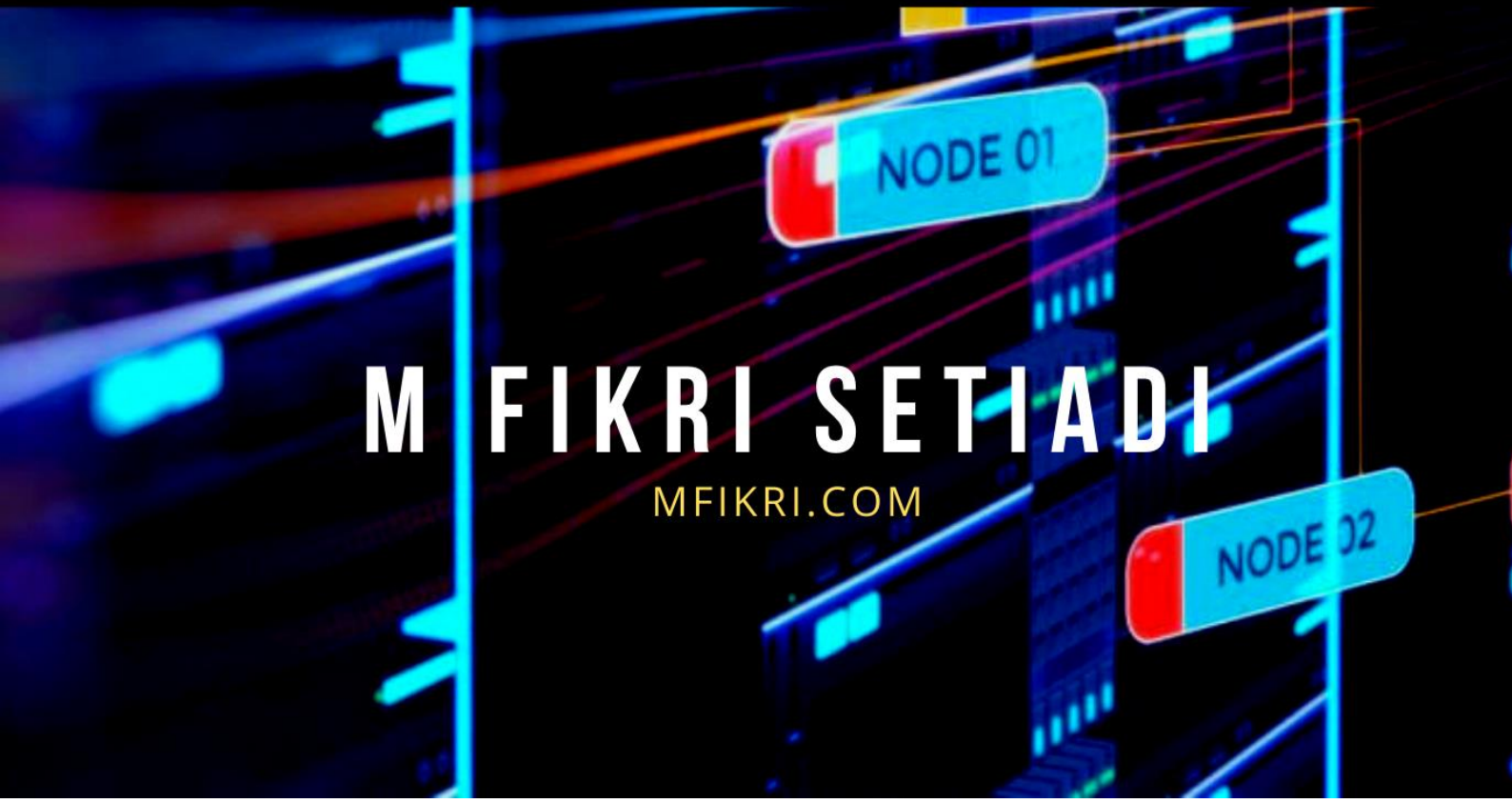




SISTEM DATABASE



M FIKRI SETIADI

MFIKRI.COM

Kata Pengantar

Puji syukur penulis panjatkan kehadiran Tuhan Yang Maha Esa yang telah melimpahkan rahmat, karunia, serta hidayah-Nya sehingga penulis dapat menyelesaikan E-Book yang berjudul “**Sistem Database**” ini.

Sasaran dari E-Book ini adalah agar programmer, engineer, software developer, dan database programmer dapat memahami konsep database serta perancangan database dengan baik dan benar khususnya pemula.

Penulis berharap agar E-Book ini bermanfaat bagi pembacanya. Saran dan kritik sangat penulis harapkan untuk perbaikan E-Book ini.

Penulis,

M Fikri Setiadi

Daftar isi

Kata Pengantar.....	i
Daftar Isi.....	ii
Daftar Tabel.....	iv
Daftar Gambar	v
 Chapter 1. Konsep dasar database	
1.1 Pendahuluan.....	1
1.2 Basis Data (<i>database</i>)	2
1.3 Manfaat Basis Data	3
1.4 Operasi Database	5
1.5 Aturan Database	6
1.6 Integritas Data	8
 Chapter 2. Sistem Database	
2.1 Definisi Database	10
2.2 Abstraksi Data	13
2.3 Bahasa Database	16
 Chapter 3. Model Data	
3.1 Entity Relationship Diagram (ERD).....	19
3.2 Varian Relasi.....	23
3.3 Kardinalitas.....	25
 Chapter 4. Normalisasi Data	
4.1 Definisi	29
4.2 Bentuk Normalisasi	29
4.3 Konsep Ketergantungan (<i>dependency</i>)	34

4.4 Kunci (<i>key</i>)	39
4.5 Anomali	41

Chapter 5. Penerapan Normalisasi

5.1 Teknik Normalisasi.....	43
-----------------------------	----

Daftar Tabel

Tabel 4.1	Bentuk Tidak Normal (<i>Unnormalized</i>)	30
Tabel 4.2	Bentuk Normal Pertama (1 NF)	31
Tabel 4.3	Bentuk Normal Kedua (2 NF)	32
Tabel 4.4	Bentuk Normal Ketiga (3 NF)	33
Tabel 4.5	Functional Dependency	35
Tabel 4.6	Dependency Transitive	36
Tabel 4.7	Dependency Trivial	37

Daftar Gambar

Gambar 3.1	Simbol – Simbol ERD	20
Gambar 3.2	Relasi Binary	23
Gambar 3.3	Relasi Unary.....	24
Gambar 3.4	Relasi N-ary	24
Gambar 3.5	Kardinalitas satu ke satu (one to one)	25
Gambar 3.6	Kardinalitas satu ke banyak (one to many).....	26
Gambar 3.7	Kardinalitas banyak ke banyak (many to many).....	27
Gambar 5.1	Faktur Pembelian Barang.....	44
Gambar 5.2	Bentuk Normalisasi Kedua (2 NF).....	48

CHAPTER 1

KONSEP DASAR DATABASE



1.1 Pendahuluan

Basis data atau *database* dapat dibayangkan seperti sebuah lemari arsip.

Jika kita memiliki lemari arsip dan bertugas mengelolanya, maka akan melakukan hal-hal seperti: memberi sampul, mengelompokkan arsip berdasarkan periode, memberi nomor, lalu menempatkan arsip tersebut dengan urutan tertentu dalam lemari tersebut.

Jika berbicara tentang database, maka seluruh data disimpan dalam sebuah database pada masing-masing tabel sesuai dengan fungsinya, sehingga dengan mudah dapat melakukan penelusuran data yang diinginkan dengan mudah dan cepat.

Masalah yang dihadapi oleh lemari arsip adalah keterlambatan dalam menelusuri data yang diinginkan.

Misalnya kita ingin mencari data karyawan tertentu, maka proses pencarian data karyawan tersebut akan lambat dikarenakan petugas penjaga lemari arsip harus melihat lembar per lembar data dari semua data karyawan.

Sedangkan jika berbicara database, maka semua data disimpan dalam database pada masing-masing tabel sesuai dengan fungsinya, sehingga akan mudah dalam pencarian data yang diinginkan dengan cepat dan akurat.

Itulah kenapa diperlukan sebuah database untuk mengelola data yang jumlahnya banyak dan kompleks.

1.2 Basis Data (*database*)

Basis dapat diartikan sebagai markas atau gudang, tempat bersarang atau berkumpul.

Sedangkan data merupakan representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, pelanggan, mahasiswa, dosen), barang, hewan, kejadian, konsep, keadaan, dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, atau kombinasinya.

Basis data atau *database* merupakan kumpulan data yang saling berelasi (berhubungan).

Relasi biasanya ditunjukan dengan *key* (kunci) dari tiap tabel yang ada. Dalam satu tabel terdapat *record-record* yang sejenis yang merupakan satu kumpulan entitas yang seragam.

Satu *record* terdiri dari *field-field* yang saling berhubungan dan menunjukkan dalam satu pengertian lengkap dalam satu *record*.

Berdasarkan pengertian diatas dapat disimpulkan bahwa basis data (*database*) mempunyai beberapa kriteria yaitu: bersifat data oriented

bukan program oriented, dapat digunakan oleh beberapa program aplikasi tanpa perlu mengubah basis datanya.

Hal ini juga dapat dikembangkan dengan mudah baik volume maupun strukturnya sehingga dapat memenuhi kebutuhan sistem-sistem baru dengan mudah.

Prinsip utama basis data atau database adalah pengaturan data dengan tujuan utama fleksibilitas dan kecepatan akses.

Adapun tujuan basis data diantaranya sebagai efisiensi yang meliputi *speed*, *space*, dan *accuracy*, mengenai data dalam jumlah besar, kebersamaan pemakai (*sharebility*), dan meniadakan duplikasi dan inkonsistensi data.

1.3 Manfaat Basis Data (*database*)

Adapun manfaat dari basis data atau database adalah sebagai berikut:

- 1) Kecepatan dan kemudahan (*Speed*), pemanfaatan basis data memungkinkan untuk dapat menyimpan, merubah, dan menampilkan kembali data tersebut dengan lebih cepat dan mudah.
- 2) Efisiensi ruang penyimpanan (*space*), dengan basis data efisiensi penggunaan ruang penyimpanan dapat dilakukan, karena penekatan

jumlah redundansi data, baik dengan sejumlah pengkodean atau dengan membuat tabel-tabel yang saling berhubungan.

- 3) Keakuratan (*accuracy*), pembentukan relasi antar data bersama dengan penerapan aturan / batasan (*constraint*) tipe, domain dan keunikan data dapat diterapkan dalam sebuah database.
- 4) Ketersediaan (*availability*), dapat memilah data utama / master, transaksi, data histori hingga data kadaluwarsa. Data yang jarang atau tidak digunakan lagi dapat diatur dari sistem basis data aktif.
- 5) Kelengkapan (*completeness*), lengkap / tidaknya data dalam sebuah database bersifat relatif. Bila pemakai sudah menganggap sudah lengkap yang lain belum tentu sama.
- 6) Keamanan (*security*), untuk menentukan siapa saja yang berhak menggunakan database beserta objek-objek di dalamnya dan menentukan jenis-jenis operasi apa saja yang boleh dilakukan.
- 7) Kebersamaan pemakai (*sharebility*), basis data dapat digunakan oleh beberapa pemakai dan beberapa lokasi. Basis data yang dikelola oleh sistem (aplikasi) yang mendukung *multiuser* dapat memenuhi kebutuhan, akan tetapi harus menghindari inkonsistensi data.

1.4 Operasi database

Database atau basis data dapat diciptakan dan dapat pula dimusnahkan. Pada sebuah penyimpanan (*disk*) dapat menempatkan beberapa basis data (*database*), misalnya basis data akademik, penjualan, perpustakaan, dan lain-lain.

Sementara dalam sebuah basis data terdapat satu tabel atau lebih. Misalnya dalam database penjualan terdiri dari tabel barang, pelanggan, kategori, dan transaksi. Operasi dasar yang dapat dilakukan database adalah:

- 1) Pembuatan database baru (*CREATE DATABASE*)
- 2) Pemusnahan database (*DROP DATABASE*)
- 3) Pembuatan tabel baru dalam database (*CREATE TABLE*)
- 4) Penghapusan tabel dari suatu database (*DROP TABLE*)
- 5) Pengisian data (*record*) kedalam tabel (*INSERT*)
- 6) Menampilkan data yang ada pada tabel (*SELECT*)
- 7) Mengubah data yang ada pada tabel (*UPDATE*)
- 8) Penghapusan data yang ada pada table (*DELETE*)

Operasi pembuatan database dan tabel merupakan operasi awal yang hanya dilakukan sekali dan berlaku seterusnya.

Sedangkan untuk operasi pengisian data merupakan operasi rutin yang dilakukan berulang-ulang.

1.5 Aturan database

Ketentuan yang harus diperhatikan pada pembuatan tabel pada database agar memenuhi kriteria sebagai database, yaitu: redundansi data, inkonsistensi data, pengaksesan data, data terisolasi untuk standarisasi, masalah keamanan, masalah integritas data, dan *multiuser*.

1.5.1 Redudansi dan Inkonsistensi Data

Penyimpanan data di beberapa tempat yang berbeda disebut redundansi, hal ini akan menyebabkan pemborosan dan menimbulkan inkonsistensi data (data tidak konsisten) apabila terjadi perubahan data di beberapa tempat.

Tujuan sistem database bukan untuk menghilangkan redundansi data, melainkan meminimalisir redundansi data, karena suatu tabel tidak dapat berelasi (berhubungan) dengan tabel lain jika tidak ada redundansi sama sekali.

1.5.2 Pengaksesan Data

Data di dalam database harus siap diakses oleh siapa saja yang membutuhkan dan mempunyai hak untuk mengaksesnya.

Oleh karena itu perlu dibuat suatu program pengelolaan atau suatu aplikasi untuk mengakses data yang dikenal sebagai *Database Management System* (DBMS).

1.5.3 Data Terisolasi untuk Standarisasi

Jika data tersebar dalam beberapa tabel dalam bentuk format yang sama, maka akan menyulitkan dalam menulis program aplikasi, baik untuk mengambil dan menyimpan data.

Oleh karena itu data dalam satu *database* harus dibuat satu format yang sama, sehingga mudah dibuat program aplikasinya.

1.5.4 Masalah Keamanan (*Security*)

Setiap pemakai sistem database tidak semua bagian mengakses semua data, misalnya data mengenai gaji pegawai hanya boleh dibuka oleh bagian keuangan, sedangkan bagian gudang dan bagian lain tidak diperkenankan untuk membukanya.

Keamanan dapat diatur dan disesuaikan baik ditingkat database atau aplikasinya.

1.5.5 Multiple User

Salah satu database dibangun karena nantinya data tersebut akan digunakan oleh banyak pengguna (*user*), baik dalam waktu berbeda maupun bersamaan.

Oleh karena itu diperlukan database yang handal dan dapat mendukung banyak pemakai atau *multiuser*.

1.6 Integritas Data

1.6.1 Domain Integrity

Domain adalah suatu himpunan yang dapat diberikan pada suatu atribut, yang mencakup nama, arti, tipe data, ukuran, dan nilai yang diijinkan.

Semua nilai yang disimpan pada kolom-kolom harus memiliki domain yang sama.

Integritas data ini merujuk pada batasan nilai yang diperbolehkan untuk setiap atribut dalam tabel yang dimiliki oleh suatu atribut.

1.6.2 Integrity Entity

Integrity entity dirancang untuk memastikan setiap relasi atau tabel yang sudah memiliki kunci primer (*primary key*), dan memastikan bahwa nilai-nilai data untuk kunci primer adalah sah atau valid, dan menjamin bahwa setiap atribut kunci primer tidak boleh kosong atau NOT NULL.

Nilai NULL adalah nilai yang diberikan ke suatu atribut saat tidak ada nilai yang dimasukan atau nilai yang diterapkan tidak diketahui.

Nilai NULL adalah bukan nilai melainkan ketidakhadiran nilai pada suatu atribut tertentu. Nilai NULL disini bukan angka 0 dan juga bukan *space* (untuk jenis data *alpha*, *alphanumeric*, dan *string*), melainkan mewakili nilai data kosong.

Sebagai contoh setiap kolom dalam pendefinisian tabel dapat dilengkapi dengan NULL (boleh kosong) dan NOT NULL (tidak boleh kosong).

```
CREATE TABLE tbl_mahasiswa(  
    Nim varchar(15) NOT NULL,  
    Nama varchar(30) NOT NULL,  
    Prodi varchar(20) NOT NULL,  
    Gender varchar(3) NOT NULL,  
    Alamat varchar(40)  
);
```

1.6.3 Integrity Referensial

Integrity referensial adalah garis yang menghubungkan antara kunci tamu (*foreign key*) di suatu tabel dengan kunci primer (*primary key*).

Integrity referensial merupakan aturan yang memelihara konsistensi antara baris-baris pada 2 relasi.

Aturan ini menerapkan bahwa jika ada kunci tamu disatu relasi tertentu, maka kunci tamu itu harus sesuai dengan nilai kunci primer di relasi lain.

CHAPTER 2

SISTEM DATABASE



2.1 Definisi Database

Sistem basis data atau *database* merupakan sistem yang terdiri dari kumpulan tabel yang saling berhubungan (*berelasi*) dan memungkinkan beberapa pengguna mengakses dan memanipulasinya.

Sistem database juga merupakan suatu sistem yang menyusun dan mengelola data organisasi perusahaan, sehingga mampu menyediakan informasi yang diperlukan oleh pengguna.

2.2 Komponen sistem database

Sistem database terdapat komponen-komponen utama yaitu perangkat keras (*hardware*), sistem operasi (*operating system*), basis data (*database*), program aplikasi (*application program*), DBMS (*Database Management System*), dan pengguna (*user*).

1) Perangkat Keras (*Hardware*)

Perangkat keras (*hardware*) yang biasanya terdapat dalam sebuah sistem database adalah komputer untuk sistem *stand alone*, sistem jaringan (*network*), memori sekunder yang online (*harddisk*), memori sekunder yang offline (*disk*), dan perangkat komunikasi untuk jaringan.

2) Sistem Operasi (*Operating System*)

Sistem operasi merupakan program yang mengaktifkan sistem komputer, mengendalikan seluruh sumber daya dalam komputer dan

melakukan operasi-operasi dasar dalam komputer, pengelolaan file, dan lain-lain.

Program pengelola basis data akan aktif (*running*) jika sistem operasi yang dikehendaki sesuai. Sistem operasi yang biasa digunakan Microsoft DOS, Microsoft Windows, UNIX, LINUX, Mac OS, dan lain-lain.

3) Basis Data (*Database*)

Basis data (*database*) merupakan koleksi dari data yang terorganisasi dengan cara sedemikian rupa sehingga data tersebut mudah disimpan dan dimanipulasi.

Sebuah sistem database dapat memiliki beberapa database. Setiap database dapat memiliki sejumlah objek database seperti table, indeks, dan lain-lain.

4) DBMS (*Database Management System*)

Database Management System (DBMS) merupakan kumpulan program aplikasi yang digunakan untuk membuat dan mengelola database.

DBMS berisi suatu koleksi data dan set program untuk mengakses data. DBMS merupakan perangkat lunak (*software*) yang menentukan bagaimana data tersebut diorganisasi, disimpan, diubah dan diambil kembali.

Perangkat lunak ini juga menerapkan mekanisme pengamanan data, pengguna data bersama, dan konsistensi data.

Perangkat lunak yang termasuk kedalam DBMS diantaranya: Microsoft Access, SQL Server, MySQL atau MySQLi, DB2, Oracle, PostgreSQL, SQLite, Mongo DB, dan Maria DB.

5) Pengguna (*User*)

Terdapat beberapa tipe users atau pengguna pada sistem database, berdasarkan cara mereka berinteraksi pada basis data, diantaranya program aplikasi, pengguna mahir, pengguna umum, dan pengguna khusus.

❖ Programmer Aplikasi

Programmer aplikasi adalah pengguna yang berinteraksi dengan database melalui DML (*Data Manipulation Language*), yang disertakan dalam program yang ditulis dalam bahasa pemrograman induk (seperti pascal, cobol, clipper, dan lain-lain).

❖ User Mahir

User mahir adalah pengguna yang berinteraksi dengan sistem tanpa menulis modul program. Pengguna menggunakan *query* (bahasa standard yang digunakan oleh setiap DBMS) untuk mengakses data.

❖ User Umum

User umum adalah pengguna yang berinteraksi dengan sistem database melalui pemanggilan satu program aplikasi permanen yang ditulis atau disediakan sebelumnya.

❖ User Khusus

User khusus adalah pengguna yang menulis aplikasi database non konvensional untuk keperluan khusus, seperti untuk aplikasi sistem pakar, dan lain-lain.

6) Administrator database

Suatu lingkup database seharusnya memiliki satu orang atau kelompok orang pada bagian struktur database untuk menangani administrasi database yang biasa disebut administrator database atau *Database Administrator* (DBA).

Administrator database adalah orang yang bertanggung jawab dan bekerja sama dengan analisis sistem dan user-user lain guna melengkapi berbagai macam tugas seperti pendefinisian data, pemodelan data, desain database, serta menjamin kerahasiaan integritas data.

2.3 Abstraksi Data

Pada abstraksi data bayangan mengenai data tidak lagi diperhatikan kondisi sesungguhnya bagaimana suatu data masuk ke database dan

disimpan ke dalam sektor mana, tetapi menyangkut secara menyeluruh bagaimana data tersebut dapat diabstraksikan.

Kegunaan utama sistem database adalah agar pengguna mampu menyusun suatu pandangan abstraksi dari data tersebut.

Database dapat dipandang dari 2 yaitu: sudut pandang pengguna dan perancang database.

Pengguna dapat diartikan sebagai orang-orang yang menggunakan database.

Sedangkan perancang database adalah orang yang berperan sebagai perancang dan pengelola database.

Seorang perancang database memiliki pandangan secara konseptual dan secara fisik.

Pandangan tersebut menunjukkan level pandangan terhadap suatu database.

Pandangan terhadap database sering disebut sebagai arsitektur database atau abstraksi database dan terbagi kedalam level fisik, level konseptual, dan level pandangan pengguna.

1) Level Fisik (*Physical Level*)

Level fisik merupakan level abstraksi yang paling rendah menjelaskan secara detail bagaimana data disimpan dan kondisi sebenarnya atau diorganisasikan secara fisik atau aktual.

Pada level ini struktur data yang diperlukan gambaran secara rinci yang dibutuhkan oleh *system enginner*, dan level ini umumnya digunakan oleh para pakar software dan hardware.

Physical level sering disebut sebagai level eksternal merupakan bentuk implementasi konseptual, yaitu suatu pandangan perancang yang berkaitan dengan permasalahan teknik penyimpanan data dalam database ke dalam media penyimpanan yang digunakan.

Pandangan ini bersifat sangat teknis dan lebih berorientasi pada mesin, yaitu berkaitan dengan organisasi berkas database.

2) Level Konseptual (*Conceptual Level*)

Level konseptual merupakan level abstraksi yang lebih rendah dari level logika dan merupakan level abstraksi yang lebih tinggi dari level fisik.

Level ini memberikan gambaran tentang data apa yang sebenarnya perlu disimpan dalam database, serta hubungan atau relasi yang terjadi diantara data dari keseluruhan database.

Level Conceptual / Global Logical Data juga merupakan suatu pandangan perancang database yang berkaitan dengan data apa saja yang perlu disimpan dalam database dan penjelasan mengenai hubungan antara data yang satu dengan yang lainnya.

Pengguna tidak mempedulikan kerumitan dalam struktur level fisik lagi, penggambaran cukup dengan memakai kotak, garis, dan hubungan secukupnya.

3) Level Pandangan Pengguna

Level pandangan pengguna atau level eksternal merupakan pandangan para pengguna database pada masing-masing pengguna database, sehingga memiliki cara pandang yang berbeda-beda tergantung pada macam data apa saja yang tersedia atau dapat diakses oleh pemakai.

Level ini merupakan level abstraksi yang mendeskripsikan bagian tertentu dari keseluruhan database secara logika kepada pengguna tentang data yang dibutuhkan.

Level ini merupakan level abstraksi data tertinggi yang menggambarkan sebagian saja tentang data yang dapat dilihat dan dipakai dari keseluruhan database.

Pengguna tidak perlu tahu bagaimana sebenarnya data-data tersebut disimpan.

2.4 Bahasa Database (*database language*)

Bahasa database adalah suatu cara untuk berinteraksi atau berkomunikasi antara pengguna dengan database yang diatur dalam bahasa khusus yang ditetapkan oleh perusahaan.

Database language dibagi menjadi 3 bagian yaitu *Data Definition Language* (DDL), *Data Manipulation Language* (DML), *Data Control Language* (DCL).

1) Data Definition Language (DDL)

Data Definition Language (DDL) merupakan satu paket bahasa DBMS yang berguna untuk melakukan spesifikasi terhadap skema database.

DDL juga merupakan struktur database yang menggambarkan desain database secara keseluruhan.

Dengan DDL dapat membuat tabel baru dan mengubah tabel, dan hasil dari kompilasi perintah DDL adalah kumpulan tabel yang disimpan dalam file khusus yang disebut kamus data (*data dictionary*).

Kamus data merupakan suatu meta data (super data) yaitu data yang mendeskripsikan data sesungguhnya.

Kamus data akan selalu diakses dalam suatu operasi database sebelum suatu file data yang sesungguhnya diakses.

2) *Data Manipulation Language* (DML)

Data Manipulation Language (DML) merupakan satu paket DBMS yang memperbolehkan pengguna untuk mengakses atau memanipulasi data sebagaimana yang telah diorganisasikan sebelumnya dalam model data yang tepat.

DML dapat mengambil informasi yang tersimpan dalam database, menyisipkan informasi baru dalam database dan menghapus informasi dari tabel.

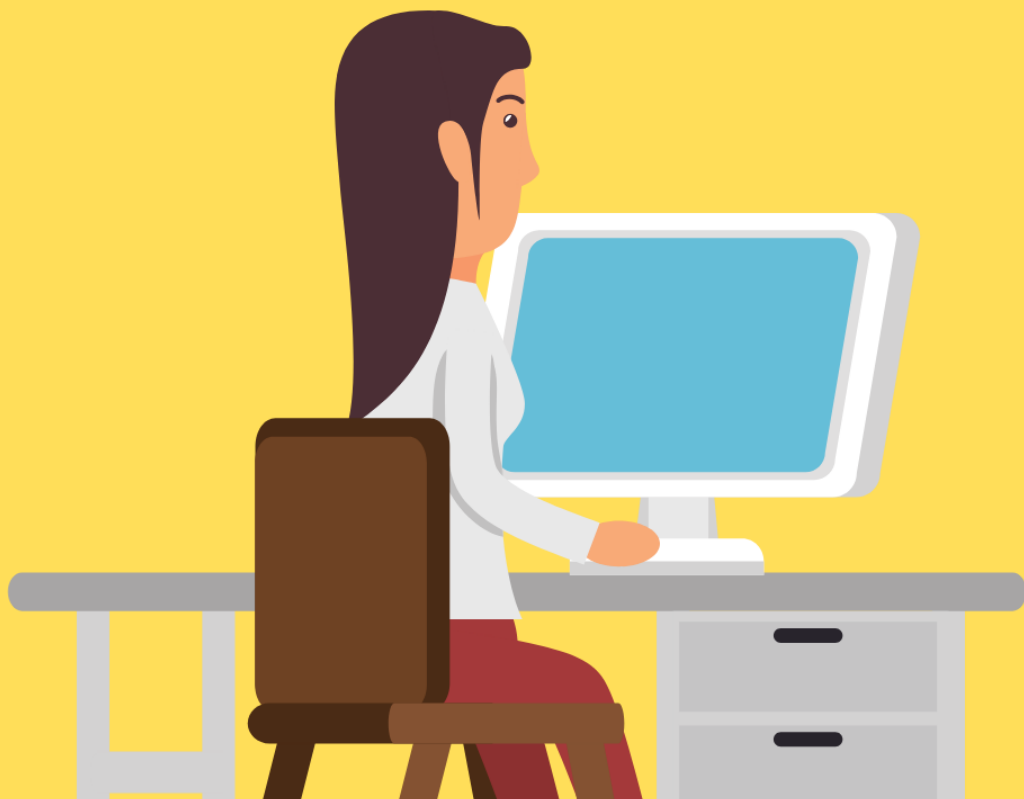
DML mempunyai 2 tipe yaitu *procedural* dan *non procedural*. *Procedural* digunakan oleh pengguna pada saat membutuhkan spesifikasi data apa yang dibutuhkan dan bagaimana cara mendapatkannya, sedangkan *non procedural* digunakan oleh pengguna pada saat membutuhkan spesifikasi data tanpa harus mengetahui bagaimana cara mendapatkannya.

3) *Data Control Language* (DCL)

Data Control Language (DCL) adalah kumpulan bahasa SQL (*structured query language*) atau bahasa database yang berfungsi untuk mengatur hak akses pengguna database.

CHAPTER 3

MODEL DATA



3.1 *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) untuk mendokumentasikan data perusahaan dengan mengidentifikasi jenis entitas (*entity*) dan hubungannya.


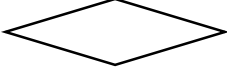


ERD merupakan suatu model jaringan yang menggunakan susunan data yang disimpan pada sistem secara abstrak.

ERD juga menggambarkan hubungan antara satu entitas yang memiliki sejumlah atribut dengan entitas yang lain dalam suatu sistem yang terintegrasi.

ERD digunakan oleh perancang sistem untuk memodelkan data yang nantinya akan dikembangkan menjadi database.

Model data ini juga akan membantu pada saat melakukan analisis dan perancangan database, karena model data ini akan menunjukkan bermacam-macam data yang dibutuhkan dan hubungan antar data.

ERD ini juga merupakan model konseptual yang dapat mendeskripsikan hubungan antara file yang digunakan untuk memodelkan struktur data serta hubungan antar data.

Simbol	Keterangan
	Entitas, yaitu kumpulan dari objek yang dapat diidentifikasi secara unik
	Relasi, yaitu hubungan yang terjadi antara salah satu lebih entitas. Jenis hubungan antara lain. one to one, One to many, dan many to many.
	Atribut, yaitu karakteristik dari entitas atau relasi yang merupakan penjelasan detail tentang entitas.
	Hubungan antara entitas dengan atributnya dan himpunan entitas dengan himpunan relasinya.

Gambar 3.1 Simbol – Simbol ERD

Selain dari simbol yang ada di tabel 3.1, sebenarnya ada simbol lain lagi yaitu entitas asosiatif, atribut derivat, dan lainnya.

Akan tetapi secara garis besar ERD terdiri dari atas tiga komponen, yaitu entitas (*entity*), atribut (*attribute*), dan relasi atau hubungan (*relation*). Entitas merupakan dasar yang terlibat dalam sistem.

Atribut atau field berperan sebagai penjelas dari entitas, dan relasi atau hubungan yang terjadi antara dua entitas.

1) Entitas (*entity*)

Entitas menunjukkan objek-objek dasar yang terkait di dalam sistem.

Objek dasar dapat berupa orang, benda, atau hal lain yang keterangannya perlu disimpan dalam database.

Untuk menggambarkan entitas dilakukan dengan cara mengikuti aturan berikut:

- ❖ Entitas dinyatakan dengan simbol persegi panjang.
- ❖ Nama entitas berupa kata benda tunggal
- ❖ Nama entitas sebisa mungkin menggunakan nama yang mudah dipahami dan menyatakan maknanya dengan jelas.

2) Atribut (*attribute*)

Atribut sering juga disebut sebagai properti, merupakan keterangan-keterangan yang terkait pada sebuah entitas yang perlu disimpan sebagai database.

Atribut berfungsi sebagai penjelas sebuah entitas untuk menggambarkan atribut yang dilakukan dengan mengikuti aturan sebagai berikut:

- ❖ Atribut dinyatakan dengan simbol ellipps.
- ❖ Nama atribut ditulis dalam simbol ellipps.
- ❖ Nama atribut berupa kata benda tunggal.
- ❖ Nama atribut sebisa mungkin menggunakan nama yang mudah dipahami dan padat menyatakan maknanya dengan jelas.
- ❖ Atribut dihubungkan dengan entitas yang sesuai dengan menggunakan garis.

3) Relasi (*relation*)

Relasi atau hubungan adalah kejadian atau transaksi yang terjadi di antara dua entitas yang keterangannya perlu disimpan dalam database.

Aturan penggambaran relasi antar entitas adalah:

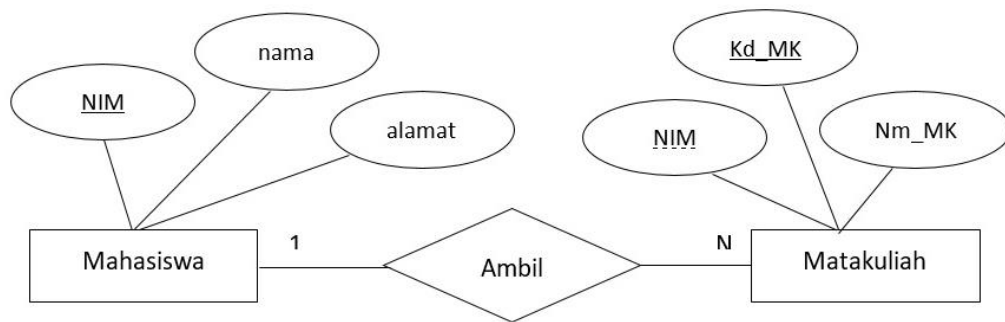
- ❖ Relasi dinyatakan dengan simbol belah ketupat.
- ❖ Nama relasi dituliskan di dalam simbol belah ketupat.
- ❖ Relasi menghubungkan dua entitas.
- ❖ Nama relasi menggunakan kata kerja aktif (diawali awalan me) tunggal.
- ❖ Nama relasi sebisa mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.

3.2 Varian Relasi

1) Relasi Binery

Relasi binary merupakan relasi yang terjadi antara 2 himpunan entitas yang berbeda.

Relasi ini merupakan relasi yang umum digunakan. Relasi antara mahasiswa mengambil matakuliah yang menunjukkan *binery relation*.



Gambar 3.2 Relasi Binary

2) Relasi Unary

Relasi Unary merupakan variasi relasi yang terjadi dari sebuah himpunan entitas ke himpunan entitas yang sama, dan unary sering disebut dengan relasi tunggal.

Relasi antara dosen dan mendampingi yang menunjukkan *unary relation*.



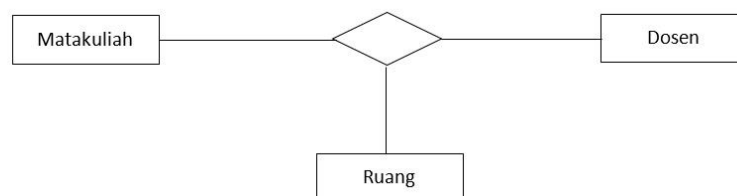
Gambar 3.3 Relasi Unary

3) Relasi N-ary

Relasi N-ary merupakan relasi dari 3 entitas atau lebih. Relasi ini untuk menghubungkan dari tiga entitas yang dimasukan ke relasi multi entitas.

N-ary relation menunjukkan secara lebih jelas bahwa bahwa beberapa entitas berpartisipasi dalam sebuah relasi tunggal.

Bentuk relasi semacam ini sebisa mungkin dihindari karena akan mengaburkan derajat relasi yang ada dan akan menyebabkan perencanaan database semakin kompleks.



Gambar 3.4 Relasi N-ary

3.3 Kardinalitas (Derajat Relasi)

Model relasi ini berdasarkan persepsi dunia nyata diantaranya himpunan objek dasar dan relasi antara entitas.

Entitas dapat diartikan sebagai objek dan diidentifikasi secara unik, dan objeknya dapat berbentuk orang, barang, dan sebagainya.

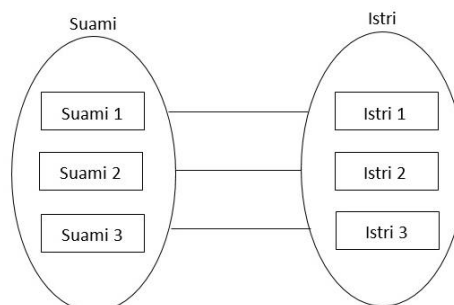
Kardinalitas relasi menunjukkan maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas lain.

kardinalitas relasi yang terjadi di antara dua himpunan entitas dapat berupa satu ke satu (*one to one*), satu ke banyak (*one to many*), dan banyak ke banyak (*many to many*).

1) Satu ke satu (*one to one*)

Hubungan satu ke satu (*one to one*) berarti setiap himpunan entitas hanya boleh berhubungan dengan satu himpunan entitas lainnya.

Sebagai contoh himpunan suami dan istri berikut:



Gambar 3.5 Kardinalitas satu ke satu (*one to one*)

Pada gambar 3.5 dapat dilihat bahwa satu himpunan entitas suami hanya berhubungan tepat dengan satu himpunan entitas istri.

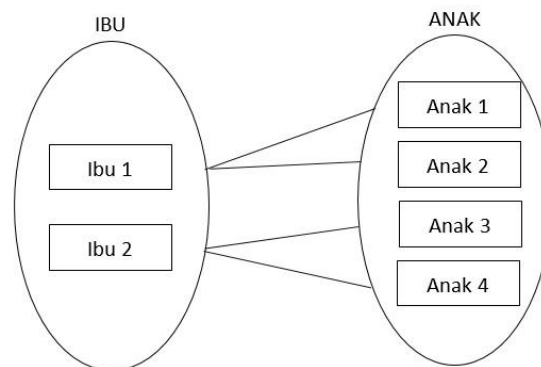
Dalam arti kata suami atau istri tidak boleh selingkuh.

Karena penulis orang baik jadi penulis tidak mengajarkan perselingkuhan disini.

2) Satu ke Banyak (*one to many*)

Hubungan satu ke banyak (*one to many*) berarti satu dari setiap himpunan entitas boleh berhubungan dengan banyak himpunan entitas lainnya.

Sebagai contoh himpunan ibu dan anak berikut:



Gambar 3.6 Kardinalitas satu ke banyak (*one to many*)

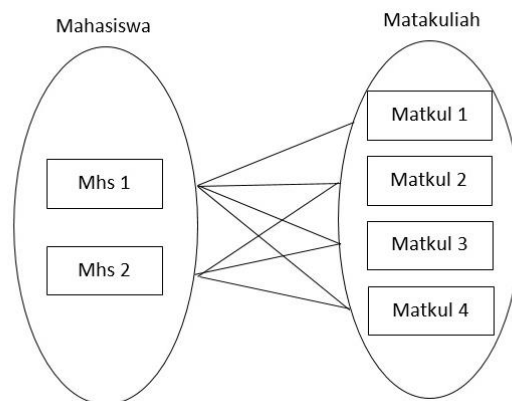
Pada gambar 3.6 diatas dapat dilihat bahwa satu himpunan ibu memiliki banyak hubungan ke himpunan entitas anak.

Dalam arti kata satu ibu bisa memiliki banyak anak dan satu anak hanya dimiliki oleh satu ibu.

3) Banyak ke Banyak (*Many to Many*)

Hubungan banyak ke banyak (*many to many*) berarti setiap himpunan entitas boleh berhubungan dengan banyak himpunan entitas lainnya dan sebaliknya.

Sebagai contoh himpunan matakuliah dan mahasiswa berikut:



Gambar 3.7 Kardinalitas banyak ke banyak (*many to many*)

Pada gambar 3.7 diatas dapat dilihat bahwa satu himpunan mahasiswa memiliki banyak hubungan ke himpunan entitas matakuliah dan satu dari himpunan matakuliah memiliki banyak hubungan ke himpunan entitas mahasiswa.

Dalam arti kata satu mahasiswa bisa memiliki banyak matakuliah dan satu matakuliah bisa dimiliki oleh banyak mahasiswa.

CHAPTER 4

NORMALISASI DATA



4.1 Definisi

Normalisasi (*normalize*) merupakan salah satu cara pendekatan atau teknik yang digunakan dalam membangun desain logik *database relation* dengan menerapkan sejumlah aturan dan kriteria standard.

Tujuan dari normalisasi adalah untuk menghasilkan struktur tabel yang normal atau baik.

Teknik normalisasi adalah upaya agar desain logik tabel-tabel berada dalam bentuk normal (*normal form*) yang dapat didefinisikan dengan menggunakan ketergantungan fungsi (*functional dependency*).

4.2 Bentuk Normalisasi

Aturan-aturan normalisasi dinyatakan dalam istilah bentuk normal.

Bentuk normal adalah suatu aturan yang dikenakan pada relasi-relasi atau tabel-tabel dalam database dan harus dipenuhi oleh relasi atau tabel tersebut pada level-level normalisasi.

Suatu relasi dikatakan dalam bentuk normal tertentu jika memenuhi kondisi tertentu juga. Beberapa bentuk normalisasi diantaranya adalah bentuk tidak normal (*unnormalize*), bentuk normal pertama (1NF), bentuk normal kedua (2NF), normal ketiga (3NF), dan seterusnya.

1) Bentuk Tidak Normal (*unnormalize*)

Bentuk tidak normal (*unnormalized*) merupakan kumpulan data yang direkam tidak ada keharusan dengan mengikuti suatu format tertentu. Pada bentuk tidak normalisasi terdapat *repeating group*, sehingga pada kondisi ini data menjadi permasalahan dalam melakukan manipulasi data (*insert, update, dan delete*) atau biasa disebut anomali.

Tabel 4.1 Bentuk Tidak Normal (*Unnormalized*)

No. Faktur	Tgl	Kode Pelanggan	Nama	Kode Barang	Nama Barang	Harga	Qty
F-001	12/12/2016	P-001	M Fikri Setiadi	B-001	Sampo	12.000,-	1
				B-002	Kopi	15.000,-	1
F-002	13/12/2016	P-002	Jack	B-002	Kopi	15.000,-	1
				B-003	Teh	7.000,-	2

2) Normal Pertama (1 NF)

Dalam relational database tidak diperkenankan adanya repeating group karena dapat berdampak terjadinya anomali.

Oleh karena itu tahap unnormal akan menghasilkan bentuk normal tahap pertama (1 NF) yang dapat di definisikan sebagai berikut:

Normal pertama (1 NF), suatu relasi atau tabel memenuhi normal pertama jika dan hanya jika setiap setiap atribut dari relasi tersebut hanya memiliki nilai tunggal dalam satu baris (*record*).

Tiap *field* hanya satu pengertian, bukan merupakan kumpulan kata yang mempunyai arti ganda dan tidak ada set atribut yang berulang-ulang atau atribut bernilai ganda.

Pada data tabel sebelumnya data belum normal sehingga harus diubah kedalam bentuk normal pertama dengan cara membuat baris berisi kolom jumlah yang sama dan setiap kolom hanya mengandung satu nilai.

Tabel 4.2 Bentuk Normal Pertama (1 NF)

No. Faktur	Tgl	Kode Pelanggan	Nama	Kode Barang	Nama Barang	Harga	Qty
F-001	12/12/2016	P-001	M Fikri Setiadi	B-001	Sampo	12.000,-	1
F-001	12/12/2016	P-001	M Fikri Setiadi	B-002	Kopi	15.000,-	1
F-002	13/12/2016	P-002	Jack	B-002	Kopi	15.000,-	1
F-002	13/12/2016	P-002	Jack	B-003	Teh	7.000,-	2

Bentuk normalisasi pertama (1 NF) ini mempunyai ciri yaitu setiap data dibentuk file daftar atau rata (*flat file*), data dibentuk dalam satu record demi satu record dan nilai-nilai dari field-field berupa nilai yang tidak dapat dibagi-bagi lagi.

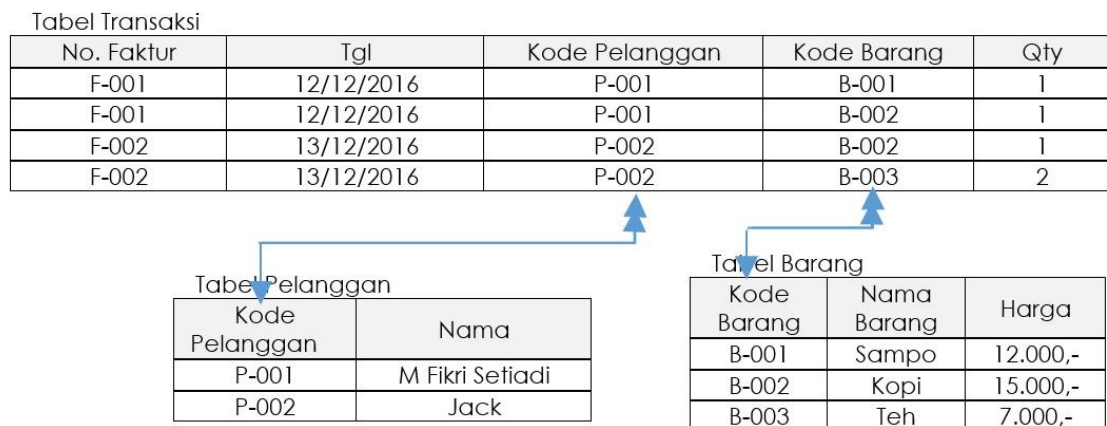
3) Normal Kedua (2 NF)

Dalam perancangan database relational tidak diperkenankan adalah partial functional dependency kepada primary key, karena dapat berdampak terjadinya anomali.

Oleh karena itu tahap normalisasi pertama akan menghasilkan bentuk normal kedua (1 NF) yang dapat didefinisikan sebagai berikut:

Normalisasi kedua (2 NF), suatu relasi memenuhi relasi kedua jika dan hanya jika relasi tersebut memenuhi normal pertama dan setiap atribut yang bukan kunci (*non key*) bergantung secara fungsional terhadap kunci utama (*Primary key*).

Tabel 4.3 Bentuk Normal Kedua (2 NF)



Bentuk normal kedua ini mempunyai syarat yaitu bentuk data yang telah memenuhi kriteria bentuk normal pertama.

Atribut bukan kunci haruslah bergantung secara fungsional pada kunci utama (*primary key*), sehingga untuk membentuk normal kedua haruslah sudah ditentukan kunci-kunci field.

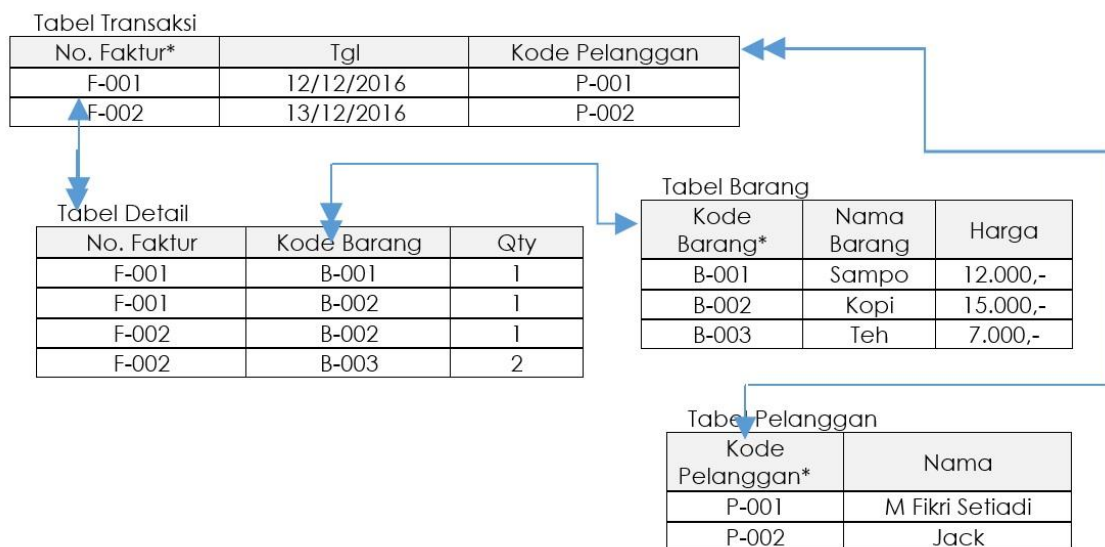
4) Normal Ketiga (3 NF)

Dalam perancangan database relational tidak diperkenankan adanya transitive dependency karena dapat berdampak terjadinya anomali.

Oleh karena itu harus dilakukan normalisasi tahap ketiga (1 NF) yang dapat didefinisikan sebagai berikut:

Normalisasi ketiga (1 NF), suatu relasi memenuhi normal ketiga jika dan hanya jika relasi tersebut memenuhi normal kedua dan setiap atribut bukan kunci (*non key*) tidak mempunyai transitive functional dependency kepada kunci utama (*primary key*).

Tabel 4.4 Bentuk Normal Ketiga (3 NF)



Bentuk normal ketiga (3 NF) ini relasi haruslah dalam bentuk normal kedua dan semua atribut bukan kunci utama tidak punya hubungan transitif.

Artinya setiap atribut bukan kunci harus bergantung hanya pada primary key secara keseluruhan, dan bentuk normalisasi ketiga sudah didapat tabel yang optimal.

4.3 Konsep ketergantungan (*dependency*)

Ketergantungan (*dependency*) merupakan konsep yang mendasari normalisasi.

Dependency menjalankan hubungan antara atribut atau secara lebih khusus menjelaskan nilai suatu atribut yang menentukan nilai atribut lainnya.

Dependency akan mencari acuan untuk pendekomposisian data ke dalam bentuk paling efisien.

1) Functional Dependency

Ketergantungan fungsional (*functional dependency*) adalah suatu kondisi dimana satu atribut atau beberapa atribut dari suatu relasi yang keberadaannya bergantung pada atribut lain.

ketergantungan fungsional didefinisikan sebagai berikut: suatu atribut Y mempunyai ketergantungan fungsional terhadap atribut X jika dan hanya jika setiap nilai X berhubungan dengan sebuah nilai Y.

Definisi diatas biasanya dituangkan dalam bentuk notasi: $X \rightarrow Y$

Artinya : X secara fungsional tergantung Y

Y secara fungsional menentukan X

Tabel 4.5 *Functional Dependency*

Kode_Barang	Nama_Barang	Qty
B-001	Sampo	2
B-002	Kopi	2
B-003	Teh	3

Sebagai keterangan mengenai notasi diatas bahwa sebelah kiri notasi disebut *determinant*, sedangkan bagian sebelah kanan disebut dependent. Relasi ini mengandung atribut kode_barang, nama_barang, harga, dan qty.

Artinya kode_barang secara fungsional menentukan nama_barang, telah terlihat bahwa untuk kode_barang yang sama dan nama_barang juga sama, sehingga $\text{kode_barang} \rightarrow \text{nama_barang}$.

2) Dependency Transitive

Ketergantungan transitive (*dependency transitive*) adalah ketergantungan secara fungsional suatu atribut kepada atribut lainnya melalui atribut yang lain pula.

Misalnya R adalah relasi yang memiliki 3 atribut A, B, dan C yang memiliki ketergantungan fungsional:


$A \rightarrow B$

$B \rightarrow C$

Artinya fungsional dependency $\rightarrow C$ disebut dependency transitive.

Tabel 4.6 *Dependency transitive*

No_Faktur	Kode_Pelanggan	Nama	Kode Barang	Qty
F-001	P-001	Joko	B-001	1
F-002	P-002	Ani	B-002	1
F-003	P-003	Jack	B-002	1
F-004	P-004	Fikri	B-001	2



Pada relasi di atas terdapat ketergantungan transitif antara no_faktur \rightarrow kode_pelanggan dan kode_pelanggan \rightarrow nama, sehingga dapat dinotasikan no_faktur \rightarrow kode_pelanggan \rightarrow nama.

3) Trivial Dependency

Ketergantungan trivial (trivial dependency) terjadi apabila suatu relasi terdapat banyak set fungsional dependency, maka dapat dilakukan penghapusan trivial dependency, misalnya relasi sebagai berikut:

Tabel 4.7 *Dependency trivial*

No_Faktur	Nama	Kode Barang	Qty
F-001	Joko	B-001	1
F-002	Ani	B-002	1
F-003	Jack	B-002	1
F-004	Fikri	B-001	2

Suatu trivial fungsional dependency jika dan hanya jika bagian sebelah kiri, maka himpunan tersebut ditulis sebagai berikut: {no_faktur, kode_barang} \rightarrow no_faktur.

Sebaliknya, non trivial fungsional dependency adalah fungsional dependency bukan trivial, dan yang sebetulnya adalah bentuk integrity rules / constrains (aturan-aturan integritas).

4.4 Kunci (*key*)

Kunci (*key*) adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris (*record*) dalam tabel secara unik, artinya jika suatu atribut dijadikan sebagai *key*, maka tidak boleh ada dua atau lebih record data dengan nilai yang sama untuk atribut tersebut.

Key yang diterapkan pada suatu tabel dan dapat dibedakan menjadi beberapa macam kunci sebagai berikut:

1) Super Key

Super key adalah satu atribut atau kumpulan atribut yang secara unik untuk mengidentifikasi sebuah record di dalam relasi atau himpunan dari satu atau lebih entitas.

Super key dapat digunakan untuk mengidentifikasi secara unik sebuah entitas.

2) Candidat Key

Calon kunci (*candidat key*) suatu set atribut dari sebuah entitas yang memiliki nilai unik.

Biasanya salah satu dari atribut di dalam suatu relasi mempunyai nilai yang unik. Salah satu dari *candidat key* akan menjadi *primary key*.

Candidat key juga merupakan salah satu atribut yang mengidentifikasi secara unik suatu kejadian spesifik dari entitas.

Misalnya pada tabel mahasiswa yang merupakan kandidat key adalah NoBP, No_HP, dan No_KTP.

3) Primary Key

Kunci utama (primary key) adalah kandidat key dalam suatu relasi yang dipilih salah satu menjadi primary key atau nilai atribut yang unik dan dipakai untuk membedakan satu kolom dengan yang lainnya.

Primary key juga merupakan atribut yang tidak mengidentifikasi secara unik suatu kejadian spesifik, tapi juga dapat mewakili setiap kejadian dari suatu entitas.

Misalnya dari tabel mahasiswa yang merupakan primary key adalah NoBP.

4) Foreign Key

Kunci tamu (*foreign key*) adalah satu atau sejumlah atribut yang melengkapi suatu *relationship* (hubungan yang menunjukkan ke induknya).

Foreign key ditempatkan pada entitas anak dan sama dengan primary key induk direlasikan.

Hubungan antara entitas induk dengan anak adalah hubungan satu lawan ke banyak (*one to many*).

4.5 Anomali

Anomali adalah suatu proses pada database yang tidak memberikan efek samping (misal menyebabkan inkonsistensi data atau membuat sesuatu data menjadi hilang ketika data lain dihapus).

Anomali database dapat dibedakan menjadi 3 yaitu anomali pengubahan (update), anomali penghapusan (delete), dan anomali penyisipan (insert).

1) Anomali pengubahan (update)

Anomali pengubahan terjadi apabila ada perubahan pada sejumlah data yang mubazir pada suatu tabel tetapi tidak seluruhnya diubah.

2) Anomali penyisipan (Insert)

Anomali penyisipan terjadi apabila pada saat penambahan data dilakukan, ternyata ada elemen data yang masih kosong, dan elemen data tersebut justru menjadi kunci.

3) Anomali penghapusan (delete)

Anomali penghapusan terjadi apabila suatu record yang tidak terpakai dihapus, dan sebagainya akibat adanya data yang hilang atau dihapus.

CHAPTER 5

PENERAPAN NORMALISASI



5.1 Teknik Normalisasi

Teknik normalisasi merupakan proses pengelompokan data elemen menjadi tabel yang menunjukkan entitas dan relasinya.

Pada proses normalisasi selalu di uji pada beberapa kondisi, apakah ada kesulitan pada saat menambah, menghapus, mengubah, dan membaca data pada database.

Jika ada kesulitan dalam pengujian tersebut, maka tabel tersebut dipecahkan menjadi beberapa tabel lagi.

Secara umum relasi dalam database jika telah memenuhi kriteria bentuk normalisasi ketiga sudah didapat tabel yang optimal.

Tujuan dari normalisasi data adalah agar menghasilkan struktur record yang konsisten secara logic, mudah dimengerti, sederhana dalam pemeliharaan, mudah ditampilkan kembali, dan meminimalkan kerangkapan data guna meningkatkan kinerja sistem.

Pada proses perancangan database dapat dimulai dari dokumen dasar yang dapat dipakai dalam sistem sesuai dengan lingkup sistem yang akan dibuat rancangan databasenya.

Berikut ini adalah contoh dokumen mengenai faktur pembelian barang pada CV. Rantau Batuah Interindo.

CV. RANTAU BATUAH INTERINDO

JALAN BUNDA VI NO 04

PADANG

FAKTUR PEMBELIAN BARANG

Kode Supplier : S02

Nama Supplier : Hitachi

Tanggal : 22/04/2016

Nomor : 779

KODE	NAMA BARANG	QTY	HARGA	JUMLAH
R02	RICE COOKER CC3	10	150.000	1.500.000
TOTAL FAKTUR				1.500.000
Jatuh Tempo Faktur : 26/05/2016				

CV. RANTAU BATUAH INTERINDO

JALAN BUNDA VI NO 04

PADANG

FAKTUR PEMBELIAN BARANG

Kode Supplier : G01

Nama Supplier : Gobel Nusantara

Tanggal : 24/04/2016

Nomor : 998

KODE	NAMA BARANG	QTY	HARGA	JUMLAH
A01	AC SPLIT ½ PK	10	1.350.000	13.500.000
A02	AC SPLIT 1 PK	5	2.000.000	10.000.000
TOTAL FAKTUR				23.500.000
Jatuh Tempo Faktur : 26/05/2016				

Gambar 5.1 Faktur pembelian Barang

1) Bentuk tidak normal (Unnormalized Form)

Langkah pertama dalam melakukan normalisasi data adalah dimulai dengan bentuk tidak normal, dengan cara ini mencancumkan semua atribut data yang apa adanya.

Bentuk tidak normal ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti suatu format tertentu, dapat saja data tidak lengkap atau duplikasi.

Tabel 5.1 Bentuk tidak normal (*Unnormalized Form*)

Nofak	Kode Supp	Nama Supplier	Kode Brg	Nama Barang	Tanggal	Jatuh Tempo	QTY	Harga	Jumlah	Total
779	S02	HITACHI	R02	RICE COOKER CC3	22/04/2016	26/04/2016	10	150000	1500000	1500000
998	G01	GOBEL NUSANTARA	A01	AC SPLIT ½ PK	26/04/2016	26/04/2016	10	1350000	13500000	23500000
			A02	AC SPLIT 1 PK			5	2000000	10000000	

Pada tabel diatas adalah dengan menuliskan semua data yang ada yang akan direkam, data yang double tidak perlu ditulis.

Terlihat baris atau *record* yang tidak lengkap. Tabel diatas dikatakan tidak normal jika ada relasi mempunyai bentuk tidak datar (*non flat*), data disimpan apa adanya, tidak memiliki struktur file yang sama, dan relasi memuat atribut berulang-ulang (*repeating group*).

2) Bentuk Normal Pertama (1 NF)

Bentuk normal pertama yaitu, dengan memisah-misahkan data pada atribut-atribut yang tepat dan bernilai atomik, juga seluruh record harus lengkap dan bentuk file adalah datar (*flat file*).

Berikut bentuk normal pertama (1 NF).

Tabel 5.2 Bentuk Normal Pertama (1 NF)

Nofak	Kode Supp	Nama Supplier	Kode Brg	Nama Barang	Tanggal	Jatuh Tempo	QTY	Harga	Jumlah	Total
779	S02	HITACHI	R02	RICE COOKER CC3	22/04/2016	26/04/2016	10	150000	1500000	1500000
998	G01	GOBEL NUSANTARA	A01	AC SPLIT ½ PK	26/04/2016	26/04/2016	10	1350000	13500000	13500000
998	G01	GOBEL NUSANTARA	A02	AC SPLIT 1 PK	26/04/2016	26/04/2016	5	2000000	20000000	20000000

Pada normal pertama masih terjadi banyak kelemahan, terutama pada proses anomali insert, update, dan delete.

- Penyisipan (*Insert*)

Pada proses penyisipan tidak dapat memasukan kode dan nama suplier saja tanpa adanya transaksi pembelian, sehingga suplier baru bisa dimasukan jika ada transaksi pembelian.

- Pengubahan (*update*)

Kode dan nama suplier terlihat ditulis berkali-kali, bila nama suplier berubah, maka di setiap baris yang ada harus diubah, bila tidak data menjadi tidak konsisten.

- Penghapusan (*delete*)

Bila satu record diatas dihapus, misalnya nomor faktur 779, maka berakibat pada penghapusan data suplier S02 (Hitachi) padahal data tersebut masih diperlukan.

Permasalahan dalam normalisasi parsial, terhapusnya informasi ketika menghapus record, dan pembaharuan atribut bukan kunci

mengakibatkan sejumlah record berubah. Berdasarkan permasalahan diatas sehingga perlu dibuatkan normal kedua.

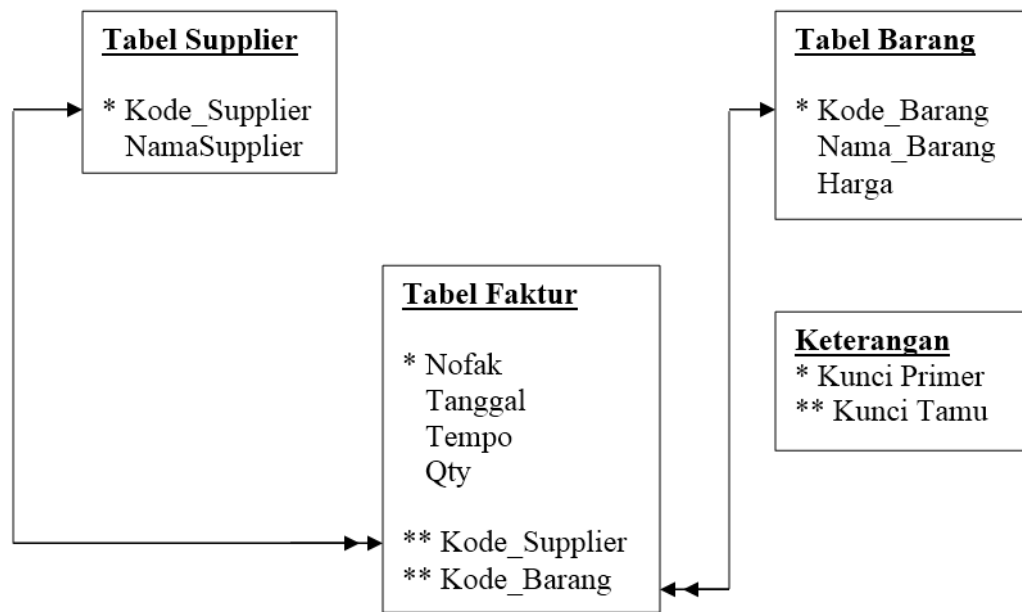
3) Bentuk Normal Kedua (2 NF)

Bentuk normal kedua yaitu dengan melakukan dekomposisi tabel di atas menjadi beberapa tabel dan mencari kunci primer dari setiap tabel tersebut dan atribut kunci haruslah unik.

Melihat permasalahan faktur diatas, maka dapat diambil beberapa calon kunci yaitu: Nofak, kode Suplier, Kode Barang.

Calon kunci tersebut nantinya bisa menjadi kunci primer pada tabel hasil dekomposisi. Dengan melihat normal pertama, kita dapat mendekomposisi menjadi tiga tabel beserta kunci primer yang ada yaitu; Tabel Suplier (kode_suplier), barang (kode_barang), dan faktur (nofak).

Dengan melihat ketergantungan fungsional atribut-atribut lain terhadap atribut kunci, maka didapatkan 3 tabel.



Gambar 5.2 Relasi Normal Kedua (2 NF)

Pemecahan tabel diatas, maka untuk pengujian bentuk normal pertama (1 NF) yaitu; insert, update, dan delete akan terjawab.

Kode suplier dan nama suplier baru dapat masuk kapan saja tanpa adanya transaksi pada tabel faktur.

Demikian pula untuk proses update dan delete untuk tabel suplier dan barang. Pada bentuk normal kedua tersebut masih terjadi permasalahan pada tabel faktur, yaitu:

- Atribut qty pada tabel faktur, tidak tergantung pada kunci utama, atribut tersebut bergantung fungsi kepada kode barang, hal ini dinamakan ketergantungan transitif dan harus dipecah menjadi dua tabel.

- Masih terdapat pengulangan, yaitu setiap kali satu faktur yang terdiri dari 5 macam barang maka harus 5 kali juga ditulis no faktur, tanggal, dan tempo.

4) Bentuk normal Ketiga (3 NF)

Bentuk normal ketiga mempunyai syarat, setiap tabel tidak mempunyai atribut yang bergantung transitif harus bergantung penuh pada kunci utama dan harus memenuhi bentuk normal kedua (2 NF).