

MODEL RELASI DAN NORMALISASI DATABASE

**Fakultas Ilmu Administrasi
Administrasi Bisnis
Universitas Brawijaya
Malang 2013**

DAFTAR ISI

Halaman Judul.....	i
Daftar Isi.....	ii
A. Relasi.....	1
B. Normalisasi	7
C. Denormalisasi.....	22

A. RELASI

1. Pengertian

Konstruksi utama untuk merepresentasikan data dalam model relasional adalah relasi. Relasi terdiri dari skema relasi dan contoh relasi. Contoh relasi adalah tabel, dan skema relasi mendeskripsikan kepala kolom dari tabel tersebut. Sebuah database relasional adalah kumpulan item data yang diatur sebagai satu set tabel resmi dijelaskan dari mana data dapat diakses dengan mudah. Sebuah database relasional dibuat menggunakan model relasional. Perangkat lunak yang digunakan dalam database relasional disebut sistem manajemen database relasional (RDBMS). Sebuah database relasional adalah pilihan utama dalam menyimpan data, lebih dari model lain seperti model database hirarkis atau model jaringan. Ini terdiri dari tabel nomor n dan meja masing-masing memiliki kunci sendiri utamanya.

2. Karakteristik Relasi

Relasi dalam model basis data relasional memiliki karakteristik :

- a. Semua entry / elemen data pada suatu baris dan kolom tertentu harus mempunyai nilai tunggal (single value), atau suatu nilai yang tidak dapat dibagi lagi (atomic value), bukan suatu kelompok pengulangan
- b. Semua entry / elemen data pada suatu kolom tertentu dalam relasi yang sama harus mempunyai jenis yang sama
- c. Masing-masing kolom dalam suatu relasi mempunyai nama yang unik
- d. Pada suatu relasi / tabel yang sama tidak ada dua baris yang identik

3. Tahapan-Tahapan Relasi

Ada tiga langkah dalam merancang database, yaitu :

3.1 Perancangan Database Konseptual (*Conceptual Database Design*)

Perancangan secara konsep merupakan langkah pertama dalam merancang database. Sesuai dengan namanya, pada tahap ini hanya menentukan konsep-konsep yang berlaku dalam sistem database yang akan dibangun. Dalam tahap ini, setidaknya harus mengetahui :

1. Prosedur kerja secara keseluruhan yang berlaku pada sistem yang sedang berjalan.

2. Informasi (output) apa yang diinginkan dari database ?
3. Apa saja kelemahan-kelemahan dari sistem yang sedang berjalan ?
4. Pengembangan sistem di masa yang akan datang.
5. Bagaimana tingkat keamanan data saat ini ?
6. Siapa saja yang terlibat dalam sistem yang sedang berjalan.
7. Apa saja input yang perlukan ?

Seorang perancang database harus paham benar terhadap sistem yang sedang berjalan dan harus mengetahui sistem yang bekerja pada database yang akan dibangun serta output apa yang diharapkan. Jika anda akan merancang Sistem Informasi Akademik, maka anda harus mengetahui bagaimana prosedur kerja dalam dunia akademik secara keseluruhan, siapa-siapa saja yang terlibat didalamnya, apa saja peraturan-peraturan yang berlaku, apakah sebuah ruang kuliah dibatasi siswanya ?, apakah mahasiswa dengan IPK < 2.00 boleh lulus atau tidak ?, bagaimana prosedur transformasi dari nilai angka ke nilai huruf ?, apakah semua karyawan boleh mengakses database tersebut ?, dan lain sebagainya. Pemahaman seorang perancang database terhadap sistem yang akan dibangun sangat menentukan baik atau tidaknya hasil rancangan databasenya.

3.2 Perancangan Database Logik

Perancangan database logik merupakan tahapan untuk memetakan proses perancangan konseptual kedalam model database yang akan digunakan, apakah model data hirarki, jaringan atau relasi. Perancangan database secara logik ini tidak tergantung pada DBMS yang digunakan, sehingga tahap perancangan ini disebut juga pemetaan model data. Berikut langkah-langkah dalam merancang database logik :

- Mendefinisikan Entity yang Dibutuhkan

Entity adalah sesuatu yang mudah diidentifikasi dengan mudah dari suatu sistem database, bisa berupa objek, orang, tempat, kejadian atau konsep yang informasinya akan disimpan. Hal-hal yang terlibat dalam suatu sistem database dapat dijadikan entity. Dari sekian banyak kemungkinan entity yang ada, maka harus memilih-milah entity mana saja yang sesuai dan mampu mengakomodasi Informasi Akademik, ada banyak kemungkinan yang bisa dijadikan entity, Misalnya entity mahasiswa, matakuliah, dosen, fakultas, jurusan, lokal dan lain

sebagainya. Maka secara sederhana, dapat ditentukan tiga entity utama yang terlibat dalam proses kegiatan akademik, yaitu :

1. Entity Mahasiswa, berfungsi untuk menyimpan data mahasiswa.
2. Entity Dosen, berfungsi untuk menyimpan data dosen, dan
3. Entity Matakuliah, untuk menyimpan data matakuliah.

- Menentukan Attribut setiap Entity Beserta Kuncinya

Setelah menentukan entity-entity yang terlibat pada sistem database yang dirancang, langkah berikutnya adalah menentukan attribut yang melekat pada entity tersebut. Attribut adalah ciri khas yang melekat pada suatu entity dan menunjukkan item sejenis. Sama halnya dalam menentukan entity, dalam menentukan attribut ini juga banyak kemungkinan, maka harus memilah-memilah attribut apa saja yang diperlukan oleh sistem database yang dirancang. Berikut beberapa entity yang mungkin pada entity mahasiswa :

- Nopb
- Nama
- Tempat Lahir
- Tanggal Lahir
- Fakultas
- Jurusan
- Agama
- Jenis Kelamin
- Tanggal Masuk
- Nama Pembimbing Akademik
- Asal SMU
- Alamat
- Nama Orang Tua
- Pendidikan Orang Tua
- Pekerjaan Orang Tua
- Alamat Orang Tua
- Dan seterusnya.

Anda boleh saja menggunakan semua kemungkinan attribut tersebut, selama attribut-attribut tersebut dibutuhkan dalam sistem database. Berikutnya adalah menentukan attribut kunci (*key*) dari entity. Kunci ini bersifat unik, sehingga antara satu tuple dengan tuple yang lainnya tidak boleh sama, disebut juga primary key. Namun perlu juga diketahui bahwa tidak semua kunci bersifat unik, tergantung kepada keberadaan attribut tersebut pada suatu entity. Sebuah kunci dapat saja berupa satu attribut, bisa juga terdiri dari beberapa attribut. Kunci ini akan digunakan nantinya dalam relasi antar entity. Dalam entity mahasiswa, attribut nobp dapat dijadikan sebagai kunci, karena antara satu mahasiswa dengan mahasiswa lainnya tidak akan ada mempunyai nobp yang sama (unik). Beda halnya, kalau attribut nama anda jadikan kunci, maka bisa saja ada dua mahasiswa atau lebih yang mempunyai nama sama, artinya attribut nama tidak unik. Kalau dalam suatu entity tidak ada attribut yang bersifat unik, maka boleh menambahkan satu buah attribut lagi, sebagai kunci yang bersifat unik. Misalnya dalam entity matakuliah, terdapat attribut sebagai berikut :

- Nama matakuliah
- Sks
- Semester
- Pra syarat
- Sifat matakuliah, wajib ambil atau hanya matakuliah pilihan.
- Dan sebagainya.

Terlihat pada daftar attribut diatas, tidak ada attribut yang bersifat unik, maka anda tambahkan satu buat attribut lagi dengan nama kode matakuliah, attribut ini yang aka dijadikan kunci, karena masing matakuliah mempunya kode yang berbeda. Dalam entity dosen, banyak juga attribut yang dapat ditentukan, seperti :

- NIP
- Nama Dosen
- Pendidikan dosen (S.1, S.1, S.3)
- Status Perkawinan
- Jenis kelamin
- Nama Istri
- Jumlah Anak

- Alamat
- Bidang Keahlian
- Jurusan
- Tanggal diangkat
- Dan sebagainya

Pada prinsipnya, semakin banyak attribut dari suatu entity yang ditentukan, semakin banyak pula informasi detail yang diperoleh terhadap entity tersebut, tentunya hal ini juga berimbang kepada semakin besarnya kapasitas daya tampung dalam media penyimpanan database. Namun, yang penting adalah menggunakan entity dan attribut yang diperlukan saja. Berikut daftar attribut dan kunci dari entity Mahasiswa, matakuliah dan dosen yang digunakan dalam perancangan Sistem Informasi Akademik Fakultas Ekonomi Universitas Helga Jaya Padang :

Entity	Attribute
Mahasiswa	1 <u>NoBP</u>
	2 Nama Mahasiswa
	3 Tempat Lahir mahasiswa
	4 Tanggal Lahir Mahasiswa
	5 Agama Mahasiswa
	6 Jenis kelamin Mahasiswa
	7 Pembimbing Akademik
	8 Alamat
Matakuliah	1 <u>Kode Matakuliah</u>
	2 Nama Matakuliah
	3 Sks
	4 Semester
	5 Pilihan
Dosen	1 <u>NIP</u>
	2 Nama Dosen
	3 Jurusan
	4 Bidang Keahlian
	5. Alamat
Keterangan : attribut yang di cetak tebal dan bergaris bawah merupakan kunci utama (Primary Key).	

- Menentukan Relasi antar Entity Beserta Kunci Tamunya.

Setelah menentukan entity dan attribut beserta kuncinya, maka selanjutnya adalah menentukan relasi antar entity. Bisa saja antara satu entity dengan entity yang lainnya tidak saling berhubungan, tapi entity tersebut berhubungan dengan entity yang satu lagi. Jika antara satu entity dengan entity yang lain saling berhubungan, maka hubungan tersebut dinyatakan sebagai entity baru, dan harus ditentukan pula

attribut dan field kuncinya. Entity hasil relasi pasti mempunyai kunci tamu (*foreign key*). Kunci tamu adalah attribute yang berfungsi sebagai kunci pada entity yang lain, tapi digunakan juga sebagai kunci pada entity hasil relasi, maka keberadaan attribut tersebut pada entity hasil relasi di sebut kunci tamu. Untuk menentukan relasi antar tabel, dapat dilakukan dengan merelasikan secara satu per satu (*one by one*). Dalam merancang sistem informasi akademik, ada 3 buah entity, yaitu :

1. Mahasiswa
2. Matakuliah
3. Dosen

Maka dapat melakukan relasi :

1. Entity Dosen dengan Entity Matakuliah
2. Entity Mahasiswa dengan Entity Dosen
3. Entity Mahasiswa dengan Entity Matakuliah

3.3 Perancangan Database Fisik

Perancangan database secara fisik merupakan tahapan untuk mengimplementasikan hasil perancangan database secara logis menjadi tersimpan secara fisik pada media penyimpanan eksternal sesuai dengan DBMS yang digunakan. Dapat disimpulkan bahwa proses perancangan fisik merupakan transformasi dari perancangan logis terhadap jenis DBMS yang digunakan sehingga dapat disimpan secara fisik pada media penyimpanan.

4. Tipe-tipe Kunci

Ada dua jenis kunci, yakni utama (primary) dan tamu (foreign). Kunci utama adalah suatu kolom (atau kelompok kolom) di mana nilai unik digunakan untuk mengidentifikasi setiap baris di dalam tabel. Oleh karena nilai kunci selalu unik (unique), bisa digunakan untuk menghapus dan mencegah/menjaga baris-baris duplikat. Selain itu ada kunci kandidat (candidate key) adalah atribut/kombinasi atribut yang memiliki kemungkinan untuk dipilih menjadi kunci primer, kunci sederhana (simple key) adalah kunci primer yang terdiri dari satu atribut. Kunci komposit (composite key) adalah kunci primer yang terdiri dari dua atribut atau lebih, kunci asing/ tamu (foreign key) adalah atribut (tunggal atau komposit) pada suatu relasi yang merupakan kunci primer di relasi lain pada basis

data yang sama, kunci asing memungkinkan suatu relasi (dependent relation) merujuk kepada relasi induk (parent relation), kunci biasanya digunakan sebagai indeks untuk mempercepat tanggapan terhadap query.

B. NORMALISASI

Jenis-jenis dependensi:

➤ Macam-macam dependensi, yaitu :

a. Dependensi fungsional

Definisi : Suatu atribut Y mempunyai dependensi fungsional terhadap atribut X jika dan hanya jika setiap nilai X berhubungan dengan sebuah nilai Y.

Notasi : $X \longrightarrow Y$ (X secara fungsional menentukan Y)

b. Dependensi fungsional sepenuhnya

Definisi : Suatu atribut Y mempunyai dependensi fungsional penuh terhadap X jika

- ❖ Y mempunyai dependensi fungsional terhadap X dan/atau
- ❖ Y tidak memiliki dependensi terhadap bagian dari X

c. Dependensi Total

Definisi : Suatu atribut Y mempunyai dependensi total terhadap atribut X jika

- ❖ Y memiliki dependensi fungsional terhadap X dan
- ❖ X memiliki dependensi fungsional terhadap Y
- ❖ Notasi : $X \longleftrightarrow Y$

d. Dependensi Transitif

Definisi : Atribut Z mempunyai dependensi transitif terhadap X bila :

- ❖ Y memiliki dependensi fungsional terhadap X
- ❖ Z memiliki dependensi fungsional terhadap Y

e. Dependensi Partial

Definisi : Suatu atribut Y dikatakan memiliki dependensi parsial terhadap X apabila memenuhi dua kondisi sebagai berikut:

1. Y adalah atribut non-kunci primer dan X adalah kunci primer

2. Y memiliki dependensi terhadap bagian dari X (tetapi tidak terhadap keseluruhan dari X)

f. Dependensi Multivalued

Definisi : merupakan kendala penuh antara dua set atribut dalam relasi.

Normalisasi Database

Proses normalisasi merupakan proses pengelompokan data elemen menjadi *table-table* yang menunjukkan *entity* dan relasinya. Pada proses normalisasi selalu diuji pada beberapa kondisi, apakah ada kesulitan pada saat menambah / *insert*, menghapus / *delete*, mengubah / *update*, dan membaca / *retrieve* pada suatu *database*. Bila ada kesulitan pada pengujian tersebut, maka relasi tersebut dipecahkan menjadi beberapa *table* lagi, sehingga diperoleh *database* yang optimal, sedangkan tujuan dari normalisasi adalah untuk membuat agar data yang ada tidak redundant dan memiliki data integrity yang kuat sehingga ketika kita melakukan relasi antara table akan dengan mudah kita menjaga data integrity dan mendapatkan datanya, selain itu normalisasi juga digunakan untuk mengeliminasi anomali

Anomali seringkali disebut juga dengan *update anomaly* yaitu efek samping yang tidak dikehendaki, yang terjadi jika relasi tidak pada bentuk normal tertentu. Terdapat 3 bentuk anomali yaitu:

- **Anomali penyisipan**, terjadi ketika dilakukan penyisipan tuple pada suatu relasi. Biasanya terjadi karena nilai *primary key* tidak diketahui saat penyisipan.
- **Anomali penghapusan**, terjadi sewaktu dilakukan penghapusan tuple dari relasi, padahal tuple tersebut mengandung sebagian informasi yang penting dan tidak boleh dihilangkan.
- **Anomali pengubahan**, terjadi karena adanya redundansi data. Ketika akan mengubah nilai data suatu atribut, maka seluruh pemunculan data tersebut harus ikut diubah.

Proses Normalisasi

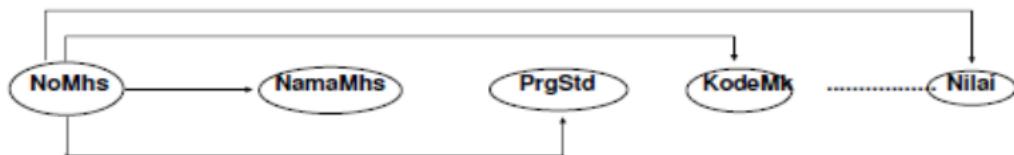
Proses Normalisasi data dapat diringkas sebagai berikut:

1. Menemukan entitas-entitas utama dalam model data
2. Menemukan hubungan antara setiap entitas
3. Menentukan atribut yang dimiliki masing-masing entitas

Bentuk Tidak Normal

Berupa data yang diterima, tanpa membaginya kedalam tabel-tabel yang ditentukan:

NoMhs	NamaMhs	PrgStd	KodeMk	NamaMk	NamaDs	Kantor	Nilai
0231	Cahyono	Statistik	MSM350	Kalkulus Lanjut	SUPOMO	R12	A
0231	Cahyono	Statistik	MSM465	Struktur Data	BUDIONO	R05	C
0232	Basuki	Ilmu Komp.	MSM465	Struktur Data	BUDIONO	R05	A
0232	Basuki	Ilmu Komp.	MSM300	Met. Statistik 1	SUBANAR	R04	B
0232	Basuki	Ilmu Komp.	MSM400	Analisis Data	JANOE	R10	C



Tahun 1970 E.F. Codd salah seorang perintis perintis teknologi basis data, mendefinisikan tiga bentuk normal yaitu:

1. Bentuk normal pertama (1NF)
- Relasi berada pada bentuk normal pertama jika tidak terdapat group-group berulang.
 - Adalah suatu keadaan yang membuat setiap perpotongan baris dan kolom dalam relasi hanya berisi satu nilai
 - Untuk membentuk 1NF perlu dilakukan langkah-langkah menghilangkan atribut-atribut yang memiliki nilai ganda.

MAHASISWA

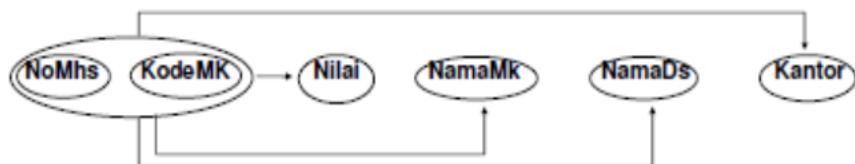
NoMhs	NamaMhs	PrgStd
0231	Cahyono	Statistik
0232	Basuki	Ilmu Komputer

MHS-MK

NoMhs	KodeMk	NamaMk	NamaDs	Kantor	Nilai
0231	MSM350	Kalkulus Lanjut	SUPOMO	R12	A
0231	MSM465	Struktur Data	BUDIONO	R05	C
0232	MSM465	Struktur Data	BUDIONO	R05	A
0232	MSM300	Met. Statistik 1	SUBANAR	R04	B
0232	MSM400	Analisis Data	JANOE	R10	C

2. Bentuk Normal kedua (2NF)

- Relasi berada pada normal kedua jika bentuk normal pertama dan pada relasi tersebut berlaku *full functional dependency*.
- Relasi MHS-MK meskipun berbentuk normal pertama tetapi masih terdapat anomali.
- Misal: jika tuple ke-2 dihapus, yang berarti mahasiswa 0231 membatalkan mata kuliah PAM261, maka informasi yang hilang tidak hanya mahasiswa membatalkan mata kuliah tetapi informasi tentang siapa yang mengajar mata kuliah tersebut juga ikut terhapus.
- Hal ini terjadi karena NamaMk, NamaDs, dan Kantor tidak bergantung pada *primary key*, tetapi hanya bergantung pada salah satu atribut komponen *primary key*, dengan kata lain pada relasi tersebut terdapat *partial dependency*



- Hal ini dapat diatasi dengan memecah relasi tersebut menjadi dua relasi yang lebih kecil.

Tabel 2. Relasi bentuk normal kedua

NoMhs	KodeMk	Nilai
0231	MSM350	A
0231	MSM465	C
0232	MSM465	A
0232	MSM300	B
0232	MSM400	C

KodeMk	NamaMk	NamaDs	Kantor
MSM350	Kalkulus Lanjut	SUPOMO	R12
MSM465	Struktur Data	BUDIONO	R05
MSM300	Met. Statistik 1	SUBANAR	R04
MSM400	Analisis Data	JANOE	R10

3. Bentuk Normal ketiga (3NF)

- Relasi berada pada bentuk normal ketiga jika memenuhi syarat bentuk normal kedua dan tidak mempunyai *transitive dependency*

KodeMk	NamaMk	NamaDs	Kantor
MSM350	Kalkulus Lanjut	SUPOMO	R12
MSM465	Struktur Data	BUDIONO	R05
MSM300	Met. Statistik 1	SUBANAR	R04
MSM400	Analisis Data	JANOE	R10

- Relasi di atas meskipun memenuhi syarat bentuk normal kedua tetapi masih terdapat anomali.
- Misalnya jika tuple pertama dihapus, yang berarti mata kuliah Kalkulus Lanjut 1 dihilangkan, ada informasi lain yang ikut hilang yaitu informasi tentang nama dosen dan kantornya.
- Demikian juga sewaktu akan disisipkan nama dosen baru, hal ini tidak akan bisa dilakukan sebelum dosen baru tersebut mengajar minimal satu mata kuliah karena *primary key* relasi tersebut adalah Kode_Mk.
- Anomali-anomali tersebut terjadi karena adanya *transitive dependency* berikut:

Kd_Mk ® Nama_Mk, Nama_Dsn

Nama_Dsn ® Kantor

atau

Kd_Mk ® Nama_Dsn ® Kantor

- Untuk menghilangkan anomali dari relasi tersebut, *transitive dependency* harus dihilangkan dengan memecah relasi menjadi dua buah relasi yang lebih kecil sbb:

KodeMk	NamaMk	NamaDs
MSM350	Kalkulus Lanjut	SUPOMO
MSM465	Struktur Data	BUDIONO
MSM300	Met. Statistik 1	SUBANAR
MSM400	Analisis Data	JANOE

NamaDs	Kantor
SUPOMO	R12
BUDIONO	R05
SUBANAR	R04
JANOE	R10

Tetapi kemudian muncul bentuk-bentuk normal yang baru, yaitu:

1. Boyce-Codd Normal Form (BCNF)
 - Bentuk normal ini merupakan perluasan dari bentuk normal ketiga.
 - Suatu relasi berada pada bentuk BCNF apabila setiap determinan merupakan *candidate key*.
 - Masalah BCNF akan muncul apabila suatu relasi mengandung tiga keadaan berikut:
 - ✓ Minimal terdapat dua *candidate key*
 - ✓ Seluruh *candidate key* bersifat komposit
 - ✓ Ada satu atribut yang berpartisipasi pada kedua *candidate key*.

Relasi non BCNF

DAFT_NILAI (No_Mhs, No_Tlp, Kd_Mk, Nilai)

No_Mhs	No_Tlp	Kode_Mk	Nilai
0231	88681	AM211	A
0231	88681	PAM261	B
0231	88681	PAM367	A
0232	88682	PAM333	C
0232	88682	PAM241	A
0233	88683	PAM345	B
0233	88683	PAM337	A
0345	88685	PAM522	B
0347	88687	PAM432	B

- Diasumsikan bahwa setiap mahasiswa mempunyai telepon dengan nomor yang berbeda.
- Relasi DAFT_NILAI mempunyai dua buah *candidate key*, yaitu **No_Mhs+Kd_Mk** dan **No_Tlp+Kd_Mk**. Atribut **Kd_Mk**

berpartisipasi pada kedua *candidate key*. Sehingga ketiga keadaan sebagai penyebab munculnya BCNF ada pada relasi tersebut.

- Relasi tersebut tidak memuat *transitive dependency* jadi relasi memenuhi syarat bentuk normal ketiga (3NF)
- Anomali:
 - ✓ Jika akan disisipkan No_Tlp untuk seorang mahasiswa baru, hal ini tidak akan bisa dilakukan sebelum mahasiswa tersebut mengikuti minimal satu mata kuliah. Kegagalan ini karena nilai Kd_Mk, sebagai salah satu atribut yang berperan dalam *primary key*, tidak diketahui pada saat penyisipan.
 - ✓ Jika akan dilakukan penghapusan pada tuple mahasiswa nomor 0347 karena mahasiswa tersebut membatalkan mata kuliah PAM432, informasi tentang nomor telepon mahasiswa tersebut ikut hilang.
- Untuk menghilangkan anomali, relasi dipecah menjadi dua buah relasi yang lebih kecil.
- Dalam memecah relasi tersebut perlu diperhatikan *functional dependency* yang ada yaitu:

No_Mhs, Kd_Mk ® Nilai

No_Tlp, Kd_Mk ® Nilai

No_Mhs ® No_Tlp

- Pada *functional dependency* ketiga, No_Mhs merupakan determinan tetapi bukan merupakan *candidate key*, sehingga relasi dipecah menjadi relasi NILAI dan TELEPON pada Tabe

a. NILAI (No_Mhs, Kode_Mk, Nilai)

No_Mhs	Kode_Mk	Nilai
0231	PAM211	A
0231	PAM261	B
0231	PAM367	A
0232	PAM333	C
0232	PAM241	A
0233	PAM345	B
0233	PAM337	A
0345	PAM522	B
0347	PAM432	B

b. TELEPON (No_Mhs, No_Tlp)

No_Mhs	No_Tlp
0231	88681
0232	88682
0233	88683
0345	88685
0347	88687

2. Bentuk Normal keempat (4NF)

Relasi berada pada bentuk normal keempat apabila memenuhi syarat BCNF dan tidak mempunyai multivalue *dependency*. Pada relasi BAHASA - Tabel 8 - perlu diperhatikan duplikasi data yang terjadi. Mahasiswa 0232 disimpan dalam 4 tuple masing-masing merupakan kombinasi antara Prg_StdI dan Bhs_Prg. Jika dilakukan dengan cara lain, misalnya seperti Tabel III. 8, maka informasi yang diperoleh akan berubah. Seolah-olah mahasiswa 0232 menguasai bahasa C ketika dia bertindak sebagai mahasiswa Komputer dan menguasai bahasa Cobol ketika bertindak sebagai mahasiswa Akuntansi. Padahal semestinya tidak demikian, sehingga penyimpanan dilakukan dengan mengkombinasikan kedua atribut tersebut.

Tabel 8 Contoh relasi dengan multivalue tanpa duplikasi tuple

BAHASA (No_Mhs, Prg_StdI, Bhs_Prg)

No_Mhs	Prg_StdI	Bhs_Prg
0232	Komputer	C
0232	Akuntansi	Cobol
0236	Statistik	Pascal
0236	Hukum	Pascal

Relasi pada Tabel 8 memenuhi syarat BCNF, akan tetapi didalamnya masih terdapat anomali. Diantaranya, jika mahasiswa 0232 menambah bahasa pemrograman yang dikuasai yaitu assembly, maka harus disisipkan

dua buah tuple sebagai kombinasi kedua atribut. Hal yang sama akan terjadi apabila terjadi penghapusan.

Untuk menghilangkan anomali tersebut maka *multivalued dependency* harus dihilangkan dengan memecah relasi menjadi dua buah relasi yang lebih kecil. Masing-masing relasi untuk menyimpan data dari atribut yang bernilai ganda, seperti pada Tabel berikut.

a. **PROGRAM_STUDI (No_Mhs, Prg_Std, Bhs_Prg)**

No_Mhs	Prg_Std
0232	Komputer
0232	Akuntansi
0236	Statistik
0236	Hukum

b. **BAHASA (No_Mhs, Prg_Std, Bhs_Prg)**

No_Mhs	Bhs_Prg
0232	Cobol
0232	C
0236	Pascal
0236	Pascal

3. Bentuk Normal kelima (5NF)

Relasi bentuk normal kelima sering disebut PJNF (*Projection Join Normal Form*), penyebutan PJNF karena untuk suatu relasi akan berbentuk normal kelima jika relasi tersebut dapat dipecah atau diproyeksikan menjadi beberapa relasi dan dari proyeksi-proyeksi itu dapat disusun kembali (JOIN) menjadi relasi yang sama dengan keadaan semula. Jika penyusunan ini tidak mungkin dilakukan dikatakan pada relasi itu terdapat *join dependencies* dan dikatakan bersifat *lossy join*.

BHS_KULIAH (No_Mhs, Kd_Mk, Bhs_Prg)

No_Mhs	Kd_Mk	Bhs_Prg
2342	PAM369	BASIC
2342	PAM369	PASCAL
2342	PAM260	PASCAL
2342	PAM260	ADA
2546	PAM260	PASCAL
2542	PAM369	BASIC

Relasi di atas berisi informasi bahasa pemrograman yang dipergunakan oleh mahasiswa untuk mengerjakan tugas-tugas mata kuliah. Diasumsikan tidak ada *functional dependencies* antara mata kuliah dan bahasa pemrograman yang dipakai. Selanjutnya relasi tersebut diproyeksikan menjadi tiga relasi berikut

a. KULIAH (No_Mhs, Kd_Mk)

No_Mhs	Kd_Mk
2342	PAM369
2342	PAM260
2546	PAM260
2542	PAM369

b. ALAT (No_Mhs, Bhs_Prg)

No_Mhs	Bhs_Prg
2342	BASIC
2342	PASCAL
2342	ADA
2546	PASCAL
2542	BASIC

c. BAHASA (Kd_Mk, Bhs_Prg)

Kd_Mk	Bhs_Prg
PAM369	BASIC
PAM369	PASCAL
PAM260	ADA
PAM260	PASCAL
PAM369	BASIC

:
Selanjutnya apabila tabel semula dihapus kemudian dilakukan operasi JOIN pada relasi-relasi itu, akan diperoleh tiga kemungkinan berikut ini.

a. JOIN (KULIAH, BAHASA) OVER No_Mhs

No_Mhs	Kd_Mk	Bhs_Prg
2342	PAM369	BASIC
2342	PAM260	PASCAL
2342	PAM369	ADA *
2342	PAM260	BASIC *
2342	PAM369	PASCAL
2342	PAM260	ADA
2546	PAM260	PASCAL
2542	PAM369	BASIC

b. JOIN (KULIAH, ALAT) OVER Kd_Mk

No_mhs	Kd_Mk	Bhs_Prg
2342	PAM369	BASIC
2342	PAM369	PASCAL
2342	PAM260	PASCAL
2342	PAM260	ADA
2546	PAM260	PASCAL
2546	PAM260	ADA *
2542	PAM369	BASIC
2542	PAM369	PASCAL *

c. JOIN (ALAT, BAHASA) OVER Bhs_Prg

No_Mhs	Kd_Mk	Bhs_Prg
2342	PAM369	BASIC
2342	PAM369	PASCAL
2342	PAM260	PASCAL

2342	PAM260	ADA
2546	PAM369	PASCAL *
2546	PAM260	PASCAL
2542	PAM369	BASIC

Pada masing-masing relasi hasil JOIN, ternyata diperoleh suatu relasi yang tidak sama persis dengan relasi awal tuple-tuple dengan tanda asterisk merupakan tuple tambahan/kelebihan yang tidak terdapat pada

relasi awal atau pada relasi awal terdapat *join dependency*. Hal ini terjadi karena atribut untuk operasi JOIN bukan merupakan determinan dari relasi awal.

Sintesis Relasi

Pada bagian ini, kita akan melihat desain relasional dari perspektif yang berbeda-perspektif sintesis. Dari perspektif tersebut, kita bertanya, “Untuk suatu himpunan atribut dengan ketergantungan fungsional tertentu, relasi apa yang harus kita bentuk?”

Pertama, amati bahwa dua atribut (misalnya, A dan B) dapat dihubungkan dalam tiga cara:

1. Keduanya menentukan satu sama lainnya:

$$A \rightarrow B \text{ dan } B \rightarrow A$$

Karena itu, A dan B memiliki relasi atribut one-to-one

2. Yang pertama menentukan yang lain.

$$\text{Jika } A \rightarrow B, \text{ tetapi } B \text{ bukan } \rightarrow A$$

A dan B memiliki relasi atribut many-to-one

3. Keduanya secara fungsional tidak berhubungan

$$\text{Jika } A \text{ bukan } \rightarrow B \text{ dan } B \text{ bukan } \rightarrow A$$

A dan B mempunyai relasi atribut many-to-many

Tipe-tipe Hubungan Atribut

Figure 4.21 Summary of Three Types of Attribute Relationships

Type of Attribute Relationship			
	One to One	Many to One	Many to Many
Relation Definition*	R(A,B)	S(C,D)	T(E,F)
Dependencies	A → B B → A	C → D D → C	E ↔ F F ↔ E
Key	Either A or B	C	(E,F)
Rule for Adding Another Attribute	Either A or B → C	C → E	(E,F) → G

* The letters used in these relation definitions match those used in Figure 4-22.

Hubungan Atribut One-To-One

Jika A menentukan B dan B menentukan A, maka nilai atribut memiliki hubungan one-to-one. Ini pasti karena jika A menetukan B, maka hubungan antara A dan B adalah many-to-one. Akan tetapi, juga benar bahwa jika B menentukan A, maka hubungan antara B dan A pasti many-to-one. Agar kedua pernyataan itu dapat diterima pada saat yang sama, hubungan antara A dan B sebetulnya pasti one-to-one (yang merupakan suatu kasus khusus dari many-to-one), dan hubungan antara B dan A sebenarnya juga adalah one-to-one. Oleh karena itu, hubungannya adalah one-to-one.

Kasus tersebut diilustrasikan oleh FID dan Fname dalam Contoh 2 serta 3 pada bagian sebelumnya untuk form normal domain. Key. Masing-masing atribut tersebut secara unik mengidentifikasi person fakultas. Akibatnya, satu nilai FID tepat berhubungan dengan satu nilai Fname, dan sebaliknya.

Tiga statement atau persamaan yang sama dapat diambil dari contoh FID dan Fname:

- Jika dua atribut secara fungsional saling menentukan satu sama lainnya, maka hubungan nilai datanya adalah one-to-one.
- Jika dua atribut secara unik mengidentifikasi entitas yang sama, maka hubungan nilai datanya adalah one-to-one.
- Jika dua atribut mempunyai hubungan one-to-one, maka keduanya secara fungsional saling menentukan satu sama lain.

pada saat membuat database dengan atribut-atribut yang memiliki hubungan one-to-one, kedua atribut tersebut harus muncul bersamaan setidaknya pada satu relasi. Atribut lainnya yang secara fungsional ditentukan oleh hal tersebut (suatu atribut yang secara fungsional ditentukan oleh atribut lainnya secara fungsional juga ditentukan oleh atribut yang lainnya) juga dapat berada pada relasi yang sama.

Mari pertimbangkan FACULTY (FI, Fname, GradFacultyStatus) dalam Contoh 3 pada bagian sebelumnya. FID dan Fname saling menentukan satu sama lainnya.

GradFacultyStatus juga dapat terjadi dalam relasi ini karena GradFacultyStatus ditentukan oleh FID dan Fname. Atribut-atribut yang secara fungsional tidak ditentukan oleh atribut-atribut tersebut mungkin tidak terjadi dalam suatu relasi dengannya. Mari kita perhatikan relasi FACULTY dan PREPARATION pada Contoh 2, di mana baik FID maupun Fname terjadi pada FACULTY, tetapi Class (dari PREPARATION) mungkin tidak muncul, sehingga Class tidak bergantung pada FID atau Fname. Jika kita menambahkan Class pada relasi FACULTY, maka key FACULTY akan memerlukan baik (FID, Class) ataupun (Fname, Class). Akan tetapi, pada kasus tersebut FACULTY tidak akan ada pada DK/NF karena ketergantungan antara FID dan Fname secara logis tidak akan tersirat oleh key-key mana pun yang mungkin.

Statement-statement tersebut dirangkum pada kolom pertama dari Peraga 4-21, dan aturan-aturan definisi record dicantumkan pada Peraga 4-22. Jika A dan B memiliki hubungan one-to-one, maka keduanya dapat berada pada relasi yang sama, katakan R. A menentukan B dan B menentukan A. Key Relasi dapat berupa A atau B. Suatu atribut baru, C, dapat ditambahkan ke R jika A atau B secara fungsional menentukan C.

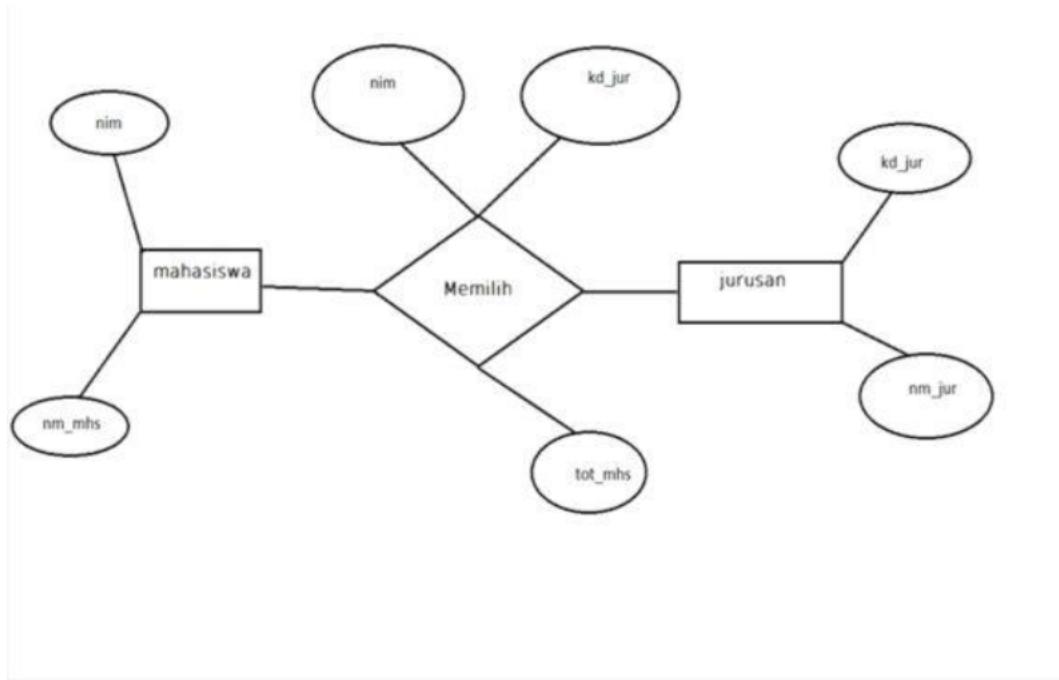
Atribut-atribut yang mempunyai hubungan one-to-one harus ada bersama sedikitnya pada satu relasi untuk menetapkan ekuivalensinya (sebagai contoh, FID sebesar 198 merujuk pada Professor Heart). Akan tetapi, umumnya kita tidak menginginkan atribut tersebut muncul bersama pada lebih dari satu relasi, karena hal tersebut menyebabkan duplikasi data yang tidak diperlukan. Seringkali, satu atau kedua atribut tersebut muncul pada relasi yang lain. Pada Contoh 2, Fname muncul baik pada PREPARATION maupun STUDENT. Meskipun mungkin untuk menempatkan Fname pada PREPARATION dan FID dalam STUDENT, namun hal tersebut umumnya adalah praktik yang buruk karena pada saat atribut-atribut dipasangkan dengan cara ini, salah satunya harus dipilih untuk menampilkan pasangan pada semua relasi yang lain. Fname dipilih pada Contoh 2.

Relasi Banyak ke Satu (Many to One)

Ini adalah kebalikan dari relasi satu ke banyak, dimana setiap record pada entity A hanya dapat berrelasi paling banyak 1 record pada entity B, tapi tidak sebaliknya, satu record pada entity B dapat berrelasi dengan beberapa record pada entity A. Dalam diagram E-R, relasi ini disimbolkan dengan angka **1** untuk menyatakan satu dan huruf **M** atau **N** untuk menyatakan banyak.

Contoh:

Dalam dunia akademik misalnya, beberapa (banyak) mahasiswa hanya mempunyai satu pilihan jurusan, sebaliknya satu jurusan dapat dipilih oleh beberapa (banyak) mahasiswa.



Keterangan :

- Entity mahasiswa mempunyai dua attribute, yaitu nomor buku pokok (**nim**) yang berfungsi sebagai field kunci, dan nama mahasiswa (**nm_mhs**).
- Entity jurusan juga mempunyai dua attribute, yaitu kode jurusan (**kode_jur**) sebagai field kunci dan nama jurusan (**nm_jur**).

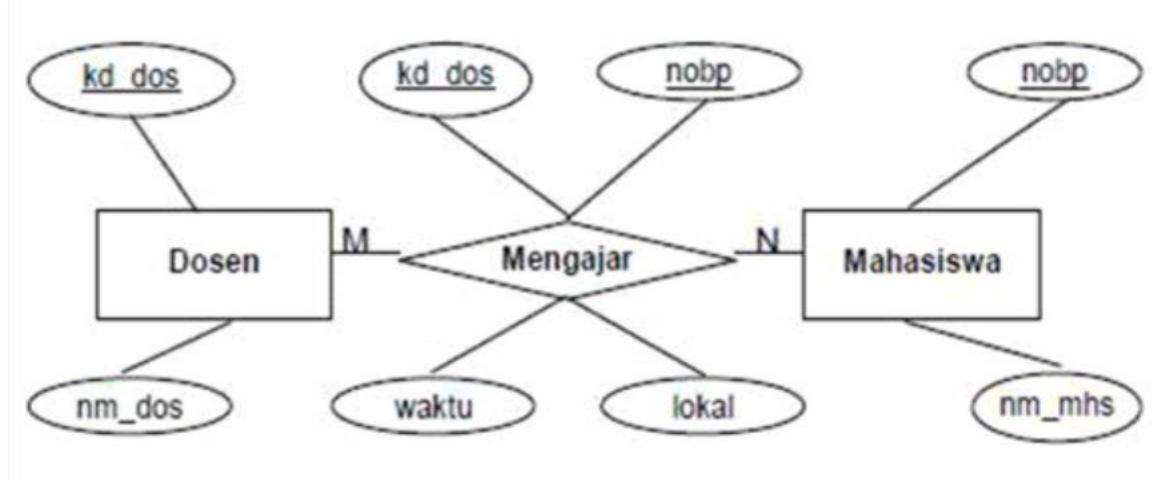
Relasi Banyak ke Banyak (Many to Many)

Artinya beberapa record pada entity A dapat berelasi dengan beberapa record juga pada entity B, begitu juga sebaliknya, beberapa record pada entity B dapat berelasi dengan beberapa record juga pada entity A. Dalam diagram E-R, relasi ini disimbolkan dengan huruf **M** atau **N** untuk menyatakan banyak.

Contoh:

Dalam hubungan antara mahasiswa dengan dosen pada perguruan tinggi, yaitu seorang dosen mengajar banyak mahasiswa, sebaliknya seorang mahasiswa dapat diajar oleh beberapa dosen, sehingga terjadi hubungan **banyak ke banyak**. Dalam diagram E-R, hal ini dapat digambar sebagai berikut:

Gambar



Keterangan:

1. Entity dosen mempunya dua attribute, yaitu kode dosen (**kd_dos**) yang berfungsi sebagai field kunci, dan nama dosen (**nm_dos**).
2. Entity mahasiswa juga mempunyai dua attribute, yaitu No. Buku Pokok (**NoBP**) sebagai field kunci dan nama mahasiswa (**nm_mhs**).
3. Hubungan antara kedua entity tersebut dinyatakan dalam entity mengajar, yang mempunyai 4 attribute, yaitu kode dosen (**kd_dos**) dan No.Buku Pokok

mahasiswa (**NoBP**) yang berfungsi sebagai kunci tamu (*foreign key*) pada entity mengajar serta attribute waktu mengajar (**waktu**) dan tempat mengajar (**lokal**).

4. Derajat relasi dinyatakan dengan **M : N**, yang menandakan bahwa hubungan antar entity adalah banyak ke banyak, seperti terlihat pada gambar diatas.

Denormalisasi Desain

Apakah denormalisasi database itu? denormalisasi database adalah pelanggaran aturan normalisasi atau menjabarkan suatu tataan database yang telah normal untuk meningkatkan performa pengaksesan data pada database. Database yang telah normal disini dimaksudkan database yang redundansi datanya minim sehingga data yang disimpan tidak mengalami kerancuan dalam proses pengaksesan.

C. DE-NORMALISASI

Apakah perbedaan normalisasi dan de-normalisasi? perbedaan normalisasi dan de-normalisasi adalah terletak pada redundansi data dan kompleksitas query. Pada redundansi data normalisasi lebih strik atau harus dihilangkan se bisa mungkin sehingga mengakibatkan apabila kita akan mengakses data dalam suatu database membutuhkan query yang kompleks. Berbeda dengan denormalisasi, denormalisasi disini tidak terlalu memikirkan tentang data yang redundant sehingga dalam mengakses data lebih cepat.

Apa sih pentingnya de-normalisasi dalam database? Apabila kita menilik lebih lanjut tentang proses pengaksesan yang dilakukan database sewaktu data yang berada dalam suatu tabel ada 1000 baris dengan 100 juta baris. Hal itu akan terasa sangat beda proses kita menunggu untuk dapat melihat data. Itupun apabila kita mengaksesnya dari beberapa tabel yang setiap tabel berisikan jutaan data dan kita hanya menginginkan sebagian saja. Dari situ denormalisasi diperlukan, untuk menjaga kestabilan performa suatu sistem.

Bagaimanakah cara melakukan denormalisasi? Kita dapat melakukan denormalisasi dalam 2 jenis :

1. melalui pembuatan kolom baru pada tabel / mengabungkan kolom pada tabel satu dengan yang lain.
2. melalui pembuatan tabel baru.

cara yang pertama dilakukan apabila data yang didenormalisasi hanya kecil dan digunakan untuk mempermudah pengaksesan data apabila diakses dalam satu tabel. Sedangkan yang kedua dilakukan apabila data yang terdapat dalam tabel tersebut merupakan rangkuman / rekapitulasi dari satu atau beberapa tabel yang pengaksesannya terpisah dari tabel yang ada.

contoh :

de-normalisasi pertama : total sks yang telah diambil seorang mahasiswa. ini dibentuk dari jumlah sks matakuliah yang pernah diambil.

denormalisasi kedua : pembuatan tabel jumlah kehadiran mahasiswa dalam satu semester. data ini dibentuk dari penjumlahan data harian mahasiswa.