memiliki redudansi yang lebih sedikit dari pada relasi 1-NF, namun relasi tersebut masih mungkin mengalami kendala bila terjadi anomaly peremajaan (update) terhadap relasi tersebut.

Misalkan kita akan melakukan update terhadap nama dari seorang Pemilik (pemilik), seperti Durki (No_Pemilik: CO93), kita harus melakukan update terhadap dua baris dalam relasi Property_Pemilik (lihat Tabel 9.5, (c) relasi Property_Pemilik). Jika kita hanya mengupdate satu baris saja, sementara baris yang lainnya tidak, maka data di dalam database tersebut akan inkonsisten / tidak teratur. Anomaly update ini disebabkan oleh suatu ketergantungan transitif (transitive dependency). Kita harus menghilangkan ketergantungan tersebut dengan melakukan normalisasi ketiga (3-NF).

Transitive Dependency (ketergantungan transitif)

Suatu kondisi dimana A, B, dan C adalah atribut-atribut dari suatu relasi sedemikian sehingga A ➔ B dan B ➔ C, maka A ➔ C (C memiliki ketergantungan transitif terhadap A melalui B), dan harus dipastikan bahwa A tidak memiliki ketergantungan fungsional (functional dependent) terhadap B atau C).
Sebagai contoh dapat dilihat ketergantungan fungsional yang ditunjukkan pada relasi Staff_Brach di Tabel 9.3 sebagai berikut.

NIK ➔ Kd_Cabang dan Kd_ Cabang ➔ Alamat_Cabang

Dapat dilihat ketergantungan transitif NIK ➔ Alamat_Cabang dapat terjadi melalui atribut Kd_Cabang.

Syarat normal ketiga (Third Normal Form / 3 NF) sebagai berikut.

1. Bentuk data telah memenuhi kriteria bentuk normal kedua.
2. Atribute bukan kunci (non-key) harus tidak memiliki ketergantungan transitif, dengan kata lain suatu atribut bukan kunci (non_key) tidak boleh memiliki ketergantungan fungsional (functional dependency) terhadap atribut bukan kunci lainnya, seluruh atribut bukan kunci pada suatu relasi hanya memiliki ketergantungan fungsional terhadap priamry key di relasi itu saja.

Sebagai contoh kita dapat melihat beberapa ketergantungan fungsional (functinal dependencies) pada relasi Pelanggan, Biaya, dan Property_Pemilik berikut ini.

- Relasi / Tabel Pelanggan terdiri dari atribut-atribut:
  No_Pelanggan ➔ Nama
  {No_Pelanggan sebagai primary key}

- Relasi / Tabel Biaya terdiri dari atribut-atribut:
  No_Pelanggan, No_property ➔ Tgl_Pinjam, Tgl_Selesai
  {No_Pelanggan, dan No_property sebagai primary key}
- Relasi / Tabel Property_Pemilik terdiri dari atribut-atribut
  No_property ➔ Alamat_Property, Biaya, No_Pemilik, Nama_Pemilik
  No_Pemilik ➔ Nama_Pemilik
  {No_property sebagai primary key}

Seluruh atribut non-primary key pada relasi Pelanggan dan Biaya di atas terlihat memiliki ketergantungan fungsional (functional dependency) terhadap primary key dari masing-masing tabel / relasi. Relasi / tabel Pelanggan dan Biaya di atas tidak memiliki ketergantungan transitif (transitive dependency), sehingga tabel tersebut telah memenuhi kriteria normal ketiga (3-NF).

Seluruh atribut non-primary key pada relasi Property_Pemilik di atas terlihat memiliki ketergantungan fungsional (functional dependency) terhadap primary key, kecuali Nama_Pemilik yang masih memiliki ketergantungan fungsional (functional dependency) terhadap No_Pemilik. Inilah contoh ketergantungan dari transitif (transitive dependency), yang terjadi ketika atribut non-primary key (Nama_Pemilik) bergantung secara fungsi terhadap satu atau lebih atribut non-primary key lainnya (No_Pemilik). Kita harus menghilangkan ketergantungan transitif (transitive dependency) tersebut dengan menjadikan relasi Property_Pemilik menjadi 2 relasi / tabel dengan format / bentuk sebagai berikut.

- Relasi / Tabel Property_Untuk_Pemilik yang terdiri dari atribut-atribut:
  No_property ➔ Alamat_Property, Biaya, No_Pemilik
  {No_property sebagai primary key}
- Dan relasi Pemilik yang terdiri dari atribut-atribut:
  No_Pemilik ➔ Nama_Pemilik
  {No_Pemilik sebagai primary key}

Bila digambarkan ke dalam tabel menjadi seperti berikut ini.

Tabel 9.7 *Tabel / Relasi yang memenuhi kriteria normal ketiga (3-NF) yang berasal dari relasi Property_Pemilik*

No_Property	Alamat_Property	Biaya	No_Pemilik
PG4	Jl. Aai / 07, Jakarta	3530	CO40
PG16	Jl.Huzai /12,Jakarta	450	CO93
PG36	Jl.Suciana / 68, Bogor	375	CO93

(c 1) Relasi Property_Untuk _Biaya

No_Pemilik	Nama_Pemilik
CO40	Ewin
CO93	Durki
CO93	Durki

(c 2) Relasi Pemilik

Kita dapat membuat suatu diagram dekomposisi yang akan menjelaskan proses / tahapan uji normalisasi dari bentuk normal kesatu sampai dengan normal ketiga pada relasi Pelanggan_Biaya seperti pada gambar 9.5.

Gambar 9.5 *Diagram dekomposisi bentuk normal kesatu sampai dengan normal ketiga dari tabel/relasi Pelanggan_Biaya*

Hasil akhir normalisasi tabel Pelanggan_Biaya sampai ke bentuk normal ketiga adalah sebagai berikut.

Tabel 9.8 Hasil akhir uji normalisasi sampai ke bentuk (3-NF) yang berasal dari relasi Pelanggan_Biaya

No_Pelanggan	Nama
CR76	Badi
CR56	Sirajuddin

(a) Relasi Customer

No_Pelanggan	No_Property	Tgl_Pinjam	Tgl_Selesai
CR76	PG4	1-Jul-93	31-Aug-95
CR76	PG16	1-Sep-95	1-Sep-96
CR56	PG4	1-Sep-95	10-Jun-93
CR56	PG36	10-Oct-93	1-Dec-94
CR56	PG16	1-Jan-95	10-Aug-95

(b) Relasi Rental

No_Property	Alamat_Property	Biaya	No_Pemilik
PG4	Jl. Aai / 07, Jakarta	3530	CO40
PG16	Jl.Huzai /12,Jakarta	450	CO93
PG36	Jl.Suciana / 68, Bogor	375	CO93

(c) Relasi Property_for _Rent

No_Pemilik	Nama_Pemilik
CO40	Ewin
CO93	Durki
CO93	Durki

(d) Relasi Owner

9.5.5 Boyce-Codd Normal Form (BCNF)

Suatu relasi dalam basis data harus dirancang sedemikian rupa sehingga mereka tidak memiliki ketergantungan sebagian (partial dependecy), maupun ketergantungan transitif (transitive dependecy), seperti telah dibahas pada subbab sebelumnya.

Boyce-Codd Normal Form (BCNF) didasari pada beberapa ketergantungan fungsional (functional dependencies) dalam suatu relasi yang melibatkan seluruh candidate key di dalam relasi tersebut. Jika suatu relasi hanya memiliki satu candidate key, maka hasil uji normalisasi sampai ke bentuk normal ketiga sudah identik dengan Boyce-Codd Noormal Form (BCNF).

Boyce-Codd Normal Form (BCNF)

Suatu relasi dikatakan telah memenuhi kriteria Boyce-Codd Normal Form (BCNF), jika dan hanya jika setiap determinan adalah suatu candidate key.

Boyce-Codd Normal Form (BCNF) tidak mengharuskan suatu relasi harus sudah dalam bentuk normal ketiga (3-NF), baru bisa di buatkan ke dalam BCNF. Oleh karena itu untuk melakukan uji BCNF kita hanya mengidentifaksi seluruh determinan yang ada pada suatu relasi, lalu pastikan determinan-determinan tersebut adalah candidate key. Dengan demikian, bisa dikatakan bahwa BCNF lebih baik dari bentuk normal ketiga (3-NF), sehingga setiap relasi di dalam BCNF juga merupakan relasi dalam 3-NF, tetapi tidak sebaliknya, suatu relasi di dalam 3-NF belum tentu merupakan relasi di dalam BCNF.

Sebelum kita berlanjut pada contoh kasus berikutnya, mari kita lihat kembali relasi Pelanggan, Biaya, Property_Untuk_Pemilik, dan Pemilik (pada tabel 9.8 di atas). Keempat relasi di atas sudah dalam bentuk BCNF, di mana masing-masing relasi memiliki hanya memiliki satu determinan yaitu candidate keynya, walaupun relasi Biaya kenyataannya memiliki3 determinan yaitu (No_Pelanggan, No_Property), (No_Pelanggan, Biaya), dan 9(No_Property, Tgl_Pinjam) seperti terlihat di bawah ini.

No_Pelanggan, No_Property → Tgl_Pinjam, Tgl_Selesai
No_Pelanggan, Tgl_Pinjam → No_Property, Tgl_Selesai
No_Property, Tgl_ Pinjam → No_Pelanggan, Tgl_Selesai

Ketiga determinan dari relasi Biaya tersebut juga sebagai candidate key sehingga relasi Biaya tersebut juga telah memenuhi kriteria BCNF.

Kita dapat saja melakuakan dekomposisi terhadap beberapa relasi untuk manjadi BCNF. Walaupun demikian tidak semua relasi perlu ditransformasi sampai ke BCNF. Adakalanya suatu relasi sudah normal sampai uji normal ketiga (3-NF), dan tidak perlu lagi dilanjutkan untuk ke BCNF.

9.6 CONTOH KASUS NORMALISASI

9.6.1 Tabel Nama_Mahasiswa di bawah ini akan dilakukan uji normalisasi dari bentuk tidak normal (unnormalized) ke dalam bentuk normal kedua (2-NF)

Terdapat sebuah tabel dengan komposisi sebagai berikut.

Tabel 9.9 *Tabel/relasi Mahasiswa yang belum memenuhi kriteria 1-NF*

Nama_mahasiswa	NIM	Kd_MKul
Jones	61521	MAT231, ECO220,HST211
Diana	61300	HST211
Tony	61425	ENG202, MAT231
Paula	61230	MAT231, ENG202

Tabel Mahsiswa di atas belum memenuhi kriteria 1-NF sebab atribut Kd_MKul masih memiliki nilai ganda dalam satu baris. Untuk mengkonversikan tabel Mahasiswa tersebut ke dalam bentuk 1-NF, maka kita harus menyusun kembali baris-baris pada Kd_MKul, sehingga setiap baris memiliki nilai tunggal, seperti tabel di bawah ini.

Tabel 9.10 *Tabel/relasi Mahasiswa yang sudah dalam bentuk 1-NF*

Nama_mahasiswa	NIM	Kd_MKul
Jones	61521	MAT231
Jones	61521	ECO220
Jones	61521	HST211
Diana	61300	HST211
Tony	61425	ENG202
Tony	61425	MAT231
Paula	61230	MAT231
Paula	61230	ENG202

Tabel Mahsiswa di atas sudah memenuhi kriteria 1-NF, tetapi belum memenuhi kriteria 2-NF sebab atribut Mahasiswa bergantung fungsional pada NIM, dan atribut Kd_MKul juga bergantung fungsional pada NIM, sehingga tabel Mahaaiawa di atas perlu dipecah menjadi dua tabel agar setiap atribut bukan primary key hanya bergantung sepenuhnya terhadap atribut primary key saja, seperti tabel di bawah ini.

Tabel 9.11 Tabel/relasi Mahasiswa yang sudah dalam bentuk 2-NF

Tabel Mahasiswa-1

Nama_mahasiswa	NIM
Jones	61521
Jones	61521
Jones	61521
Diana	61300
Tony	61425
Tony	61425
Paula	61230
Paula	61230

dan

Tabel Mahasiswa-2

NIM	Kd_MKul
61521	MAT231
61521	ECO220
61521	HST211
61300	HST211
61425	ENG202
61425	MAT231
61230	MAT231
61230	ENG202

9.6.2 TABEL NAMA_MAHASISWAC DI BAWAH INI AKAN DILAKUKAN UJI NORMALISASI DARI BENTUK BENTUK NORMAL KE SATU (1-NF) SAMPAI KE BENTUK KETIGA (3- NF)

Terdapat sebuah tabel yang sudah memenuhi kriteria 1-NF dengan komposisi sebagai berikut.

Tabel 9.12 Tabel/relasi Mahasiswa-C yang sudah dalam bentuk 1-NF

Tabel Mahasiswa-C

Nama_mahasiswa	NIM	Tgl_Lahir	Kuliah	Kd_MKul	SKS	Nilai	Bobot
Jones	61521	12/05/77	Kalkulus	MAT231	3	B	3
Jones	61521	12/05/77	Ekonomi-1	ECO220	3	A	4
Jones	61521	12/05/77	History	HST211	2	B	3
Diana	61300	14/28/78	History	HST211	2	A	4
Tony	61425	11/01/76	Bhs.Ingrris	ENG202	2	C	2
Tony	61425	11/01/76	Kalkulus	MAT231	3	B	3
Paula	61230	06/14/77	Kalkulus	MAT231	3	B	3
Paula	61230	06/14/77	Bhs.Ingrris	ENG202	2	C	2

Dari tabel Mahasiswa-C di atas terdapat beberapa ketergantungan fungsional di anatara atribut-atribut sebagai berikut.

NIM ➔ Nama_Mahasiswa, Tgl_Lahir
Kd_MKul ➔ Kuliah, SKS
NIM, Kd_Mkul ➔ Nilai
Nilai ➔ Bobot

Tabel Mahasiswa-C di atas belum memenuhi kriteria 2-NF, selama terdapat beberapa atribut seperti Tgl_Lahir, Kuliah yang tidak memiliki ketergantungan fungsional terhadap primary key (NIM, Kd_Mkul). Untuk mengkonversi tabel tersebut menjadi 2-NF, maka Tabel Mahasiswa-C perlu dipecah menjadi 3 tabel yaitu: Tabel Mahasiswa-C1 = (NIM, Nama_Mahasiswa, Tgl_Lahir), Mahasiswa-C2 = (Kd_MKul, Kuliah, SKS), dan Mahasiswa-C3 = (NIM, Kd_MKul, Nilaid, Bobot) dengan komposisi tabel sebagai berikut.

Tabel 9.13 (a, b, c) Tabel/relasi Mahasiswa-C yang sudah dalam bentuk 2-NF

Nama_mahasiswa	NIM	Tgl_Lahir
Jones	61521	12/05/77
Diana	61300	14/28/78
Tony	61425	11/01/76
Paula	61230	06/14/77

(a) Tabel StudentC1

NIM	Kd_MKul	Nilai	Bobot
61521	MAT231	B	3
61521	ECO220	A	4
61521	HST211	B	3
61300	HST211	A	4
61425	ENG202	C	2
61425	MAT231	B	3
61230	MAT231	B	3
61230	ENG202	C	2

(b) Tabel StudentC2

Kuliah	Kd_MKul	SKS
Kalkulus	MAT231	3
Ekonomi-1	ECO220	3
History	HST211	2
Bhs.Ingrris	ENG202	2

(c) Tabel StudentC3

Tabel Mahasiswa-C1 dan Mahasiswa-C2 telah memenuhi Kriteria 3-NF, namun tabel Mahasiswa-C3 belum memenuhi kriteria 3-NF, selama atribut non-key /bukan kunci Nilai dan Bobot masih saling memiliki ketergantungan fungsional. Untuk mengkonversinya menjadi bentuk 3-N, maka Tabel Mahasiswa-C3 tersebut perlu dipecah menjadi 2 tabel yaitu: Tabel Mahasiswa-C3 = (NIM, Kd MKul, Nilai) dan Mahasiswa-C3B = (Nilai, Bobot) dengan komposisi tabel sebagai berikut.

Tabel 9.14 (a, dan b) Tabel/relasi Mahasiswa-C3 yang sudah dalam bentuk 3-NF

Nilai	Bobot
A	4
B	3
C	2
D	1
E	0

(a) Tabel StudentC3A

NIM	Kd_MKul	Nilai	Bobot
61521	MAT231	B	3
61521	ECO220	A	4
61521	HST211	B	3
61300	HST211	A	4
61425	ENG202	C	2
61425	MAT231	B	3
61230	MAT231	B	3
61230	ENG202	C	2

(b) Tabel StudentC3B

9.36 TABEL WAWANCARA_PELANGGAN DI BAWAH INI AKAN DILAKUKAN UJI NORMALISASI SAMPAI BOYCE-CODD NORMAL FORM (BCNF)

Terdapat relasi Wawancara_Pelanggan yang berisi aturan-aturan rinci untuk mewawancarai client-client yang akan dilakukan oleh beberapa orang staff dari perusahaan Rumah_Impian. Staff-staff tersebut akan melakukan wawancara dengan para client di suatu ruang khusus untuk wawancara. Seorang client hanya diwawancarai satu kali pada tanggal yang telah ditetapkan, namun mungkin saja dilakukan wawancara berikutnya pada tanggal yang akan ditentukan. Relasi ini mempunyai dua candidate key, yang bernama (No_Pelanggan, Tgl_Wawancara) dan (NIK, Tgl_Wawancara, Waktu_Wawancara). Kita tentukan (No_Pelanggan, Tgl_Wawancara) sebagai primary key pada relasi ini. Format/ bentuk relasi Wawancara_Pelanggan dapat dilihat sebagai berikut.

Relasi Wawancara_Pelanggan dengan atribut-atribut.
(No_pelanggan, Tgl_Wawancara, Waktu_Wawancara, NIK, No_Ruang)
dalam bentuk tabel sebagai berikut.

Tabel 9.15 Tabel/relasi Wawancara_Pelanggan

No_Pelanggan	Tgl_Wawancara	Waktu_Wawancara	NIK	No_Ruang
CR76	13-May-95	10.30	SG5	G101
CR56	13-May-95	12.00	SG5	G101G
CR74	13-May-95	12.00	SG37	G102
CR56	1-Jul-95	10.30	SG5	G102

Relasi Wawancara_Pelanggan di atas memiliki ketergantungan fungsional (fungctional dependency) sebagai berikut.

- No_Pelanggan, Tgl_Wawancara → Waktu_Wawancar, NIK, No_Ruang
- NIK, Tgl_Wawancara, Waktu_Wawancara → No_Pelanggan
- NIK, Tgl_Wawancara → No_Ruang

- Oleh karena itu (No_Pelanggan, Tgl_Wawancara) dan (NIK, Tgl_Wawancara, Waktu_Wawancara) adalah composite candidate key dari relasi Wawancara_Pelanggan yang meliputi atribut umum Tgl_Wawancara. Relasi Wawancara_Pelanggan tersebut tidak memiliki ketergantungan sebagian (partial dependency) dan ketergantungan transitif (transitive dependency) terhadap primary key (No_Pelanggan, Tgl_Wawancara). Dengan demikian, relasi tersebut telah berada dalam bentuk 3-NF, namun relasi tersebut belum memenuhi kriteria BCNF, karena keberadaan determinan (NIK, Tgl_Wawancara) yang tidak menjadi candidate key untuk relasi Wawancara_Pelanggan. Untuk mentransformasikan relasi Wawancara_Pelanggan ke BCNF, kita harus membuatnya menjadi dua relasi/tabel

yang diberinama relasi Wawancara dan relasi Ruang_Staff dengan bentuk sebagai berikut.

- Relasi Wawancara memiliki atribut-atribut sebagai berikut.
  (No_pelanggan, Tgl_Wawancara, Waktu_Wawancara, NIK)
  dengan primary keynya adalah No_pelanggan dan Tgl_Wawancara
- Relasi Ruang_Staff memiliki atribut-atribut sebagai berikut.
  (NIK, Tgl_Wawancara, No_Ruang)
  dengan primary keynya adalah NIK, dan Tgl_Wawancara
  Dalam bentuk tabel dapat dilihat sebagai berikut.

Tabel 9.16 (a, dan b) Tabel/relasi Ruang_Staff dan Wawancara yang telah memenuhi kriteria BCNF

No_Pelanggan	Tgl_Wawancara	Waktu_Wawancara	NIK
CR76	13-May-95	10.30	SG5
CR56	13-May-95	12.00	SG5
CR74	13-May-95	12.00	SG37
CR56	1-Jul-95	10.30	SG5

(a) Tabel/relasi Interview

NIK	Tgl_Wawancara	No_Ruang
SG5	13-May-95	G101
SG5	13-May-95	G101G
SG37	13-May-95	G102
SG5	1-Jul-95	G102

(b) Tabel/relasi Staff_Room

-oo0oo-

Bab 10

Sistem Basis Data Terdistribusi

10.1 PENDAHULUAN

Tujuan utama di balik perkembangan sistem basis data adalah suatu keinginan untuk mengintegrasikan berbagai data-data operasional dari suatu orfanisasi dan menyediakan pengaksesan data terkontrol. Walaupun integrasi dan pengontrolan pengaksesan data akan mengimplikasikan suatu sentralisasi data dan sistem. Pada kenyataanya perkembangan jaringan komputer mengarah kepada model kerja yang desentralisasi.

Desentralisasi tersebut akan memperkenalkan struktur organisasi banyak perusahaan, yang secara logis terdistribusi ke dalam divisi-divisi, departemen-departemen, proyek-proyek, dll, dan secara fisik terdistribusi ke dalam offices (kantor-kantor), plants (bangunan-bangunan untuk pekerjaan produksi suatu perusahaan), factories (pabrikpabrik), dll., di mana masing-masing unit akan merancang dan membiayai pengolahan data mereka masing-masing.

Perkembangan sistem basis data terdistribusi akan merefleksikan dan mencerminkan struktur organisasional tersebut, membuat data diseluruh unit dapat diakses dengan baik, dan menyimpan data-data penting dan sering digunakan ke tempat-tempat yang paling sring digunakan dan mudah ditemukan, serta meningkatkan kemampuan data untuk digunakan secara bersama berikut efisiensi pengaksesannya.

10.2 DEFINISI

Sebelum membahas jauh tentang sistem terdistribusi, kita akan mendefinisikan basis data terdistribusi terlebih dahulu.

10.2.1 BASIS DATA TERDISTRIBUSI

Basis data terdistribusi adalah kumpulan data logic yang saling berhubungan, secara fisik terdistribusi dalam jaringan komputer, yang tidak tergantung dari program aplikasi sekarang maupun pada masa yang akan datang.

Sedangkan File merupakan kumpulan data yang dirancang untuk suatu aplikasi atau sekumpulan aplikasi yang dekat hubungannya.

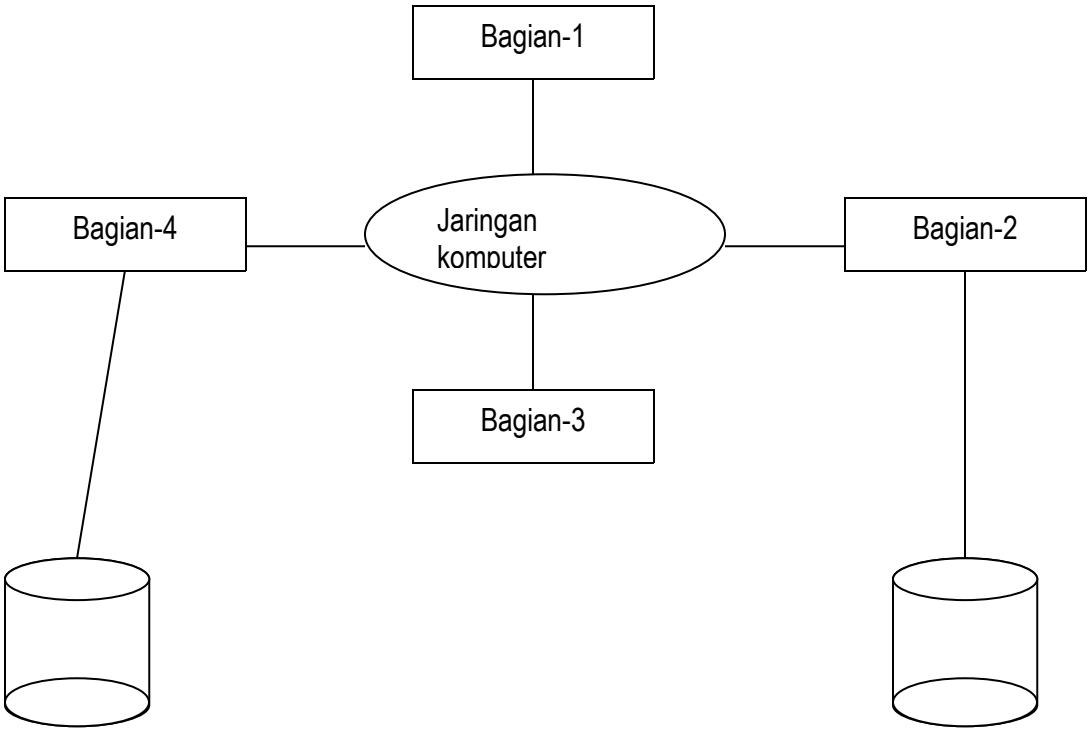
10.2.2 DISTRIBUTED DATABASE MANAGEMENT SISTEM (DBSM TERDISTRIBUSI)

DBSM terdistribusi adalah sistem software yang memungkinkan ditatanya suatu basis data terdistribusi dan membuat distribusi tersebut transparan bagi setiap pemakai (user).

Suatu DBMS terdiri dari sistem basis data tunggal yang terbagi ke dalam beberapa penggalan, masing-masing pengalan tersebut akan diletakkan pada suatu atau beberapa komputer di bawah pengontrolan terpisah oleh DBMS. Komputer-komputer tersebut terkoneksi dalam suatu jaringan komunikasi.

User mengakses basis data terdistribusi tersebut melalui sebuah aplikasi yang berbasis database. Aplikasi-aplikasi tersebut tidak diperoleh atau diakses oleh user secara lokal (dari work-station tempat mereka bekerja), namun diakses secara global dari suatu server yang terpusat. Kita mengasumsikan bahwa DBMS terdistribisi memiliki paling tidak satu aplikasi yang global. Oleh karena itu, DBMS terdistribusi akan memiliki beberapa karakteristik berupa antara lain.

a. Kumpulan data-data logic (yang dapat digunakan secara bersama) terdistribusi pada beberapa unit komputer yang berbeda.
b. Komputer tersebut terkoneksi ke dalam suatu jaringan komunikasi.
c. Data pada masing-masing unit komputer (work-station) terkontrol oleh suatu DBMS.
d. DBMS pada masing-masing bagian dapat menangani aplikasi-aplikasi local, secara otomatis.
e. Masing-masing DBMS berpartisipasi paling tidak pada satu apliakasi global.



Gambar 10.1 *DBMS Terdistribusi*

Beberapa fungsi khusus /spesifik yang disediakan oleh DBMS terdistribusi antara lain sebagai berikut.

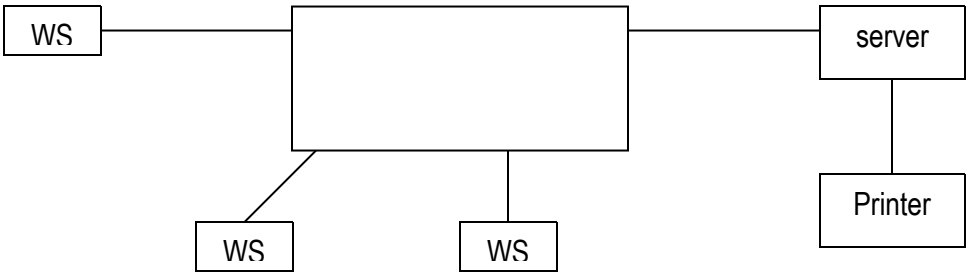
1. Schema Integration (skema yang terintegrasi)
2. Location transparency and distributed query processing (transparansi lokasi dan pemrosesan queri terdistribusi).
3. Concurrency Control and failure handling (pengontrolan dan penanganan kesalahan secara bersama).
4. Administration (administrasi).

10.2.3 SISTEM BASIS DATA HETEROGEN

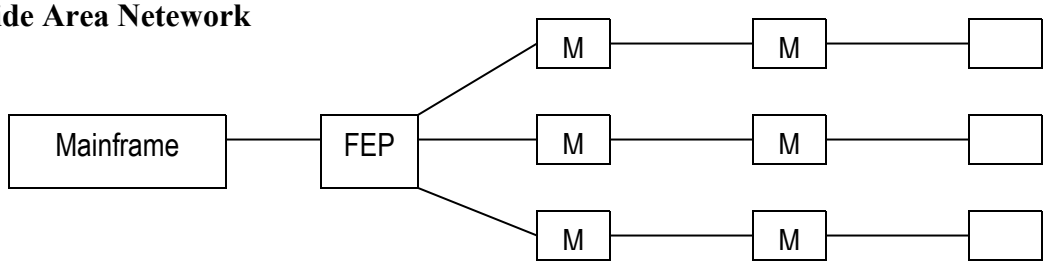
Sistem basis data terdistribusi yang heterogen bisa saja berada di beberapa tempat, masing-masing tempat memelihara sistem basis data lokalnya, masing-masing tempat dapat memproses transaksi lokal. Data disimpan pada setiap komputer, tidak ada hubungan antara organisasi data yang berbeda. Pemakaian dapat mengakses ke kmputer lain, namun harus tahu bagaimana data diorganisasi. Heterogen basis data di dalam sebuah jaringan berbentuk seperti berikut ini.

1. Basis data dapat mendukung perbedaan model data (hirarki, jaringan, relasional atau objek oriented).
2. Platform komputer dapat berbeda (Micro, Mini, Mainframe).
3. DBMS dapat disediakan oleh vendor yang berbeda.
4. Perbedaan bahasa query dapat digunakan dalam basis data yang berbeda.
5. Istilah lain yang sering digunakan untuk basis data heterogen adalah;
   - MDBS (Multi Database System)
   - FDBS (Federated Database System)
   - FDBMS (Federated Database Management System)
   - DDBS (Distributed Data Base Management)
6. Konfigurasi jaringan komunikasi juga berbeda (seperti LANs, WANs, TPC /IP, SNA, DECNET, OSI), seperti terlihat dalam beberapa topologi jaringam di bawah ini:

a. Local Area Network

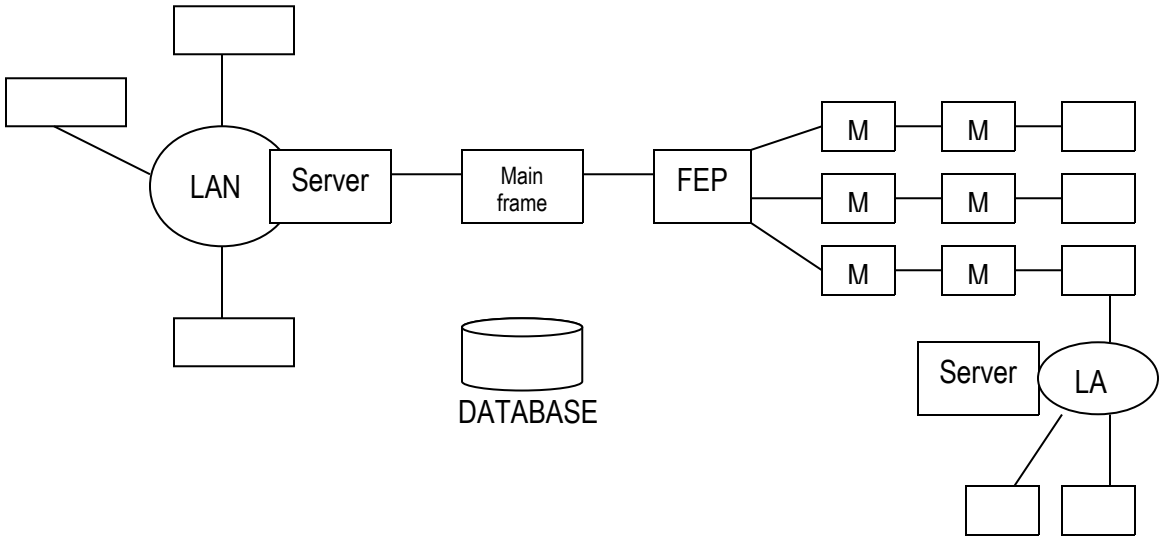


b. Wide Area Netework

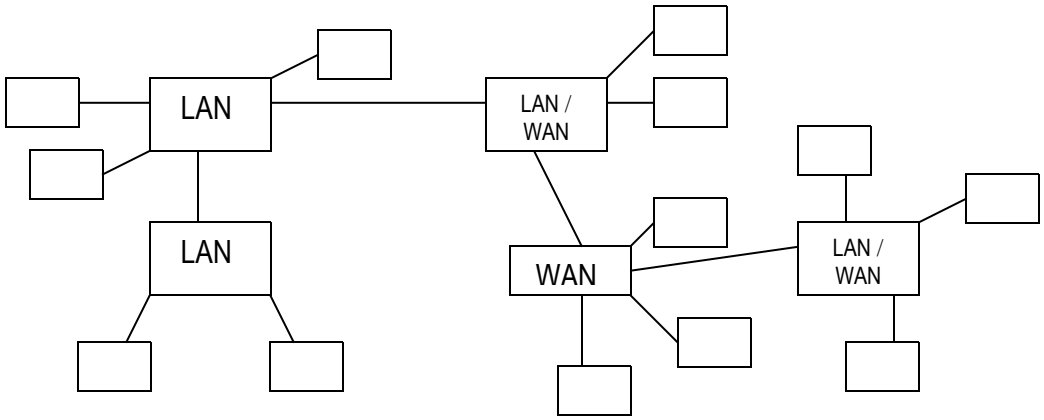


- FEP = Font – End Processor

c. **LAN + WAN**



d. Super Network (Internal)



Gambar 10.2 (a, b, dan c) *Gambar Topologi Jaringan*

10.2.4 DDTMS (DISTRIBUTED DATA TRANSACTION MANAGEMENT SISTEM / MANAJEMEN SISTEM TRANSAKSI DATA TERDISTRIBUSI)

Manajemen Sistem Transaksi Data Terdistribusi merupakan sekumpulan modul-modul software di mana pengaturan semua distribusi data (file dan database) dan menggabungkan beberapa transaksi yang berhubungan dan tersebar pada banyak computer. Manajemen Sistem Transaksi Data Terdistribusi merupakan system yang kompleks antara Client dan Server.

1. Tantangan Disain dan implementasi dari Manajemen Sistem Data Terdistribusi berupa:
   a. bagaimana cara mengakses data secara langsung dalam suatu jaringan komunikasi.
   b. bagaimana cara mengupdate data di sebuah jaringan komunikasi.
   c. bagaimana cara menjaga kebersamaan dan standardisasi data, sementara data tersebut selalu diakses oleh banyak pemakai (user).
   d. bagaimana menyusun transaksi dan meningkatkan respon terhadap setiap keinginan pemakai (user).
2. DTM (Distributed Transaction Manager / Manajer Transaksi Terdistribusi)
   Manjager Transaksi Terdistribusi berfungsi mengatur pelaksanaan sebuah transaksi melalui banyak sistem (sistem yang heterogen).
3. Keunggulan DDTMS (Distributed Data Transaction Management sistem/Manajemen Sistem Transaksi Data Terdistribusi) antara lain sebagai berikut.
   a. Mengatur biaya komunikasi dengan menempatkan data di tempat yang sering diakses.
   b. Meningkatkan kehandalan dan juga keberadaan dari sistem.
   c. Meningkatkan kapasitas sebuah sistem.
   d. Menigkatkan kinerja sistem.
   e. Memperbolehkan pengguna untuk mengontrol.
4. Kelemahan Manajemen Sistem Transaksi Data Terdistribusi antara lain sebagai berikut.
   a. Meningkatkan kompleksitas dari sistem.
   b. Membuat pengendalian terpusat lebih sulit dibandingkan dengan pengendalian terdistribusi.
   c. Penjagaan (security) terhadap data item lebih sulit dilakukan, karena semakin banyak yang mempunyai hak untuk menakses data.
   d. Membuat hasil kinerja suatu sistem transaksi lebih sulit dilakukan.
   e. Menurunkan hasil kinerja suatu sistem secara menyeluruh.
5. Distributed Operating Sistem (DISOS)
   Distributed Operating Sistem (DISOS) menyediakan sarana Opoerating sistem (sistem operasi) secara lengkap dan transparan kepada user / pemakai, juga menyediakan transparansi kepada user dengan jaringan komputer yang terdistribusi. Distributed Operating Sistem (DISOS) adalah bagian dari Distributed Data and Transaction Management sistem (DDTMS).
   Contoh adalah sebagai berikut.
   a. Distributed Computing Platform
      Distributed Computing Platform (DCP) merupakan gabungan antara distributed operating sistem (sistem operasi terdistribusi), distributed data base manager (manajer basis data terdistrbusi), distributed transaction manager (manajer transaksi terdistribusi), dan fasilitas jaringan komunikasi untuk mendukung program aplikasi yang berbeda spesifikasinya.
   b. Sistem Operasi Terpusat, merupakan kebalikan dari sistem operasi terdistribusi.
   c. Tingkatan transparan yang disediakan oleh Distributed Computing Platform (DPC) antara lain sebagai berikut.

1. End user transparasy
2. Developer transparansy
3. Designer transparansy
4. Manager transparansy

10.2.5 DFM (Distribusi File Management / Manajemen File Terdistribusi)

Distribusi Manajemen File akan menyediakan akses yang transparan, dan manipulasi fiel (membuka, membaca, menulis, dan menutup lokasi file yang diakses) jarak jauh dari program aplikasi dab atau pengguna, serta dapat melakukan manipulasi dan administrasi data pada komputer yang berbeda dalam suatu jaringan komunikasi.

1. Layanan DFM antara lain meliputi:
2. seleksi dan deseleksi file jarak jauh untuk diakses,
3. pembuatan dan penghapusan file jarak jauh. Membaca dan merubah file data jarak jauh,
4. pengontrolan fungsi-fungsi melalui penguncian / pembukaan,
5. keamanan dari akses yang tidak berhak.

10.2.6 DDBM (Distribusi Database Manager / Manajer Sistem Basis Data Terdistribusi)

Manajer Sistem Basis Data Terdistribusi akan menyediakan akses yang transpasran dalam memanipulasi basis data dan administrasi jarak jauh dari basis data dalam sebuah jaringan komunikasi.

10.3 ORGANISASI SISTEM TERDISTRIBUSI

Organisasi sistem terdistribusi terdiri atas tiga dimensi yang dapat dilihat sebagai berikut.

10.3.1 Tingkat Kebersamaan (lavel of sharing)

Kebersamaan yang dimaksud pada tingkatan ini adalah kebersamaan data (data sharing), serta kebersamaan data dan program (data plus program sharing).

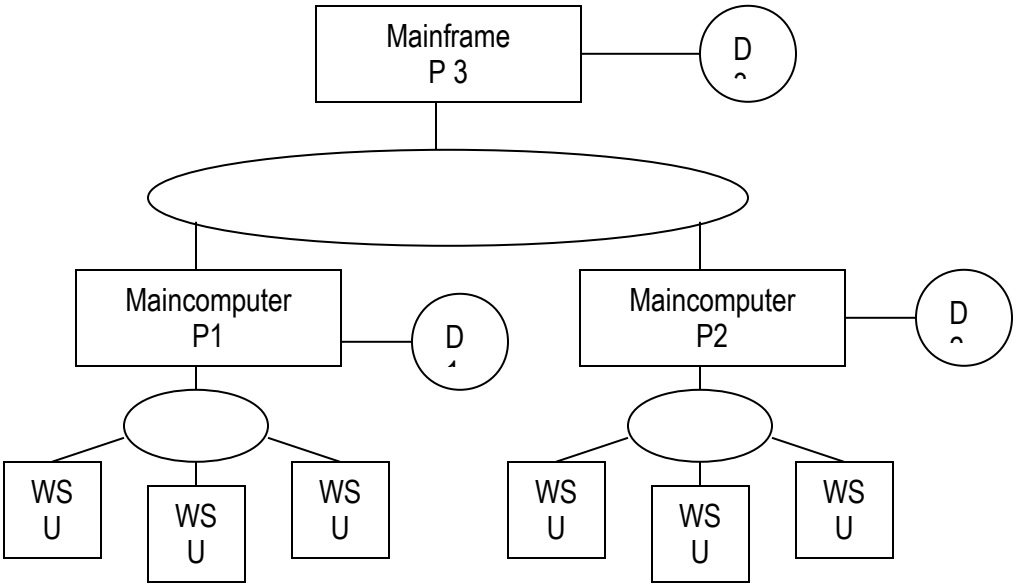
10.3.2 Tingkah laku pola akses (behavior of access pattern)

Tingkah laku pola akses (behavior of access pattern) akan dilihat dari dua sisi yakni pola akses yang dilakukan secara dinamis, yang akan memerlukan pengetahuan terbaru (up to date) yang selalu berkembang dalam ilmu komputer, sedangkan pola akses yang kedua lebih bersifat statis.

10.3.3 Tingkat pengetahuan atas tingkah laku pola

AKSES (LEVEL OF KNOWLEDGE ON BEHAVIOR OF ACCESS PATTERN BEHAVIOR)

Di bawah ini merupakan gambar dari aplikasi sistem terdistribusi:



Gambar 10.3 *Aplikasi Sistem Terdistribusi*

10.4 REPLICATION (REPLIKASI)

Replikasi dapat didefinisikan sebagai berikut, jika suatu relasi r direplikasi, maka sebuah salinan dari relasi disimpan pada dua atau lebih lokasi. Dalam kasus yang ekstrem, kita bisa juga mamiliki replikasi penuh (full replication), di mana sebuah salinan dari relasi r dapat disimpan pada setiap lokasi dalam sistem. Beberapa kelebihan dan kekurangan replikasi adalah sebagai berikut.

10.4.1 AVAILABILITY

Jika salah satu dari lokasi yang berisi relasi r mengalami kegagalan, maka relasi r dapat ditemukan pada lokasi lain, sehingga sistem dapat melanjutkan pemrosesan query yang berisi r, walaupun terjadi kegagalan.

10.4.2 MENINGKATKAN PEMROSESAN PARALLEL

Jika sebagian besar pengaksesan terhadap relasi r hanya menghasilkan pembacaan relasi, maka beberapa lokasi dapat memproses query yang berisi r dalam bentuk parallel. Makin banyak replikasi r, maka makin besar kemungkinan data yang dibutuhkan dapat ditemukan pada lokasi di mana transaksi diaksekusi. Dengan demikian replikasi data meminimalkan perpindahan data antar lokasi.

10.4.3 MENINGKATKAN OVER-HEAD TERHADAP UPDATE.

Sistem harus menjamin bahwa seluruh replikasi dari sebuah relasi r adalah konsisten. Jika tidak demikian, maka akan terjadi kesalahan perhitungan sehingga bila r di update, hasil update tersebut harus disebarkan ke seluruh lokasi yang berisi replikasi. Akibatnya akan meningkatkan over-head. Sebagai contoh: pada sebuah sistem pembayaran mahasiswa secara on-line (melalui bank), di mana informasi tunggakan di replikasikan pada berbagai lokasi, diperlukan jaminan bahwa total tunggakan mahasiswa sesuai pada seluruh lokasi.

Secara Umum replikasi meningkatkan unjuk kerja operasi baca, dan meningkatkan keberadaan data terhadap transaksi baca. Tetapi transaksi up-date memerlukan over-head yang lebih besar. Masalah pengontrolan konkurensi untuk mengatasi up-date oleh beberapa transaksi untuk replikasi data adalah jauh lebih kompleks dibandingkan dengan pendekatan sentralisasi untuk mengontrol konkurensi tersebut. Kita dapat menyederhanakan manajemen replikasi dari r dengan memilih salah satu replikasi sebagai salinan primer dari r (primary copy of r).

	Replikasi Penuh	Replikasi parsial	Partisi
Proses Query	Mudah	Kesulitan ◄——► sama	
Manajemen direktori	Mudah (atau nonexitensi)	Kesulitan ◄——► sama	
Pengendalian Concurrency	Moderat	sulit	Mudah
Keandalan	sangat tinggi	tinggi	Rendah
Realitas	Kemungkinan penerapan	realitas	Kemungkinan penerapan

Gambar 10.4 *Matriks Perbandingan Alternatif Replikasi*

10.5 FRAGMENTATION (FRAGMENTASI)

Jika relasi r di fragmentasikan, maka dibagi ke dalam sejumlah fragmen r_1, r_2,…., rn. Fragmen ini akan berisi informasi yang cukup untuk menyusun kembali relasi r mula-mula.

Seperti yang akan dilihat pada tabel di bawah ini, penyusunan kembali akan terjadi melalui penerapan operasi union.

Tabel 10.1 *Tabel/Relasi Deposit*

Nama_Cabang	No_Rekening	Nama_Pelanggan	Saldo
Citra Raya	305	Ruswadi	500
Citra Raya	226	Hudzaifah	336
Cikupa	177	Hudzaifah	205
Cikupa	402	Zahidah	10000
Citra Raya	155	Zahidah	62
Cikupa	408	Zahidah	1123
Cikupa	639	abdul	750

Terdapat dua alternatif yang dapat dilakukan dalam fragmentasi, antara lain sebagai berikut.

10.5.1 HORIZONTAL FRAGMENTATION (FRAGMENTASI HORISONTAL)

Fragmentasi Horisontal didefinisikan oleh suatu operasi seleksi atas 'Pemilik relation' R1 dengan notasi: $R_1^j = {}_Q F_j (R_1)$, $1 \le j \le F_j$.

Notasi di atas merupakan formula seleksi untuk mendapatkan fragmentasi R_1^j.

Ada dua aspek penting untuk simple predicate (predikat sederhana) adalah sebagai berikut.

- Completencess (Kelengkapan)
  Suatu himpunan simple predicate P_r dikatakan 'complete' jika dan hanya jika terdapat suatu 'equal probabillity of access (probabiliytas akses yang identik satu sama lain)' oleh setiap aplikasi pada setiap dua tuple miliki suatu 'mintern fragment (fragmentasi minterm)' yang didefinisikan menurut P_r.
- Minimality (Standardisasi Terendah)
  Aspek ini sangat intuitif sehingga jika suatu predicate (predikat) mempengaruhi fragmentasi (misalkan: suatu fragmentasi atas. F_i dan f_j), maka harus ada paling sedikit satu aplikasi yang mengakses f_i dan f_j secara berbeda.

1. **Aturan Dasar**
   Aturan dasar completeness dan minimality, yang menyatakan bahwa suatu relasi atau fragmentasi harus dipartisi "ke dalam paling sedikit dua bagian yang dapat diakses secara berbeda oleh paling sedikit satu aplikasi "adalah F_i dari P_r: fragment f_i didefinisikan menurut suatu minterm predicate (predikat minterm) yang didefinisikan atas predicate (predikat) dari p_r.
2. **Derived Horizontal Fragmentation (Fragmentasi Horisontal Berasal)**
   Diberikan suatu link L dengan Pemilik (L) = S dan member (L) = R, derived horizontal fragment (Asal dari fragmentasi horizontal) R didefinisikan sebagai berikut.

$$R_I = R \text{ X } S_I, \ 1 \le I \le w$$

W merupakan hasil penjumlahan maksimum dari fregmentasi yang didefinisikan atas

$$R_1 \text{ dan } S_I = {}_Q F_1 (S_1)$$

dengan catatan bahwa, F_i adalah formula pendefinisian primary horizontal fragment S_1.

3. Checking For Correctness (Validasi Kebenaran)

Tiga kriteria yang dapat digunakan untuk melakukan validasi terhadap kebenaran suatu algoritma fragmentasi (fragmentasi horizontal) adalah sebagai berikut.

a. Completeness (kelengkapan)
b. Reconstruction (Rekonstruksi)
c. Disjoinness (Dibuat dalam beberapa bagian).

4. Kriteria Fragmentasi Horizontal

Fragmentasi horizontal memiliki dua kriteria sebagai berikut.

a. Setiap fragmen memiliki attribute yang sama dengan relasi.
b. Setiap fragmen menerima beberapa record dari relasi yang difragmentasi.

10.5.2 FRAGMENTASI VERTIKAL

Fragmentasi Vertikal akan memproduksi relasi dari fragmen-fragmen R_1, R_2, ..., ... Rr dengan setiap fragmen berisi suatu subset, dan beberapa atribut dari R termasuk primary key.

1. Information requirements (Komposisi yang perlu diketahui)

a. Frekuensi akses

Q = {q1, q2, ..., ... q_q} sebagai himpunan query yang berlaku atas relasi R(A_1, A_2, ..., ... An), maka 'attribute usage value (Nilai atribut yang terpakai)' didefinisikan:

$$\text{Use/gunakan } (q_1, A_j) = \begin{cases} 1 \text{ jika attribute Aj dirujuk oleh query q1} \\ 0 \text{ lainnya} \end{cases}$$

b. Attribute affinity measure

Notasi aff (A_1, A_j) akan menyatakan 'measure the bond (keterikatan / rangkaian ukuran)' antara dua atribut dari suatu relasi brdasarkan bagaimana mereka diakses oleh suatu aplikasi.

$$\text{Off}(A_i, A_j) = \sum \sum_{k \mid \text{user}(q_k, A_t) = 1_A \text{ user } [q_k, A_j] = 1}^{\forall S_i} \text{refj}(q_k)\, \text{accj}(q_k)$$

refe (q_k) : jumlah akses pada attribute [A_1, A_j) untuk setiap ekssekusi aplikasi qk pada site S_e

acce (q_k) : measure frekwensi akses dari aplikasi qk pada site S_e

2. Checking For Correctness (Validasi Kebenaran)

Tiga kriteria yang dapat digunakan untuk melakukan validasi terhadap kebenaran suatu algoritma fragmentasi (fragmentasi vertical) adalah sebagai berikut.

a. Completeness (kelengkapan).
b. Reconstruction (rekonstruksi).
c. Disjointness (dibuat dalam beberapa bagian).

3. Allocation (Penempatan)

'Penempatan' lebih merupakan masalah tata letak 'individual files' pada suatu jaringan komputer. Diasumsikan himpunan fragmen F = {F_1, F_2, ... Fr}, dan suatu jaringan berisi 'sites/ S = {S_1, S_2, ... Sm}, dengan suatu himpunan aplikasi Q = {Q_1, Q_2, ... Qq}. Maka masalah alokasi adalah mencari distribusi optimal F dan S.

Dua ukuran optimalitas yang dapat digunakan adalah sebagai berikut.

a. Minimal cost (biaya yang dikeluarkan minimal), sementara akan menghasilkan.
b. Performace (penampilan dan output) yang optimal dan dapat didistribusikan dengan baik.

4. **Kriteria Fragmentasi Vertikal**

Fragmentasi vertikal akan memiliki tiga kriteria sebagai berikut.

a. Setiap fragmen memiliki jumlah record sama dengan relation yang difragmentasi.
b. Setiap fragmen memiliki attribute primary key dari relation yang difragmentasi.l
c. Setiap fragmen menerima beberapa attribute bukan key.

10.5.3 Derajat Fragmentasi

Derajat Fragmentasi akan berada dalam Biaya yang tidak ada fragmentasi hingga fragmentasi pada tingkat tuple individual (horizontal) atau pada tingkat atribut individual (vertikal).

10.5.4 CORRECTNES RULES OF FRAGMENTATION (ATURAN KEBENARAN FRAGMENTASI)

1. **Kelengkapan (Completeness)**
   Jika relasi R didekomposisi menjadi fragmen R_1, R_2, … R_n, maka setiap item data yang dijumpai pada R dapat dijumpai juga pada salah satu R atau lebih.
2. **Rekonstruksi (Reconstruction)**
   Jika relasi R didekomposisi menjadi fragmen R_1, R_2, … R_n maka dapat didefinisikan satu operator ∇ sehingga R = ∇ R_1 $\forall R_1 \check{I} F_R$
3. **Disjoitness (Dibuat dalam beberapa bagian)**
   Jika relasi R didekomposisi secara horizontal menjadi fragmen R_1, R_2, … R_n dan item data di dalam Rj, maka di tidak terdapat di dalam fragmen lain kecuali R_R (k=j)

10.5.5 Simple Predicate (Predikat sederhana)

Diberikan relasi R(A_1, A_2, ...A_n), dimana A_1 adalah suatu atribut yang didefinisikan atas dominan Dengan, maka suatu 'simple predicate (predikat sederhana), P_j yang didefinisikan pada R akan memiliki bentuk sebagai berikut.

$$P_j: A_i \theta \text{ value (nilai)}$$

dengan $\theta \in \{ =, < \neq, \leq, >, \geq\}$ dan (value $\in D_1$).

Notasi Pr_i menyatakan himpunan semua 'simple predicate (predikat sederhana)' pada relasi R_i, dan anggota Pri dinyatakandengan Pr_j.

10.5.6 Minterm Predicate (Predikat Minterm)

Minterm Predicate (predikat minterm) merupakan hasil 'conjunction (penghubung)' dari beberapa 'simple predicates (predikat sederhana)'. Jika diberikan Pri = {Pi_1, Pi_2, ... Pi_m} sebagai himpunan 'simple predicates (predikat sederhana)' untuk relasi R, maka himpunan 'minterm predicates(predikat sederhananya' nya adalah M_1 ={m_i, mi_2, ... mi_z}.

Klausa di atas dapat didefinisikan sebagai berikut.

$$Mi = \{ m_{ij} \mid m_{ij} = pik \in Pr_i \; Pik^* \}, \; 1 \le k \le j \le z$$

dengan $Pik = Pik$ atau $Pik = \neg pik$

Jadi setiap 'simple predicate (predikat sederhana)' dapat muncul dalam suatu 'minterm predicate (predikat minterm)' dalam bentuk asli atau dalam bentuk negasi.

10.5.7 DUA HIMPUNAN DATA UNTUK 'QUANTITATIVE INFORMATION'

1. **Minterm selectivity**
   Minterm selectivity (Kepandaian Memilih Minterm) merupakan jumlah suatu relasi yang akan diakses oleh suatu query berdasarkan suatu minterm predicate (predikat minterm). Notasi minterm selectivity atas mi adalah sel ($\underline{m_1}$).
2. **Acces frequency**
   Acces frequency (frekuensi pengaksesan) merupakan frekwensi aplikasi user dalam mengakses data. Jika Q = {q_1, q_2, ... q_q}adalah himpunan query user, maka acc(qj) menyatakan frekwensi akses suatu query di dalam suatu periode.

10.6 IMPLEMENTASI ALJABAR RELASIONAL DALAM FRAGMENTASI

Terdapat suatu fragmentasi relasi Deposit dengan skema berikut.

Deposit_Skema = (Nama_Cabang , No_Rekening, Nama_Pelanggan, Saldo)

Relasi tersebut diatas akan terlihat seperti pada Tabel 10.1 Tabel / Relasi Deposit) di atas. Relais / tabel deposit tersebut akan di konversikan ke dalam kasus fragmentasi horizontal, vertikal, dan gabungan keduanya.

10.6.1 KASUS FRAGMENTASI HORIZONTAL

Relasi r dipartisikan ke dalam sejumlah subset, r1, r2, ..., rn. Setiap tupel dalam relasi r harus termasuk ke dalam salah satu fragmen sehingga relasi mula-mula dapat disusun kembali jika diperlukan.

Sebuah fragnmen dapat ditentukan sebagai suatu seleksi pada relasi global r, yakni sebuah predicate (predikat) Pi yang akan digunakan untuk menyusun fragmen ri sebagai berikut.

$r_1 = \sigma P_i (r)$

Penyusunan kembali relasi r dapat diperoleh dengan melakukan penggabungan (union) dari seluruh fragmen, yakni:

$$r = r_1 \cup r_2 \cup .....r_n$$

Sebagai gambaran, anggap saja bahwa relasi r merupakan relasi deposit seperti tabel 10.1 di atas. Relasi ini dapat dibagi ke dalam n fragmen yang berbeda, masing-masing berisi tupel rekening yang berasal dari sebuah cabang ((branch) tertentu. Seandainya sistem perbankan hanya mempunyai dua cabang, Citra Raya dan Valley-view, maka akan muncul dua fragmen yang berbeda.

$$\text{Deposit1} = \sigma_{\text{Nama\_Cabang}} = \text{“Citra Raya”}\ ^{(\text{deposit})}$$
$$\text{Deposit2} = \sigma_{\text{Nama\_Cabang}} = \text{“Cikupa”}\ ^{(\text{deposit})}$$

Tabel yang dihasilkan dari dua sintaks aljabar relasional di atas adalah sebagai berikut.

Tabel 10.2 (a dan b) *Fragmnetasi Horisotal dari Tabel / Relasi Deposit*

Nama_Cabang	No_Rekening	Nama_Pelanggan	Saldo
Citra Raya	305	Ruswadi	500
Citra Raya	226	Hudzaifah	336
Citra Raya	155	Zahidah	62

(a) Tabel Relasi Deposit1

Nama_Cabang	No_Rekening	Nama_Pelanggan	Saldo
Cikupa	177	Hudzaifah	205
Cikupa	402	Zahidah	10000
Cikupa	408	Zahidah	1123
Cikupa	639	abdul	750

(b) Tabel / Relasi Deposit2

Fragmen deposit 1 disimpan pada lokasi Citra Raya, dan fragmen deposit2 disimpan pada lokasi Cikupa. Pada contoh di atas, fragmen adalah disjoin. Dengan mengubah predikat seleksi (Selection Predicate / s) yang digunakn untuk menyusun fragmen, kita dapat mempunyai sebuah tupel tertentu dari r yang muncul dari sebuah r_i.

10.6.2 KASUS FRAGMENTASI VERTIKAL

Dalam bentuk yang sederhana, fragmen vertikal adalah sama seperti dekomposisi. Fragmentasi vertiakal dari r(R) mencakup ketentuan / definisi beberapa subset R_1, R_2, … R_n dari R sedemikian rupa sehingga:

$$R = R_1 \cup R_2 \cup \ldots \cup R_n$$

Setiap fragmen r_1 dan r di tentukan oleh:

$$r_1 = \pi R_1 (r)$$

Relasi r dapat disusun kembali dari fragmen-fragmen dengan melakukan natural joint:

$$r = r_1 \theta r_2 \theta \ldots \theta r_n$$

Pada umumnya fragmentasi vertikal dikerjakan dengan penambahan suatu atrbut khusus yang disebut tuple-id pada skema R. Tuple-id adalah suatu physical atau logical address untuk sebuah tupel. Karena setiap tupel pada r harus mempunyai sebuah alamat yang unik (unique address), maka atribut tuple-id merupakan sebuah key untuk argumentasi scheme (skema). Pada tabel 10.3 berikut, ditunjukkan relasi deposit1 yang merupakan relasi deposit dari tabel 2 dengan penambahan tuple-id.

Tabel 10.3 *Tabel / Relasi Deposit dari tabel 10.2 dengan tuple-id*

Nama_Cabang	No_Rekening	Nama_Pelanggan	Saldo
Citra Raya	305	Ruswadi	500
Citra Raya	226	Hudzaifah	336
Cikupa	177	Hudzaifah	205
Cikupa	402	Zahidah	10000
Citra Raya	155	Zahidah	62
Cikupa	408	Zahidah	1123
Cikupa	639	abdul	750

Tabel 10.4 berikut ini menunjukkan dekomposisi vertikal dari deposite-scheme. Dua relasi yang ditunjukkan pada tabel 10.4 berikut, merupakan hasil dari:

$$\text{Deposit3} = \sigma_{\text{deposit-scheme-3}}{}^{(\text{deposit1})}$$
$$\text{Deposit4} = \sigma_{\text{deposit-scheme-4}}{}^{(\text{deposit2})}$$

Untuk menyusun kembali relais deposit mula-mula dari fragmen-fragmen yang ada, kita harus lakukan dengan operasi natural join. Sitaks dalam aljabar relasional sebagai berikut.

$$\pi_{\text{deposit-scheme-3}}{}^{(\text{deposit3 q deposit4})}$$

Tabel yang akan dihasilkan dari sintaks aljabar relasional di atas adalah sebagai berikut.

Tabel 10.4 (a dan b) *Fragmentasi vertikal dari relasi deposit*

Nama_Cabang	Nama_Pelanggan	Tuple-id
Citra Raya	Ruswadi	1
Citra Raya	Hudzaifah	2
Cikupa	Hudzaifah	3
Cikupa	Zahidah	4
Citra Raya	Zahidah	5
Cikupa	Zahidah	6
Cikupa	Abdul	7

(a) Tabel Relasi Deposit3

No_Rekening	Saldo	Tuple-id
305	500	1
226	336	2
177	205	3
402	10000	4
155	62	5

(b) Tabel / Relasi Deposit4

Perhatikan ekspresi (deposit3 θ deposit4). Ekspresi tersebut merupakan bentuk khusus dari operasi natural join, seperti yang telah kita bahas pada bab IV (Aljabar Relasional). Atribut joint adalah Tuple-id karena tuple-id menyatakan suatu alamat sehingga memungkinkan untuk menyamakan sebuah tuple dari deposit3 yang bersesuaian dengan tuple dari deposit4 dengan menggunakan alamat yang diberikan oleh nilai tuple-id. Alamat tersebut akan memberikan pencarian langsung(direct retrieval) terhadap tuple tanpa memerlukan sebuah index sehingga natural join dapat dihitung lebih efisien. Hal yang harus diwaspadai pada fragmentasi vertical adalah terjadinya lossy decomJabatan.

10.6.3 MIXED FRAGMENTATION (FRAGMENTASI GABUNGAN)

Relasi r dibagi kedalam sejumlah fragmen relasi r_1, r_2, … r_n. Setiap fragmen diperoleh sebagai hasil penggunaan skema fragmentasi horizontal atau vertical pada relasi r, atau sebuah fragmen dari relasi r yang diperoleh sebelumnya.

Sebagai gambaran anggap bahwa relasi r adalah relasi deposit seperti pada tabel11. Relasi ini pada awalnya dibagi ke dalam fragmen deposit 3 dan deposit 4. Pada

fragmentasi gabungan kita dapat membagi fragmen deposit3 dengan menggunakan skema fragmentasi horizontal ke dalam dua buah fragmen, yaitu:

$$\text{Deposit3a} = \sigma_{\text{Nama\_Cabang}} = \text{“Citra Raya”}^{(\text{deposit3})}$$
$$\text{Deposit3b} = \sigma_{\text{Nama\_Cabang}} = \text{“Cikupa”}^{(\text{deposit3})}$$

Jadi, relasi r dibagi ke dalam tiga buah fragmen, yaitu deposit3a, deposit3b, deposit4, di mana masing-masing deposit menempati lokasi yang berbeda.

-oo0oo-